# ScanToVR: An RGB-D to VR Reconstruction Framework

Hiranya Garbha Kumar†
The University of Texas at Dallas
Richardson, Texas, USA
hiranya@utdallas.edu

Ninad Arun Khargonkar
The University of Texas at Dallas
Richardson, Texas, USA
nxk180069@utdallas.edu

Balakrishnan Prabhakaran
The University of Texas at Dallas
Richardson, Texas, USA
bprabhakaran@utdallas.edu

## ABSTRACT

Growing public interest in Virtual Reality (VR) has made hardware like VR headsets and 3D sensors commonplace, but the resource intensive and tedious nature of developing interactive VR experiences continues to be a limiting factor for VR becoming mainstream. To this end, we propose a framework to convert input RGB-D scans taken from commodity RGB-D sensors into an interactive VR scene in a 3D environment in a few seconds. We use state-of-the-art 3D instance segmentation algorithms to extract object instances from the RGB-D scan. We then retrieve a matching CAD (Computer Aided Design) model using 3D shape embeddings from a common embedding space learnt using CAD models and RGB-D scan instances. We align retrieved CAD models to scan objects using a novel 7-DoF (Degrees of Freedom) pose estimation approach and replicate the structure of the scene using plane segmentation algorithms and recreate the scene in a Unity environment using the matched CAD models. We evaluate and compare our approach on key metrics, such as instance segmentation accuracy, object retrieval accuracy, CAD model alignment and total runtime, on a test set of over 300 scenes, taking an average of 1.8 seconds for the entire conversion across our test dataset. We also perform detailed runtime analysis on various aspects of our approach to understand potential limitations of existing and proposed algorithms, while comparing total runtime against existing works.

## CCS CONCEPTS

• Computing methodologies~Artificial intelligence~Computer vision~Computer vision problems~Reconstruction, Matching • Computer graphics ~ Graphics systems and interfaces ~ Virtual reality

## KEYWORDS

Semantic reconstruction, CAD model retrieval, 3D pose estimation, Virtual Reality

## 1 Introduction

Recent public interest in Virtual Reality (VR) has been driven by a combination of commodity VR headsets like Occulus Quest 2, Valve Index, HTC Vive and popular VR games and experiences such as Beat Saber, VR Chat, Half-Life: Alyx, etc. 3D sensors (such as Microsoft Kinect and Intel Realsense) and devices utilizing 3D camera systems (like Apple's recent iPad Pro and iPhone Pro) have also become commonplace. VR and Augmented Reality (AR) experiences are also used in various applications such as 3D virtual tours of Museums [2] and real estate [3], education, designing and prototyping of automobile [4], and even medical purposes like management of Phantom Pain from amputations [5][6]. There has also been significant interest in the development of large-scale VR experiences in the industry from different companies, Meta being one of the most prominent ones.

Despite the current adoption of VR and AR systems, several challenges remain on its path to becoming mainstream. One of the challenges is associated with the development of VR experiences. 3D experiences such as VR and AR interactions typically require a team of experts to spend a significant amount of time and resources to develop. These requirements increase significantly if we consider tasks such as indoor 3D reconstruction, which is our primary focus in this article, where there are a large number of objects and the layout of the scene as well as the placement, orientation, color, and texture of the objects need to be matched to the real-world scene. Thus, partial, or complete automation of such tasks is important to make creation of VR experiences accessible.

3D reconstruction techniques can be broadly classified into two types based on their output: dense 3D reconstruction and semantic reconstruction. There has been a significant amount of research in the scope of dense 3D reconstructions [1][10-18]. Early works such as KinectFusion [1][10] and StereoScan[11] introduced algorithms to create dense, accurate and smooth 3D surface reconstructions from RGB-D videos in real-time, while later works such as ElasticFusion [13] and BundleFusion [15] have improved upon various aspects of the process. While dense 3D reconstruction methods output a photorealistic output and are well-suited for certain VR experiences, there are some inherent issues with the approach. One of the primary issues with such dense 3D reconstruction methods is that missing data during capture can cause objects in the reconstruction to be incomplete. Due to the nature of RGB-D images, surfaces of objects not directly in sight of the sensor are not captured in the data. While this is partly mitigated by using an RGB-D video which allows capturing the scene from different perspectives to the blind spots, a significant

*Article Title Footnote needs to be captured as Title Note
†Author Footnote to be captured as Author Note
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
*WOODSTOCK'18, June, 2018, El Paso, Texas USA*
© 2018 Copyright held by the owner/author(s). 978-1-4503-0000-0/18/06...$15.00

amount of data remains missing in the reconstruction due to the cluttered nature of indoor scenes, for example, surfaces of objects placed against a wall are not captured. Dense 3D reconstructions also tend to be heavy in terms of data and static, i.e., individual objects in the scene cannot be interacted with. While recent semantic segmentation methods can partly solve this issue, such reconstructions are surface-level reconstructions, and as such, yield hollow objects. These drawbacks make them not an ideal choice for any interactive VR experience.

Semantic reconstruction techniques [22][23][24][25][30] overcome these issues by replacing the objects in the scene with semantically and geometrically similar CAD (Computer Aided Design) models, but they haven't been explored as much owing to technological and computational limitations. Their outputs tend not to be as photorealistic compared to dense 3D reconstructions, although recent advancements in game engines, specifically improvements in global illumination techniques such as Ray Tracing, have brought them very close if not on par with dense reconstructions. Representing objects in the scene by individual CAD models allows the resulting VR environment to include details such as texture, material, and other physical characteristics of each object. This can be pivotal for VR experiences and 3D simulations where the physical characteristics of objects in the scene play an important role.

There are various challenges associated with semantic 3D reconstruction, the primary ones being detecting objects in the scene, and retrieving and aligning CAD models corresponding to each object. Following recent advancements in efficient 3D convolutions [7] and the increase in computational power of GPUs, 3D Convolutional Neural Networks (CNNs) have become more accessible, leading to significant improvements in the performance of 3D object detection and segmentation algorithms [19][20][21]. Recent works in semantic reconstruction [22][23][24] have made significant strides towards addressing challenges involved with model retrieval and alignment but show minimal improvements in terms of object segmentation as most of them do not leverage state-of-the-art 3D object segmentation algorithms, bottlenecking the overall performance of such methods in some key metrics. Although there have been works pertaining to CAD model retrieval and alignment using a single RGB (2D) or RGB-D image [39][56][57][58], such methods tend to focus on a single object [56] or rely on inferred noisy depth data for reconstruction [39][57][58], leading to lower overall performance.

## 1.1 Proposed Approach

To address these challenges, we propose a framework to convert an input RGB-D scan into a VR scene. We utilize state-of-the-art 3D object segmentation algorithms to extract object instances from the scan, followed by 3D shape encoding-based model retrieval process to fetch matching CAD models for each object instance. To align the matched CAD models to object instances, we propose a novel 7-DoF (scale x, y, z, position x, y, z, and rotation along Z-axis) pose estimation approach. Following this, we segment and replicate planar structural components to recreate the layout of the indoor scene and recreate the scene in VR using a 3D game engine. We evaluate our proposed approach on a test dataset of more than 300 scenes, with the entire conversion taking an average of 1.8 seconds across our test dataset, on key metrics while performing extensive runtime analysis to identify potential bottlenecks. Although there are prior works that output an alignment of CAD models, to our knowledge, there are none that convert an input RGB-D scan into a VR scene. Owing to our design decisions, our approach demonstrates notable improvements in performance over existing methods on key metrics while being similar in total runtime.

# 2 Related Work

There exists a rich line of research on 3D object reconstruction from diverse kinds of input data such as multi-view RGB images, fused RGB-D scans, and point clouds. Extending it further, prior works have looked at reconstructing entire scenes in an online fashion and real-time constraints, as seen in KinectFusion [1][10], BundleFusion[15] and NeuralRecon [18]. However, most of the prior works focus on producing a mesh or TSDF output with pre-defined quality constraints and rely on color information for accurate representation. CAD model-based reconstruction differs from the above as the geometry of the detected object plays a key role and the end goal also differs as CAD models allow for better user interaction with the environment, more freedom to the designer of a VR environment to edit and change the object's parameters, while also not requiring post-processing for gaps in output due to noisy inputs. CAD model-based reconstructions are also different in the fact that they are volumetric reconstructions, as opposed to the mentioned methods, which are surface-level reconstructions and can completely miss surfaces that are not completely captured by the camera.

**Instance Segmentation and Object Detection.** For CAD-based retrieval methods, the initial step involves separating out shapes of interest from the input scene by using some variation of an object detection and segmentation method as seen in Gupta et al. [25]. There has been an increasing focus on leveraging RGB-D data for the task of 3D instance and semantic segmentation with recent works using modified 3D convolutional neural networks with customized operators for this task. Since we use existing, pre-trained models for this part of our framework, we only briefly discuss some of the more recent approaches working on point clouds. The unstructured nature of point clouds makes it difficult to directly adapt convolutional architectures but recent approaches like PointNet [19] tackle it via aggregating local neighborhood information for per-point feature extraction. Sparse convolution-based approaches like MinkowskiNet [7] define efficient convolution kernels for dealing with point clouds for feature extraction. Instance segmentation methods on point clouds typically perform semantic segmentation and then perform a grouping operation to separate out the class-wise instances like seen in JSIS3D [20], PointGroup [26] and OccuSeg [27]. SoftGroup [21] tries to leverage the advantages of both proposal-based and grouping-based methods in a two-stage pipeline

consisting of bottom-up grouping for proposal generation followed by top-down refinement.

**CAD Model Retrieval and Alignment.** Once a desired object is segmented out from the input RGB-D scan, the core problem is aligning a CAD model from an existing dataset to the object. Initial machine learning approaches for aligning CAD models to 3D scenes have been studied both from a classical, hand-tuned features [28][29][30] perspective and deep learning-based methods [25]. Song et al. [28] use linear SVMs (Support Vector Machines) corresponding to each model in a CAD dataset and iteratively go over the scene via a sliding window approach. Each SVM classifier is evaluated on the window to find the best match model from the dataset. Li et al. [27] obtain shape descriptors and key points for both noisy scans and 3D CAD models, encoding their local and global geometric features for efficient matching. Gupta et al. [25] leverage CNNs to output probable poses for objects already segmented by an existing segmentation method. Using some refinement over the pose outputs for the segmented object, the closest matching CAD model is inserted into the scene. ASIST [30] differs from the previous method as it tries to have a single, unified pipeline. It considers the semantic labeling and CAD model's retrieval and placement scene as a single optimization problem via an energy-based formulation. Additionally, it only considers point cloud inputs instead of RGB-D seen in previous works.

Deep learning approaches have found favor in comparison to earlier models primarily due to their robustness and reducing the need for hand-crafted shape features. Features descriptors are usually then obtained from deep neural networks trained on point cloud [19][32] and volumetric [31] shape classification tasks or based on implicit shape representation [33][34]. 3DMatch [35] developed a Siamese neural network as a feature extractor for matching input scans to CAD models based on establishing correspondences using the detected features. Similarly, Scan2CAD [22] proposes a 3D CNN approach to target similar correspondences and address some issues with domain gap of real-world scans and CAD models. They also introduce an annotated dataset for CAD model retrieval and alignment based on ScanNet [36] and ShapeNet [37]. Dahnert et al. [24] proposed a joint embedding space between CAD models and scanned objects that is learned via a triplet loss formulation based on existing scan-to-CAD model annotated datasets. SceneCAD [23] considers the layout reconstruction part of the overall problem where instead of independently assessing each object, a graph neural network is used to form connections between them to enforce consistency in the scene's reconstruction.

Other approaches have used a single RGB image of a scene to retrieve and align CAD models. Lim et al.[56] use an RGB image to estimate the relative pose between a provided CAD model and an object in the image. Huang et al. [58], Manni et al. [39], and Izadinia et al. [57] use an RGB image to perform semantic 3D reconstruction, relying on depth data inferred from deep learning models. As such predicted depth maps tend to be noisy and inaccurate, the algorithms tend to miss out on smaller 3D geometric details, resulting in errors in retrieval, positioning and alignment leading to the overall poor performance of the final algorithm.

**Virtual World Reconstruction.** Some of the algorithmic ideas have been primarily used in applications pertaining to Virtual and Augmented Reality (VR/AR) where creating an environment from scratch might be time consuming. RealitySkins [38] tries to address this problem by dynamically generating the environment based on the input scan from a user's head-mounted display (HMD). Snap2Cad [39] shows a system that utilizes the built-in RGB sensors on modern smartphones to reconstruct an object in AR via matching with CAD models for use in online multiplayer scenarios. VRFromX [40] also uses neural network-based methods for object retrieval and alignment with human-in-the-loop as a part of an interactive content creation tool. In a similar vein, TransforMR [41] is a mixed reality system for object substitution that leverages segmentation and accurate pose estimation for consistent replacement and use in character animation.

## 3 ScanToVR Design

The proposed framework aims to convert an input indoor RGB-D scan into a semantically and visually similar, interactable Unity 3D world. Figure 1 gives an overview of the working of the framework. To achieve this, we first replace objects in the RGB-D scan with CAD models. Contrary to dense 3D reconstructions such as [1][10][15][13], where the output is a photo-realistic, precise but static reconstruction of the input RGB-D scene, this method of replacing objects in the scene with CAD models allows the output to be interactable for VR experiences (since each CAD model can be manipulated independently in the VR scene). To replicate the layout of objects in the RGB-D scene, we estimate the 7-DoF pose of objects and use it to align CAD models to their corresponding object instances in the scene. Finally, we detect, segment, and reconstruct the structural components of the indoor scene, such as the floor, walls, etc. To summarize, given an input indoor RGB-D scan, our framework aims to:

1. Identify and segment instances of objects of interest in the scan.
2. For each instance, find and retrieve the closest matching CAD model, which in cases where there are no good matches, can be geometrically different to the instance.
3. Find 7-DoF transformations to align matched CAD models to respective object instances.

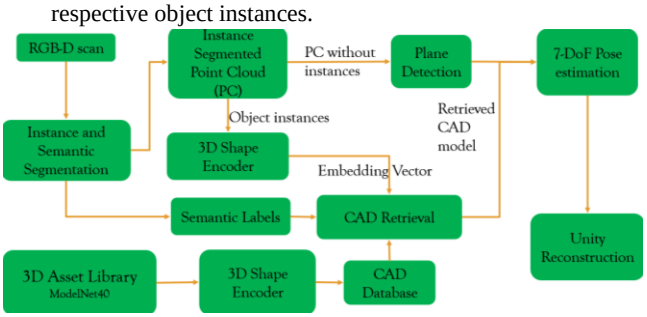


**Figure 1: Workflow of the proposed approach**

4. Generate a Unity VR scene that semantically resembles the input RGB-D scan using the matched CAD models.

### 3.1 Input Data

RGB-D is a widely used input modality to capture 3D data, with high-quality annotated indoor RGB-D datasets such as S3DIS [42], ScanNet [36], and SUN-RGB-D [43] available today. With growing interest and recent advances in 3D object detection, the number of annotated RGB-D datasets for tasks like 3D object segmentation, 3D semantic segmentation, etc., are also increasing. Two types of RGB-D data are widely available: RGB-D images and RGB-D scans. RGB-D images are similar to a 2D RGB image combined with depth data from a depth sensor. As such, they can have blind spots, i.e., missing depth data, due to occlusion from different objects in the scene. In addition, they have a restricted, single-perspective view of the scene. This is not ideal for reconstruction since objects with missing parts can be easily misidentified and the limited field of view would result in just a small part of the actual scene to be captured. RGB-D scans overcome this issue by combining several RGB-D images, either from an RGB-D video or individual RGB-D images taken from different viewpoints to cover for blind spots. As a result, RGB-D scans have better point density (and by extension capture finer details), capture complete objects more consistently, and capture more, if not all, of the indoor scene in a single scan. Due to these reasons, we use RGB-D scans as input to the proposed system.

## 3.2 3D Object Detection and Segmentation

In order to semantically reconstruct a given 3D scene, we first need to detect, and extract objects present in the input RGB-D scan. Although 2D object detection has matured significantly, the requirement of depth information to accurately place objects in the virtual 3D scene makes these algorithms non-ideal for 3D reconstruction. Despite the recent advances in methods to estimate depth in 2D images, they tend to be noisy and inaccurate [39][44] and are still a work in progress. 3D object detection and segmentation algorithms can overcome these issues as they directly work on 3D point clouds, and thus can provide accurate and precise depth information, but until recently, their computational requirements have been too high for widespread adoption. Due to recent breakthroughs in efficient 3D convolutions [7] and an increase in computational capacities of consumer grade GPUs, CNN-based 3D object detection and segmentation methods have made significant advances [21][20][45]. Recent works on 3D object detection and segmentation [7][21][19][32][20][21][45][46] can broadly be classified into bounding-box based methods (object detection algorithms) [7][21][19][32] and mask-based methods (semantic segmentation algorithms) [20][21][45][46]. Bounding box-based methods output a 3D bounding box for each object instance in the scene, while mask-based methods output a semantic and instance label for each point in the input point cloud. In indoor scenes, it is common to find objects in close proximity to one another. In such cases, a bounding box-based method can be imprecise for extracting points corresponding to a specific instance, since, in case of an overlap, the box can contain points from an adjacent object. These points can alter the geometric characteristics of the extracted object instance and cause objects to be misidentified into different classes. To avoid this problem, we opt for a mask-based method that outputs precise segmentation masks for each instance in the scene. In a segmentation mask, there can be only one instance and semantic label associated with each point $P_i$ in the scene point cloud. This eliminates the possibilities of overlap between segmented point clouds of different objects instances, solving our earlier problem.

## 3.3 Model Retrieval and alignment

Since our goal is to output an interactable VR scene, we aim to replace objects in the scene with semantically and geometrically similar 3D object models while aligning the CAD model to the replaced object. To this end, we propose a pipeline that matches, retrieves, and estimates the pose of the matched CAD model relative to the corresponding object instance from the RGB-D scan.

*3.3.1 Model Retrieval.* After extracting object instances from the RGB-D scene, our next challenge lies in fetching a CAD model that is semantically and geometrically similar to the object instance from our CAD model dataset. Previous works have explored different ways to address this challenge. Some early approaches have used template matching with hand crafted per-class templates [47]. More recent works have utilized large model datasets like ShapeNet [37] and ModelNet [48] while using 3D CNNs for their model retrieval tasks [22][24] using 3D shape vectors, while using a CNN to encode the 3D shape vectors. This ensures that the system can handle a very large number and wide variety of models for each semantic class while keeping the runtime computational costs low.

*3.3.2 7-DoF Pose Estimation* After fetching matching CAD models for each object instance in the scene, we need to re-position and transform the models to align them with their corresponding object instances. For this, transformation parameters for scale, position and rotation need to be calculated for the CAD model relative to the object instance in the scene. Various approaches have been explored to address this challenge. Early approaches have used 2D template matching [47], ICP or a variant of ICP [30], while more recent works have made use of deep learning algorithms to predict transformation parameters using 3D data [49] [22] or between 2D images and 3D data [39]. Although deep learning methods are fast (with a GPU), they can be inconsistent and inaccurate. Deterministic 3D approaches, like ICP and its variants, are accurate but tend to be computationally expensive and often require tuning various parameters to get desired results. With this in mind, we propose a novel approach to estimate 7-DoF pose that is both fast and accurate (refer Section 4.4).

*3.3.3 Layout Reconstruction* To output a complete VR scene, we aim to also reconstruct the structural components of the scene such as walls, floor, and ceiling. We use a deterministic plane detection algorithm utilizing Random sample consensus (RANSAC) to segment various planar structural components using plane detection methods and replicate these structures in VR using a pipeline similar to the one used for CAD model retrieval and pose estimation.

## 3.4 VR Reconstruction

After fetching CAD models corresponding to the object instances in the scene and replicating the primary structural components, our next step involves creating a VR world based on this information.

Among the wide variety of 3D game engines available today, Unity and Unreal Engine are widely used, and thus, have extensive support and documentation available. We chose Unity for our implementation, but with minor modifications, the framework can be made to work with Unreal Engine or most other game engines as well.

To summarize our design choices for different parts of the system (overview show in Figure 1), we:

1. Detect object instances in input RGB-D scans using a state-of-the-art 3D instance segmentation model
2. Retrieve closest matching CAD models using learned 3D shape encoding vectors
3. Estimate the relative 7-DoF pose between the CAD model and object instance using 2D and 3D algorithms
4. Detect and replicate planar structural components in the scene using RANSAC
5. Construct a VR environment in Unity using retrieved CAD models, their corresponding pose estimations and the structural components of the indoor scene.

# 4 Implementation

## 4.1 Datasets

Due to a growing interest in 3D object segmentation, the number of annotated indoor RGB-D datasets have significantly increased in recent years. Although there is an abundance of synthetic indoor segmentation datasets like SunCG [50], RoboTHOR [51], Structured3D [52] and Hypersim [53] owing to the semi-autonomous nature of generating virtual scene annotations, point clouds generated by such datasets using model sensors tend to be much better in quality compared to RGB-D scans captured in the real world. This discrepancy can severely impact the performance of algorithms trained on such synthetic data, when testing on real-world data. Due to the challenges associated with manual annotations, real-world indoor RGB-D datasets are limited, with Stanford 3D Indoor Scene Dataset [42], ScanNet [36] and SUN RGB-D [43] being some of the widely used datasets. Among these, ScanNet [36] is by far the largest dataset with 1513 scenes and has been used to evaluate various state-of-the-art 3D segmentation algorithms. We also found ScanNet data to be better than existing datasets in annotations and quality of reconstruction. Thus, it is the dataset we chose to evaluate our method on.

Among CAD model datasets, the most widely used datasets for object classification and CAD model retrieval tasks are ModelNet [48] and ShapeNet Core [37]. Although annotations in ShapeNet are more information rich, the extra information doesn't benefit our approach. As such, we chose to use ModelNet[48] for our implementation due to its significantly larger collection of models (over 150k models) resulting in a more varied collection of models for each semantic category.

## 4.2 Instance Segmentation

The introduction of efficient 3D convolutions has made training 3D deep learning algorithms more accessible, leading to a significant increase in research on this topic in recent years. Notable recent works on 3D instance segmentation include [21][20][54[26][46]. For our approach, we select the semantic categories common to both the RGB-D and CAD model datasets we are using. Further, we only focus on large object classes like furniture, as the general layout of a scene is primarily defined by such large objects. This also helps us output a consistent and higher quality reconstruction, as we've found smaller objects to introduce more errors in the system (refer **Error! Reference source not found.**). This narrows down our selected number of semantic categories to 8: bathtub, bed, bookshelf, chair, desk, sofa, table, and toilet.

Based on the performance of existing 3D instance segmentation approaches on the selected classes, we opted to implement the algorithm proposed by Vu et al. [21]. This work builds upon the work by Chen et al. [46] by modifying and improving on the instance proposal pipeline. We keep the architecture proposed in the paper unchanged and train the network on top of the existing checkpoint from [46] with the reduced number of classes (8). The model is implemented using PyTorch, which is a Python based deep-learning framework, and trained on 120k iterations using Adam optimizer with a learning rate of 0.04 and voxel size of 2cm. The instance segmentation pipeline outputs a mask and a semantic label corresponding to each predicted instance in the scene. Figure 2 shows a qualitative comparison between ground truth annotations and the generated semantic segmentation mask for a test scene from ScanNet v2 dataset. Note that predicted segmentation is only shown for categories of interest, not for all the categories output by the algorithm.

Used exactly as proposed in the original work, the algorithm often erroneously labels small cluster of points in the scene. While these small clusters don't affect the overall mean average precision (a key performance metric for such algorithms) of the algorithm due to their relatively small size, they can severely impact our reconstruction, resulting in the final VR scene littered by random small objects around the scene. To filter out these noisy predictions, which were primarily caused by low confidence instances and instances with a small number of points, we add a threshold for confidence score (Tconf) and number of points in the instance (Tpoints). We evaluate the algorithm on different configurations of Tconf and Tpoints to quantify the effects of varying these parameters on the performance, and the results can be found in [refer table]. Based on our studies, we selected Tconf=0.5 and Tpoints=512 as the ideal configuration for our pipeline.

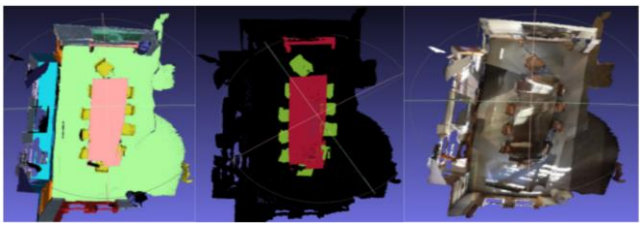

**Figure 2: Left to right: RGB-D scan data, ground truth semantic annotations, predicted semantic mask**

## 4.3 Model Retrieval

After we extract object instances, we find the closest matching CAD models for each of them. Note that we don't use any thresholds to distinguish a match as good or bad, and in some instances, the closest CAD model to a given instance can have significant geometrical differences. For a set of $m$ extracted object instances $I = \{I_0, I_1, I_2, I_3, \dots I_{m-1}\}$, we find a set of matching CAD models $O = \{O_0, O_1, O_2, O_3, \dots O_{m-1}\}$, such that $O \in O_{ds}$, where $O_{ds}$ represents our 3D object dataset, and for $x \in [0, m)$, $O_x$ matches $I_x$ semantically and geometrically. To keep our retrieval process fast while being able to fetch from a large collection of models, we opt to utilize 3D shape embeddings with vector based nearest-neighbor search. Although hand-crafted 3D shape descriptors based on local geometric features have been used in previous works, recent advances in 3D CNNs have enabled training deep learning models for object classification which produce 3D embedding that outperform such hand-crafted features. Even for such methods, object retrieval poses a significant challenge as both clean and complete CAD models, and incomplete and noisy real-world object instances, need to be mapped to a common embedding space. We evaluate semantic label assisted model-retrieval in contrast to purely encoding based retrieval.

*4.3.1 3D Shape Encoding.* For generating 3D shape embeddings, we adapt the work by Choy et al.[7] The network is an implementation of the work by Pratt et al. [55] built on a versatile and efficient framework for 3D convolutions, Minkowski Engine, and performs close to state-of-the-art on object classification tasks. We modify the network by passing the embedding layer of the network through a max pooling layer followed by a sigmoid layer. The resulting 1024 length vector is then used as a 3D shape embedding vector. Since the network takes point clouds as input and CAD models are mesh files, we first convert them to point clouds by sampling 2048 points uniformly from the surface of the CAD models. The number of points is fixed to 2048 due to the requirement of the network's architecture. Object instances with more points than 2048 are randomly down sampled, whereas those less than 2048 points are up sampled by duplication. The point clouds are normalized and their mean is shifted to origin (0,0,0). We pass the resulting point cloud through the network to get semantic predictions and encoding vectors.

**Training.** Training the network solely on CAD models or on object instances yields poor results on the other dataset, while CAD models followed by training on object instances and vice versa results in the network's performance on the prior dataset degrading significantly. To effectively learn a common embedding space for both CAD models and object instances, we employ a mixed training strategy where we train the network on both object instances and CAD models together. The results of these experiments are shown in Table 2. We train the network using a voxel size of 1cm, batch size of 32, for 10k steps, with the forward pass switching between samples from each dataset after each step.

*4.3.2 CAD Model Database.* To make CAD model retrieval faster at runtime, we preemptively pass all the models in our CAD model dataset through our encoding network and add the vectors to a K-Nearest Neighbor (KNN) based tree, which forms our CAD model database. We use cosine distance as the distance metric to measure similarity between vectors in the database. Cosine distance between two vectors A and B is defined as

$$cosine\ distance = 1 - \frac{A \cdot B}{\|A\|\|B\|} = 1 - \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} - (1)$$

At runtime, an object instance is encoded into a vector and queried against the database, which returns the CAD model with the lowest cosine distance to the instance. Since there is no threshold distance set to distinguish a match as good or bad, in some instances, the closest CAD model to a given instance can be geometrically different.

## 4.4 Pose Estimation

To align matched CAD models to object instances, we need to estimate the relative pose of the CAD model with respect to the object instance. To this end, we propose a fast and accurate method to estimate the 7-DoF pose (position x, y, z, scale x, y, z, and rotation along the Z axis) of the CAD model relative to the object instance. We make the assumption that all objects in the scene are placed upright, i.e., they are only rotated along the Z-axis. Since our goal is to re-create the layout of a room or indoor scene with the object categories being large objects like furniture, which are predominantly placed in an upright position, this assumption works in most cases for our situation. In cases where the objects are not placed upright, the objects will still be placed in the correct position in the reconstruction but will be placed upright as the algorithm doesn't account for rotation along X and Y axes. This can be easily corrected in the final output by manually rotating these objects along the required axes.

To estimate the scale and position, we draw a 3D bounding box around the object instance. Since an axis-aligned bounding box can misrepresent the scale of the instance significantly unless it is aligned with all the axes, we utilize a 3D Oriented Bounding Box (OBB).

*4.4.1 3D Oriented bounding box calculation.* Although there are different ways to estimate a 3D minimum-volume OBB around an object, accurate 3D methods tend to computationally expensive while other methods trade-off accuracy for being faster. For our approach, we utilize a hybrid 2.5D method to compute 3D OBB around an object. We project the object point cloud to the XY plane (top-down view) and draw a minimum area rectangle around the projection using a rotating caliper approach [10]. The 2D box is represented by:

$$minAreaRect = (X_{\text{center}}, Y_{\text{center}}, X_{\text{extent}}, Y_{\text{extent}}, \theta) - (2)$$

where $(X_{\text{center}}, Y_{\text{center}})$ are the coordinates for the center, $(X_{\text{extent}}, Y_{\text{extent}})$ are the dimensions of the rectangle in the XY plane, and $\theta$ is the angle of rotation of the rectangle along the Z-axis. Due to our assumption of the rotation of the object being constrained to the Z-axis, the dimensions and center of this rectangle accurately represent the dimensions and center of the object in X and Y axes. We convert this 2D rectangle into a 3D OBB using the following equations to calculate the Z coordinates and yaw:

$$Z_{extent} = \frac{Z_{max} - Z_{min}}{2} - (3)$$

$$Z_{center} = Z_{min} + Z_{extent} - (4)$$
$$yaw = \theta - (5)$$

where $(Z_{max}, Z_{min})$ are the Z-axis bounds of the object point cloud. This yields us coordinates $(X_{\text{center}}, Y_{\text{center}}, Z_{center})$ for the center and $(X_{\text{extent}}, Y_{\text{extent}}, Z_{extent})$ for the scale of the 3D OBB. Figure 3 shows OBBs calculated using the proposed approach for various point clouds corresponding to both CAD models (in black) and object instances from RGB-D scene (in red). The illustration demonstrates that our algorithm calculates accurate minimum-volume OBBs for each point cloud.

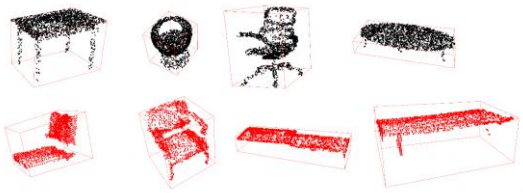

**Figure 3: Calculated OBBs using our approach. The points in black (first row) are from CAD models, points in red (second row) are from segmented object instances from RGB-D scans.**

*4.4.2 Scale and Position Calculation.* The center of the 3D OBB is considered as the position of the object. Note that we avoid taking the mean of all points in the instance point cloud as the center since, depending on the geometric shape of the object and variation in density of the captured point cloud, it might not represent the true center of the instance. We calculate the relative 3D scale of the object instance with respect to the scale of the fetched CAD model using the following equation

$$scale = \left(\frac{S_x^i}{S_x^m}, \frac{S_y^i}{S_y^m}, \frac{S_z^i}{S_z^m}\right) - (6)$$

where, $(S_x^i, S_y^i, S_z^i)$ are the dimensions of the oriented bounding box for the object instance and $(S_x^m, S_y^m, S_z^m)$ are the dimensions of the oriented bounding box for the CAD model.

*4.4.3 Rotation estimation.* Next, we apply the previously calculated position and scale transformations to the CAD model to align it to the object instance. To calculate the Z-axis rotation of the model with respect to the instance, we rotate the model around the Z-axis in steps (keeping the object instance static) and calculate Chamfer distance between the CAD model and object point clouds at each step. We select the best alignment angle as the one with the minimum Chamfer distance. The algorithm used is detailed in Algorithm 1. The iterations in Algorithm 1 are executed in parallel to improve runtime. In addition, we further minimize the amount of time taken by Chamfer distance calculations by uniformly down sampling the input point clouds to 512 points, which we found to have a good balance between low calculation time while not losing finer geometric details based on empirical studies. Figure 4 shows the results of applying our alignment algorithm for qualitative analysis. Each row demonstrates the process for a different CAD model and object instance pair. The first column containing black (CAD model), and red (object instance) point clouds shows the initial positions without any scaling or repositioning of the CAD model. The second column shows the CAD model point cloud rescaled and repositioned to match the instance. The last column shows the final point clouds after 7-DoF alignment. As shown, it demonstrates good performance on incomplete object instances as well (shown in the second row).

Input: Model point cloud $P_m$, Instance point cloud $P_i$
Output: alignment angle α
Algorithm:
For $\alpha_1 \leftarrow 0\ to\ 360, increments\ of\ 30$:
    $P_{m\_rot} \leftarrow$ Rotate $P_m$ by $\alpha_1$
    Calculate and record Chamfer distance between $P_{m\_rot}$ and $P_i$
$\alpha_2 \leftarrow \alpha_1$ with minimum Chamfer distance in the above loop
For $\alpha_3 \leftarrow \alpha_2 - 30\ to\ \alpha_2 + 30, increments\ of\ 10$:
    $P_{m\_rot} \leftarrow$ Rotate $P_m$ by $\alpha_3$
    Calculate and record Chamfer distance between $P_{m\_rot}$ and $P_i$
$\alpha_4 \leftarrow \alpha_3$ with minimum Chamfer distance in the above loop
For $\alpha_5 \leftarrow \alpha_4 - 5\ to\ \alpha_4 + 5, increments\ of\ 2$:
    $P_{m\_rot} \leftarrow$ Rotate $P_m$ by $\alpha_5$
    Calculate and record Chamfer distance between $P_{m\_rot}$ and $P_i$
$\alpha \leftarrow \alpha_5$ with minimum Chamfer distance

**Algorithm 1: Proposed algorithm to find best alignment angle**

## 4.5 Layout detection

To recreate the structural layout of the indoor environment in the VR scene, we detect and estimate the orientation of different planar structural components in the scene. We use RANSAC to detect and segment these structures. Based on empirical study on parameters for RANSAC, for our case, we found minimum number of points of 10000, distance threshold of 10cm, and 10000 iterations to provide the best results. We follow this up with our pose estimation method (Section 4.3) to estimate the position, scale, and orientation of the extracted planes. Note that since there are no CAD models associated with structural components, we use a unit cube point cloud $P_u$ to find the relative pose instead. The components are then formed by deforming a unit cube model in Unity.

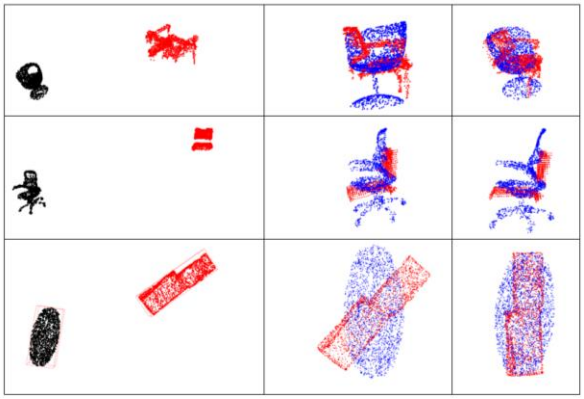

**Figure 4: Visualization of our 7-DoF alignment process.**

## 4.6 Unity Reconstruction

The object-CAD associations and their corresponding transformation parameters for each input RGB-D scan are written to a JSON file which is parsed by Unity. Inside Unity, we load the CAD models (use a unit cube for structures) and apply transformation parameters and color the CAD models based on their corresponding object instances. Color for each object instance is determined by averaging the color values of all points in the instance point cloud (or points corresponding to the plane for structures). The result is a VR scene that semantically resembles the input RGB-D scan. Figure 5 shows an example VR scene produced (right) using an input RGB-D scan from ScanNet dataset (left).

## 5 Results

We evaluate our approach on various metrics while comparing with existing works, as detailed in the following sub-sections. The following evaluations are done on the test set of ScanNet (containing 315 scenes) using the entire ModelNet dataset as the model database and the results are averaged across all test scenes.

### 5.1 Instance segmentation

To evaluate the performance of the selected instance segmentation algorithm with varying confidence threshold (Tconf) and number of minimum points per instance (Tpoints), we compare voxel-wise precision, recall and F1 scores of the algorithm in **Error! Reference source not found.**. These results show that Tpoints has a larger effect on mAP than Tconf, especially beyond Tpoints=1024, but tuning Tconf also results in considerable improvement in performance.

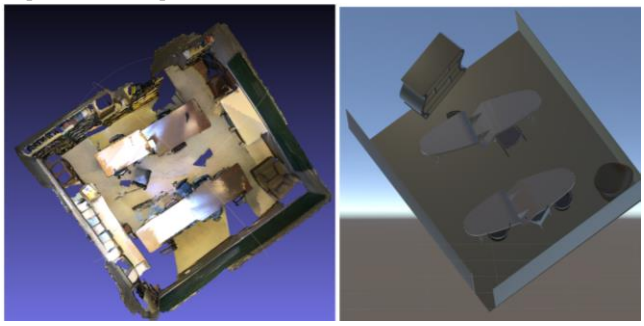

**Figure 5: Visualization of a scene from ScanNet converted into a Unity scene.**

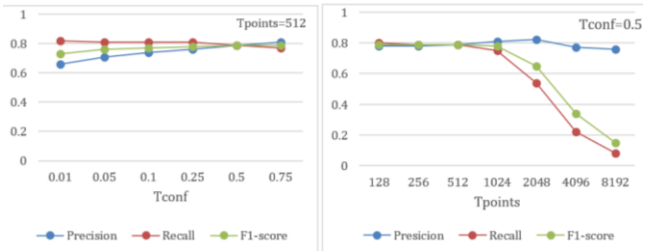


**Figure 6: Evaluating the effect of changing Tconf with Tpoints=512 (left) and changing Tpoints with Tconf=0.5 (right) on performance**

In Table 1, we compare the network's segmentation performance to the best model from Scan2CAD [22] on common categories, which is the most recent work on model retrieval and alignment utilizing and evaluating any form of instance segmentation. The applied approach outperforms [22] by 15% on average.

**Table 1: Comparison of voxel-wise F1 scores for object segmentation.**

| | Bathtub | bookshelf | chair | table | sofa | average |
|---|---|---|---|---|---|---|
| Scan2CAD [22] | 0.6 | 0.59 | 0.74 | 0.57 | 0.75 | 0.65 |
| Applied approach | 0.87 | 0.71 | 0.85 | 0.81 | 0.75 | 0.80 |

### 5.2 Encoder performance

We measure the performance of our model retrieval pipeline by comparing the semantic labels of the fetched CAD model to the ground truth instance annotation from our RGB-D scan dataset, ScanNet. To quantify this, we use a per-instance F1 score, which is defined as follows:

$$F1\ score = \frac{2 * precision * recall}{precision + recall}$$

$$precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

$$recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

Table 2 shows the results of our retrieval pipeline on data from ScanNet and ModelNet. Note that the F1 scores here are per-instance, not per voxel, i.e., a match is considered successful with the IoU is above 0.5 and the semantic label of the matched model matches that of the ground truth instance. The results show that although relatively low recall score on ScanNet instances brings down the F1 score, the network shows good performance over both datasets. This demonstrates the success of our mixed dataset training strategy for the encoder network. Weighted average (based on number of samples per class) of F1 scores shows that the network poor performance is on classes with low number of occurrences.

### 5.3 CAD model retrieval performance

In Table 3 we evaluate two retrieval approaches, one aided by semantic predictions from the instance segmentation network, and the other purely relying on embedding vectors for retrieval based on semantic per-instance F1 scores while comparing it to existing

works. Results indicate the approach aided by semantic labels significantly outperforms purely embedding based retrieval, hinting at further possible improvements to the training strategy or architecture of the encoder network, performing on par with existing approaches while handling more categories.

**Table 2: Performance of the shape encoder network on CAD dataset (ModelNet) and object instances from ScanNet.**

| | ScanNet | | | ModelNet | | |
|---|---|---|---|---|---|---|
| Class | Precision | Recall | F1-score | Precision | Recall | F1-score |
| bathtub | 1 | 0.52 | 0.68 | 1 | 0.72 | 0.84 |
| bed | 0.74 | 0.4 | 0.52 | 0.88 | 0.98 | 0.92 |
| bookshelf | 1 | 0.69 | 0.82 | 1 | 0.87 | 0.93 |
| chair | 0.91 | 1 | 0.95 | 0.95 | 0.91 | 0.93 |
| desk | 0.73 | 0.38 | 0.5 | 0.76 | 0.79 | 0.77 |
| sofa | 0.88 | 0.29 | 0.44 | 0.89 | 1 | 0.94 |
| table | 0.8 | 0.96 | 0.87 | 0.81 | 0.85 | 0.83 |
| toilet | 0.92 | 0.98 | 0.95 | 0.96 | 0.96 | 0.96 |
| Balanced Average | 0.8725 | 0.6525 | **0.72** | 0.91 | 0.88 | **0.89** |
| Weighted Average | 0.88 | 0.88 | **0.87** | 0.9 | 0.9 | **0.9** |

**Table 3: Semantic (per-instance) F-1 scores for two configurations of the model retrieval pipeline compared to existing work.**

| Method | ASIST [30] | Label-assisted | Encoding only |
|---|---|---|---|
| bathtub | - | 1 | 0.68 |
| bed | 0.96 | 0.97 | 0.52 |
| bookshelf | - | 0.8 | 0.82 |
| chair | 0.96 | 0.99 | 0.95 |
| desk | - | 0.94 | 0.5 |
| table | 0.91 | 0.94 | 0.44 |
| sofa | 1 | 0.92 | 0.87 |
| toilet | 1 | 1 | 0.95 |
| Average | 0.96 | 0.95 | 0.72 |

## 5.4 Alignment

To quantify the performance of our alignment algorithm, we calculate the voxel-wise F1-scores between ground truth object instances and matched CAD models with a voxel size of 10 cm. The score can vary between 0 and 1, 0 indicating no common voxels between the instance and CAD model, and 1 indicating a perfect match. While this is not the ideal metric to quantify alignment, as object instances from scan data can still have missing parts, leading to a lower score despite perfect alignment, in general we found it to be a good representation of alignment. Table 4 shows the average voxel-wise F1 scores on a per-class basis.

**Table 4: Voxel-wise F1 scores between aligned CAD models and ground truth instances.**

| Class | Voxel F1 score |
|---|---|
| bathtub | 0.87 |
| bed | 0.89 |
| bookshelf | 0.51 |
| chair | 0.41 |
| desk | 0.59 |
| sofa | 0.63 |
| table | 0.72 |
| toilet | 0.39 |
| Average | 0.63 |

## 5.5 Runtime Analysis

To evaluate the runtime of our framework, we measure time taken for various steps of the framework averaged across all scans in ScanNet. We run our pipeline on a system equipped with an Intel Core i7-9700K processor and a RTX 2070 GPU. Table 5 shows the individual runtimes of various algorithms involved.

**Table 5: Runtime analysis for various processes in the algorithm in milliseconds.**

| Task | Average Time taken (ms) |
|---|---|
| Instance segmentation | 544.2 |
| 3D model retrieval | 1.2 |
| Shape encoding | 13.6 |
| 3D pose estimation | 1187.3 |
| Unity reconstruction | 70.6 |
| **Total** | **1817.1** |

The time taken by the encoding and retrieval algorithms don't vary significantly as they only operate on a single instance at a time. Meanwhile, as shown in Figure 7 and Figure 8, the time taken by our instance segmentation and 3D pose estimation algorithms scale up with the number of instances. Note that the scale of number of instances is different for both graphs as we consider number of total ground truth objects for instance segmentation, but only instances of selected classes go through the pose estimation method. Our pose estimation algorithm is partially sequential (limited by the number of CPU threads), leading to the runtime scaling almost linearly with the number of instances in the scene and contributing most to the overall time taken. Note that more recent hardware can potentially result in lower runtimes. In Table 6 we compare the overall runtime of the proposed framework in different scenarios to recent works on CAD model retrieval and alignment. Across the ScanNet test dataset of 315 scenes, our framework takes about 1.8 seconds to convert an input RGB-D scan into a Unity VR scene on average, which is comparable to the current state-of-the-art. Table 6 shows a comparison with existing works for different number of instances. Note that the number of instances used by SceneCAD [23] (# objects 1, 5, 26) don't exactly match with ours, [22] and [49], but we've aligned them to the nearest corresponding number in our evaluation.

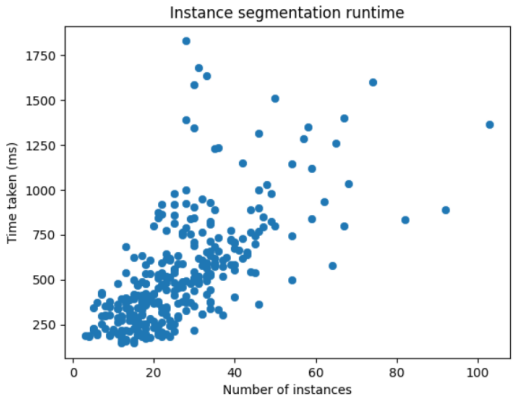


**Figure 7: A plot showing the runtimes (y-axis) versus number of instances (x-axis) of Instance segmentation algorithm.**

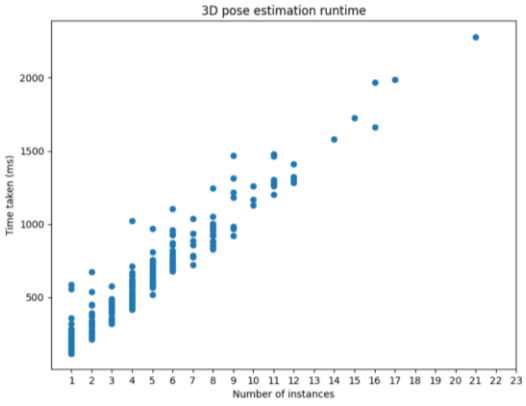


**Figure 8: : A plot showing the runtimes (y-axis) versus number of instances (x-axis) of the 7-DoF pose estimation algorithm**

**Table 6: Comparison of total runtime for varying number of instances, # instances for SceneCAD mentioned in parenthesis.**

| Number of objects | 7 | 16 | 20 |
|---|---|---|---|
| Scan2CAD [22] | 288.6 | 565.86 | 740.3 |
| SceneCAD [23] | 2 (5) | - | 2.6 (26) |
| End-to-End [49] | 0.62 | 1.11 | 2.6 |
| Our approach | 1.14 | 1.98 | 2.58 |

## 6 Discussions and Conclusion

We present an approach to convert an RGB-D scan to a 3D Unity VR scene. The proposed method detects and segments objects in the RGB-D scene, retrieves semantically and geometrically similar CAD models, aligns the CAD models to object instances, replicates the layout and generates a VR scene in Unity, taking an average of 1.8 seconds for the entire reconstruction across our test dataset. We evaluate the proposed approach on several key metrics on a dataset of over 300 scenes, while comparing it to existing works. Our approach leverages state-of-the-art instance segmentation architecture to output a more semantically accurate scene than existing algorithms while maintaining a similar runtime. The results demonstrate the effectiveness of our approach in reconstructing a VR scene from an RGB-D scan. The framework has several applications, including but not limited to, helping VR developers and designers quickly generate a base design to work on (especially for indoor replication tasks), modelling real-world indoor environments in 3D for simulations, or making development of VR games and experiences more accessible to smaller teams. The low total runtime of the algorithm, with further optimizations and improvements, is promising for real-time virtual applications like remote VR meetings in a common physical space.

Despite the framework's performance, closer examination of failure cases suggests that there are some methods to improve upon in our proposed approach. The retrieval process finds the closest matching CAD model, and without any checks, can find geometrically dissimilar matches. Thresholding based on cosine distance, or a human-in-the-loop approach can help in this regard. The retrieved CAD models sometimes contain significant differences in finer geometric details, suggesting that an improved retrieval pipeline, maybe using a shape encoder with better architecture or training strategy, can further improve performance. The relatively small number of classes of objects we can detect and replicate is a limitation, as this results in a considerable percentage of the total number of objects in the scene, especially smaller objects, being not replicated in the final VR scene. Improvements in instance segmentation and shape encoding algorithms can help in this regard. The overall runtime performance of our approach is primarily bottlenecked by the pose estimation approach and better optimizations to the algorithm can improve the runtime significantly. Although RANSAC is a proven approach for plane detection, it needs some parameter tuning based on the dataset and doesn't yield ideal results in some scenarios, for example, it misses small wall sections or walls with large windows. In addition, layout detection using plane segmentation often yields to segments of walls disconnected to the floor in the reconstructed VR scene. Edge or corner-based layout detection methods could yield better results in this case.

With this method, we take a step towards automating the replication of real-world scenes in VR and autonomous generation of VR environments, making generating a VR scene more accessible. Despite the promising applications of such methods and recent advances in 3D deep learning algorithms, there has been limited work done in this scope, and we hope there will be more interest in this field in the future.

## REFERENCES

[1] Newcombe, Richard A., et al. 'KinectFusion: Real-Time Dense Surface Mapping and Tracking'. *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, IEEE, 2011, https://doi.org10.1109/ismar.2011.6162880.

[2] https://www.louvre.fr/en/online-tours

[3] https://home3ds.com/3d-scan-home/

[4] https://www.bmw.com/en/events/nextgen/global-collaboration.html

[5] Annaswamy, Thiru M., et al. 'Clinical Feasibility and Preliminary Outcomes of a Novel Mixed Reality System to Manage Phantom Pain: A Pilot Study'. *Pilot and Feasibility Studies*, vol. 8, no. 1, Springer Science and Business Media

LLC, Oct. 2022, p. 232, https://doi.org10.1186/s40814-022-01187-w.

[6] Sano, Yuko, et al. 'Reliability of Phantom Pain Relief in Neurorehabilitation Using a Multimodal Virtual Reality System'. 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), IEEE, 2015, https://doi.org10.1109/embc.2015.7318897.

[7] Choy, Christopher, Junyoung Gwak, et al. '4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks'. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2019, https://doi.org10.1109/cvpr.2019.00319.

[8] Works Cited

[9] Choy, Christopher, Junha Lee, et al. 'High-Dimensional Convolutional Networks for Geometric Pattern Recognition'. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2020, https://doi.org10.1109/cvpr42600.2020.01124.

[10] Izadi, Shahram, et al. 'KinectFusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera'. Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, 2011, pp. 559–568.

[11] Geiger, Andreas, et al. 'StereoScan: Dense 3d Reconstruction in Real-Time'. 2011 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2011, https://doi.org10.1109/ivs.2011.5940405.

[12] Nießner, Matthias, et al. 'Real-Time 3D Reconstruction at Scale Using Voxel Hashing'. ACM Transactions on Graphics, vol. 32, no. 6, Association for Computing Machinery (ACM), Nov. 2013, pp. 1–11, https://doi.org10.1145/2508363.2508374.

[13] Whelan, Thomas, et al. 'ElasticFusion: Dense SLAM without A Pose Graph'. Robotics: Science and Systems XI, Robotics: Science and Systems Foundation, 2015, https://doi.org10.15607/rss.2015.xi.001.

[14] Choi, Sungjoon, et al. 'Robust Reconstruction of Indoor Scenes'. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2015, https://doi.org10.1109/cvpr.2015.7299195.

[15] Dai, Angela, et al. 'Bundlefusion: Real-Time Globally Consistent 3d Reconstruction Using on-the-Fly Surface Reintegration'. ACM Transactions on Graphics (ToG), vol. 36, no. 4, 2017.

[16] Han, Lei, et al. 'Real-Time Globally Consistent Dense 3D Reconstruction with Online Texturing'. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 3, Institute of Electrical and Electronics Engineers (IEEE), Mar. 2022, pp. 1519–1533, https://doi.org10.1109/TPAMI.2020.3021023.

[17] Xu, Yabin, et al. 'HRBF-Fusion: Accurate 3D Reconstruction from RGB-D Data Using on-the-Fly Implicits'. ACM Transactions on Graphics, vol. 41, no. 3, Association for Computing Machinery (ACM), June 2022, pp. 1–19, https://doi.org10.1145/3516521.

[18] Sun, Jiaming, et al. 'NeuralRecon: Real-Time Coherent 3D Reconstruction from Monocular Video'. ArXiv [Cs.CV], 1 Apr. 2021, http://arxiv.org/abs/2104.00681. arXiv.

[19] Qi, Charles R., et al. 'PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation'. ArXiv [Cs.CV], 2 Dec. 2016, http://arxiv.org/abs/1612.00593. arXiv.

[20] Pham, Quang-Hieu, Thanh Nguyen, Binh-Son Hua, Gemma Roig, and Sai-Kit Yeung. "JSIS3D: Joint semantic-instance segmentation of 3D point clouds with multi-task pointwise networks and multi-value conditional random fields." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8827-8836. 2019.

[21] Vu, Thang, et al. 'Scalable SoftGroup for 3D Instance Segmentation on Point Clouds'. ArXiv [Cs.CV], 17 Sept. 2022, http://arxiv.org/abs/2209.08263. arXiv.

[22] Avetisyan, Armen, Manuel Dahnert, et al. 'Scan2CAD: Learning CAD Model Alignment in RGB-D Scans'. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2019, https://doi.org10.1109/cvpr.2019.00272.

[23] Avetisyan, Armen, Tatiana Khanova, et al. 'SceneCAD: Predicting Object Alignments and Layouts in RGB-D Scans'. ArXiv [Cs.CV], 27 Mar. 2020, http://arxiv.org/abs/2003.12622. arXiv.

[24] Dahnert, Manuel, et al. 'Joint Embedding of 3D Scan and CAD Objects'. ArXiv [Cs.CV], 19 Aug. 2019, http://arxiv.org/abs/1908.06989. arXiv.

[25] Gupta, Saurabh, et al. 'Aligning 3D Models to RGB-D Images of Cluttered Scenes'. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2015, https://doi.org10.1109/cvpr.2015.7299105.

[26] Jiang, Li, et al. 'PointGroup: Dual-Set Point Grouping for 3D Instance Segmentation'. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2020, https://doi.org10.1109/cvpr42600.2020.00492.

[27] Han, Lei, et al. 'OccuSeg: Occupancy-Aware 3D Instance Segmentation'. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2020, https://doi.org10.1109/cvpr42600.2020.00301.

[28] Song, Shuran, and Jianxiong Xiao. 'Sliding Shapes for 3D Object Detection in Depth Images'. Computer Vision – ECCV 2014, Springer International Publishing, 2014, pp. 634–651, https://doi.org10.1007/978-3-319-10599-4_41. Lecture Notes in Computer Science.

[29] Li, Yangyan, et al. 'Database-Assisted Object Retrieval for Real-Time 3D Reconstruction'. Computer Graphics Forum: Journal of the European Association for Computer Graphics, vol. 34, no. 2, Wiley, May 2015, pp. 435–446, https://doi.org10.1111/cgf.12573.

[30] Litany, Or, et al. 'ASIST: Automatic Semantically Invariant Scene Transformation'. Computer Vision and Image Understanding: CVIU, vol. 157, Elsevier BV, Apr. 2017, pp. 284–299, https://doi.org10.1016/j.cviu.2016.08.002.

[31] Qi, Charles R., Hao Su, et al. 'Volumetric and Multi-View CNNs for Object Classification on 3D Data'. 2016 IEEE

Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016, https://doi.org10.1109/cvpr.2016.609.

[32] Qi, Charles R., Li Yi, et al. 'PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space'. ArXiv [Cs.CV], 7 June 2017, http://arxiv.org/abs/1706.02413. arXiv.

[33] Park, Jeong Joon, et al. 'DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation'. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2019, https://doi.org10.1109/cvpr.2019.00025.

[34] Mescheder, Lars, et al. 'Occupancy Networks: Learning 3D Reconstruction in Function Space'. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2019, https://doi.org10.1109/cvpr.2019.00459.

[35] Zeng, Andy, et al. '3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions'. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017, https://doi.org10.1109/cvpr.2017.29.

[36] Dai, Angela, et al. 'ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes'. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017, https://doi.org10.1109/cvpr.2017.261.

[37] Chang, Angel X., et al. 'ShapeNet: An Information-Rich 3D Model Repository'. ArXiv [Cs.GR], 9 Dec. 2015, http://arxiv.org/abs/1512.03012. arXiv.

[38] Shapira, Lior, and Daniel Freedman. 'Reality Skins: Creating Immersive and Tactile Virtual Environments'. 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), IEEE, 2016, https://doi.org10.1109/ismar.2016.23.

[39] Manni, Alessandro, et al. 'Snap2cad: 3D Indoor Environment Reconstruction for AR/VR Applications Using a Smartphone Device'. Computers & Graphics, vol. 100, Elsevier BV, Nov. 2021, pp. 116–124, https://doi.org10.1016/j.cag.2021.07.014.

[40] Ipsita, Ananya, et al. 'VRFromX: From Scanned Reality to Interactive Virtual Experience with Human-in-the-Loop'. Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems, ACM, 2021, https://doi.org10.1145/3411763.3451747.

[41] Kari, Mohamed, et al. 'TransforMR: Pose-Aware Object Substitution for Composing Alternate Mixed Realities'. 2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), IEEE, 2021, https://doi.org10.1109/ismar52148.2021.00021.

[42] Armeni, Iro, et al. '3D Semantic Parsing of Large-Scale Indoor Spaces'. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016, https://doi.org10.1109/cvpr.2016.170.

[43] Shuran, Samuel P., and Jianxiong Lichtenberg. 'Sun Rgb-d: A Rgb-d Scene Understanding Benchmark Suite'. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 567–576.

[44] Chen, Zhao, et al. 'Estimating Depth from RGB and Sparse Sensing'. ArXiv [Cs.CV], 8 Apr. 2018, http://arxiv.org/abs/1804.02771. arXiv.

[45] Rukhovich, Danila, et al. 'FCAF3D: Fully Convolutional Anchor-Free 3D Object Detection'. Lecture Notes in Computer Science, Springer Nature Switzerland, 2022, pp. 477–493, https://doi.org10.1007/978-3-031-20080-9_28. Lecture Notes in Computer Science.

[46] Chen, Shaoyu, et al. 'Hierarchical Aggregation for 3D Instance Segmentation'. ArXiv [Cs.CV], 4 Aug. 2021, http://arxiv.org/abs/2108.02350. arXiv.

[47] Nan, Liangliang, et al. 'A Search-Classify Approach for Cluttered Indoor Scene Understanding'. ACM Transactions on Graphics (TOG), vol. 31, no. 6, 2012, pp. 1–10.

[48] Wu, Zhirong, et al. '3D ShapeNets: A Deep Representation for Volumetric Shapes'. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2015, https://doi.org10.1109/cvpr.2015.7298801.

[49] Avetisyan, Armen, et al. 'End-to-End CAD Model Retrieval and 9DoF Alignment in 3D Scans'. ArXiv [Cs.CV], 10 June 2019, http://arxiv.org/abs/1906.04201. arXiv.

[50] Song, Shuran, et al. 'Semantic Scene Completion from a Single Depth Image'. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017, https://doi.org10.1109/cvpr.2017.28.

[51] Deitke, Matt, et al. 'RoboTHOR: An Open Simulation-to-Real Embodied AI Platform'. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2020, https://doi.org10.1109/cvpr42600.2020.00323.

[52] Zheng, Jia, et al. 'Structured3D: A Large Photo-Realistic Dataset for Structured 3D Modeling'. Computer Vision – ECCV 2020, Springer International Publishing, 2020, pp. 519–535, https://doi.org10.1007/978-3-030-58545-7_30. Lecture Notes in Computer Science.

[53] Roberts, Mike, et al. 'Hypersim: A Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding'. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE, 2021, https://doi.org10.1109/iccv48922.2021.01073.

[54] Liang, Zhihao, et al. 'Instance Segmentation in 3D Scenes Using Semantic Superpoint Tree Networks'. ArXiv [Cs.CV], 17 Aug. 2021, http://arxiv.org/abs/2108.07478. arXiv.

[55] Pratt, Harry, et al. 'FCNN: Fourier Convolutional Neural Networks'. Machine Learning and Knowledge Discovery in Databases, Springer International Publishing, 2017, pp. 786–798, https://doi.org10.1007/978-3-319-71249-9_47. Lecture Notes in Computer Science.

[56] Lim, J. J., Pirsiavash, H., & Torralba, A. (2013). Parsing ikea objects: Fine pose estimation. In Proceedings of the IEEE international conference on computer vision (pp. 2992-2999).

[57] Izadinia, H., Shan, Q., & Seitz, S. M. (2017). Im2cad. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5134-5143).

[58] Huang, S., Qi, S., Zhu, Y., Xiao, Y., Xu, Y., & Zhu, S. C. (2018). Holistic 3d scene parsing and reconstruction from a single rgb image. In Proceedings of the European conference on computer vision (ECCV) (pp. 187-203).