Three-dimensional shape reconstruction of objects from a single depth view using deep U-Net convolutional neural network with bottle-neck skip connections

Edwin Valarezo Añazco\textsuperscript{1} | Patricio Rivera Lopez\textsuperscript{1} | Tae-Seong Kim\textsuperscript{1,2}

\textsuperscript{1}Department of Electronics and Information Convergence Engineering, College of Electronics and Information, Kyung Hee University, Republic of Korea
\textsuperscript{2}Department of Biomedical Engineering, College of Electronics and Information, Kyung Hee University, Republic of Korea

Abstract

Three-dimensional (3D) shape reconstruction of objects requires multiple scans and complex reconstruction algorithms. An alternative approach is to infer the 3D shape of an object from a single depth image (i.e. single depth view). This study presents such a 3D shape reconstructor based on U-Net 3D-convolutional neural network (3D-CNN) with bottle-neck skipped connection blocks (U-Net BNSC 3D-CNN) to infer the 3D shapes of objects from only a single depth view. The BNSC block is a fully convolutional block that uses skip connections to improve the performance of the sequential 3D-convolutional layers of U-Net. The primary advantage of U-Net BNSC 3D-CNN is improving the accuracy of shape reconstruction while reducing the computational load. The evaluation of the proposed U-Net BNSC 3D-CNN uses unseen views from trained and untrained objects with two public databases, ShapeNet and Grasp database. Our reconstructor achieves 72.17% and 69.97% accuracy in terms of the Jaccard similarity index for trained and untrained objects, respectively, with the ShapeNet database, whereas previous reconstructor based on 3D-CNN achieves 66.43% and 58.35%. With Grasp database, our reconstructor achieves 87.03% and 85.35%, whereas 3D-CNN 76.52% and 76.02%. Also, our U-Net BNSC 3D-CNN reduces the computational load of the standard 3D-CNN reconstructor by 6.67% in the computation time and by 98.69% in the number of trainable parameters.

1 | INTRODUCTION

In general, three-dimensional (3D) shape reconstruction of various objects involves scanning the objects from multiple view angles with RBG-D sensors\textsuperscript{[1–4]}, a laser scanner\textsuperscript{[5]}, or stereo cameras\textsuperscript{[6,7]}. The views are then patched together via reconstruction algorithms to generate the 3D shape of an object\textsuperscript{[8,9]}. Using multiple scanning and patch-up algorithms to reconstruct a 3D shape is time consuming and effort intensive. The initial works for 3D shape reconstruction used point clouds or meshes to generate a rough 3D shape of an object by assuming symmetry\textsuperscript{[10–15]}. Thus, those approaches have difficulty reconstructing irregularly shaped objects (i.e. asymmetric objects). Other works reconstructed 3D shapes by classifying an object according to its similarity with a database of CAD models\textsuperscript{[16–20]}. Those approaches cannot reconstruct objects without a matching model in the dataset. It would be beneficial to have an intelligent method by which to reconstruct 3D objects from unseen views of learned (i.e. trained) objects, and it would be even more beneficial if the method could reconstruct 3D objects from unseen views of unlearned (i.e. untrained) objects.

Recently, 3D shape reconstruction via deep learning (DL) artificial intelligence from a single depth view of an object has offered an alternative solution\textsuperscript{[21,22]}. These approaches require less time and effort than conventional 3D shape reconstruction techniques\textsuperscript{[8,9]}. The basic idea is to train DL intelligence with a variety of learning objects and then use that intelligence to infer or reconstruct 3D shapes from unseen views of both learned and unlearned objects. Han et al.\textsuperscript{[23]} presented an algorithm built in two modules that could reconstruct the 3D shapes of six objects. The first module was a 3D fully convolutional network combined with...
long short-term memory. It was used to infer the global structure of an object from an incomplete point cloud and multi-view depth information. The second module based on an autoencoder generated the complete 3D shape of the object from the output of the first module. That reconstruction algorithm achieved 96.1% of completeness with 50% of the complete volume as input. They defined completeness as the fraction of points in the reconstructed volume within a distance \( \alpha \) (i.e., 0.001 times the maximum shape diameter) of any point in the ground-truth. The work of Smith and Meger [21] used a variational autoencoder and generative adversarial networks (GANs). Their system focused on 3D shape reconstruction from a single depth view of the object. However, their GAN architecture suffered from instability, especially when training involved multiple views from multiple objects [21]. Also, they provided only qualitative results showing some sample reconstructed objects from the training database and an actual Kinect scan. In their latest work, Varley et al. [22] implemented a 3D-convolutional neural network (3D-CNN) for 3D shape reconstruction from a single depth view of an object. Their 3D-CNN is a large neural network involving 148.4 million of trainable parameters. Also, their system offered limited performance in reconstructing untrained objects. The reconstruction accuracy (Jaccard similarity index) dropped from 74.86% with trained objects to 64.96% with untrained objects.

In this study, we present a novel 3D shape reconstructor to infer the 3D shapes of seen and unseen objects. The proposed shape reconstructor uses only a single depth view of an object to infer or reconstruct its full 3D shape. The design of our reconstructor is based on U-Net CNN because its connections between the contractive and expansive paths provide more feature information to the expansive path for 3D shape reconstruction [24]. In our architecture, the U-Net structure is extended from a 2D-CNN to a 3D-CNN to increase the receptive field through a 3D-CNN kernel [25]. In addition, we add two novel components to the design of the reconstructor: a skip connection block (SC block) and bottle-neck layers. The skip connections are pathways between non-consecutive layers that propagate the training error in the backward pass [26]. The propagation of the training error improves the training and thus the 3D reconstruction performance. To reduce the computational load, bottle-neck layers are added to the SC block (BNSC block). The bottle-neck layers reduce the training time of the SC block by reducing the number of trainable parameters and computational cost.

We evaluated our U-Net 3D-CNN with BNSC blocks (U-Net BNSC 3D-CNN) against three baseline networks including 3D-CNN, 3D-CNN with SC block, and 3D-CNN with BNSC block. The evaluations were performed using unseen views from seen (learned) objects and unseen views from unseen (unlearned) objects. The results show that our 3D shape reconstructor achieves superior Jaccard similarity index values (i.e., reconstruction accuracy) for both trained and untrained objects compared with the baseline networks.

2 | METHODS

Our proposed 3D-shaped reconstructor is shown in Figure 1. The input of our reconstructor is a single depth view of an object expressed in voxels. The proposed U-Net BNSC 3D-CNN is shown with a U-Net 3D-CNN, skip connections, and bottle-neck layers. The output of the proposed U-Net BNSC 3D-CNN is an inferred and reconstructed 3D shape of an object, also expressed in voxels.

2.1 | Baseline networks

2.1.1 | 3D-CNN

The 3D-CNN is a neural network based on the classic CNN structure that contains some convolutional and pooling layers followed by fully connected layers and output layers [27]. The shape reconstruction algorithm in Varley et al. [22] is based on such a 3D-CNN with three 3D-convolutional layers and two dense layers. The output layer is a dense layer with 64,000 nodes used to reconstruct the shape of the object in a voxel grid of \( 40 \times 40 \times 40 \).

In this study, we implement a 3D-CNN with five 3D-convolutional layers, three fully connected layers (i.e., dense layers), and one 2D-convolutional layer as the output layer. Figure 2a shows our implemented 3D-CNN. Each 3D-convolutional layer extracts 64 features (@64) with a kernel size of \( 4 \times 4 \times 4 \). A drop-out layer was added right after the last 3D-convolutional layer to prevent overfitting. The output layer is a 2D-convolutional layer (instead of a dense layer with 64,000 nodes) to reduce the number of trainable parameters from that found in Varley et al. [22]. The 2D-convolutional layer extracts 160 features (@160) with a kernel size of \( 3 \times 3 \).

2.1.2 | Skip connection block 3D-CNN

The skip connections are called a shortcut connection in Ref. [28], which is a connection between non-consecutive layers in a neural network. The key idea of a skip connection is to improve the training, especially in deep or wide networks [29–31]. Skip connections are pathways that transfer the training error back to the initial layers (i.e., backpropagation error), which produces a training improvement [26,32].

One of the critical requirements for 3D shape reconstruction is reducing the computational load because conventional 3D shape reconstructors generally use very deep networks (i.e., many trainable parameters) with prolonged training time. SC block is constructed as shown in Figure 2b with the following characteristics. First, the classic residual blocks (i.e., SC blocks) are generally used in very deep networks [28]. Then, the SC block was modified to skip one convolutional layer instead of two convolutional layers, similar to Huang et al. [32]. Second, different operations to combine the output of \( X_2 \) with the output of \( X_1 \) were explored: addition
3D depth sensor

**Figure 1** Overview of our three-dimensional shape reconstruction process

![Figure 1](image1.png)

**Figure 2** Architecture of the baseline three-dimensional (3D) reconstructor based on 3D-convolutional neural network (3D-CNN). (a) 3D-convolutional block (3D-CNN), (b) skip connection block (SC 3D-CNN), (c) bottle-neck block (BNSC 3D-CNN)

![Figure 2](image2.png)

[28], concatenation [32], and average [33,34]. In our initial tests, the average operation performed slightly better than the others in terms of reconstruction accuracy.

To evaluate the effect of the SC block for 3D shape reconstruction, we performed evaluations with and without the skip connections, as shown in Figure 2b,a, respectively.

### 2.1.3 BNSC block 3D-CNN

We designed our BNSC block to reduce the computational load of the SC block by adding bottle-neck layers. A bottle-neck layer is a convolutional layer with a kernel size of $1 \times 1 \times 1$ that reduces or restores the feature map [28,35]. The bottle-neck layers reduce the number of trainable parameters and computational cost through their $1 \times 1 \times 1$ kernels. Figure 2c shows our BNSC block, which has two bottle-neck layers. The first bottle-neck layer reduces the feature maps from @64 to @16. At the end of the block, another bottle-neck layer increases (i.e. restores) the feature maps from @16 to @64. Even though the number of convolutional layers increases from two in the SC block to four in the BNSC block, the training time of our BNSC block should be less than the SC block because of the reduction in the computational cost. Also, the BNSC block has fewer trainable parameters than the SC block.

The number of trainable parameters can be computed using Equation (1) [36,37]:

$$\text{Number of trainable parameters} = \text{Input Features} \times \text{Kernel Size} \times \text{Output Features} + \text{Bias},$$  \hspace{1cm} (1)
Computations in a convolutional layer, as calculated using Equation (1), the number of trainable parameters at the end of the SC block is 266,368. The input volume for all convolutional layers in the SC and BNSC blocks is 40 × 40 × 40. According to Equation (2), the computational cost of the SC block, as shown in Figure 2c.

### TABLE 1 The number of trainable parameters and computational cost of the SC block

| SC block | Number of trainable parameters | Total | Computational cost |
|----------|-------------------------------|-------|-------------------|
| 1° convolution | 1 × (4 × 4 × 4) × 64 + 64 | 4160 | (40 × 40 × 40) × 1 | 262,144,000 |
| 2° convolution | 64 × (4 × 4 × 4) × 64 + 64 | 262,208 | (4 × 4 × 4) × 64 | 16,777,216,000 |
| Total number of trainable parameters | 266,368 | | Total number of computations | 17,039,360,000 |

### TABLE 2 The number of trainable parameters and computational cost of the BNSC block

| BNSC block | Number of trainable parameters | Total | Computational cost |
|------------|-------------------------------|-------|-------------------|
| 1° convolution | 1 × (4 × 4 × 4) × 64 + 64 | 4160 | (40 × 40 × 40) × 1 | 262,144,000 |
| 2° convolution | 64 × (1 × 1 × 1) × 16 + 16 | 1040 | (4 × 4 × 4) × 16 | 1,048,576,000 |
| 3° convolution | 16 × (4 × 4 × 4) × 16 + 16 | 16,400 | (4 × 4 × 4) × 16 | 65,536,000 |
| 4° convolution | 16 × (1 × 1 × 1) × 64 + 64 | 1088 | (4 × 4 × 4) × 64 | 1,441,792,000 |
| Total number of trainable parameters | 22,688 |

The computational cost is defined as the number of computations in a convolutional layer, as calculated using Equation (2) [37,38]:

\[
\text{Computational cost} = \text{Input Volume} \times \text{Input Features} \times \text{Kernel Size} \times \text{Output Features,}
\]

where, Input Volume is the size of the input information for the 3D-convolutional layer (40 × 40 × 40 before the first pooling layer).

For instance, the first SC block in Figure 2 has two convolutional layers with a kernel size of 4 × 4 × 4. The input feature is @1 and @64 for each convolution. The output features of both convolutional layers are @64. Using Equation (1), the number of trainable parameters at the end of the first SC block is 266,368. The input volume for all convolutional layers in the SC and BNSC block is 40 × 40 × 40. According to Equation (2), the computational cost of the SC block is 17,039 million computations. In contrast, the BNSC block has four convolutional layers with kernel sizes of 4 × 4 × 4, 1 × 1 × 1, 4 × 4 × 4, and 1 × 1 × 1. The input features are thus @1, @64, @16, and @16, and the output features are @64, @16, @16, and @64. Using Equation (1), the number of trainable parameters at the end of the BNSC block is 22,688, 11.74 times fewer than the SC block. Using Equation (2), the computational cost of our BNSC block is 1441 million computations, 11.82 times fewer computations than the SC block. Thus, in the BNSC block, fewer trainable parameters need to be optimized comparing with the SC block, and the computational cost is also less. The comparison details in terms of the number of trainable parameters and computational costs are given in Tables 1 and 2, respectively.

The number of trainable parameters and computational cost of the first SC block is 266,368. The input volume for all convolutional layers in the SC and BNSC blocks is 40 × 40 × 40. According to Equation (2), the computational cost of the SC block, as shown in Figure 2c.

### Proposed U-Net BNSC 3D-CNN

U-Net is a network based on convolutional layers with two paths: a contractive one that captures context information from the feature extracted by the convolutional layers and an expansive path that reconstructs the object shape based on the features extracted by the contractive path [39]. Furthermore, U-Net uses some skip connections to connect the layers from the contractive path with their equivalents in the expansive path, providing more feature information for shape reconstruction.

In our reconstructor, only 3D-convolutional layers are used in the U-Net because their larger 3D receptive field improves the performance [25]. Furthermore, the BNSC block is used because BNSC 3D-CNN outperforms the 3D shape reconstruction performance of 3D-CNN. Finally, to propagate the feature information to the whole network, extra connections are added in the contractive path of our U-Net BNSC 3D-CNN reconstructor, as shown in Figure 3.

Figure 3 shows the proposed U-Net BNSC 3D-CNN. The input layer is a 3D-convolutional layer that extracts 64 features (@64) with a kernel size of 4 × 4 × 4. Then, two BNSC blocks and two pooling layers are used in the contractive path, along
with two skip connections, as in Ref. [40]. However, we use an addition operation instead of the concatenation operation used in Ref. [40]. In the middle of our U-Net BNSC 3D-CNN, a third BNSC block is used before the first up-sampling layer. The expansive path uses two BNSC blocks, two up-sampling layers, and skip connections between the contractive and expansive paths with a concatenate operation, as in Refs. [41,42]. The output layer is a 3D-convolutional layer with a kernel size of $4 \times 4 \times 4$ and one feature, corresponding to the reconstructed 3D object shape expressed in voxels.

Previous 3D shape reconstructors are based on an autoencoder with the U-Net connections between the encoder and decoder [43,44]. These previous reconstructors [43,44] differ from the original U-Net architecture because of the dense layers at the end of the encoder. The proposed U-Net BNSC 3D-CNN differs from the 3D autoencoder with the U-Net connections because our reconstructor uses BNSC blocks and skip connections in the contractive path.

The loss function used to train the proposed U-Net BNSC 3D-CNN is a weighted version of the cross-entropy function in Ref. [44]. It was designed to update the values with more emphasis when the voxel grid has information about the object. Equation (3) describes the weighted loss function:

$$\mathcal{L} = -\frac{1}{N} \sum_{n=1}^{N} \left[ y_n \log \hat{y}_n + (1 - y_n) \log \left(1 - \hat{y}_n\right)\right], \quad (3)$$

where, $N$ represents the total number of voxel grids times the number of samples in each training batch; $y_n$ is the true voxel grid value; $\hat{y}_n$ is the predicted voxel grid value; and $\gamma$ is the weight value (three in our case).

For the final output of the 3D shape reconstructor, all voxel grid values with a probability of less than 0.75 are converted to 0 and those with a probability greater than 0.75 are converted to 1, producing a reconstructed 3D object shape.

### 2.3 Databases

To train and evaluate all the implemented 3D shape reconstructors, we used two public databases. First, the Grasp database [45] offers 590 objects as mesh models. The objects include tools, toys, groceries, drugstore products, and household objects. Second, the ShapeNet database [46] offers 3135 categories of CAD models, classified into warehouse objects.
such as toys and household objects such as furniture. The public databases were chosen because of their variety of objects and popularity in the research community. All objects are expressed in voxels as binvox files [22] and can be read using Binvox software, which rasterizes 3D object models into binary 3D voxel grids [47,48].

The binvox files were created using the following guidelines [22]. First, the meshes collected from the Grasp database were placed in Gazebo [49], which is 3D software for robotic simulations. Using a virtual depth camera, several partial depth views of the imported mesh objects were created. Second, using the partial depth views and Binvox, an occupancy grid of the visible pixels was created, with 1s assigned to object voxels and 0s to the background. Finally, the shape corresponding to the visible pixels was created, with 1s assigned to object voxels using the partial depth views and Binvox, an occupancy grid of 2.4 views of the imported mesh objects were created. Second, simulations. Using a virtual depth camera, several partial depth views were placed in Gazebo [49], which is 3D software for robotic lines [22]. First, the meshes collected from the Grasp database narrowly 3D voxel grids [47,48]. Binvox software, which rasterizes 3D object models into binary 3D voxel grids [47,48] and can be read using such as toy's and household objects such as furniture. The public databases were chosen because of their variety of objects and popularity in the research community. All objects are expressed in voxels as binvox files [22] and can be read using Binvox software, which rasterizes 3D object models into binary 3D voxel grids [47,48].

The binvox files were created using the following guidelines [22]. First, the meshes collected from the Grasp database were placed in Gazebo [49], which is 3D software for robotic simulations. Using a virtual depth camera, several partial depth views of the imported mesh objects were created. Second, using the partial depth views and Binvox, an occupancy grid of the visible pixels was created, with 1s assigned to object voxels and 0s to the background. Finally, the shape corresponding to the partial view was down-sampled to a 3D voxel grid of 40 × 40 × 40 and saved as a Binvox file.

2.4 Evaluation methodologies

To evaluate the baseline networks (i.e. 3D-CNN, SC 3D-CNN, and BNSC 3D-CNN) and our proposed U-Net BNSC 3D-CNN, two evaluation methodologies were used. First, we conducted unseen views testing (unseen views of learned objects) to evaluate the performance of our reconstructors with trained objects. Second, we conducted unseen objects testing (unseen views of unlearned objects) to evaluate the performance of the implemented networks with untrained objects. Both evaluation methodologies were implemented using Grasp and ShapeNet databases.

From the Grasp database, three datasets were derived. First, the training dataset includes 250 partial views from 76 different objects. Second, the unseen views dataset contains 50 views not included in the training of each of the same 76 objects used in the training dataset. Third, the unseen objects dataset contains 50 unseen views of each of 15 objects not included in the training dataset.

From the ShapeNet database, we used the same categories of objects evaluated by Yang et al. [44]. The categories include a chair, couch, table, bench, faucet, firear, car, and plane. From the categories of chair, couch, table, and bench, the training dataset was derived using 2400 partial views of each category. The unseen views dataset contains 1000 partial views of the same categories included in the training dataset. The unseen objects dataset contains 1000 partial views of the categories not included in the training dataset (i.e. faucet, firear, car, and plane).

To assess the implemented 3D shape reconstructors, we used the Jaccard similarity index, as expressed in Equation (4):

\[ J(A, B) = \frac{A \cap B}{A \cup B}, \]

where, \( A \) and \( B \) indicate two volumes, as in Refs. [5,50–52]. A Jaccard value (\( J \)) of 0 represents no intersection between the ground-truth and the reconstructed object, and a \( J \) of 1 represents a perfect match between the ground-truth and reconstructed object.

The computational load of the reconstructors was evaluated based on the training time and network size. The training time was defined as the time needed by the reconstructors to process all the data in the training dataset one time (i.e. one epoch). The network size was defined as the total number of trainable parameters (i.e. weights, bias) in each reconstructor.

2.5 Implementation

All networks presented in this work were built using Keras and Tensorflow and trained with no pre-training or fine-tuning. Training and testing were done with an Intel Xeon CPU E5-2620 v3 2.40 GHz and two NVIDIA GeForce GTX 1080 GPUs, each with 8 GB of RAM.

To train our networks, we used weighted cross-entropy except for Varley’s, which used cross-entropy [22]. AdamOptimizer was used to optimize the weights with a learning rate of 0.00,003. The number of training steps (epochs) was 160 for Varley’s, 200 for 3D-CNN, 170 for SC 3D-CNN, and BNSC 3D-CNN, and 140 for U-Net BNSC 3D-CNN. The number of training steps for each network was chosen to prevent over-fitting after the network achieved the smallest loss value.

2.6 Conventional 3D reconstructors

Our proposed U-Net BNSC 3D-CNN is compared against two previous works. First, a 3D-CNN implemented in Varley et al. [22] (i.e. Varley’s), which used the reconstructed 3D objects for object grasping. Second, Yang et al. [44] introduce a 3D reconstructor based on a 3D encoder-decoder network combined with GAN (i.e. 3D-RegGAN++). 3D-RegGAN++ was trained and evaluated using the same categories of the ShapeNet database used in our work. Then for 3D-RegGAN++, the original trained weights [44] were utilized. To perform a fair comparison with our proposed U-Net BNSC 3D-CNN and the baseline networks, the output of 3D-RegGAN++ was down-sampled to a voxel grid of 40 × 40 × 40.

3 EXPERIMENTAL RESULTS

3.1 Results with the grasp database

Table 3 shows the performance of our proposed U-Net BNSC 3D-CNN in term of the Jaccard similarity index compared with that of Varley’s and the baseline networks. The Unseen Views of Seen Objects column indicates the average Jaccard index value for all objects in the unseen views dataset. The avocado, banana, and box columns are example objects from the unseen views dataset; each column value is the average Jaccard index value of 50 unseen views of each object. The Unseen Views of Unseen Objects column is the average Jaccard similarity index value for all objects in the unseen objects
dataset. The book, cell phone, and toy columns are example objects from the unseen objects dataset; each column value is the average Jaccard similarity index value of 50 unseen views of each object.

As shown in Table 3, 3D-CNN performed slightly better than Varley's, 1.3% better with the unseen views dataset and 1.2% better with the unseen objects dataset. SC 3D-CNN outperformed Varley's and 3D-CNN, 4% better than Varley's with the unseen views dataset and 2.2% with the unseen objects dataset. BNSC 3D-CNN outperformed Varley's by 3%–10%, and 3D-CNN by 2%–4% in all tests. Comparing the proposed U-Net BNSC 3D-CNN against Varley's, the Jaccard similarity index improved from 6% to 23% in all evaluations. U-Net BNSC 3D-CNN also outperformed the baselines networks by 4%–16% in all tests.

The reconstructed shapes of some example objects from the unseen views and unseen objects tests are presented in Figure 4. The proposed U-Net BNSC 3D-CNN clearly produced the best results.

Table 4 compares the computational load of the reconstructor. The Training Time column gives the time needed to process one epoch, 250 views of 76 objects. The Network Size column gives the total number of trainable parameters per reconstructor in millions [M] of parameters. Our proposed U-Net BNSC 3D-CNN has the minimum size. The Normalized Network Size column gives the normalized value of each reconstructor using Varley's as the reference. The 3D-CNN and SC 3D-CNN are 21.29% of the size of Varley's network. Thus, there is a reduction of 78.71% of Varley's network size. BNSC 3D-CNN is slightly smaller than SC 3D-CNN because of the BNSC block in BNSC 3D-CNN has fewer trainable parameters than the SC block in SC 3D-CNN. The largest reduction is shown by our proposed U-Net BNSC 3D-CNN, which is 98.69% smaller than Varley's network.

Figure 5 summarizes the unseen views test, unseen objects test, and network size. The network size (millions [M] of trainable parameters) of each 3D reconstructor is shown using bubble size. The X-axis represents the Jaccard similarity index value in the unseen objects test. The Y-axis is the Jaccard similarity index value in the unseen views test. As shown in Figure 5, the proposed U-Net BNSC 3D-CNN outperforms the other networks in both the unseen views and unseen objects testing. Figure 5 also shows a significant reduction in network size. Our U-Net BNSC 3D-CNN has 93.76% fewer trainable parameters than all baseline networks and 98.69% fewer trainable parameters than Varley's.

3.2 | Results with the ShapeNet database

Table 5 shows the performance of the proposed U-Net BNSC 3D-CNN in term of the Jaccard similarity index compared with the baseline networks and the previous works. The column of Unseen Views of Seen Objects is the average Jaccard index for all the object categories in the unseen views dataset. The columns of the chair and couch show the average results of the Jaccard index over 1000 objects from each category. As shown in Table 5, U-Net BNSC 3D-CNN outperforms by 22% the Varley's reconstructor, 6% 3D-RecGAN++, 12% 3D-CNN, 10% SC 3D-CNN, and 7% BNSC 3D-CNN. The column of the Unseen Views of Unseen Objects is the average Jaccard index with all the object categories included in the unseen objects dataset. The column of faucet and car are the average Jaccard index of 1000 objects from each category. Likewise, with the unseen views dataset, U-Net BNSC 3D-CNN outperforms all previous networks: by 28% over Varley, 11% over 3D-RecGAN++, 21% over 3D-CNN, and 20% over SC 3D-CNN and BNSC 3D-CNN.

Figure 6 provides the results of a qualitative evaluation of all 3D reconstructors with ShapeNet. The proposed U-Net BNSC 3D-CNN reconstructs the 3D volumes most similar to the ground truth compared with all other networks, especially for the unseen objects.

4 | DISCUSSION

Typically, the performance of neural networks (e.g., CNNs) improves when more layers are added. 3D-CNN slightly outperforms Varley's in terms of reconstruction accuracy (Jaccard similarity index) because 3D-CNN has more convolutional layers than Varley's. As shown in Figure 2a,b SC 3D-CNN has the same number of convolutional layers as 3D-CNN. The only difference in their architectures is the skip connections.
However, SC 3D-CNN had a Jaccard index of 80.57% in the unseen views test and 78.4% in the unseen objects test, while 3D-CNN had a Jaccards index of 77.9% and 77.2%, respectively. The increment in reconstruction accuracy reflects the effect of skip connections and the advantages of a 3D shape reconstructor based on SC blocks.
The 3D reconstruction performance of our U-Net BNSC 3D-CNN improved upon that of classical reconstructors based on 3D-CNN because of the BNSC blocks and the skip connections in our U-Net architecture. Our experiments show that BNSC 3D-CNN outperformed the 3D reconstruction performance of 3D-CNN. For instance, the reconstruction performance increased from 77.9% with 3D-CNN to 81.63% with BNSC 3D-CNN in the unseen views test and from 77.2% to 79.08% in the unseen objects test with the Grasp database; from 50.94% to 55.86% in the unseen views test and from 48.99% to 49.83% in the unseen object test with the ShapeNet dataset. Adding the skip connections between the BNSC blocks increased the reconstruction performance, because the skip connections create pathways to propagate feature information to the whole network. For instance, the skip connections in the contractive path of our U-Net BNSC 3D-CNN propagate features to the second and third BNSC blocks, and the skip connections between the contractive and expansive paths propagate the features to the fourth and fifth BNSC blocks. To preserve and use the features from the contractive path in the 3D reconstruction, the skip connections between the contractive and expansive paths use the concatenate operation. As a result of the feature information propagation, U-Net BNSC 3D-CNN is 6% more accurate than the BNSC 3D-CNN in shape reconstruction with the Grasp database and 17% more accurate with the ShapeNet dataset, as shown in Tables 3 and 5, respectively.

It is important to mention that the training time of U-Net BNSC 3D-CNN depends on the operation used with the skip connections in the contractive path. In our work, we used addition operations because addition does not increase the number of features at the input of the next BNSC blocks. Using the concatenation operation instead of addition imposes an increment in the network size and computational cost because of the increase in the feature maps at the input of the second, third, fourth, and fifth BNSC blocks. In consequence, the training time increases, as in the case of the training time of the SC block compared with the BNSC block.

The 3D reconstruction algorithms based on 3D-convolutional layers produce a large number of trainable parameters because of the 3D-kernel. To reduce the network size, we use the BNSC block, which adds bottle-neck layers to the SC block. The BNSC block has the same advantages as the SC block (i.e. training is improved by skip connections), but a reconstructor based on BNSC blocks is faster and has fewer trainable parameters than a reconstructor based on SC blocks because of the bottle-neck layers, as shown in Table 4.

Our U-Net BNSC 3D-CNN is only 20 s faster than Varley’s in training time, despite the large difference in network size between the two reconstructors. The training time does

| Average Jaccard similarity (%) | Unseen views of seen objects | Chair | Couch | Unseen views of unseen objects | Faucet | Car |
|---------------------------------|-------------------------------|-------|-------|-------------------------------|--------|-----|
| Varley et al. [22]              | 50.05 ± 8.34                  | 4.84 ± 10.15 | 69.67 ± 11.67 | 41.46 ± 5.79 | 34.61 ± 7.67 | 56.39 ± 10.14 |
| 3D-RecGAN++                     | 66.43 ± 7.91                  | 65.04 ± 11.64 | 76.30 ± 12.25 | 58.35 ± 7.97 | 57.78 ± 12.09 | 68.68 ± 8.85 |
| 3D-CNN                          | 50.94 ± 7.67                  | 47.71 ± 10.25 | 70.67 ± 8.13 | 48.99 ± 6.41 | 44.27 ± 10.39 | 60.27 ± 9.43 |
| SC 3D-CNN                       | 52.38 ± 7.88                  | 49.11 ± 10.71 | 72.63 ± 8.51 | 49.50 ± 6.71 | 44.40 ± 10.87 | 62.40 ± 9.41 |
| BNSC 3D-CNN                     | 55.86 ± 7.68                  | 51.81 ± 10.38 | 74.48 ± 7.95 | 49.83 ± 6.04 | 44.71 ± 10.43 | 69.75 ± 7.64 |
| U-Net BNSC 3D-CNN               | 72.17 ± 7.26                  | 70.59 ± 9.32 | 80.18 ± 6.9 | 69.97 ± 6.08 | 67.47 ± 9.54 | 72.91 ± 7.56 |

Abbreviation: 3D-CNN, 3D-convolutional neural network; BNSC, BNSC, bottle-neck skipped connection; SC, skipped connection.
not reflect the network sizes because of the parallel processing with GPUs. However, the 20 s of difference in training time is magnified when the training is extended for several epochs or when the training dataset includes more objects or views. The main advantage of our proposed reconstructor, comparing to Varley's network, is its improvement in the Jaccard index without a penalty in the computational load. The network size of 3D-RecGANS++ is 153.2 million trainable parameters, which is slightly more parameters than Varley's. Thus, our U-Net BNSC 3D-CNN widely reduces the network size of 3D-RecGANS++.

Our U-Net BNSC 3D-CNN can be implemented on a graphic card with less memory consumption than previous works due to its network sizes. The difference in network size between our proposed U-Net BNSC 3D-CNN and the previous works is caused by the dense layers in Varley's and 3D-RecGANS++. Replacing the last dense layers with convolutional layers decreased the number of trainable parameters, as shown in Table 4 for the baseline networks. Therefore, our U-Net BNSC 3D-CNN has no dense layers.

The 3D reconstruction performance of our U-Net BNSC 3D-CNN might be improved by making some changes in its architecture. For instance, adding more BNSC blocks in the contractive and expansive paths would extract more features for 3D reconstruction. Also, increasing the number of feature maps extracted in each convolutional layer might provide more information for 3D reconstruction because the input would be decomposed into higher dimensional features. More investigations are also need for 3D reconstruction regarding the kernel size. In our experiments, we observed that a larger kernel size (e.g., 6 × 6 × 6 or 5 × 5 × 5) improved 3D reconstruction of the general shape of the object, whereas a
small kernel size (e.g., $3 \times 3 \times 3$) reconstructed the details of the object better. However, an increment in training time should be considered because of the number of layers (i.e., more BNSC blocks), feature maps, or kernel size will increase the computational load of the reconstructor.

The reconstruction performance with untrained objects depends on the partial view. For instance, whether the partial view provides information about the object details, U-Net BNSC 3D-CNN reconstructs those details in the object with higher reconstruction accuracy, as shown by the arms and legs of a toy in Figure 4 or the tires of a car in Figure 6. However, the reconstruction accuracy (i.e., Jaccard similarity index) for untrained objects would be affected if the partial view did not show any information about the object details.

5 | CONCLUSION

In terms of reconstruction accuracy (i.e., Jaccard similarity index), the proposed U-Net BNSC 3D-CNN outperforms the baseline and previous reconstructor tested in this work by 6%–11% in the unseen views test with the Grasp database and by 6%–22% with the ShapeNet database. In the unseen objects test, our proposed reconstructor improves by 6%–9% with the Grasp database and by 10%–28% with the ShapeNet database. This improvement is because the skip connections propagate the error and the features information to the whole network. Our U-Net BNSC 3D-CNN reduces the network size by 98.69% compared with Varley’s and by 93.76% compared to the error and the features information to the whole network.

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (NRF-2019R1A2C1003713). Edwin Valarezo Añazco gratefully acknowledges Escuela Superior Politécnica del Litoral (ESPOL) and its excellence program Walter Valdano Raffo.

ACKNOWLEDGEMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (NRF-2019R1A2C1003713). Edwin Valarezo Añazco gratefully acknowledges Escuela Superior Politécnica del Litoral (ESPOL) and its excellence program Walter Valdano Raffo.

REFERENCES

1. Steinbrucker, F., et al.: Large-scale multi-resolution surface reconstruction from RGB-D sequences. IEEE International Conference on Computer Vision, Sydney, Australia, pp. 3264–3271 (2013)

2. Newcombe, R.A., et al.: KinectFusion: real-time dense surface mapping and tracking. 10th IEEE International Symposium on Mixed and Augmented Reality, Basel, Switzerland, pp. 127–136 (2011)

3. Dai, A., et al.: ScanComplete: large-scale scene completion and semantic segmentation for 3D scans. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, USA (2018)

4. Riegler, G., Bischof, G., Geiger, A.: OctNetFusion: learning depth fusion from data. International Conference on 3D Vision (3DV), Qingdao, China, pp. 57–66 (2017)

5. Stutz, D., Geiger, A.: Learning 3D shape completion from laser scan data with weak supervision. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, USA (2018)

6. Ondruska, P., Kohli, P., Izadi, S.: MobileFusion: real-time volumetric surface reconstruction and dense tracking on mobile phones. IEEE Trans. Visual Comput. Graph. 21(11), 1251–1258 (2015)

7. Xu, S., et al.: PM-PM: PatchMatch with Potts model for object segmentation and stereo. IEEE Trans. Image Proc. 24(7), 2182–2196 (2015)

8. Nieltner, M., et al.: Real-time 3D reconstruction at scale using voxel hashing. ACM Trans. Graph. 32(6), 1–11 (2013)

9. Zavala, J.P., et al.: Three-dimensional reconstruction of objects obtained by two orthogonal cameras. IEEE Latin Am. Trans. 13(9), 3162–3168 (2015)

10. Sipiran, I., Gregor, R., Schreck, T.: Approximate symmetry detection in partial 3D meshes. Comput. Graph. Forum. 33(7), 131–140 (2014)

11. Bohg, J., et al.: Mind the gap - robotic grasping under incomplete observation. IEEE International Conference on Robotics and Automation, Shanghai, China, pp. 686–693 (2011)

12. Huam, A., et al.: Exploiting symmetries and extrusions for grasping household objects. IEEE International Conference on Robotics and Automation (ICRA), Seattle, USA, pp. 3702–3708 (2015)

13. Thrun, S., Wegbreit, B: Shape from symmetry. Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1, Beijing, China, pp. 1824–1831 (2005)

14. Miltra, N.J., Guibas, L.J., Pauly, M: Partial and approximate symmetry detection for 3D geometry. ACM Trans. Graph. 25(3), 560–568 (2006)

15. Pauly, M., et al.: Discovering structural regularity in 3D geometry. ACM Trans. Graph. 27(3), 43 (2008)

16. Li, Y., et al.: Database-assisted object retrieval for real-time 3D reconstruction. Comput. Graph. Forum. 34(2), 435–446 (2015)

17. Kim, Y.M., et al.: Acquiring 3D indoor environments with variability and repetition. ACM Trans. Graph. 31(6), 208 (2012)

18. Nan, L., Xie, K., Sharf, A.: A search-classify approach for cluttered indoor scene understanding. ACM Trans Graph. 31, 6 (2012)

19. Hinterstoisser, S., et al.: Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. International Conference on Computer Vision, Barcelona, Spain, pp. 858–865 (2011)

20. Papazov, C., Barschka, D.: An efficient RANSAC for 3D object recognition in noisy and occluded scenes. ACCV 2010: Computer Vision – ACCV 2010, pp. 135–148 (2011)

21. Smith, E.J., Meger, D.: Improved adversarial systems for 3D object generation and reconstruction. Conference on Robot Learning, California, USA, pp. 1–10 (2011)

22. Varley, J., et al.: Shape completion enabled robotic grasping. IEEE International Conference on Intelligent Robots and Systems, Vols. 2017-Sept, Vancouver, Canada, pp. 2442–2447 (2017)

23. Han, X., et al.: High-resolution shape completion using deep neural networks for global structure and local geometry inference. IEEE International Conference on Computer Vision (ICCV), Venice, Italia, pp. 85–93 (2017)

24. Han, Y., Ye, J.C.: Framing U-Net via deep convolutional framelets: application to sparse-view CT. IEEE Trans. Med. Imag. 2018, 37, 1418–1429

25. Ji, S., et al.: 3D convolutional neural networks for crop classification with multi-temporal remote sensing images. Rem. Sens. 10, 75 (2018)

26. Zhang, K., et al.: Residual networks of residual networks: multilevel residual networks. IEEE. Trans. Circ. Syst. Video Technol. 28, 1303–1314 (2018)
27. Lee, Y., et al.: Gradient-based learning applied to document recognition. Proc. IEEE. 86(11), 2278–2324 (1998)
28. He, K., et al.: Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, USA, pp. 770–778 (2016)
29. Zagoruyko, S., Komodakis, N.: Wide residual networks. Proceedings of the British Machine Vision Conference (BMVC), York, UK, pp. 87.1–87.12 (2016)
30. Wei, Y., et al.: Boosting the accuracy of multispectral image pansharpening by learning a deep residual network. IEEE Geosci. Rem. Sens. Lett. 14, 1795–1799 (2017)
31. Cazenave, T.: Residual networks for computer go. IEEE Trans. Comput. Intell. AI Games. 10(1), 107–110 (2018)
32. Huang, G., et al.: Densely connected convolutional networks. Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, USA, pp. 2261–2269 (2017)
33. Larsson, G., Maire, M., Shakhnarovich, G.: FractalNet: ultra-deep neural networks without residuals. International Conference on Learning Representations, Toulon, France, pp. 1–11 (2017)
34. Zhu, L., et al.: Sparsely aggregated convolutional networks. The European Conference on Computer Vision (ECCV), Munich, Germany, pp. 1–16 (2018)
35. Heravi, E.J., Aghdam, H.H., Puig, D.: An optimized convolutional neural network with bottleneck and spatial pyramid pooling layers for classification. Pattern Recogn. Lett. 105, 50–58 (2018)
36. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. International Conference on Learning Representations (ICLR), Banff, Canada, pp. 1–14 (2014)
37. Stanford CS class CS231n. http://cs231n.github.io/convolutional-networks/. Accessed September 2018
38. He, Z., et al.: BD-Net: A multiplication-less DNN with hirarchical depthwise separable convolution. IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Hong Kong, China, pp. 130–135 (2018)
39. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich Germany, pp. 234–241 (2015)
40. Jegou, S., et al.: The one hundred layers Tiramisu: fully convolutional DenseNets for semantic segmentation. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Honolulu, 2017, pp. 1175–1183 (2017)
41. Zhang, Z., Liu, Q., Wang, Y.: Road extraction by deep residual U-Net. IEEE Geosci. Rem. Sens. Lett. 15, 749–753 (2018)
42. Fan, J., et al.: Multichannel fully convolutional network for coronary artery segmentation in X-ray angiograms. IEEE Access. 6, 44635–44643 (2018)
43. Dai, A., Qi, C.R., Niessner, M.: Shape completion using 3D-encoder-predictor CNNs and shape synthesis. Proceedings - 36th IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, USA, pp. 6545–6554 (2017)
44. Yang, B., et al.: Dense 3D object reconstruction from a single depth view. IEEE Trans. Pattern Anal. Mach. Intell. 41(12), 2820–2834 (2019)
45. Kappler, D., Boehg, J., Schaal, S.: Leveraging big data for grasp planning. Proceedings - IEEE International Conference on Robotics and Automation, Seattle, USA, pp. 4304–4311 (2015)
46. ShapeNet: an information-rich 3D model repository. https://www.shapenet.org/. Accessed September 2018
47. Binvox, a 3D mesh voxelizer. https://www.patrickmin.com/binvox/. Accessed October 2018
48. Nooruddin, F.S., Turk, G.: Simplification and repair of polygonal models using volumetric techniques. IEEE Trans. Visual Comput. Graph. 9(2), 191–205 (2003)
49. Koenig, N., Howard, A.: Design and use paradigms for Gazebo, an open-source multi-robot simulator. International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan, pp. 2149-2154 (2004)
50. Yang, B., et al.: 3D object reconstruction from a single depth view with adversarial learning. Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW, Venice, Italy, pp. 679–688 (2017)
51. Hane, C., Tulsiani, S., Malik, J.: Hierarchical surface prediction for 3D object reconstruction. Proceedings - 2017 International Conference on 3D Vision, Qingdao, China, pp. 412–420 (2017)
52. Large-scale 3D shape reconstruction and segmentation from ShapeNet Core55. https://shapenet.cs.stanford.edu/tecv17/. Accessed September 2018
53. Chan, C.Y.: Advancements, prospects, and impacts of automated driving systems. Int. J. Transport. Sci. Technol. 6(3), 208–216 (2017)
54. Choi, C., et al.: Learning object grasping for soft robot hands. IEEE Robot. Autom. Lett. 3(3), 2370–2377 (2018)
55. Dai, A., Niellner, M., Theobalt, C.: BundleFusion: real-time globally consistent 3D reconstruction using on-the-fly surface reintegration. ACM Trans. Graph. 36(3), 24 (2017)

How to cite this article: Valarezo Añazco E, Rivera Lopez P, Kim TS. Three-dimensional shape reconstruction of objects from a single depth view using deep U-Net convolutional neural network with bottleneck skip connections. *IET Comput. Vis.* 2021;15:24–35. https://doi.org/10.1049/cvi2.12014