Snap2cad: 3D indoor environment reconstruction for AR/VR applications using a smartphone device

Snap2CAD: 3D indoor environment reconstruction for AR/VR applications using a smartphone device

Alessandro Manni, Damiano Oriti, Andrea Sanna, Francesco De Pace, Federico Manuri
Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino, 10129, Italy

Abstract

Indoor environment reconstruction is a challenging task in Computer Vision and Computer Graphics, especially when Extended Reality (XR) technologies are considered. Current solutions that employ dedicated depth sensors require scanning of the environment and tend to suffer from low resolution and noise, whereas solutions that rely on a single photo of a scene cannot predict the actual position and scale of objects due to scale ambiguity. The proposed system addresses these limitations by allowing the user to capture single views of objects using an Android smartphone equipped with a single RGB camera and supported by Google ARCore. The system includes 1) an Android app tracking the smartphone's position relative to the world, capturing a single RGB image for each object and estimating depth information of the scene, 2) a program running on a server that classifies the framed objects, retrieves the corresponding 3D models from a database and estimates their position, vertical rotation, and scale factor without deforming the shape. The system has been assessed measuring the translational, rotational and scaling errors of the considered objects with respect to the physical ones acting as a ground truth. The main outcomes show that the proposed solution obtains a maximum error of 18% for the scaling factor, less than nine centimeters for the position and less than 18° for the rotation. These results suggest that the proposed system can be employed for XR applications, thus bridging the gap between the real and virtual worlds.

1 Introduction

Indoor environment reconstruction is a fundamental problem in computer graphics and computer vision. Conventional computer methods for generating the digital counterpart of a scene can be based on merging a large number of depth images [41, 43, 61] or using photogrammetry [26, 38], which again requires a large number of photos from different points of view. A popular smartphone app was Autodesk 123D Catch [1], which required 26 photos from different angles to reconstruct an object.

Scanning solutions based on LiDAR or Time-of-Flight (ToF) sensors such as Kinect tend to suffer from low resolution, noise [17] and missing parts. Following the success of deep learning (DL) methods in image classification and generation tasks [45], researchers have been trying to apply these techniques to several 3D tasks [14, 21]. DL methods add semantics and can solve some of the limitations of the conventional solutions: they can reconstruct a scene from a single RGB image by predicting room layout, object location, pose and shape [24, 42], they can add missing parts of an incomplete scan [12], they can retrieve and align 3D models of objects to the scan to overcome low resolution [3].

However, when thinking about fast scene reconstruction for Virtual Reality (VR) and Augmented Reality (AR) applications using a common smartphone, a full scan of the environment, which takes time and requires depth sensors that are only available on a few recent high-end devices, should be avoided. Furthermore, position and scale of the objects are crucial, and therefore DL methods that use a single RGB image for scene reconstruction are not suitable, since they do not solve the global scale ambiguity that arises when inferring depth from a 2D image. In fact, it is an ill-posed problem and an open challenge. In addition, reconstructing the entire scene from a single RGB image would probably result in heavily occluded objects whose shapes and poses would be hard to predict.

In many disciplines, from education to entertainment, from healthcare to industry, VR and AR are some of the most promising emerging technologies. In VR, users are immersed in virtual worlds, whereas AR allows users to see an augmented version of the real world; in both cases, being able to generate a virtual version of a real object can be extremely useful. For instance, for VR users having a virtual representation of the physical space is important for safety concerns.

On the other hand, an AR user might want to share the physical environment with a remote VR user so

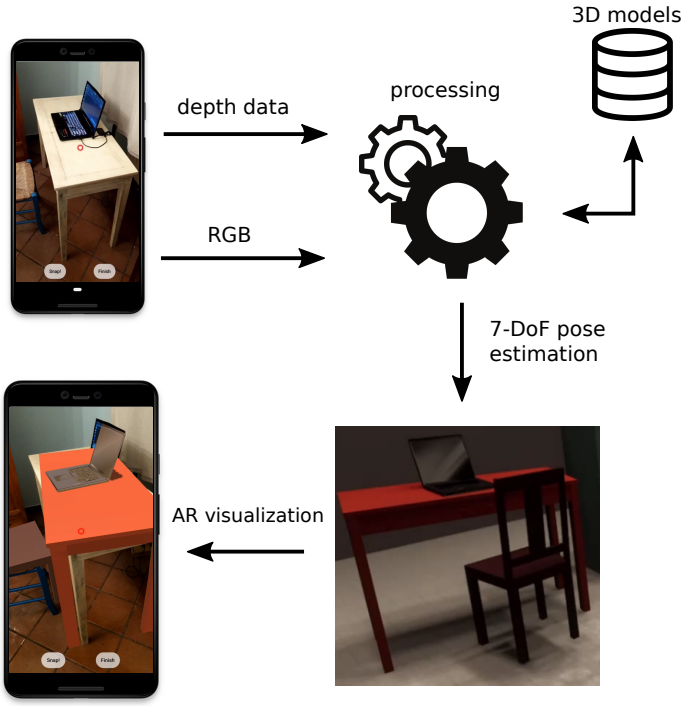


Figure 1: The proposed system. An Android smartphone is used to capture a single RGB picture for each object, which is augmented with the depth information provided by Google ARCore. Then, a server processes these data in order to classify the object in the picture and retrieve the most similar synthetic 3D object from a database; position, vertical orientation and scale factor are estimated and applied to the model. Finally, an AR app and a desktop app can be used to visualize the reconstructed scene.

that both can move and interact in a common space. In this case, scene understanding can be fundamental for a better experience, as it can provide both users with spatial and semantic information about the objects.

Building on these considerations, a novel semi-automatic system allowing users to reconstruct the digital version of a real indoor scene using a smartphone is presented in this paper. A smartphone is used to capture images of those objects for which a 3D representation is to be retrieved from a database. The images, along with depth information, are then sent to a server for processing. The system is semi-automatic, as the user still has to perform some tasks manually: 1) capturing one picture for every object and 2) specifying a point belonging to the framed object. All the other steps related to the process are automatically executed by the server. Specifically, the first step is the object segmentation and classification, which assigns a semantic label to each pixel and allows the system to distinguish among different objects in the frame. Then, after isolating the target object, the server finds the most similar 3D CAD model available in a model database. Finally, the object pose and scale factor is estimated, computing the object rotation and position in the world reference system defined by the smartphone. The object is scaled according to the computed scale factor without performing any other mesh deformation. An overview of the system is shown in Figure 1.

To the best of the authors' knowledge, this work is the first system capable of estimating the 7-DoF pose (3-DoF position, 3-DoF scale and 1-DoF rotation around the vertical axis) of objects from a single view using a smartphone with a single RGB camera to reconstruct an indoor environment.

The most relevant works related to the scene reconstruction problem are discussed in Section 2. Section 3 presents the proposed system, whereas the conducted experiments and related results are presented and discussed in Section 4 and in Section 5, respectively. Finally, the future developments are discussed in Section 6.

2 Related work

During the last two decades, the researchers have conducted extensive work on the reconstruction of 3D objects or of entire scenes from single and multiple images or RGB-D scans.

Following the success of deep learning in image-related tasks, many large-scale datasets of 3D models such as ShapeNet [6] or ModelNet [56] were made available to the academic community in order to develop new learning-based methods for 3D reconstruction. Other commonly used datasets are ScanNet [11], which contains annotated 3D scans of indoor scenes, NYU Depth Dataset [39] and SUN RGB-D [50], which contain annotated RGB-D images.

2.1 Single and Multiple View Scene Reconstruction

Some works [9, 22, 58] discretize the 3D space using voxels, which are then predicted by neural networks (NNs) similar to those used for images. When using voxels, the reconstruction quality is severely affected by memory constraints [22]; data compression structures that take advantage of the sparsity of the data, such as octrees, can been used to increase the output

resolution [46, 53].

Some researchers tried to devise networks that output meshes directly; some works [18, 19, 49] consider the object as a whole, whereas the object is managed as the union of simple parts in [8, 16]; each part is modeled separately, thus producing higher quality results. These approaches suffer from bad topology, and they do not really take advantage of the ability of meshes to efficiently scale with the shape complexity.

Some of the most recent works use implicit representations in the forms of occupancy functions [37, 47, 59] or signed distance functions [33], avoiding the discretization of the 3D space. Methods based on implicit representations do not generalize well to unseen objects (i.e., objects that do not appear in the training dataset) and poses; moreover, they tend to produce overly smooth shapes [4].

Izadinia et al. [25] use Convolutional Neural Networks (CNNs) to train comparison metrics, which are used in an iterative algorithm to optimize placement and scale of objects, in order to match the reference image of an indoor scene. Although the generated scene is similar to the reference image, it is not possible to determine the real sizes of objects from a single RGB image.

Guo et at. [20] can predict position, orientation and size of individual objects in a scene from a single RGB-D image, but small objects tend to be missing from the reconstructed scene and occlusion can lead to erroneous estimates.

Holistic approaches [24, 42] do not consider one object at a time, but they reason about the scene as a whole, trying to understand the relationships (e.g., support, symmetry, etc.) among the objects. Huang et al. [24] jointly recover room layout, camera pose, and object bounding boxes of an indoor scene, whereas the solution proposed in [42] can also reconstruct the mesh of objects.

FroDO [30] can estimate the pose of an object and reconstruct its shape, but it requires a series of localized RGB images.

CoReNet [44] can predict the shape, relative pose and class of all objects depicted in a single image, but the quality of the generated geometry is affected by the discretization step, which is required in order to reconstruct the object shape.

Given an input image containing multiple objects, Kuo et al. [27] first find object instances through segmentation, then they retrieve the most similar CAD model to any object instance by using a shared embedding space for image-CAD pairs. In addition to shape retrieval, they predict the object pose.

Vid2CAD [36] integrates single-frame predictions by NNs with globally-consistent poses obtained by solving the alignment problem as a temporally global, multi-view constraint optimization problem.

2.2 Reconstruction from RGB-D Scan

SG-NN [12] takes as input partial RGB-D scans and predicts missing geometry in a self-supervised manner. The result is an high resolution 3D reconstruction of the scene. RevealNet [23], given an incomplete RGB-D scan, detects the objects in the scene and infers their complete geometry.

Scan2CAD [3] predicts correspondence heatmaps between regions of an RGB-D scan and 3D CAD models using a 3D CNN, then it finds the 9-DoF poses for 3D CAD model alignment to the scan. The main disadvantage of this approach is that it requires to compare each model in the database to each scanned region to reconstruct the whole scene. Scan2CAD is extended in [2] by the addition of layout estimation, which also helps in improving the overall accuracy.

3 The Proposed Solution

The proposed system recognizes the framed objects and retrieves the 3D CAD models (represented as polygonal meshes) corresponding to the object classes from a database. Let $\mathbb{M} = \{m_n\}$ be the model set, with $1 \leq n \leq N$ and N be the number of 3D CAD models. Each model m_n belongs to a specific category or class $c_k \in \mathbb{C} = \{c_k\}$. For each object, the goal is to infer its class c_k, find the most similar 3D CAD model m_n and estimating its 7-DoF pose: 3-DoF for translation, 3-DoF for scale, and 1-DoF for rotation around the vertical axis.

The system operates in indoor scenes using a handheld Android device running a Google ARCore-based app [54]. ARCore is a collection of tools for creating AR experiences. For each physical object, a *snapshot* consisting of 1) the RGB image, 2) the depth data, 3) the device pose, and 4) the lowest horizontal plane is acquired.

The system does not require a depth sensor since it leverages the Visual-SLAM and the depth map estimation algorithms included in the ARCore Depth API. Specifically, a Samsung Galaxy S8 fitted with a single RGB camera was used in this work. Figure 2 illustrates the four stages of the proposed system.

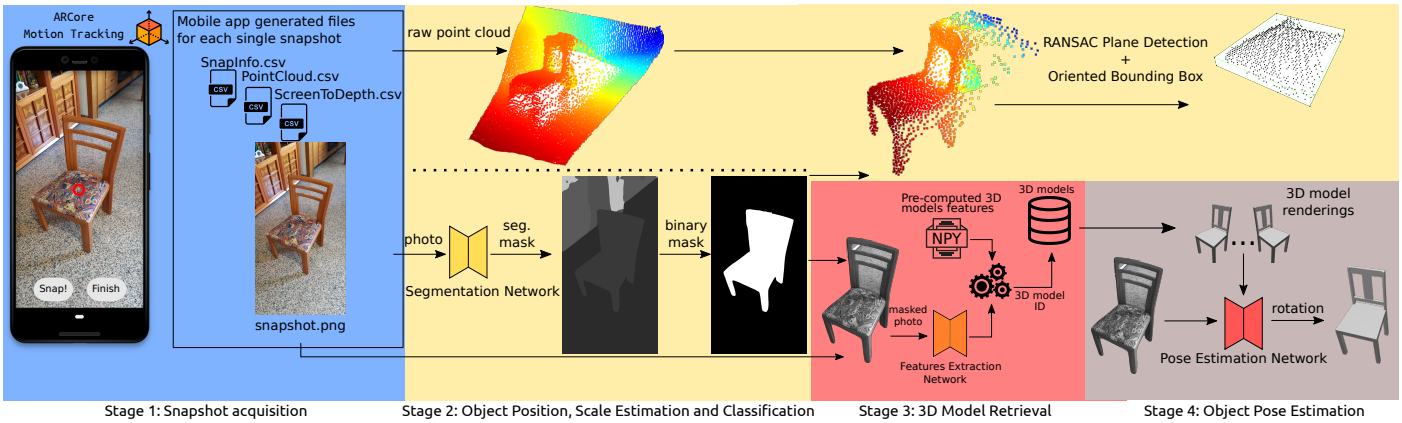


Figure 2: The reconstruction pipeline of the proposed system. Stage 1 is executed by the client whereas stages 2, 3 and 4 are executed by the server.

3.1 Snapshots acquisition

The first stage of the presented solution consists in taking snapshots of the objects by using the Android smartphone. Starting from the utilities implemented in the DepthLab app [13] provided by Google, an Android app was developed using Unity3D[1] and the ARCore Depth API SDK. The developed app tracks the smartphone in the Unity3D world coordinates, it predicts the depth maps in real-time and it detects the horizontal planes. When the app starts, ARCore defines a coordinate frame $W = \{O, xyz\}$ by fixing its origin and orientation to the starting position $P_{device}(0) \in \mathbb{R}^3$ and orientation $R_{device}(0) \in \mathbb{R}^3$ of the device:

$$O = P_{device}(time = 0) \tag{1}$$

$$xyz = R_{device}(time = 0) \tag{2}$$

By moving the smartphone, the device pose is computed by the ARCore Visual-SLAM algorithm. When the user wants to retrieve a digital version of an object, they should target the object on the smartphone screen by using the fixed virtual red cursor shown in Figure 2. Then, the user can acquire the snapshot by tapping on the "Snap!" button, which triggers four events:

1. The app saves the i-th single-view RGB image F_i^{rgb} of the scene in camera resolution Res_{cam}^{rgb}.

2. The app checks the detected horizontal planes $\mathbb{P}_i^h = \{p_{i,j}\}, \forall j \in \{1, 2, \ldots, J\}$, where J is the number of detected planes in the given snapshot. The lowest plane $h_{lp,i}$ is considered as a candidate ground plane, and its height relative to the smartphone initial position is stored in a CSV file called *SnapInfo*:

$$h_{lp,i} = \min_{j \in \{1,2,\ldots,J\}} height(p_{i,j}) \tag{3}$$

The rotation about the vertical axis $R_{device,i}^v$ of the smartphone is also stored and later used to estimate the rotation of the object.

3. The app stores in a CSV file called *ScreenToDepth* the mapping between the screen space coordinates of the captured frame F_i^{rgb} and the coordinates of the estimated depth map F_i^{depth} of that frame. This is necessary because the depth map estimated by ARCore has a resolution of 160x90px, which is different and much less than the color frame resolution Res_{cam}^{rgb}, equal to 2220x1080px.

4. In order to construct the point cloud in camera space Q_i^{cam} corresponding to a given depth map (see Section 3.2), the program uses the camera intrinsic parameters, i.e. principal point (c_u, c_v) and focal length f. If $Q(u, v)$ denotes the point of the point cloud corresponding to the pixel (u, v) of the depth map, and $z(u, v)$ is the depth value, then its x, y and z coordinates are computed using the following equation:

$$Q(u, v) = \begin{bmatrix} \frac{u - c_u}{f} z(u, v) \\ \frac{v - c_v}{f} z(u, v) \\ z(u, v) \end{bmatrix} \tag{4}$$

5. The app stores in another CSV file called *PointCloud* the mapping between the screen space coordinates of the estimated depth map F_i^{depth} and

[1] https://unity.com/

the world space 3D points. The world space 3D points have been computed starting from the camera space 3D points using the utilities provided by [13]. The generated point cloud has approximately 14000 points.

These operations are repeated for each object that the user wishes to replace with a 3D model. This stage ends after the user clicks on the "Finish" button, which instructs the smartphone to forward all generated files to the server.

3.2 Object Classification and Estimation of Object Position and Scale

The server automatically processes the files generated by the smartphone application in the second stage. In this stage, the goal is both to classify the selected object in each snapshot and to estimate its position $P_i \in \mathbb{R}^3$ and size $S_i \in \mathbb{R}^3_{>0}$ in the world space. In order to extract from the RGB image only the object targeted by the user, the system performs a semantic segmentation, which works by assigning a semantic label (i.e., an object class label) to each pixel in the image.

It is possible to find several DL solutions to accomplish this task for indoor environments. In particular, there are well-known datasets such as COCO [35], ADE20K [62] and SUN RGB-D [50] on which Deep Neural Networks are trained. The presented solution uses MSeg [28], which combines all the mentioned datasets plus Mapillary [40], IDD [55], BDD [60] and Cityscapes [10] into a single composite dataset with 194 categories. The neural network architecture is HRNet-W48 [52], which was pre-trained with Mseg-3m-1080p model shared by [28].

The output of the segmentation network is a gray scale image F_i^{gray}, whose pixels have a gray color code corresponding to a specific semantic class. Since the segmentation network returns an image with a different resolution than the input image, the image is scaled back to the resolution of the frame F_i^{rgb} acquired by the smartphone using the nearest-neighbor interpolation to avoid introducing artifacts such as gray tones not present before. The pixel in the center of the image carries the color code of the targeted object. If the color code represents a class of objects managed by the system, the server processes the image further, otherwise it discards it.

The system is now able to retrieve from the raw point cloud Q_i of the single view snapshot the point cloud of the object Q_i' highlighted in the mask, by using the pixel-by-pixel mapping between the mask, the relative estimated depth map and then the world space point cloud. The point cloud is transformed from camera space to world space using a transformation matrix $T_{cam \to world}$ as shown in the following equation:

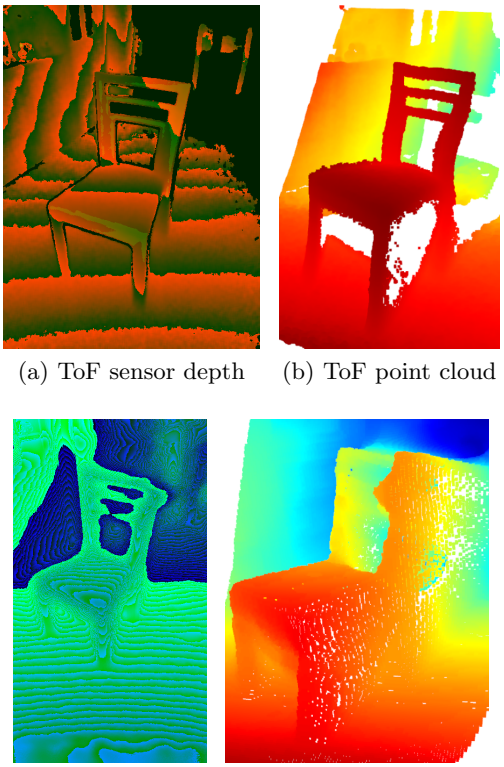
$$Q_i^{world} = T_{cam \to world} \cdot Q_i^{cam} \tag{5}$$

Then, the set of points belonging to the object is extracted from the original point cloud by using a binary mask that can be obtained by performing a flood fill operation on the segmentation image starting from its central pixel.

For small objects (e.g., mice, remotes, bowls, cups, etc.), it is sufficient to compute a 3D oriented bounding box of the segmented point cloud Q_i^{world} to obtain the centroid and the 3D size of the object. The system uses the algorithm implemented in Open3D [63], which is an approximation to the minimum volume box containing a set of points. On the other hand, large-size objects present several challenges: 1) they may consist of different parts, 2) they may have outliers in the point cloud caused by the single view of the object 3) and they may have flying points caused by the smoothing that ARCore applies to the depth maps. The ARCore smoothing process cannot be disabled and it is shown in Figure 3.

For such objects, a RANdom SAmple Consensus (RANSAC) algorithm is applied to the segmented point cloud, detecting only the horizontal planes. Then, the 3D oriented bounding box is computed considering the points positioned on the detected plane. This operation returns the extent of the object and the position of the centroid where the value of its vertical axis represents the height of the object.

When objects are composed of parts with holes, the ARCore smoothing process creates flying points that could affect the size of the 3D oriented bounding box. Chairs are particularly challenging objects because the backrest can present cavities and also because the 3D oriented bounding box would not provide information about the height of the seat (a correctly aligned collider could be useful in AR/VR applications). To overcome these problems, the system proceeds in three steps. The first step creates a mask of depth discontinuities in the depth map to remove most of the point cloud outliers: an automatic Canny edge detection algorithm similar to the one presented in [5] is applied to a gray scale version of the depth map that is normalized to enhance the depth variations. Then, two morphological transformations, dilation and closing, are used. The kernel of the Canny edge varies with the distance of



(a) ToF sensor depth (b) ToF point cloud

(c) ARCore depth (d) ARCore point cloud

Figure 3: The depth maps and point clouds. (a-b) The depth map acquired with a generic ToF sensor and the related point cloud, respectively. (c-d) The same ToF depth map processed with ARCore and the related point cloud, respectively. The ARCore processing creates flying points on the object contour.

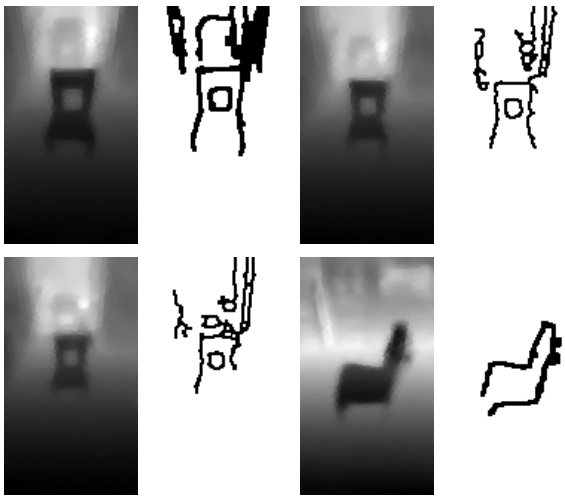


Figure 4: Depth images (represented as normalized gray scale images) and relative edge masks obtained using a Canny edge detection operator.

the object center from the smartphone, as shown in Figure 4.

The generated binary mask is used to remove the outliers in the point cloud, which are concentrated in depth discontinuities. The RANSAC algorithm detects the horizontal plane of the seat in the second step, whereas a statistical outlier removal algorithm is applied to the point cloud segmented by the plane in the third step, removing the remaining outliers left over from the previous step. The whole process is shown in Figure 5.

At this point, the system computes the 3D oriented bounding box of the segmented plane found by RANSAC, which provides the extent of the seat and the centroid falling on the seat; this gives the seat's height. The overall height of the chair is also determined by finding the highest point of the segmented point cloud data. On the other hand, cabinets, dressers, and nightstands are treated differently by the system due to three factors: 1) the segmentation network does not always predict the correct label, 2) objects lying on the small surface of a nightstand might create artifacts on the point cloud data and the RANSAC algorithm may not detect the horizontal plane and 3) the camera might not frame the top surface of a cabinet due to its height, so the RANSAC algorithm will fail to find a plane in this case either. For these classes of objects, the system finds the highest vertex of the segmented point cloud, which is assumed to be the height of the top surface, and then it creates an artificial horizontal plane by moving the vertices on the vertical axis. The algorithm provided by Open3D for the estimation of the 3D oriented bounding box is used to obtain the extent of the object and the 2D position of the centroid, which is not affected by the previous operations.

3.3 3D Model Retrieval

Image-based 3D model retrieval is the task of retrieving a relevant 3D synthetic object from a database that is similar to a query photo captured in real world. The domain gap between 3D shapes and natural images is a major challenge [32, 29, 51, 15]. There are different ways to perform 3D model retrieval: Chen et al. [7] introduced LightField Descriptors (LFDs) to represent 3D shapes, whereas other approaches use CNNs [31, 34]. In this work, A VGG-19 [48] CNN was used as it provides better fine-grained 3D model retrieval with respect to other approaches such as LFDs. A subset of 3D models from the ShapeNetCore dataset [6] is rendered from 12 viewpoints, then the VGG-19 CNN pre-trained on ImageNet is used to extract their features. These features are computed

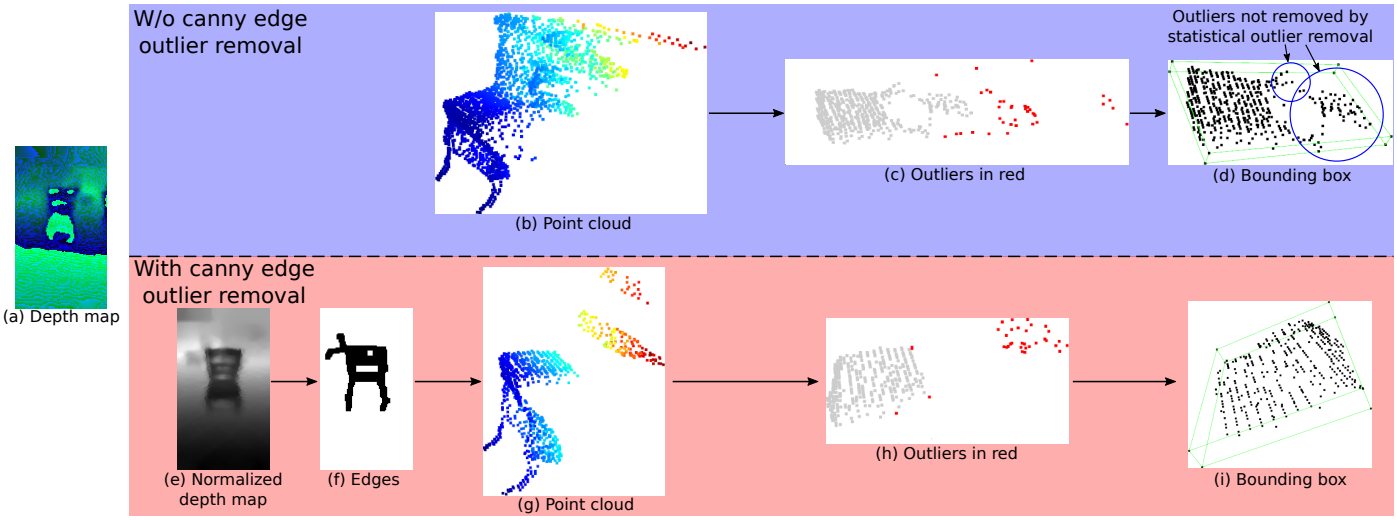


Figure 5: Canny edge-based outlier removal. The (a) input depth map estimated by ARCore is converted to a point cloud (b-g). With statistical outlier removal (SOR), the outliers highlighted in red in (c) are detected and removed; in (d) it is shown how the bounding box is not correct because some outliers were left by the SOR. By using the Canny edge detection algorithm on the (e) normalized depth map, whose result is shown in (f), the outliers generated by the ARCore smoothing process can be removed (h) and the bounding box is correctly estimated (i).

and stored once. The most similar object is determined by computing the Euclidean distance between the extracted features of the segmented target object and the features computed for the renderings associated to 3D models of the same category. To reduce the noise of natural images, the system removes most of the background and other elements by extracting the query object from the snapshot using the binary mask generated in stage 2, followed by a blurring operation to soften the borders and a final cropping operation to center the object. The features of the segmented objects are extracted by the VGG-19 CNN, then the Euclidean distance between the query image and all the renderings belonging to the objects under the category given by the segmentation network is calculated. The 3D object associated to the rendering with the shortest distance to the natural RGB image is selected (an example of 3D model retrieval is shown in Figure 6).

Figure 6: Query segmented images and top-3 models retrieved. The first column represents the segmented photos of real objects used as queries, whereas the second to forth columns represent the most similar retrieved models (best from left to right).

3.4 Object Pose Estimation

The system predicts the rotation of the object around the vertical axis in the fourth stage. Several DL solutions try to solve the problem of object pose estimation from a single RGB image, but many require that the neural network has to be trained on the specific object whose pose is to be estimated. This approach is incompatible with the conditions under which the system is intended to operate: an indoor environment where never-before-seen objects have to be considered. For this reason, the proposed system uses PoseFromShape [57]. This DL solution can predict the viewpoint of an object in an RGB image with respect to known viewpoints. For this work, PoseFromShape

uses the model provided by [57] trained on ShapeNetCore, a subset of ShapeNet (Figure 7 illustrates some models of the ShapeNet dataset). From this dataset, the 3D models of the considered category were taken and rendered from different viewpoints using Blender.

The background of the RGB snapshot could impact on the accuracy of the prediction, thus the system uses the image of the segmented object obtained in the previous stage.

Since the system has already retrieved the 3D object from the previous step, it feeds the neural network with the collection of renderings associated to that object and the segmented photo of the object.

The output of the PoseFromShape network is the camera's viewpoint, which is composed of the Euler angles for azimuth, elevation and in-plane rotation. The system only considers the azimuth that is going to be applied to the 3D model used to represent the physical object.

3.5 Scene Reconstruction and AR Visualization

For each snapshot, the system generates three CSV files containing: 1) position and size of the object, 2) vertical rotation of the object and 3) rotation of the smartphone. Furthermore, the height of the floor h_{floor} valid for the whole reconstructed scene system

Figure 7: Some 3D models from ShapeNet Core dataset. The models are in a canonical pose.

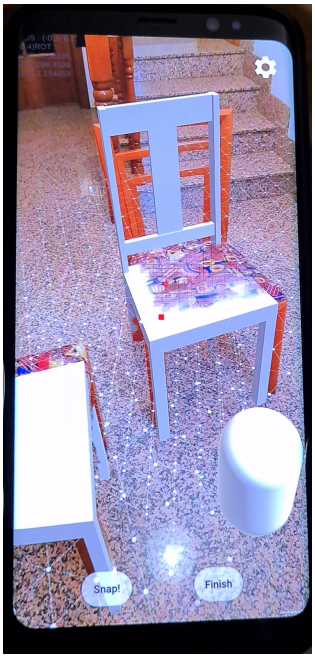


Figure 8: The AR view. Two chairs visualized by the AR app: the occlusions are taken into account.

is stored in an additional CSV file named *Floor*. The height of the floor is the average of all the values that differ from the lowest of the list $\{h_{lp,i}\}$ by 4 cm or less:

$$h_{lp} = \min_i h_{lp,i} \tag{6}$$

$$h_{floor} = \text{avg}(h_{lp,i}) \quad \text{if} \quad \text{abs}(h_{lp,i} - h_{lp} < 4cm) \tag{7}$$

It has been observed that when the tracking of a plane is lost and then detected again, there is a difference of less than 4 cm from previous detection.

A desktop application has been developed using Unity3D to visualize the reconstructed scene. The 3D models were taken from the ShapeNetCore dataset [6]. The 3D models have been pre-processed to present real world sizes and centroid positions consistent with the one estimated by the system. The Unity3D desktop app parses all the CSV files, selecting the 3D models and computing the appropriate rotation, scale and position values. To qualitatively evaluate the accuracy of the reconstruction, all 3D models are also visualized by a mobile AR app. The user can freely move around the scene to verify whether the 3D models correctly overlap with the real objects. The geometry-aware occlusion feature of ARCore facilitates the evaluation. In fact, since a dense depth map is estimated at almost every frame, it is possible to check whether a 3D model is correctly superimposed on the physical object (Fig-

ure 8). At the following link it is possible to find a video showing the proposed system[2].

4 Experimental validation

To evaluate the accuracy of the system, a new dataset is introduced. The dataset consists of 500 snapshots of several objects with different shapes. The objects are divided in categories and the snapshots were taken from different points of view.

4.1 Scaling error

The ground truth (GT) is represented by the physical object size. To determine the scaling error of each category, several snapshots have been taken from different views. Then, the root mean squared errors (RMSE) between the GT and the estimated size is computed for each snapshot. Finally, the average of the all RMSE values is computed, determining the category scaling error. The error is computed for each dimension (x, y, and z) and it is expressed as a percentage of absolute difference from the ground truth value.

4.2 Position error

The position GT of the object is determined using a marker placed at the center of the physical object. The marker is used to create a mask that segments the points at the center of the object. Hence, the 3D bounding box can be computed finding the center coordinates of the object. Then, the same procedure as the one introduced in Section 4.1 is applied to determine the category position error, expressed in meters.

4.3 Vertical rotation error

The GT value of the object vertical rotation is determined using the same marker recognition process as used in Section 4.2. However, the smartphone is mounted on a fixed support to ensure consistency during the evaluation process. The rotation assessment is computed using two different images: the GT image (with the marker, see Figure 9) and the evaluated one (without the marker). The GT rotation is determined by computing the marker rotation, whereas the rotation of the evaluated object is computed by the proposed solution. Finally, the rotational error is obtained by calculating the average of all RMSE values, expressed in degrees.

[2] `https://youtu.be/SW0JWhPBX50`

(a) Object with marker (b) The estimated pose

Figure 9: (a) Captured photo showing an object with the marker used to determine the ground truth pose; (b) the estimated pose of the same object obtained with the proposed system.

5 Discussion

The position, rotational and scaling errors are shown in Table 1. It appears that the position and scaling errors along the Y axis grow if the surface of the object is not flat or there are obstacles that partially occlude the object surface, thus interfering with the depth map estimation algorithm. Small occlusions, such as a laptop covering part of a table, are tolerated, whereas a heavily occluded table may lead to a failure in the segmentation stage. Regarding the rotation error, some inconsistencies have been detected in objects with symmetric shapes, resulting in wrong rotations of 180° around the vertical axis. These wrong predictions seem to happen regardless of the dataset used to train the network. Generally, the errors increase when the object is not fully captured in the snapshot, and thus the 3D oriented bounding box does not perfectly align to the segmented point cloud of the object itself. Matte black surfaces pose a challenge since the stereo matching algorithm [54] fails to provide consistent correspondences among frames composed by a large amount of dark pixels, thus creating reconstruction errors and artifacts on the depth map. The proposed system reconstructs a scene using single snapshots for each object, thus it is important to capture the object from views that cover the essential parts of the object (e.g., to estimate the position and size of a chair, the seat should be visible). It has been noted that chests of drawers and nightstands may be misclassified as cabinets by the segmentation network. This is probably due to the fact that the evaluated image regions considered by HRNet do not contain enough information to discern the differences among some classes of objects having similar features [52].

	Avg. RMSE Position (m)			Avg. RMSE Scaling (%)			Avg. RMSE Scaling Factor Error (%)	Avg. RMSE Rotation (°)
Object (PS/R)	x	y	z	x	y	z	(x, y, z)	y
Chair (61/11)	0.03	0.02	0.02	10	2	13	10	9
Swivel chair (24/6)	0.02	0.02	0.03	6	6	12	8	15
Armchair (33/8)	0.05	0.02	0.07	13	4	12	10	9
Table (62/9)	0.05	0.02	0.04	7	3	10	7	12
Nightstand (31/6)	0.04	0.06	0.05	13	7	23	14	18*
Chest of drawers (12/7)	0.03	0.07	0.08	6	5	8	6	8
Cabinet (57/5)	0.06	0.03	0.05	15	6	14	12	14
Bed (24/7)	0.09	0.04	0.1	7	8	17	10	14
Laptop (22/12)	0.02	0.02	0.02	7	8	21	12	10
Bowl (17/-)	0.05	0.03	0.04	14	22	7	14	-
Mouse (25/6)	0.02	0.01	0.02	11	26	11	16	2*
Remote (22/5)	0.03	0.02	0.01	6	20	25	17	6*
Book (33/8)	0.02	0.01	0.02	17	23	15	18	6*

Table 1: The main outcomes. The average of the root mean squared errors of position, scaling, vertical rotation and the average size error in 3D. Objects marked with * may have 180° rotation error. PS and R stand for the number of position-scaling and rotation snapshots, respectively.

Finally, regarding the execution time, the system has been evaluated using a laptop equipped with a CPU Intel Core i7-8750h and an NVIDIA RTX 2060 GPU. The proposed solution can reconstruct a scene with five objects in 21 seconds: 15 seconds for the segmentation task, approximately 2 seconds for 3D model retrieval, less than 2 seconds for objects processing and 2.25 seconds for the pose inference.

6 Conclusion

This paper presents a system able to reconstruct a scene captured by an Android smartphone supported by ARCore. With respect to the current state of the art, the proposed solution exploits only the RGB camera, without employing traditional depth sensors. The system consists of an ARCore-based app that takes snapshots of objects in an indoor environment from a single view. The snapshots are then processed on a server: the target object in the frame is classified and the most similar 3D model is retrieved from a database; then, scale, position and vertical rotation of the object are estimated. All DL modules can be replaced as better solutions become available. In addition, two Unity3D applications have been presented: 1) a desktop app that populates the virtual scene with instances of 3D models using the estimated pose without deforming object meshes and 2) an AR app that superimposes the virtual objects over their real counterparts. These applications can be used in online multiplayer mode, thus visualizing the AR instance of the desktop/VR player from the smartphone and vice-versa the desktop/VR instance of the tracked AR user in the reconstructed scene.

Several techniques have been described to overcome the challenges posed by the single view snapshots and the poor resolution of the depth maps. A dataset of more than 500 snapshots was introduced to evaluate the system accuracy.
Future work will include a wider model dataset, which in combination with non-trivial deformation of meshes could allow more fine-grained shape retrieval. Furthermore, the scene layout will be also determined, providing the ability to detect and reconstruct walls, floors and ceilings. Finally, other methods that overcome the vertical rotation constraint could be investigated in order to detect rotations around any axis.

References

[1] Autodesk 123d catch application. `https://www.autodesk.com/solutions/123d-apps`, 2021.

[2] A. Avetisyan, Tatiana Khanova, C. Choy, D. Dash, Angela Dai, and M. Nießner. Scenecad:

Predicting object alignments and layouts in rgb-d scans. *ArXiv*, abs/2003.12622, 2020.

[3] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X. Chang, and Matthias Nießner. Scan2cad: Learning CAD model alignment in RGB-D scans. *CoRR*, abs/1811.11187, 2018.

[4] Abhishek Badki, Orazio Gallo, Jan Kautz, and Pradeep Sen. Meshlet priors for 3d mesh reconstruction. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2846–2855, 2020.

[5] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986.

[6] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing 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. *CoRR*, abs/1512.03012, 2015.

[7] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On Visual Similarity Based 3D Model Retrieval. *Computer Graphics Forum*, 2003.

[8] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. *CoRR*, abs/1911.06971, 2019.

[9] 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. *CoRR*, abs/1604.00449, 2016.

[10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016.

[11] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. *CoRR*, abs/1702.04405, 2017.

[12] Angela Dai, Christian Diller, and Matthias Niessner. Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[13] Ruofei Du, Eric Turner, Maksym Dzitsiuk, Luca Prasso, Ivo Duarte, Jason Dourgarian, Joao Afonso, Jose Pascoal, Josh Gladstone, Nuno Cruces, Shahram Izadi, Adarsh Kowdle, Konstantine Tsotsos, and David Kim. DepthLab: Real-Time 3D Interaction With Depth Maps for Mobile Augmented Reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST. ACM, 2020.

[14] George Fahim, Khalid Amin, and Sameh Zarif. Single-view 3d reconstruction: A survey of deep learning methods. *Computers Graphics*, 94:164 – 190, 2021.

[15] Huan Fu, Shunming Li, Rongfei Jia, Mingming Gong, Binqiang Zhao, and Dacheng Tao. Hard example generation by texture synthesis for cross-domain shape similarity learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14675–14687. Curran Associates, Inc., 2020.

[16] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. SDM-NET: deep generative network for structured deformable mesh. *CoRR*, abs/1908.04520, 2019.

[17] Vincent Gay-Bellile, Adrien Bartoli, Kamel Hamrouni, Patrick Sayd, Steve Bourgeois, and Amira Belhedi. Noise modelling in time-of-flight sensors with application to depth noise removal and uncertainty estimation in three-dimensional measurement. *IET Computer Vision*, 9, 08 2015.

[18] Justin Johnson Georgia Gkioxari, Jitendra Malik. Mesh r-cnn. *ICCV 2019*, 2019.

[19] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation. *CoRR*, abs/1802.05384, 2018.

[20] Ruiqi Guo, Chuhang Zou, and Derek Hoiem. Predicting complete 3d models of indoor scenes. *CoRR*, abs/1504.02437, 2015.

[21] Xian-Feng Han, Hamid Laga, and Mohammed Bennamoun. Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era. *CoRR*, abs/1906.06543, 2019.

[22] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. *CoRR*, abs/1704.00710, 2017.

[23] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sic: 3d semantic instance completion for RGB-D scans. *CoRR*, abs/1904.12012, 2019.

[24] Siyuan Huang, Siyuan Qi, Yinxue Xiao, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. Cooperative holistic scene understanding: Unifying 3d object, layout, and camera pose estimation. In *Advances in Neural Information Processing Systems*, pages 206–217, 2018.

[25] Hamid Izadinia, Qi Shan, and Steven M. Seitz. IM2CAD. *CoRR*, abs/1608.05137, 2016.

[26] Michal Jancosek and Tomas Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR 2011*. IEEE, jun 2011.

[27] Wei-Cheng Kuo, A. Angelova, Tsung-Yi Lin, and Angela Dai. Mask2cad: 3d shape prediction by learning to segment and retrieve. In *ECCV*, 2020.

[28] John Lambert, Zhuang Liu, Ozan Sener, James Hays, and Vladlen Koltun. MSeg: A composite dataset for multi-domain semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[29] Tang Lee, Yen-Liang Lin, Hungyueh Chiang, Ming-Wei Chiu, Winston Hsu, and Polly Huang. Cross-domain image-based 3d shape retrieval by view sequence learning. In *2018 International Conference on 3D Vision (3DV)*, pages 258–266, 2018.

[30] Kejie Li, Martin Rünz, Meng Tang, Lingni Ma, Chen Kong, Tanner Schmidt, I. Reid, L. Agapito, J. Straub, S. Lovegrove, and R. Newcombe. Frodo: From detections to 3d objects. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14708–14717, 2020.

[31] Wenhui Li, Anan Liu, Weizhi Nie, Dan Song, Yuqian Li, Weijie Wang, Shu Xiang, Heyu Zhou, Ngoc-Minh Bui, Yunchi Cen, Zenian Chen, Huy-Hoang Chung-Nguyen, Gia-Han Diep, Trong-Le Do, Eugeni L. Doubrovski, Anh-Duc Duong, Jo M. P. Geraedts, Haobin Guo, Trung-Hieu Hoang, Yichen Li, Xing Liu, Zishun Liu, Duc-Tuan Luu, Yunsheng Ma, Vinh-Tiep Nguyen, Jie Nie, Tongwei Ren, Mai-Khiem Tran, Son-Thanh Tran-Nguyen, Minh-Triet Tran, The-Anh Vu-Le, Charlie C. L. Wang, Shijie Wang, Gangshan Wu, Caifei Yang, Meng Yuan, Hao Zhai, Ao Zhang, Fan Zhang, and Sicheng Zhao. Monocular Image Based 3D Model Retrieval. In Silvia Biasotti, Guillaume Lavoué, and Remco Veltkamp, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2019.

[32] Yangyan Li, Hao Su, Charles Ruizhongtai Qi, Noa Fish, Daniel Cohen-Or, and Leonidas J. Guibas. Joint embeddings of shapes and images via cnn image purification. *ACM Trans. Graph.*, 34(6), October 2015.

[33] Chen-Hsuan Lin, Chaoyang Wang, and Simon Lucey. Sdf-srn: Learning signed distance 3d object reconstruction from static images. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[34] Kyaw Zaw Lin, Weipeng Xu, Qianru Sun, Christian Theobalt, and Tat-Seng Chua. Learning a disentangled embedding for monocular 3d shape retrieval and pose estimation. *CoRR*, abs/1812.09899, 2018.

[35] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[36] K. Maninis, S. Popov, M. Nießner, and V. Ferrari. Vid2cad: Cad model alignment using multi-view constraints from videos. *ArXiv*, abs/2012.04641, 2020.

[37] Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *CoRR*, abs/1812.03828, 2018.

[38] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Adaptive structure from motion with

a contrario model estimation. In *Proceedings of the Asian Computer Vision Conference (ACCV 2012)*, pages 257–270. Springer Berlin Heidelberg, 2012.

[39] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[40] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *International Conference on Computer Vision (ICCV)*, 2017.

[41] Richard Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. pages 127–136, 10 2011.

[42] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[43] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. Graph.*, 32(6), November 2013.

[44] Stefan Georgiev Popov, Pablo Bauszat, and Vittorio Ferrari. Corenet: Coherent 3d scene reconstruction from a single rgb image. In *The European Conference on Computer Vision (ECCV)*, 2020.

[45] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and S. S. Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Comput. Surv.*, 51(5), September 2018.

[46] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6620–6629, 2017.

[47] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2020.

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[49] Ayan Sinha, Asim Unmesh, Qixing Huang, and Karthik Ramani. Surfnet: Generating 3d shape surfaces using deep residual networks. *CoRR*, abs/1703.04079, 2017.

[50] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576, 2015.

[51] Yu Su, Yu Li, Dan Song, An-An Liu, and Jie Nie. Joint intermediate domain generation and distribution alignment for 2d image-based 3d objects retrieval. *IEEE Transactions on Multimedia*, PP:1–1, 07 2020.

[52] Ke Sun, Yang Zhao, Borui Jiang, Tianheng Cheng, Bin Xiao, Dong Liu, Yadong Mu, Xinggang Wang, Wenyu Liu, and Jingdong Wang. High-resolution representations for labeling pixels and regions. *CoRR*, abs/1904.04514, 2019.

[53] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2107–2115, 2017.

[54] Julien Valentin, Adarsh Kowdle, Jonathan T. Barron, Neal Wadhwa, Max Dzitsiuk, Michael John Schoenberg, Vivek Verma, Ambrus Csaszar, Eric Lee Turner, Ivan Dryanovski, Joao Afonso, Jose Pascoal, Konstantine Nicholas John Tsotsos, Mira Angela Leung, Mirko Schmidt, Onur Gonen Guleryuz, Sameh Khamis, Vladimir Tankovich, Sean Fanello, Shahram Izadi, and Christoph Rhemann. Depth from motion for smartphone ar. *ACM Transactions on Graphics*, 2018.

[55] Girish Varma, Anbumani Subramanian, Anoop M. Namboodiri, Manmohan Chandraker, and C. V. Jawahar. IDD: A dataset for exploring problems of autonomous navigation in unconstrained environments. *CoRR*, abs/1811.10200, 2018.

[56] Zhirong Wu, Shuran Song, Aditya Khosla, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets for 2.5d object recognition and next-best-view prediction. *CoRR*, abs/1406.5670, 2014.

[57] Yang Xiao, Xuchong Qiu, Pierre-Alain Langlois, Mathieu Aubry, and Renaud Marlet. Pose from shape: Deep pose estimation for arbitrary 3D objects. In *British Machine Vision Conference (BMVC)*, 2019.

[58] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, and Shengping Zhang. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019.

[59] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomír Mech, and Ulrich Neumann. DISN: deep implicit surface network for high-quality single-view 3d reconstruction. *CoRR*, abs/1905.10711, 2019.

[60] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving video database with scalable annotation tooling. *CoRR*, abs/1805.04687, 2018.

[61] Hao Zhang and Feng Xu. Mixedfusion: Real-time reconstruction of an indoor scene with dynamic objects. *IEEE Transactions on Visualization and Computer Graphics*, 24(12):3137–3146, 2018.

[62] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ADE20K dataset. *CoRR*, abs/1608.05442, 2016.

[63] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *CoRR*, abs/1801.09847, 2018.