

3D SCENE RECONSTRUCTION SYSTEM BASED ON A MOBILE DEVICE

Damiano Oriti, Andrea Sanna, Francesco De Pace, Federico Manuri,
Francesco Tamburello and Fabrizio Ronzino

Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi, 24, Torino, I-10129, Italy

ABSTRACT

Augmented reality (AR) and virtual reality (VR) applications can take advantage of efficient digitalization of real objects as reconstructed elements can allow users a better connection between real and virtual worlds than using pre-set 3D CAD models. Technology advances contribute to the spread of AR and VR technologies, which are always more diffuse and popular. On the other hand, the design and implementation of virtual and extended worlds is still an open problem; affordable and robust solutions to support 3D object digitalization is still missing. This work proposes a reconstruction system that allows users to receive a 3D CAD model starting from a single image of the object to be digitalized and reconstructed. A smartphone can be used to take a photo of the object under analysis and a remote server performs the reconstruction process by exploiting a pipeline of three Deep Learning methods. Accuracy and robustness of the system have been assessed by several experiments and the main outcomes show how the proposed solution has a comparable accuracy (chamfer distance) with the state-of-the-art methods for 3D object reconstruction.

KEYWORDS

Machine Learning, Object Reconstruction, Object Pose Estimation, Augmented Reality, Virtual Reality

1. INTRODUCTION

In computer graphics and computer vision, 3D object reconstruction tackles the problem of generating a digital three-dimensional representation of an object given some observations (e.g., multiple images from different points of view) of it. Nowadays, digital 3D objects are extensively used in architecture, gaming, augmented reality (AR) and virtual reality (VR), filmmaking, product design, advertisement, manufacturing, cultural heritage, and in many other fields. Common techniques used to generate digital 3D objects are photogrammetry and digital sculpting; the former is the process of extracting three-dimensional information from

two-dimensional data by identifying common points in two or more photos taken from different positions (stereo photogrammetry), whereas the latter can be seen as an *artistic approach* that allows specialized artists to use software and hardware tools to manually make the digital models of objects; for games and other 3D applications modeling 3D assets using software such as Maya, 3ds Max or Blender has been a proved method for many years, but the cost and time resources involved in such a process do not scale well with a society where there is a growing demand for virtual assets.

The digitalization of many human activities in recent decades, connected to the explosion in popularity of computers and smartphones, AR and VR, Internet of Things, and robotics, have entailed a rising demand for less expensive and better 3D object reconstruction systems. For example, digital twins (DTs) can be exploited to train technicians in a safe and cost-efficient manner; virtual environments can be a place to socialize when far apart, and it can also be a safe way to meet friends or relatives when global events such as the Covid-19 pandemic happen.

The most common representations of 3D objects are polygonal meshes, voxels and point clouds (Figure 1 shows different 3D representations of the Stanford bunny¹). A polygonal mesh is a collection of vertices connected through edges and faces, whereas the point clouds are sets of points with no topological information. The voxels are elements of a regular grid similar to a pixel matrix but in 3D space. Besides the positional information, vertices and points can also have other attributes, such as normals and colors. Polygonal meshes are the preferred representation in games or other interactive applications that have to run in real-time, for several reasons: 1) they are efficient to store in memory, 2) modern GPUs are designed to efficiently process/render triangles, which are a common type of polygonal meshes where all faces are triangles, and 3) they are fast to render because almost always the only information that is required for visualization is the object surface, and polygonal meshes are very fit for representing surfaces.

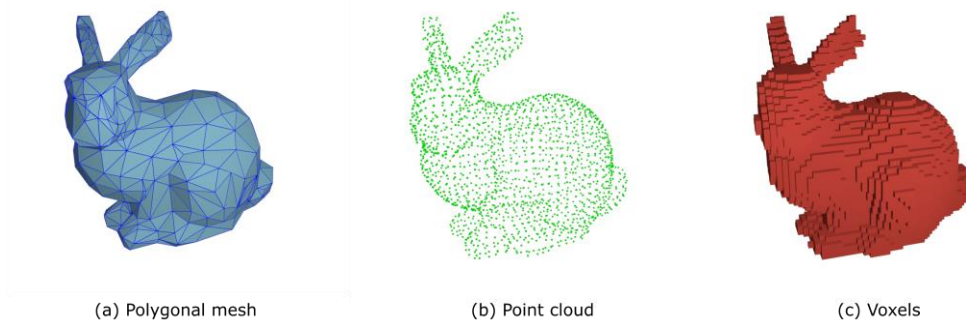


Figure 1. The Stanford bunny represented as (a) polygonal mesh, (b) point cloud and (c) voxels

Recently, some researchers have considered implicit representations for 3D objects such as implicit occupancy functions (Mescheder et al., 2019; Saito et al., 2020) and signed distance functions (Lin et al., 2020). The occupancy function of a volumetric object is the function which, for each point in the three-dimensional space, gives a single real number which is one

¹ <https://graphics.stanford.edu/software/scanview/models/bunny.html>

if the point lies on the surface of the object or inside its volume and zero if the point is outside its volume. Similarly to occupancy functions, distance functions and signed distance functions evaluate to zero if the point lies on the surface, whereas they give the distance (or signed distance) from the closest surface point in any other case. Contrary to polygonal meshes and point clouds, implicit models represent shapes continuously, this can result in higher quality reconstructions.

Following the success of Machine Learning (ML), and specifically Deep Learning (DL) and neural networks (NNs), in solving several image-related tasks (e.g., classification, segmentation, super sampling, deblurring, edge detection, keypoint detection, etc.), many researchers (Guo et al., 2015; Izadinia et al., 2016; Mescheder et al., 2019; Avetisyan et al., 2019; Han et al., 2019; Chen et al., 2020; Lin et al., 2020; Li et al., 2020; Popov et al., 2020) have been trying to apply the same basic principle of learning from data to the 3D object reconstruction problem. Since three-dimensional data can have various representations, different approaches have been proposed that have specific advantages and disadvantages in different applications (Xiao et al., 2020; Fahim et al., 2021).

This work proposes a 3D scene reconstruction system designed for indoor environments that only requires a computer and a smartphone supporting the Google ARCore framework. The smartphone is used (a) to acquire a single RGB image of each real object that the user intends to reconstruct, (b) to visualize the digital object after it has been reconstructed and, if necessary, (c) to adjust its position, rotation and scale relative to the real environment. The computer is used to compute the 3D object geometry as the smartphone is not powerful enough to support the reconstruction task, which is organized as a pipeline whose major steps are the following: (a) image masking aimed at isolating the object of interest by removing the background, (b) watertight mesh construction and (c) rotation estimation. The core of the reconstruction system is a DL solution called BSP-Net (Chen et al., 2020) capable of producing a mesh representation of an object given a single picture of it. The contribution of this paper is twofold: (a) a reconstruction pipeline based on state-of-the-art DL methods for reconstructing a 3D model of a real object given a single RGB image and (b) a smartphone-based system to generate the whole scene that can be used, for example, for AR and VR single or multi-user applications.

This paper is organized as follows: the most relevant works on object reconstruction are presented in Section 2, whereas the proposed reconstruction system is presented in Section 3. The experiments are discussed in Section 4 along with the related results; finally, some possible future work is proposed in Section 5.

2. RELATED WORK

2.1 Deep Learning-Based 3D Object Reconstruction Methods

Several methods for 3D object reconstruction based on Deep Learning that produce three-dimensional assets from a single or multiple images have been proposed in the last decade, thanks to the availability of large-scale 3D shape datasets, such as ShapeNet (Chang et al., 2015), ModelNet (Wu et al., 2015) and ScanNet (Dai et al., 2017). The first approaches used multiple view-representations and voxels and were derived from the growing literature

on machine learning applied to common 2D image-related tasks, such as classification, since generalizing the devised neural networks to the 3D case was mostly trivial; for example, using a convolutional neural network model designed to classify 2D images for voxel classification could only require to switch from 2D convolutions to 3D convolutions. Examples of these are 3D-R²N² by Choy et al. (2016), Häne et al. (2017), OctNet by Riegler et al. (2017), Tatarchenko et al. (2017) and Pix2Vox by Xie et al. (2019). Voxel-based models are easier to devise, but the very high requirement in terms of memory occupation (the memory scales cubically with the grid resolution) can severely limit the resolution of the final output.

Implicit representations usually have much lower memory occupation than voxel-based approaches. Mescheder et al. (2019) use a neural network classifier to represent an implicit 3D surface; instead of generating a volume directly, they can query their classifier to determine if a point is inside or outside the object volume. Similarly, Saito et al. (2020) devised a multi-layer perceptron to act as an occupancy function to reconstruct high resolution human models with color information. SDF-SRN (Lin et al., 2020) learns signed distance functions from 2D images and it only requires a single view of objects at the training time. Although implicit representations give good results in some cases, for example when dealing with organic objects, they are not suitable for objects with hard edges.

Mesh-based and point-based methods are harder to design because meshes and point clouds might have a varying number of vertices/points, whereas neural network-based approaches usually favor fixed data sizes. Some works (Groueix et al., 2018; Pan et al., 2018; Wang et al., 2020) use a template mesh that is deformed to match the shape of the object to be reconstructed or one of its parts, whereas Chen et al. (2020) combine a fixed number of planes whose parameters are inferred using a NN to first produce some convexes, that are then combined to produce the final concave shape. Badki et al. (2020) use meshlets, or mesh patches, and iteratively deform and transform them to match the target shape by optimizing a given loss function; since meshlets are fitted locally, this method is more robust to unseen objects and poses during training compared to approaches that favor global features.

Popov et al. (2020) can reconstruct multiple objects at the same time from a single image computing them in a common coordinate frame.

2.2 Smartphone-based 3D Object Reconstruction Systems

In this Section the most relevant object reconstruction systems that use a smartphone (or a tablet) as a capturing device are presented. Some reconstruction approaches (Tanskanen et al., 2013; Ondrůška et al., 2015) are based on creating a dense representation of the desired object by capturing several observations from different points of view. Tanskanen et al. (2013) use the integrated inertial sensors on the device to make the tracking and mapping process more robust; an efficient stereo matching algorithm is used to compute the depth values from which a dense 3D model with absolute scale is created. Ondrůška et al. (2015) propose an interactive system similar to the KinectFusion (Newcombe et al., 2011) where depth maps obtained through stereo-matching of RGB images are fused volumetrically; an implicit surface representation of the object is extracted, thus allowing the system to estimate a 6 degrees of freedom (DoF) pose for each frame. Muratov et al. (2016) can reconstruct an object with a 2-stage process: in the first stage they scan (i.e., capture multiple pictures) the real object by using the monocular camera of a smartphone while simultaneously acquiring IMU sensors data; then, in the second stage, depth maps are calculated by a bundle adjustment technique

(Triggs et al., 2000) and used to construct a textured polygonal mesh. Donlic et al. (2017) exploit the Digital Light Processing (DLP) projector available on some tablets and pair it to the commonly available RGB camera to make a depth sensor, which works similarly to the Microsoft Kinect; they also take advantage of the built-in IMU sensors to get a robust and accurate point registration that is used to fuse multiple depth maps in a single voxel representation of the object.

The work presented in this paper differs from those approaches by requiring only one picture of the real object, thus making the proposed system much simpler to use.

3. THE 3D OBJECT RECONSTRUCTION SYSTEM

The proposed system is composed of an Android smartphone, which runs an application based on Unity3D² and Google ARCore³, and a computer running an application written in Python.

Unity3D is a popular engine allowing an application to run on the major operating systems and platforms with minimal modifications for the specific target platform. Applications powered by Unity3D are usually developed in the Unity3D editor, which allows to intuitively place different assets directly in the scene by drag and drop; the logic is coded in the C# programming language.

ARCore is a framework developed by Google providing common functionalities for AR applications that run on Android devices, such as smartphones and tablets. The most important features of ARCore are cloud anchors, depth estimation, Visual SLAM-based motion tracking, environmental understanding and light estimation.

The system is designed as a client-server architecture; the client (i.e., a smartphone) is used (a) for capturing pictures of an indoor scene with embedded camera pose information and (b) for 3D object visualization in an AR visualization mode (Section 3.1), whereas the server (i.e., a computer) is used to process the captured photos of objects in order to transform them into 3D models (Section 3.2).

3.1 The Client

The smartphone application has two modes: (a) *photo mode* and (b) *AR object visualization mode* (Figure 2 illustrates the two modes). The photo mode allows users to take photos of the environment; similar to the photo application available on Android or iOS phones, this mode provides a gallery where users can visualize all the pictures they have taken and eventually delete some of them. After taking a picture, the users can save it in the gallery or discard it to take another image.

² <https://unity.com>

³ <https://arvr.google.com/arcore>

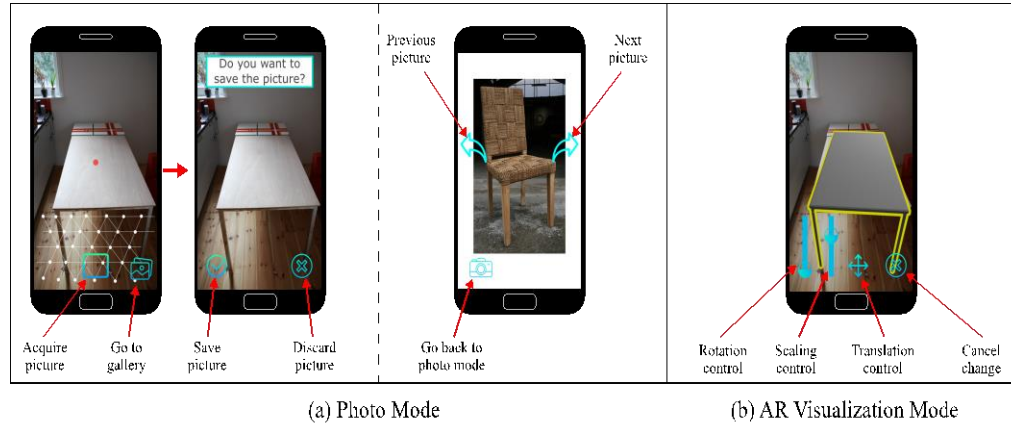


Figure 2. The smartphone application has two modes: (a) *photo mode* and (b) *AR object visualization mode*. The user can capture new photos of the environment in the photo mode (a), whereas images can be displayed in the gallery. The transformation controls can be used to move, rotate or scale virtual objects in the scene in the AR visualization mode (b). The scale is uniform across all dimensions, whereas rotation is limited to the vertical axis. When translated, virtual objects are constrained to stay within the room boundaries

The application converts a new photo from the YUV-420-888 encoding format, that ARCore returns when using the function `GoogleARCore.CameraImageBytes()`, to the RGB format that is required for the correct visualization of the pictures in the Unity3D application and for the reconstruction software running on the server.

Besides using ARCore to take pictures, the smartphone application takes advantage of the Google AR framework to map the environment using its environmental understanding and Visual SLAM-based tracking capabilities; in this way, it is possible to keep track of the smartphone position and rotation relative to a coordinate frame that is constructed at the start of the application; having a global reference frame fixed to the real world is necessary for virtual object visualization because it allows the virtual objects to stay in fixed positions relative to the real objects.

In order to place the digital objects within the boundaries of the real environment, the ARCore plane detection functionality is used to detect floors and walls, which are shown as virtual grids overlapping the actual scene. The user can place one anchor for each object, which is used as a placeholder to position the digital object, on the planes detected by ARCore; using anchors is useful because users will not have to place all virtual objects manually after they have been reconstructed, although adjustments could still be necessary in order to better align virtual and real objects. On the contrary, the rotation is automatically estimated by the server during the reconstruction process by using a DL solution, as explained in Section 3.2.

When at least one object has been reconstructed by the server and sent to the client, users can switch to the AR object visualization mode in order to visualize the reconstructed virtual objects and manipulate them by using the provided on-screen controls. By selecting an object, the user can apply three different transformation operations: translation, rotation and scaling. When an object is selected, the transformation controls appear at the bottom of the screen, as

shown in Figure 2 (b). Since the real objects are supposed to lie upright on the ground, the rotation control is limited to the vertical axis (i.e., yaw angle), allowing the user to correct the rotation in case the estimation fails in giving accurate results. Since the reconstruction process preserves the relative dimensions of the actual object, the scaling control is uniform across all the three dimensions. Translation is constrained to be on the floor and within the room walls that ARCore detects, as the virtual objects should overlap the real objects.

3.2 Data Exchange

After acquiring one or more photos, the user can send them to the server for processing; each photo has an associated identifier (i.e., a unique integer number), thus allowing the client to map a given object reconstruction to the related photo. In order to reliably exchange data between the client and the server, a Transmission Control Protocol (TCP) connection is established between the hosts.

Data is serialized using the JavaScript Object Notation (JSON) open standard file format before the transmission over the network, thus easing the communication between the C# application that runs on the smartphone (client) and the Python software running on the computer (server).

3.3 The Server

3.3.1 Server Configuration and Interprocess Communication

The computer used as server is equipped with a Nvidia GeForce 1060 GPU with 6 GB of VRAM supporting CUDA in order to speed up the DL code that handles the semantic segmentation, the reconstruction and the rotation estimation steps of the pipeline.

The reconstruction program and the program handling the data transfer between the server and the client are two separate processes and they are executed independently of each other.

Communication and synchronization between the two processes are handled through file writing and reading. When the server data transfer process receives an image and its identifier from a client through the network, it saves it in a folder, which is periodically checked by the reconstruction process. When an unprocessed image is found, the server reconstructs the corresponding object mesh through the reconstruction process and it sends the result with its corresponding identifier back to the client. Processed images are deleted after having been processed in order to avoid to process them again.

3.3.2 Reconstruction Pipeline

The server executes the reconstruction pipeline which is illustrated in Figure 3. The first step segments the image in foreground (i.e., the region of the image containing all the pixels of the object) and background (i.e., all pixels not belonging to the object), inasmuch it is not guaranteed that the user can take pictures with a perfectly uniform background, which is necessary for BSP-Net to output a clean mesh. In fact, this neural network was trained by using a synthetic dataset containing renders of 3D shapes from ShapeNet, where each render included only the object on a uniformly colored background.

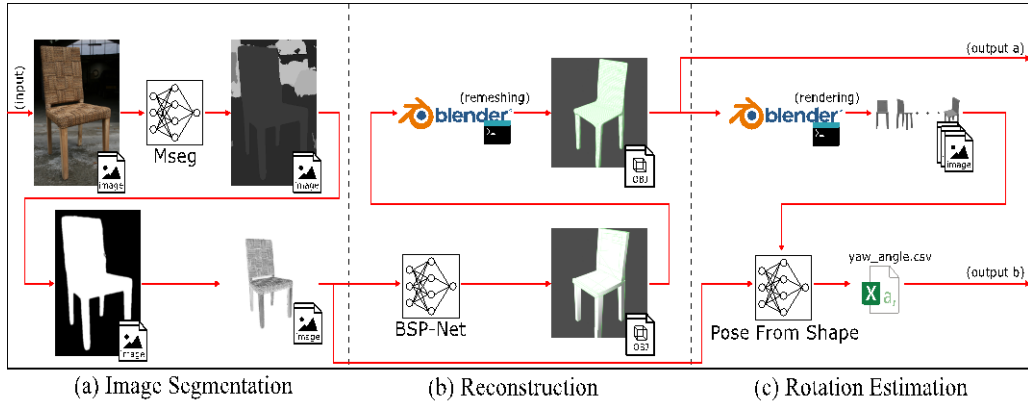


Figure 3. The reconstruction pipeline consists of three steps: (a) isolation of the object of interest in the image by removing the background, (b) construction of a watertight mesh and (c) estimation of the object rotation about the vertical axis. The three major steps are in turn divided into seven sub-steps

In order to determine which pixels belong to the foreground object, Mseg (Lambert et al., 2020) has been used. Mseg is a neural network model based on HRNet (Sun et al., 2019) designed for high resolution semantic segmentation of images; semantic segmentation is the task of assigning a class or category label (i.e., determining which class the object belongs to) to every pixel in the image. Given the original, color picture, Mseg provides a new image with gray-coded segmentation regions, where each tone of gray represents a specific object class.

In order to simplify the background removal process, assuming that the central image pixel includes the object of interest, a flood fill operation can be executed in order to convert the semantic segmentation image to a binary mask that can be later used to separate the foreground object from the background. The binary mask can present jagged edges, which may have a negative effect on the reconstruction by altering the silhouette of the object; in order to minimize the effect of jagged edges, the binary mask is blurred by using a Gaussian filter. At the current stage, if the photo contains multiple instances of the same object category, the segmentation works correctly only if the instances are clearly separated. In fact, the segmentation network employed can only separate object categories, but not object instances; when two or more objects of the same category overlap in the image, the segmentation network will return a single region embedding all instances, making it impossible to isolate any one of them by using just the segmentation mask. Before executing BSP-Net, the new image with the removed background is first converted to a grayscale image, then it is cropped, resized and padded in order to have a fixed-sized image with constant padding, which helps to have consistent results, as shown in Section 4. Adding the padding is necessary because it allows to match the format of the renders used during training. Figure 4 shows how using the wrong image format and incorrectly segmenting the object affect the reconstruction.













Original photos					
	Squared image	Non-squared image	Accurately segmented image	Wrongly segmented image	Image without padding
BSP-Net input					
BSP-Net output					

Figure 4. Examples of failure cases: for the chair, a comparison between a squared image and a non-squared image is shown; feeding a non-squared image to BSP-Net will result in a stretched 3D model. On the right, the reconstruction of a table is shown in three different cases; in the first image the background was accurately removed through a manual process and padding was added to get the best result. In the second case, the automatic segmentation performed by Mseg fails, resulting in a broken 3D model. In the third case, the image is again accurately segmented, but no padding is added, resulting in a partially broken and incorrectly scaled 3D model

The second step is the construction of a mesh given the grayscale picture of the object without the background. A grayscale image is required for BSP-Net because the neural network is designed to process grayscale images, although it could technically be modified to accept color images as input. Secondly, BSP-Net has been trained on 128×128 px images, thus the grayscale image is also downscaled to the required pixel resolution.

After converting the input to the appropriate format, BSP-Net is executed and the constructed mesh is saved on the disk as a Wavefront OBJ file.

The constructed mesh is watertight, that is, it consists of one closed surface without holes; this is particularly useful to have for 3D games because the use of backface culling (i.e., skipping triangles pointing in the same direction as the camera while rendering a given frame) would result in parts of the mesh incorrectly disappearing when rendered from specific points of view. BSP-Net has been designed to exploit the binary space partitioning idea: a 3D shape is constructed as a union of convex parts, which are in turn obtained by combining multiple planes. Since BSP-Net constructs the 3D shape as a union of convex parts, it can produce concave shapes. However, the output tends to be very messy with a lot of intersections among the convex parts. In order to get a topologically cleaner mesh, a remeshing algorithm provided by Blender⁴ is employed (Figure 5 shows a comparison between the output of BSP-Net and the remeshed object). The remeshing algorithm,

⁴ <https://www.blender.org>

implemented as part of OpenVDB⁵, is based on voxels and level sets; it first converts the mesh to voxels with a given voxel size, then it extracts a given isosurface mesh from the grid.

The final step of the reconstruction pipeline is the estimation of the object rotation about the vertical axis, since it is assumed that the objects lie upright on a horizontal plane (usually the floor). The Pose From Shape (Xiao et al., 2019) neural network is employed in order to estimate the object rotation. As its name suggests, Pose From Shape is capable of estimating the pose of a real object given several views of it, or even of a simplified 3D model approximating the real object. The proposed solution exploits the reconstructed mesh in order to produce, through rendering, twelve views of the object with a yaw in the range $[0, 360^\circ]$ with a delta angle of 60° and a roll in the range $[0, 30^\circ]$ with a delta angle of 30° .

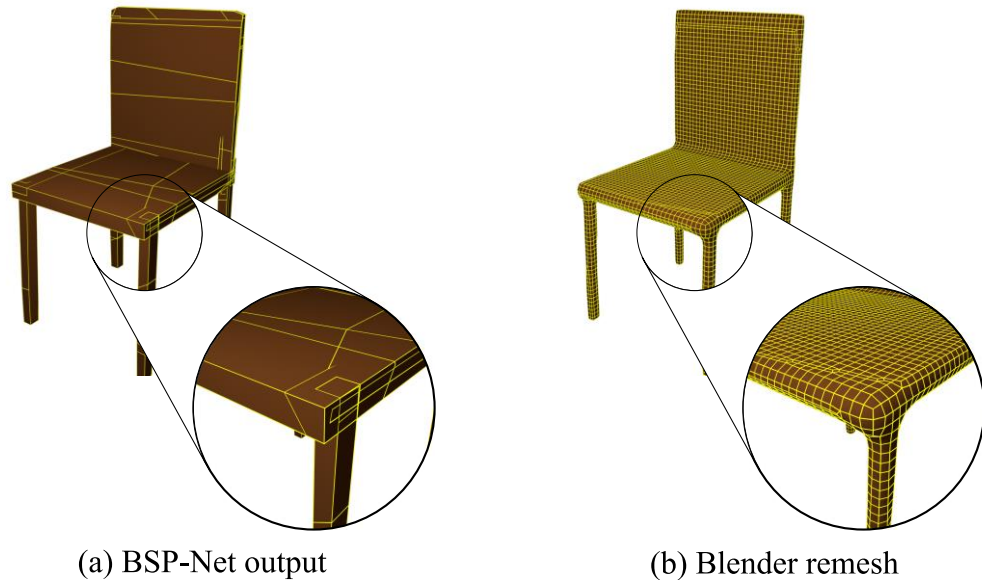


Figure 5. Comparison between the polygonal mesh model produced by BSP-Net and the remeshed model. Blender is used in command-line mode in order to remesh the output of BSP-Net. The typical mesh constructed by BSP-Net has an irregular topology (a); the new mesh has a cleaner topology after remeshing (b)

The rendered model is the remeshed object and Blender is used in command-line mode to automatically render the required views (Figure 4 (c) illustrates the described process from rendering to object pose estimation).

⁵ <https://www.openvdb.org>

4. EXPERIMENTS

Table 1. Reconstruction accuracy of BSP-Net (synthetic data) and the proposed pipeline (real photos) for three different object classes. The evaluation metric is the chamfer distance (lower is better)

Class	BSP-Net (synthetic data)	Ours (real photos)
Chair	0.067	0.073
Table	0.129	0.174
Sofa	0.090	0.071
Mean	0.092	0.093

The proposed reconstruction pipeline has been tested by using samples from the Pix3D dataset (Sun et al., 2018), which contains images of real objects and associated ground-truth (GT) 3D models and poses. Pix3D is one of the few datasets for 3D tasks that do not use synthetic images; moreover, contrary to other datasets, Pix3D provides precise alignment between 2D and 3D shapes. Three object categories have been tested on which both BSP-Net and Mseg had been trained: chairs, sofas and tables.

In order to quantitatively evaluate the reconstruction accuracy, the chamfer distance (CD) (Chen et al., 2020) metric has been used for each pair of reconstructed model and GT CAD model; this metric is commonly used to compare the reconstruction accuracy when working with three-dimensional data. The CD is defined for point clouds as the summation of minimum distances between all points of the point clouds.

$$1) \quad CD(P_1, P_2) = \frac{1}{|P_1|} \sum_{x \in P_1} \min_{y \in P_2} \|x - y\|_2 + \frac{1}{|P_2|} \sum_{x \in P_2} \min_{y \in P_1} \|x - y\|_2$$

The meshes obtained by using the described reconstruction pipeline have been initially converted to a voxel grid with resolution 32^3 , then the voxels have been converted to a point cloud by taking the center point of each voxel. The same procedure has been used to convert the GT meshes provided by Pix3D to point clouds. Before computing the CD, the point clouds have been translated in order to have the bounding box centered in the world origin, then they have been scaled in order to have the same size along the longest dimension.

Table 1 shows the average CD for the original BSP-Net when operating on synthetic images and the proposed pipeline operating on real photos. The results show that the proposed pipeline produces reasonably accurate reconstructions starting from a single real photo, despite the fact that BSP-Net had been trained only on synthetic images and Mseg does not produce an accurate segmentation (Figure 6 shows some reconstructed objects from real photos).

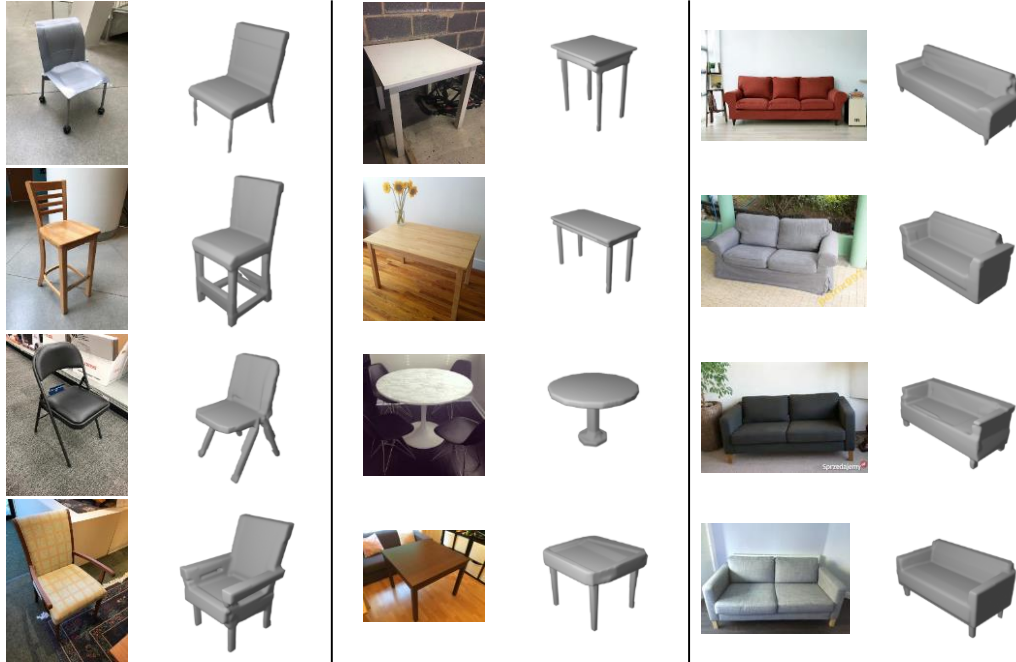


Figure 6. Some reconstructed models obtained by applying the proposed pipeline on Pix3D image samples

The pipeline has been designed to be as modular as possible, allowing to easily swap existing modules. As an example, the semantic segmentation module can be replaced with an improved version or with an alternative solution. The modular architecture of the reconstruction pipeline allows, for example, to use an occupancy network in place of BSP-Net, as long as it accepts an image as input and it outputs a 3D object representation that can be converted to a mesh. Experiments in this regard were conducted in order to find a suitable reconstruction method, which ended with the choice of BSP-Net as it performed better (Chen et al., 2020) than the tested occupancy network proposed by Mescheder et al. (2019) for the categories of interest, such as chairs, sofas and tables.

5. CONCLUSION

A new reconstruction solution is presented in this paper; AR and VR applications can take advantage of it especially when extended environments can be shared by multiple users. The proposed solution exploits three different DL methods to reconstruct a 3D model of a real object framed in a single RGB image captured by a smartphone and remotely processed using a computer.

One limitation of the proposed system is that it requires a smartphone, even if the user is using a VR or AR head-mounted display device that is equipped with a camera and that, therefore, could be used in place of the smartphone for image acquisition and object pose

adjustment. Objects belonging to the same class cannot be managed if they overlap when framed as the segmentation stage is unable to distinguish among multiple instances of the same object class. Also background parts might be included as object and this can limit the reconstruction accuracy.

Some possible future works will consider the possibility to run all the reconstruction process on the only client device used to capture the object images (e.g. a Microsoft HoloLens 2) and enhancing BSP-Net in order to take into account the object color and retraining it with real or photorealistic images.

REFERENCES

- Avetisyan et al., 2019. Scan2CAD: Learning CAD Model Alignment in RGB-D Scans. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA, pp. 2609-2618.
- Badki et al., 2020. Meshlet Priors for 3D Mesh Reconstruction. *Proceedings of 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA.
- Chang et al., 2015. ShapeNet: An Information-Rich 3D Model Repository. *Journal CoRR*, Vol. abs/1512.03012.
- Chen et al., 2020. BSP-Net: Generating Compact Meshes via Binary Space Partitioning. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA, pp. 42-51.
- Choy et al., 2016. 3D-R²N²: 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. *Proceedings of the European Conference on Computer Vision (ECCV)*. Amsterdam, The Netherlands.
- Dai et al., 2017. ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes. *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA.
- Donlic et al., 2017. On Tablet 3D Structured Light Reconstruction and Registration. *Proceedings of IEEE International Conference on Computer Vision Workshops (ICCVW)*. Venice, Italy, pp. 2462-2471.
- Fahim et al., 2021. Single-View 3D reconstruction: A Survey of deep learning methods. *Journal Computers & Graphics*, Vol. 94, pp. 164-190.
- Groueix et al., 2018. A Papier-Mache Approach to Learning 3D Surface Generation. *Proceedings of 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA.
- Guo et al., 2015. Predicting Complete 3D Models of Indoor Scenes. *Journal CoRR*, Vol. abs/1504.02437.
- Han et al., 2019. Image-Based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era. *Journal IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 43, No. 5, pp. 1578-1604.
- Häne et al., 2017. Hierarchical Surface Prediction for 3D Object Reconstruction. *Proceedings of 2017 International Conference on 3D Vision (3DV)*. Qingdao, China.
- Izadinia et al., 2017. IM2CAD. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA, pp. 2422-2431.
- Lambert et al., 2020. MSeg: A Composite Dataset for Multi-Domain Semantic Segmentation. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA, pp. 2876-2885.

- Li, K., Rünz, M., et al., 2020. FroDO: From Detections to 3D Objects. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, US, pp. 14708-14717.
- Lin et al., 2020. SDF-SRN: Learning Signed Distance 3D Object Reconstruction from Static Images. *Proceedings of Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*. USA, pp. 11453-11464.
- Mescheder et al., 2019. Occupancy Networks: Learning 3D Reconstruction in Function Space. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA, pp. 4455-4465.
- Muratov et al., 2016. 3DCapture: 3D Reconstruction for a Smartphone. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Las Vegas, NV, USA, pp. 893-900.
- Newcombe et al., 2011. KinectFusion: Real-time dense surface mapping and tracking. *Proceedings of 10th IEEE International Symposium on Mixed and Augmented Reality*. Basel, Switzerland, pp. 127-136.
- Ondruška et al., 2015. MobileFusion: Real-Time Volumetric Surface Reconstruction and Dense Tracking on Mobile Phones. *Journal IEEE Transactions on Visualization and Computer Graphics*, Vol. 21, No. 11, pp. 1251-1258.
- Pan et al., 2018. Residual MeshNet: Learning to Deform Meshes for Single-View 3D Reconstruction. *Proceedings of 2018 International Conference on 3D Vision (3DV)*. Verona, Italy.
- Popov et al., 2020. CoReNet: Coherent 3D Scene Reconstruction from a Single RGB Image. *Proceedings of European Conference on Computer Vision (ECCV)*. Glasgow, UK, pp. 366-383.
- Riegler et al., 2017. OctNet: Learning Deep 3D Representations at High Resolutions. *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA.
- Saito et al., 2020. PIFuHD: Multi-Level Pixel-Aligned Implicit Function for High-Resolution 3D Human Digitization. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA, pp. 81-90.
- Sun et al., 2019. High-Resolution Representations for Labeling Pixels and Regions. *Journal CoRR*, Vol. abs/1904.04514.
- Sun et al., 2018. Pix3D: Dataset and Methods for Single-Image 3D Shape Modeling. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA, pp. 2974-2983.
- Tanskanen et al., 2013. Live Metric 3D Reconstruction on Mobile Phones. *Proceedings of IEEE International Conference on Computer Vision*. Sydney, NSW, Australia, pp. 65-72.
- Tatarchenko et al., 2017. Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs. *Proceedings of 2017 IEEE International Conference on Computer Vision (ICCV)*. Venice, Italy.
- Triggs et al., 1999. Bundle Adjustment - A Modern Synthesis. *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*. Corfu, Greece, pp. 298-372.
- Wang et al., 2020. Pixel2Mesh: 3D Mesh Model Generation via Image Guided Deformation. *Journal IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 43, Issue 10, pp. 3600-3613.
- Wu et al., 2015. 3D ShapeNets: A deep representation for volumetric shapes. *Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA.
- Xiao et al., 2020. A survey on deep geometry learning: From a representation perspective. *Journal Computational Visual Media*, Vol. 6, pp. 113-133.
- Xiao et al., 2019. Pose from Shape: Deep Pose Estimation for Arbitrary 3D Objects. *Proceedings of 30th British Machine Vision Conference (BMVC)*. Cardiff, UK.
- Xie et al., 2019. Pix2Vox: Context-Aware 3D Reconstruction from Single and Multi-View Images. *Proceedings of 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea (South).