sensors

*Article*

# Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from *ShapeNetCore* Dataset

**Audrius Kulikajevas [1], Rytis Maskeliūnas [2], Robertas Damaševičius [3,*] and Sanjay Misra [4,5]**

[1] Department of Multimedia Engineering, Kaunas University of Technology, 51368 Kaunas, Lithuania; audrius.kulikajevas@ktu.edu
[2] Centre of Real Time Computer Systems, Kaunas University of Technology, 51368 Kaunas, Lithuania; rytis.maskeliunas@ktu.lt
[3] Department of Software Engineering, Kaunas University of Technology, 51368 Kaunas, Lithuania
[4] Department of Electrical and Information Engineering, Covenant University, Ota 1023, Nigeria
[5] Department of Computer Engineering, Atilim University, Ankara 06830, Turkey; sanjay.misra@atilim.edu.tr
* Correspondence: robertas.damasevicius@ktu.lt

Received: 28 February 2019; Accepted: 28 March 2019; Published: 31 March 2019

**Abstract:** Depth-based reconstruction of three-dimensional (3D) shape of objects is one of core problems in computer vision with a lot of commercial applications. However, the 3D scanning for point cloud-based video streaming is expensive and is generally unattainable to an average user due to required setup of multiple depth sensors. We propose a novel hybrid modular artificial neural network (ANN) architecture, which can reconstruct smooth polygonal meshes from a single depth frame, using a priori knowledge. The architecture of neural network consists of separate nodes for recognition of object type and reconstruction thus allowing for easy retraining and extension for new object types. We performed recognition of nine real-world objects using the neural network trained on the *ShapeNetCore* model dataset. The results evaluated quantitatively using the Intersection-over-Union (IoU), Completeness, Correctness and Quality metrics, and qualitative evaluation by visual inspection demonstrate the robustness of the proposed architecture with respect to different viewing angles and illumination conditions.

**Keywords:** 3D depth shape recognition; 3D depth scanning; RGB-D sensors; hybrid neural networks

## 1. Introduction

Reconstruction of three-dimensional (3D) depth-based geometry is one of the core problems in computer vision with commercial applications. These applications range from importing 3D scanned assets into video games and virtual reality (VR) applications [1], gesture recognition [2], indoor mapping [3], recreating environments in movies, recreating evidence and crime scenes in digital forensics [4], designing of dental implants and prosthetics [5], performing Building Information Modelling (BIM) in construction industry [6], environmental perception for industrial/service robots [7], and preserving cultural heritage in museums [8]. However, to this day, the systems that provide 3D scanning capabilities are expensive and are generally unattainable for an average user. Yet, the desirability to have the 3D scene reconstruction is so crucial that the researchers propose new methods that aim to transform RGB images into depth cloud, without the need for additional hardware [9], so that 3D scanning systems would be more affordable and accessible.

We cannot expect the user to either have expensive laser depth scanners which are capable of great accuracy of scanned objects, or an array of sensors which would be capable to pick up all regions occluded by other objects or even self-occlusion and we cannot expect from a general user to be bothered with taking time to precisely scan the entirety of the object so that it would be reconstructed incrementally [10,11] based on delta frames and camera localization. For example, many classical scene reconstruction algorithms rely on simultaneous localization and mapping (SLAM) [12], in order to scan the entirety of 3D objects in the environment, which is then converted either into point-cloud, voxel-cloud volume or triangulated into a mesh. Unfortunately, incremental algorithms tend suffer from one major flaw: changes in scene can create corruptions in the mesh [13]. This makes the application of such approaches unstable in real-world scenes, where objects rather than the view perspective move in space. Other methods such as space carving [14] bypass some of these issues by performing subtractive reconstruction from multiple perspectives with an addition of mask. However, this method assumes that we can accurately extract the mask of an object, which can prove to be very difficult in some aspects due to adverse illumination conditions.

Another approach employed by some of the most successful reconstruction algorithms is to use a priori knowledge of the objects to be reconstructed [15–17]. While these methods have shown great recall capabilities and are less prone to errors do to a priori knowledge, they still depend on illumination conditions as the RGB cameras only capture the visible light spectrum, which may cause distortions in case of dim light and would be impossible to use in dark environments.

With the increasing amount of available low-cost consumer-grade depth sensors such as *Microsoft Kinect* [18], *Intel RealSense* [19], and depth cameras becoming a standard feature in some flagship mobile phones, we are moving towards an era where RGB-Depth (RGB-D) sensors are as common as regular RGB cameras. Object detection and segmentation with RGB-D sensors has been widely used in recent years, such as Canonical Correlation Analysis (CCA)-based multi-view Convolutional Neural Networks (CNN) [20], using regular point clouds in addition to multi-views for point cloud recognition [21], fusing CNNs with simultaneous localization and mapping in order to perform object segmentation [22], employing multi-modal deep neural networks and Dempster Shafer evidence theory to achieve the task of object recognition [23], or adopting multifoveated point clouds [24].

Applying data acquired from RGB-D sensors is a logical evolution of the reconstruction algorithms as the non-stereoscopic (two RGB lenses side-by-side simulating binocular vision) depth sensors are less dependant on ambient light conditions and are capable of capturing even in pitch black environments using infrared projectors, albeit are still prone to speckles due to the properties of object surface [25,26]. There have already been attempts to achieve surface prediction by using depth [27] and silhouettes [28] performed experiments mostly consist of synthetic data. On the other hand, these cameras have limitations too. While RGB frames are generally difficult to segment due to different textures and colors [29] and it is generally easier to segment an object from a noisy background using RGB-D sensor from a sufficient distance, objects in close proximity to each other are difficult to be segmented due to their depth values being very similar. Furthermore, commercially used depth sensors tend to suffer from distortions when projecting infrared (IR) laser [30].

We present a hybrid neural network architecture, capable of reconstructing smooth polygonal meshes from a single depth frame, using a priori knowledge and still being capable of running on low-end RGB-D sensor devices in intractable frame rates. The aim is not to scan the 3D geometry, but rather a stream of depth data, which can later be used to recreate 3D holographic objects. The structure of this paper is organized as follows: Section 2 describes the proposed modular neural network, reconstruction algorithm for 3D object recognition, and network training; Section 3 presents experimental results of the proposed network architecture; and finally, Section 4 discusses the results and concludes the article.

## 2. Materials and Methods

### 2.1. Architecture of Hybrid Neural Network

The proposed hybrid network consists of a single classifier network that branches off into n-reconstruction networks (in comparison to standard methods of having a single neural network performing both of these tasks at once). For our hybrid ANN architecture we adopted the hierarchical approach. The ANN consists of a single predictor node and multiple reconstructor nodes, where each reconstructor node is dedicated to recognizing either a specific object or a group of similar objects. This allows more easily training additional types of objects without having to re-train for reconstruction or facing the risk of loosing existing gradients by training on additional models [31]. This adds some modularity to the system while also giving the benefit of reducing the training time due to low iteration count required as the Adam optimizer [32] manages to converge the model in very few iterations generally under 50, depending under model complexity.

For the discriminator ANN in the hybrid network (Figure 1), we use a simple one hot CNN, which takes an input of 320 × 240 depth frame and runs it through a convolution layer. In convolution layer, we create 32 samples by using a 3 × 3 kernel with max-pooling function, downsampling the original image by a factor of two. After convolution layer we add random noise to the output by using a dropout layer with a chance of $P(x) = 0.2$, which allows for better generalization. Finally, we flatten our output into 1-dimensional tensor and run it through 256 neuron density layer with the output being returned as one-hot encoded array. For all of our layers we used Rectified Linear Units (ReLUs) [33] as they have been shown to give great results in conjunction with CNNs [34]. Finally, we compute the loss using softmax cross-entropy (1) in order to discriminate between different types of object classes, where $y_0$ is ground truth value, $p$ is predicted value. Once we have the classifier result, we can select the appropriate neural network best fitting for the reconstruction of the observed object frame. Thus the hybridization of these two neural networks allows us to have desired modularity in our method.

$$H(y_0, p) = -\sum y_0 log(\frac{e^p}{\sum e^p}) \tag{1}$$

A single node that is used for reconstructing the voxel volume is shown in Figure 2. The reconstruction ANN adopts the convolutional encoder layers from *PointOutNet* [15] architecture as it has shown to have good encoding capabilities. However, we modified the decoding components. First, we added a dropout layer with $P(x) = 0.4$ for increased generalization, following a 512 neuron density connected layer. The final layer is the 32 × 32 × 32 voxel space layer. While all other layers use ReLU as activation function, the output layer uses the sigmoid function to clamp the output ranges $B \in [0; 1]$. For a loss function, mean square error (Equation (2)) is used, where $p$ is prediction, $y_0$ is ground truth, $n$ is the number of elements in batch. Please note that using the mean value instead of absolute loss creates a better network topology, as the latter may fall into a local minima and constantly generate the same output.

$$H(y_0, p) = \sum \frac{(p - y_0)^2}{n} \tag{2}$$

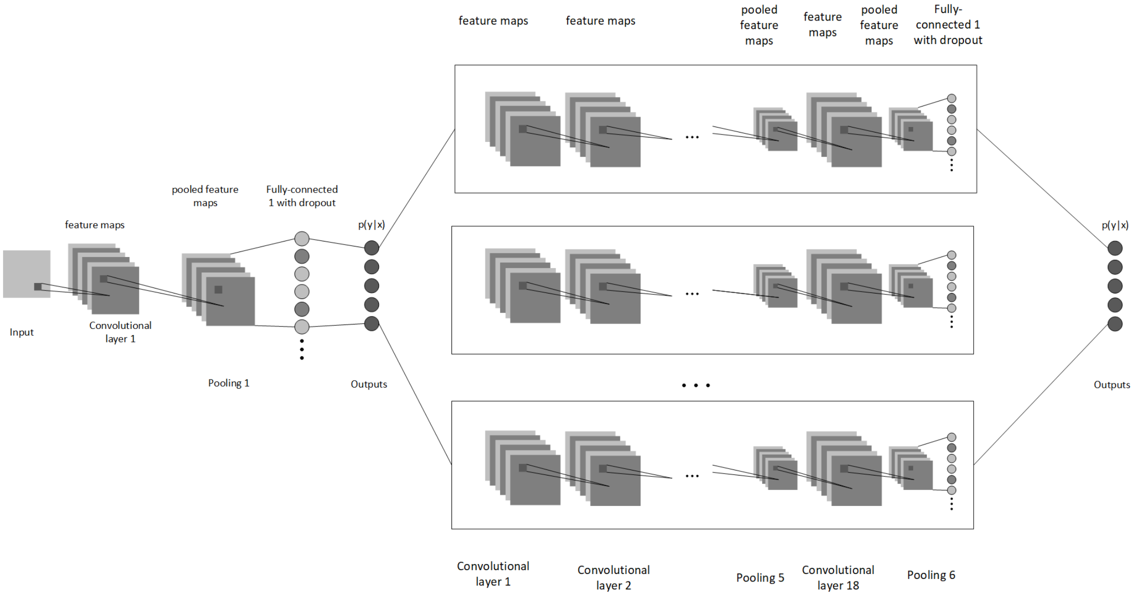


**Figure 1.** Architecture of hybrid neural network consisting of discriminator and reconstructor. The node accepts depth data as an input, applies a 3 × 3 max-pool convolution layer with a stride of 2 and 32 samples. Following the convolution layer a dropout layer of P(x) = 0.2 is applied to avoid overfitting. The final discriminator layer is the fully connected one-hot layer. The result of the classifier helps us pick the best suited reconstruction neural network for the observed object. Afterwards the same input is sent into appropriate reconstruction neural network which then performs deep convolutions and sends it to fully connected layer which predicts the voxel cloud of the object.

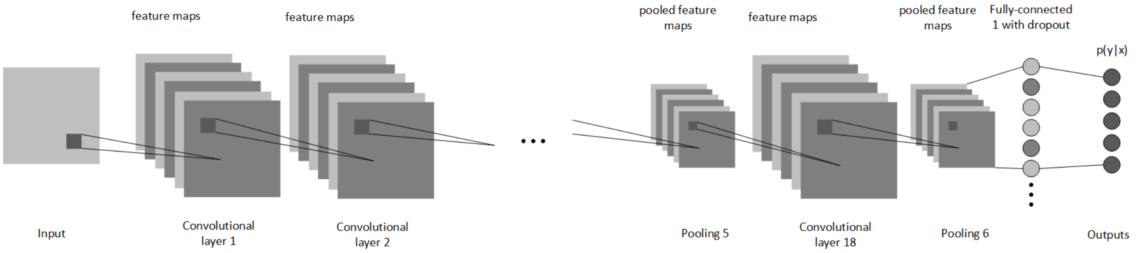


**Figure 2.** A single branch of ANN used for reconstruction of voxel space.

## 2.2. Reconstruction Algorithm

The proposed 3D reconstruction algorithm (Figure 3) consists of three main steps: prediction, reconstruction and post-processing. In the prediction step we use depth sensor data as our input in order to select the reconstruction network, if its was pre-trained. Once the reconstruction ANN is decided the input is then fed to the network to perform voxel cloud prediction.

Finally, the algorithm performs voxel cloud post-processing by turning the reconstruction network output into a polygonal mesh and applying an additional surface smoothing to eliminate noisiness. To use native rendering as provided by graphics pipeline we need to turn the voxel volume cloud into a triangle mesh, which is performed in two steps. First, we convert voxels into triangles via marching cubes [35] using an algorithm presented in Figure 4. We iterate over all voxels in the voxel cloud and create an adjacency cube that is used to determine the shape the voxel should take as follows: we calculate the edges based on adjacency cube. If the adjacency cube ege flag returns 0, we assume that the voxel is inside the mesh and skip it, otherwise we select the edge flag from the marching cube hash table and find the point of intersection of the surface with each edge, if intersections exist we compute the edge vertice positions. Finally, we construct the triangles based on triangle connection table and push them to mesh.

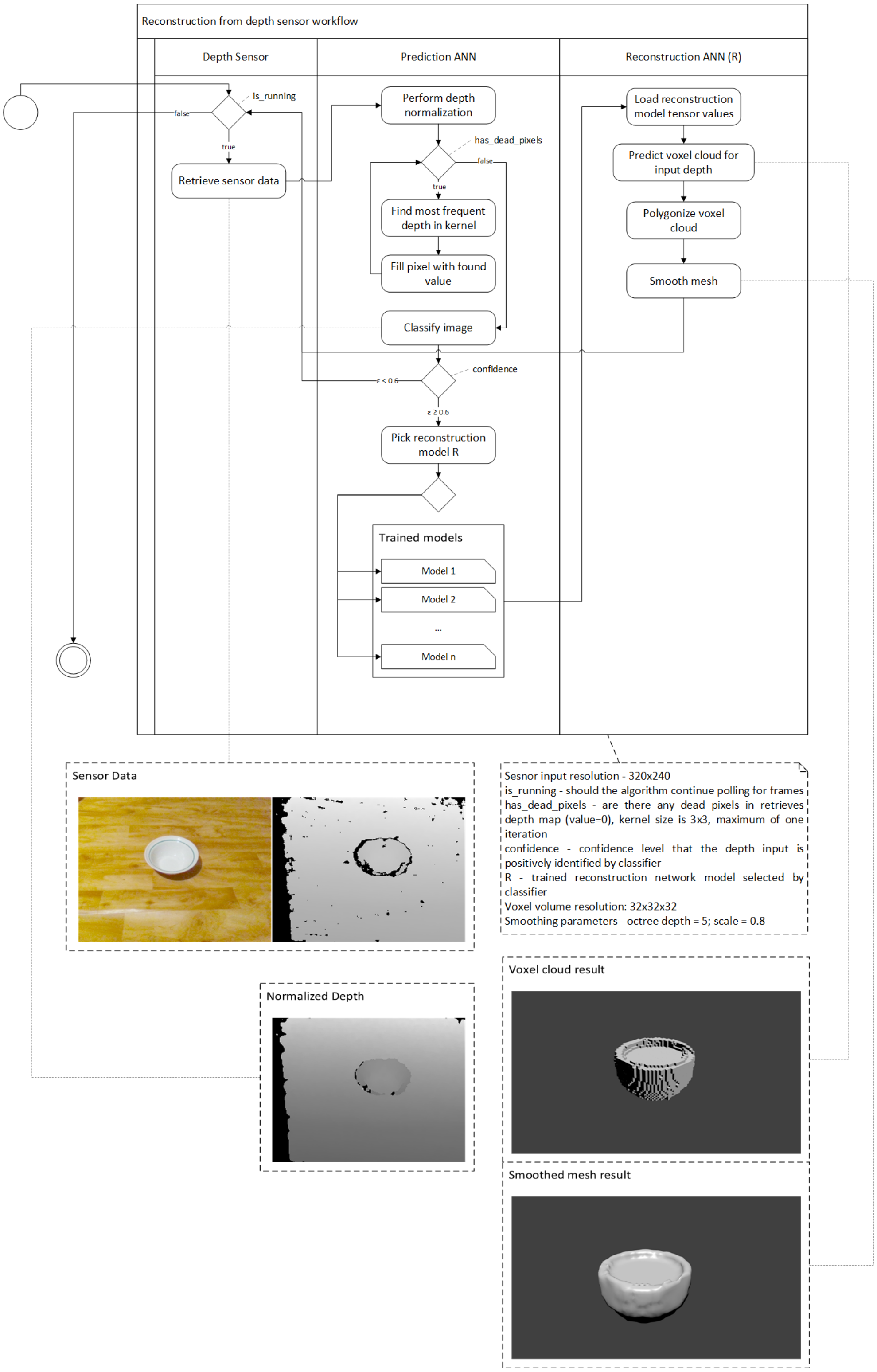
However, this approach produces ambiguities which causes holes to appear in the mesh. Due to the fundamental way non-stereoscopic depth sensors work they are prone to noise. The noise in depth frame acts as *holes* that have a depth value of zero. When using real sensor data, we add additional postprocessing in order to denoise the image as much as possible. We do this by using a kernel method that finds the most frequent value in the kernel and using that as new pixel value (see Equation (3)), where $D$ is depth field matrix, $x$ and $y$ is the coordinates of the pixel on the image.

$$D(x,y) = \begin{cases} D(x,y), & \text{if } x \neq 0 \\ \hat{D}(i,j) \text{ for } i = x-2,\ldots,x+2 \text{ and } j = y-2,\ldots,y+2, & \text{otherwise} \end{cases} \tag{3}$$

Furthermore, the generated mesh is somewhat blocky. To mitigate this issue, we further apply smoothing by applying dual contouring [36] on the generated mesh.

### 2.3. Network Training

For neural network training and validation, we use the *ShapeNetCore* dataset and *Blender* in order to generate appropriate depth images and ground truths of voxel cloud. To train the neural network, first, we find all available objects that we are working with and separate them into different objects. Once that step is complete, we pick the first category and load a single *OBJ* object from that category. After the object is loaded, we use *Blender* to render depth fields for each object from different angles. We have selected values the perspectives in such a way that the object would be rendered from all 45° and 90° angles at distances of 1 and 1.5 units, except the bottom, giving us a total of 48 perspectives. We save the perspectives as *OpenEXR* format as unlike standard image formats *OpenEXR* is linear, allowing us to retain all depth range values, which standard non-lossy image formats would loose due to the limitation of 32 bits per pixel [37]. Furthermore, as our network is trained only on depth frames, we do not encounter any problems related emulating lighting as opposed to when choosing Lambert, Phong, PBR, etc. shading models for realistic lightning in RGB enviroments. After we have rendered the given object mesh into depth fields, we perform geometry voxelization as suggested in [38]. This is done by partitioning the geometry boundaries into equal sized cells. The size of the cell is chosen based on the largest object axis. Once the space is partitioned, we iterate over all cells and calculate if the cell should be filled or not, the state is determined using ray-triangle intersection [39]. After the model is processed, we continue with all the models in the class until none are left and move on to next class, we continue this until no classes or objects are left. When data preparation step is complete we perform our training. This is done in multiple stages. First stage consists of training classifier network to recognize the object class so that an appropriate network can be chosen afterwards. Once the classification model has converged or we reach 500 iterations we train each class individually on a new neural network, while saving tensor values for each network. An UML activity diagram in Figure 5 demonstrates this process. Due to automatization of very large quantities of models, automatic depth generation may fail due to irregular object sizes, mainly very thin objects like knives. Therefore, we add an additional check to filter out invalid object inputs such as empty frames and very few clustered pixels that could potentially spoil the training gradients.



Sesnor input resolution - 320x240
is_running - should the algorithm continue polling for frames
has_dead_pixels - are there any dead pixels in retrieves depth map (value=0), kernel size is 3x3, maximum of one iteration
confidence - confidence level that the depth input is positively identified by classifier
R - trained reconstruction network model selected by classifier
Voxel volume resolution: 32x32x32
Smoothing parameters - octree depth = 5; scale = 0.8

**Figure 3.** Depth sensor data is captured and sent to classifier ANN. If classifier network recognizes the object, the sensor data is sent to reconstruction ANN, otherwise the frame is dropped. Reconstruction ANN generates the voxel cloud. Voxel cloud is turned into polygonal mesh, and mesh smoothing is applied.

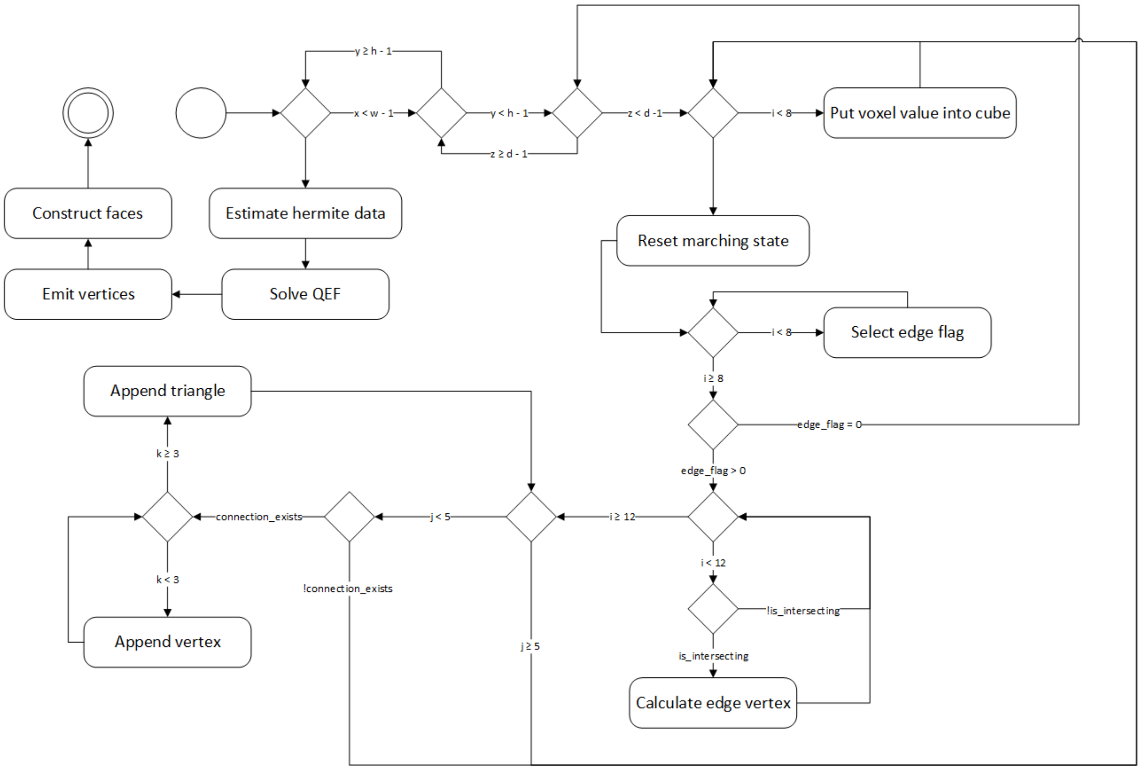
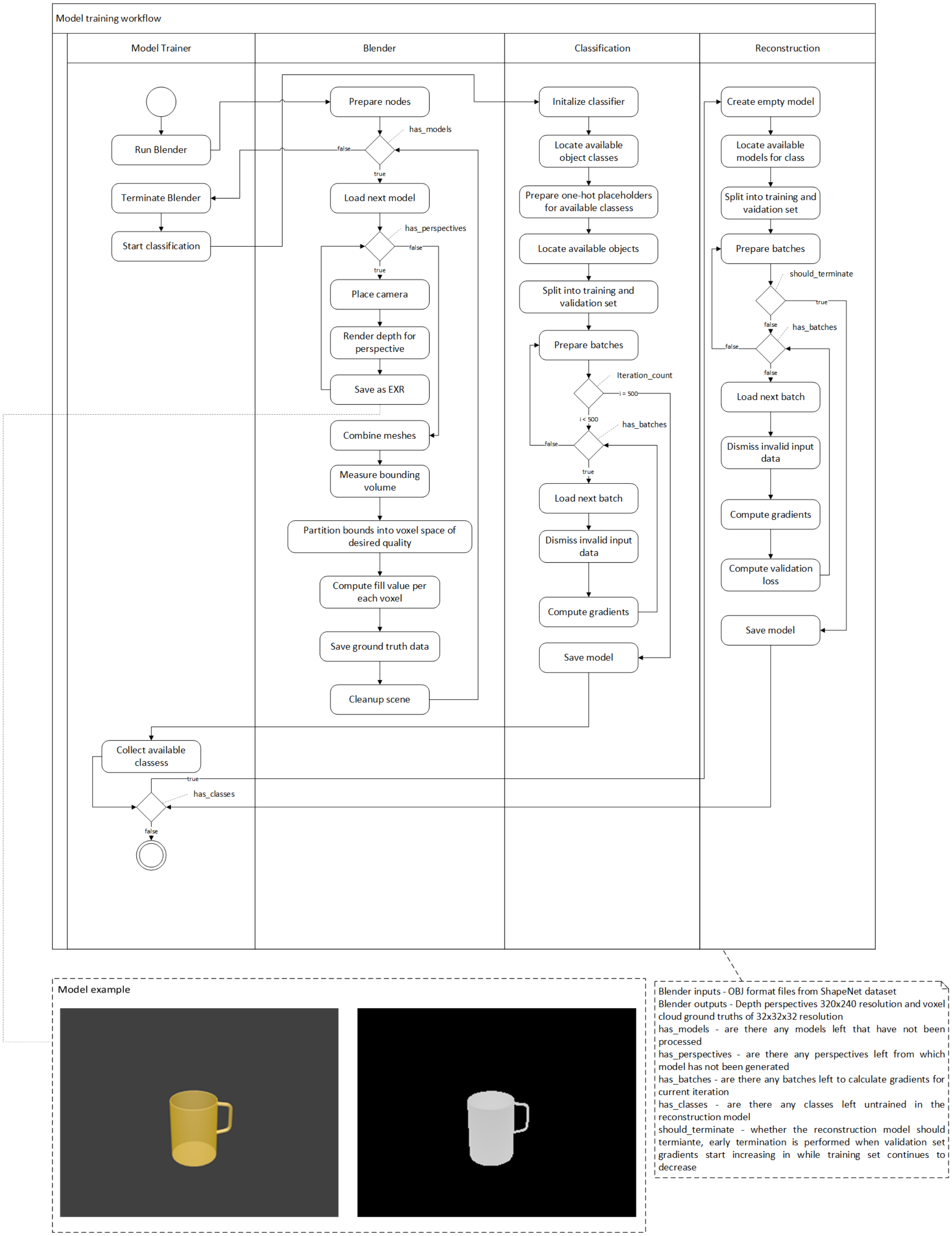


**Figure 4.** Mesh conversion into polygons via marching cubes.

### 2.4. Dataset

3D recognition depends on a priori information about desired objects. Therefore, it is a requirement to have a good labeled element dataset. However, for 3D object recognition there are a few such datasets that are more limited. Our main sources are *ShapeNetCore*, a subset of *ShapeNet* [40] dataset that has clean 3D models and manually verified categories, and real-world data captured by the *Intel RealSense ZR300* (Intel Corporation, Santa Clara, CA, USA) device. An example of 3D models provided in *ShapeNetCore* dataset is shown in Figure 6, and an example of real depth sensor data acquired by *Intel RealSense ZR300* is given in Figure 7. While we use *ShapeNetCore* as a source of training data, we also use real depth sensor data for visual validation and testing in real-life applications. This is mainly due to not having ground-truth data for real objects, which unlike virtual model datasets would allow us to extract all the necessary features.

We also have explored different subsets of the *ShapeNet* database. However, these models have proved to be problematic due to their shapes not being properly normalized and aligned as opposed to *ShapeNetCore*, which is undesired effects for training. Therefore, the only model we used from *ShapeNetSem* for our experiments was *Book*, which had the worst recall rates of all models due the problems specified previously.



Blender inputs - OBJ format files from ShapeNet dataset
Blender outputs - Depth perspectives 320x240 resolution and voxel cloud ground truths of 32x32x32 resolution
has_models - are there any models left that have not been processed
has_perspectives - are there any perspectives left from which model has not been generated
has_batches - are there any batches left to calculate gradients for current iteration
has_classes - are there any classes left untrained in the reconstruction model
should_terminate - whether the reconstruction model should termiante, early termination is performed when validation set gradients start increasing in while training set continues to decrease

**Figure 5.** Overview of workflow for *ShapeNetCore* data set preparation and model training.

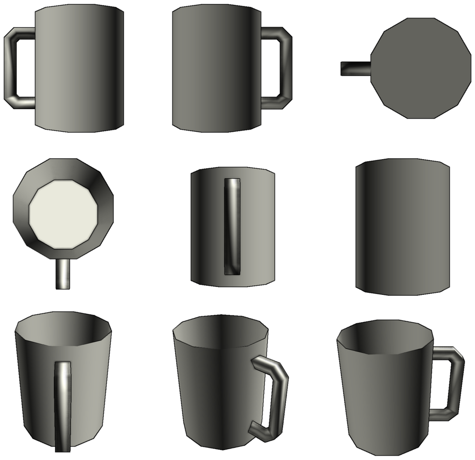


**Figure 6.** An example of a model from *ShapeNetCore* dataset.

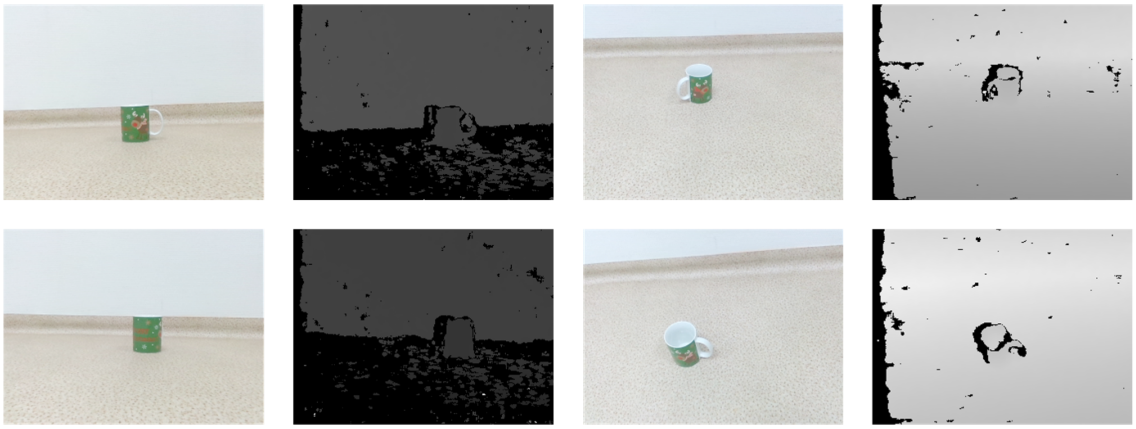


**Figure 7.** An example of real depth sensor data set captured by *Intel RealSense ZR300* captured from different vantage points.

### 2.5. Evaluation

The goal of reconstruction is usually to achieve a difference between the reconstruction and the ground truth as small as possible. We define reconstruction quality by using *Intersection-over-Union* (*IoU*) metric [41] as defined by Equation (4), where $A$ denotes a turned on voxel in ground truth and $B$ denotes a turned on voxel in prediction, and $P$ is conditional probability.

$$IoU = \frac{P(B|A)}{P(B|A) + P(\neg B|A) + P(B|\neg A)} \tag{4}$$

We also use the *Completeness*, *Correctness* and *Quality* metrics [42]. *Completeness*, also known as *Producer's Accuracy* and *Detection Rate*, is the ratio of voxels in ground truth that were reconstructed:

$$Completeness = \frac{P(B|A)}{P(B|A) + P(B|\neg A)} \tag{5}$$

The *Correctness* metric shows how well the reconstructed voxels match the ground truth:

$$Correctness = \frac{P(B|A)}{P(B|A) + P(\neg B|A)} \tag{6}$$

The *Quality* metric gives a combined value that balances both correctness and completeness as follows:

$$Quality = \frac{Completeness \cdot Correctness}{Completeness + Correctness - Completeness \cdot Correctness} \tag{7}$$

## 3. Results

### 3.1. Experimental Settings

The experiments were performed on two different computers: (1) a computer workstation containing *nVidia 1070* graphics card, *Intel i7-4790* processor and installed *16 GB of RAM* which managed to achieve an average of 151 frames per second, and (2) a laptop with a *nVidia 960M* graphics chip, *Intel i5-4210H* processor and installed *12GB of RAM*, which was still able to achieve an average of 28.88 frames per second. We think that both computers represent the range of consumer devices, while the achieved graphics processing speed should be enough for most applications that would use consumer grade depth sensors. Please note that the proposed reconstruction algorithm is GPU bound, therefore we are interested in specifications of the graphics chip.

### 3.2. Quantitative Results

The quantitative results of the proposed algorithm during classification task can be observed in Table 1, as we can see the median recall rate for classification task is close to 84% in the classification task.

**Table 1.** Quantitative results of classifier recall rate on testing set. Our entire dataset was split using 80:20 rule into training and validation sets.

| Object | Classification Correctness | Number of Unique Testing Set Objects per Class | Number of Unique Testing Set Objects per Class |
|---|---|---|---|
| Bottle | 0.850 | 399 | 99 |
| Book | 0.754 | 68 | 17 |
| Can | 0.844 | 87 | 21 |
| Chair | 0.777 | 2289 | 572 |
| Knife | 0.898 | 340 | 85 |
| Laptop | 0.772 | 460 | 115 |
| Mug | 0.949 | 172 | 42 |
| Pillow | 0.863 | 77 | 20 |
| Bowl | 0.843 | 149 | 36 |
| Mean | 0.839 | 449 | 112 |

The qualitative results for the proposed reconstruction neural network are presented in terms of the *IoU* metric in Table 2.

**Table 2.** Comparison of qualitative reconstruction comparison between prediction and ground truths using the IoU metric, and the number of different objects in training set.

| Object | $IoU_{min}$ | $\overline{IoU}$ | $IoU_{max}$ | Percentage of Good and Excellent Reconstructions |
|---|---|---|---|---|
| Bottle | 0.220 | 0.578 | 0.980 | 98.438 |
| Book | 0.104 | 0.389 | 0.707 | 79.167 |
| Can | 0.524 | 0.798 | 0.996 | 100 |
| Chair | 0.239 | 0.440 | 0.986 | 98.958 |
| Knife | 0.201 | 0.608 | 0.944 | 97.917 |
| Laptop | 0.044 | 0.333 | 0.984 | 58.201 |
| Mug | 0.149 | 0.446 | 0.952 | 89.005 |
| Pillow | 0.229 | 0.681 | 0.995 | 98.953 |
| Bowl | 0.106 | 0.617 | 0.995 | 84.896 |
| Mean | 0.202 | 0.543 | 0.949 | 89.504 |

Please note that the $IoU$ metric values presented in Table 2 do not fully capture the quality of the reconstruction due to the low minimum values skewed by failed reconstruction. Therefore, we differentiate the $IoU$ values into three groups of *Poor*, *Good* and *Excellent* quality. We have selected the $IoU$ values corresponding to said groups based on the heuristically set threshold values for the best and worst results when inspecting the models visually. We assume *Poor* quality reconstruction is not able to reach $IoU$ of 0.25, and *Excellent* quality reconstruction has the $IoU$ value exceeding 0.75, while the *Good* quality has $IoU \in (0.25, 0.75]$. As we can see from Figure 8, a majority of reconstruction results fall into *Good* category, letting us assert that we achieved the desired goal. However, we still have outliers, such as *Laptop* and *Book*. The poor quality of *Book* reconstruction can be explained by training set being the least diverse of all, which, unlike other sets, has not been properly normalized. Poor reconstruction of *Laptop* may also be caused by poor training set as all of the training models, which contain only opened laptops. To present an overall evaluation of quality, we present the percentage of good and excellent reconstructions with $IoU \geq 0.25$ in Table 2.

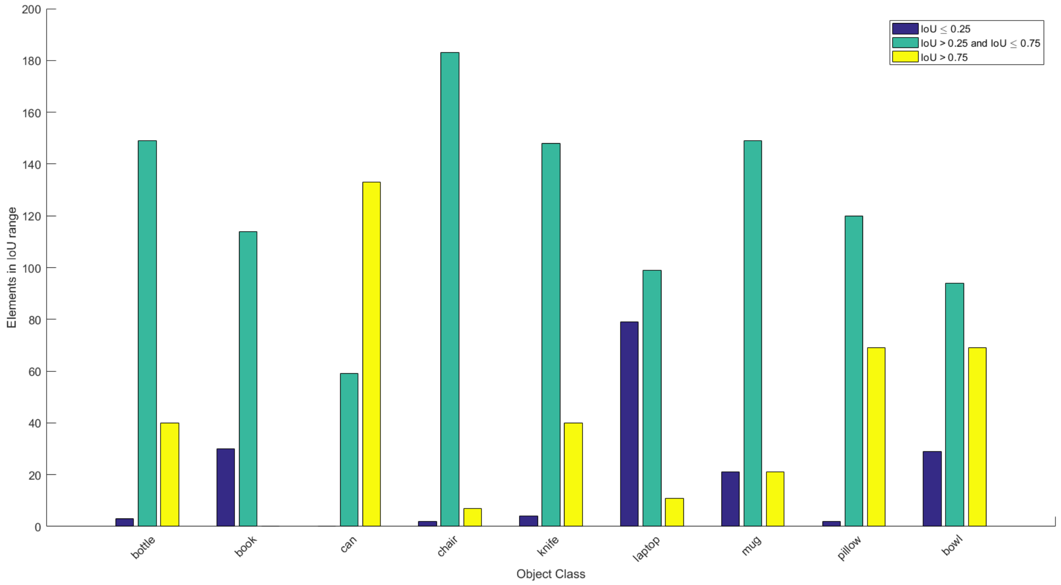


**Figure 8.** Histogram of the IoU values for heuristic comparison of reconstruction results.

The values of *Completeness*, *Corectness*, and *Quality* are summarized in Figure 9. More simple objects with round shape (such as *Can* or *Bottle*) were reconstructed with a larger accuracy.

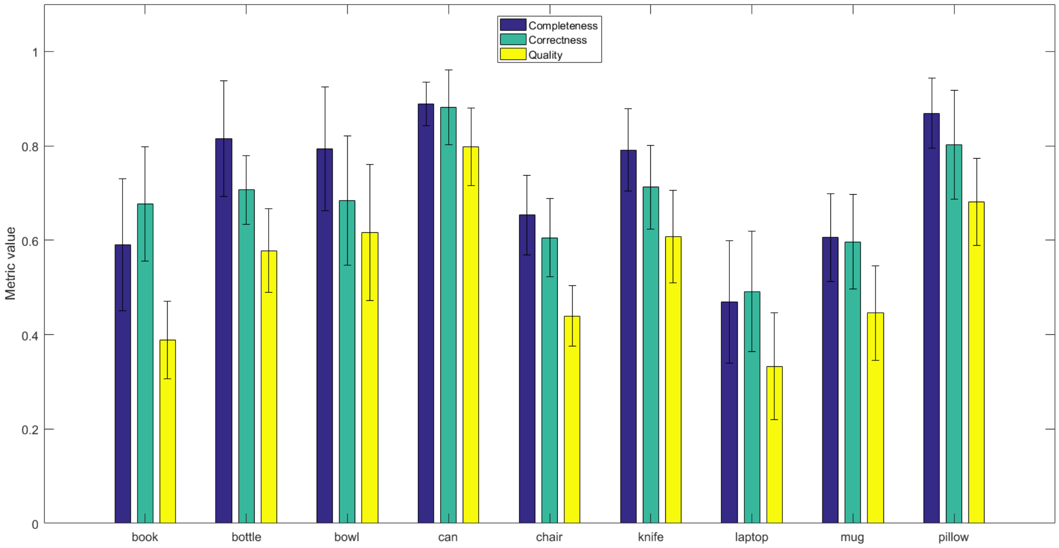


**Figure 9.** Comparison of reconstruction results using *Completeness*, *Correctness*, and *Quality*) metric values.

We also have compared object similarity based on its kernel features (see Figure 10). This was done by computing the average a sliding 3 × 3 kernel and comparing to the features found in the ground truth object. Difference between features indicates drift from the expected ground truth, while 0 indicates that features are identical.

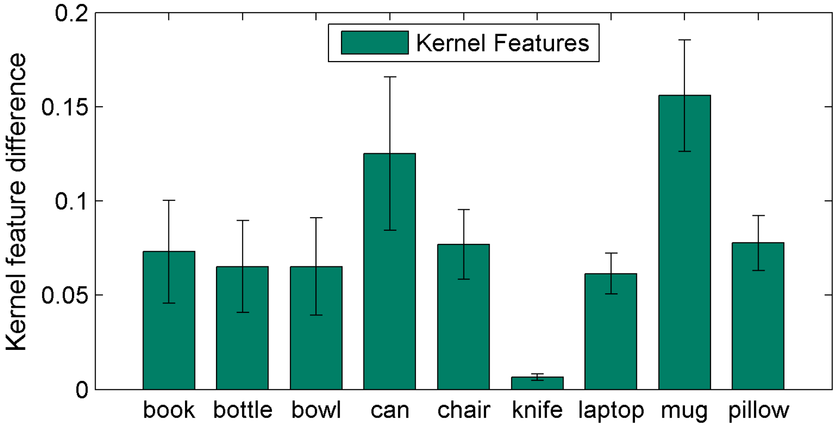


**Figure 10.** Comparison of kernel features when using 3 × 3 kernel to identify similarities between objects. Zero indicates that kernel features are identical to those of ground truth.

### 3.3. Visual Comparison of Reconstruction Results

As we do not have ground truths for real sensor data we can evaluate the results qualitatively by visual inspection. In Figure 11, we provide an example of real depth sensor data and the reconstruction results. For this example, we take an RGB-D frame using a depth sensor used for reconstruction. We use RGB sensor data only as reference point for us to inspect the quality of reconstruction, as the data is not used in the algorithm. We normalize the given depth frame as described in Figure 3, classify it in order to select the correct reconstruction network and forward data to the trained reconstruction model. Once we receive the voxel cloud, we transform it into polygonal mesh and apply smoothing. Finally, we inspect it visually using *Blender*.

We also visually compare predictions on synthetic data for our existing models. In Figure 12, we can see the validation results from multiple angles for an example of synthetic depth input of *Bowl* object. The predicted voxel cloud (*red*) value does resemble the object depth field quite well, including the inlet of the bowl, the difference between ground truth (*green*) and prediction are definitely noticeable albeit majority of differences can be considered as negligible. For example, the predicted value is slightly offset from the ground truth center, and predicted bowl is slightly higher then ground truth. However, there are some more important defects, like holes in the predicted voxel cloud and islands of disjointed voxels.

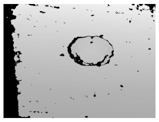
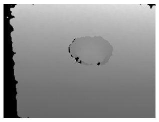
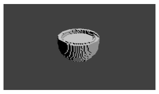
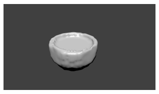
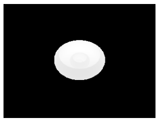
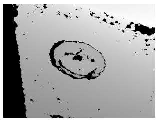
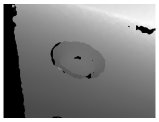
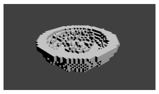
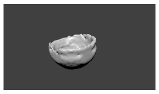
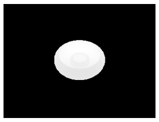
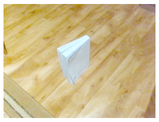
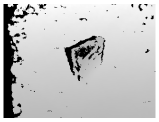
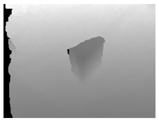
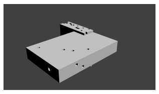
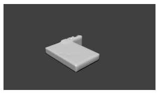
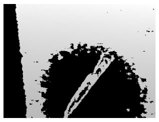
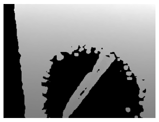
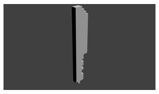
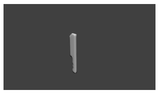
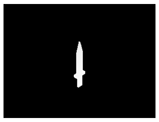
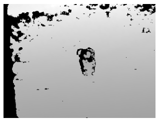
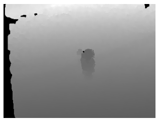
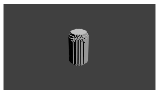
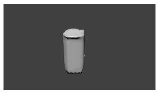
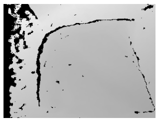
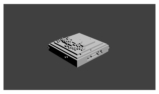
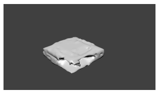
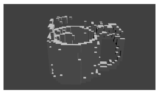
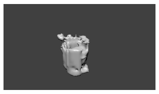
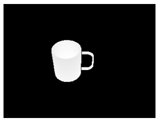
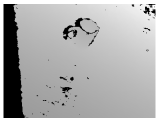
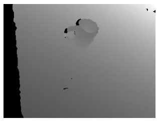
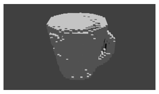
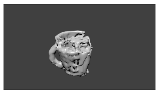
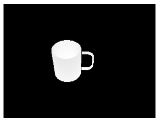
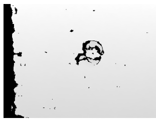
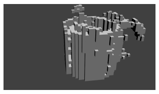
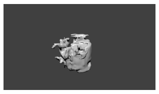
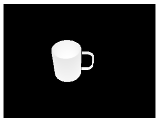
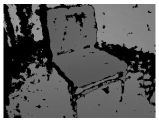
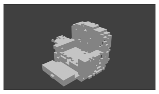
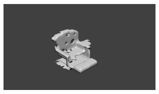
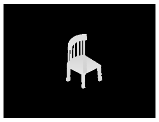
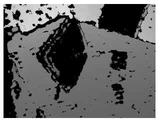
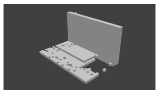
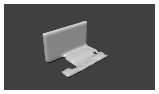
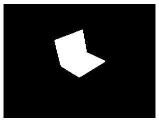
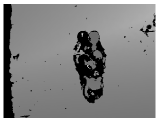
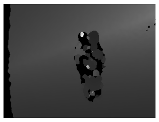
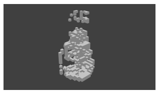
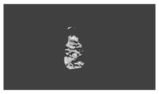
We have collected frames for each of the trained objects in order to inspect reconstruction quality for each class. The results are presented in Table 3 with frames taken at extreme angles, in an attempt to test the reconstruction network. The results show that the proposed neural network architecture is robust against such manipulations and was still able to predict the general shape of an object. The depth frames of *Bowl* captured by *Intel RealSense* have been reconstructed properly in terms of shape, albeit we can see some issues with the inner part of the bowl. In the first reconstruction (see the 1st row of Table 3), the reconstructed bowl is very shallow. In the second reconstruction (see the 2nd row of Table 3), the reconstructed shape has multiple artefacts inside it, although due to the localization of the noise it may be attributed that depth map in question being a lot noisier. The *Book* dataset managed to reconstruct the basic shape of the object. However, it contains an additional appendage which does not seem to be immediately obvious in the depth field. The *Knife* dataset has managed to reconstruct the shape very well, retaining handle-to-blade ratio and seemingly recognizing smaller dents in the handle. Neural network managed to reconstruct the *Bottle* without any obvious glitches. However, we can observe in the RGB image that the object has a narrowing at the top which the network did not manage to capture. However, we still can see that the specific part of object is noisy as well, as *Intel RealSense* was not able to capture it properly. However, due to the $32 \times 32 \times 32$ voxel density we are using, such details would most likely not be visible anyway.

ANN that is trained to reconstruct *Pillow* has managed to perform the task relatively well. However, we can see from the RGB image that the pillow had odd corners, which were not captured by reconstructing ANN. The *Mug* was one of the most complex objects in our training set. However, the reconstructed object is definitely recognizable as a mug. However, we can see that with so many errors in the reconstructed voxel cloud, the smoothing algorithm had trouble while polygonizing the mesh. The *Chair* was reconstructed well and is recognizable as a chair. Unfortunately the full detail of chair's legs was not captured. The *Notebook* also is easily recognizable, although the polarized glass screen of the notebook appears as black in *Intel RealSense* frame. While this may cause a lot of issues due to network not being trained for it, reconstruction has failed in other places instead. The final dataset consists of *Bottle*, which faced the issue with *Intel RealSense* being unable capturing PET objects, which has caused the depth map to be completely garbled, while the reconstruction results are not recognizable.

In Figure 13, we present an example of the results of reconstruction for the same object captured from different viewing angles. In one of the images we can see that the handle of a *Mug* is not present in the RGB camera. However, the ANN was still able to infer that the mug should have a handle as the network was only trained on mugs that have handles. Finally, we can see that in the particularly noisy depth fields, the reconstruction quality has dropped significantly, meaning the there was not enough data in the corrupted images for us to reconstruct from a single frame.

In Table 4, we present an example of 3D object reconstruction performed in pitch black (row 1) and low (row 2) illumination conditions. we can see that although in RGB images (column 1) *Mug* is poorly visible, the IR camera captures *Mug* (column 2) fairy well and the result of reconstruction (column 6) is easily recognizable.

**Table 3.** Visual qualitative reconstruction results. Table shows: RGB frame, infrared (depth) frame, normalized depth frame; reconstructed voxel cloud; polygonized and smoothed voxel cloud; an example of a similar object in training set.

| RGB | Depth | Normalized Depth | Voxel Cloud | Mesh | Training Data |
|---|---|---|---|---|---|
| |  |  |  |  |  |
| |  |  |  |  |  |
|  |  |  |  |  | |
| |  |  |  |  |  |
| |  |  |  |  | |
| |  | |  |  | |
| | | |  |  |  |
| |  |  |  |  |  |
| |  | |  |  |  |
| |  | |  |  |  |
| |  | |  |  |  |
| |  |  |  |  | |

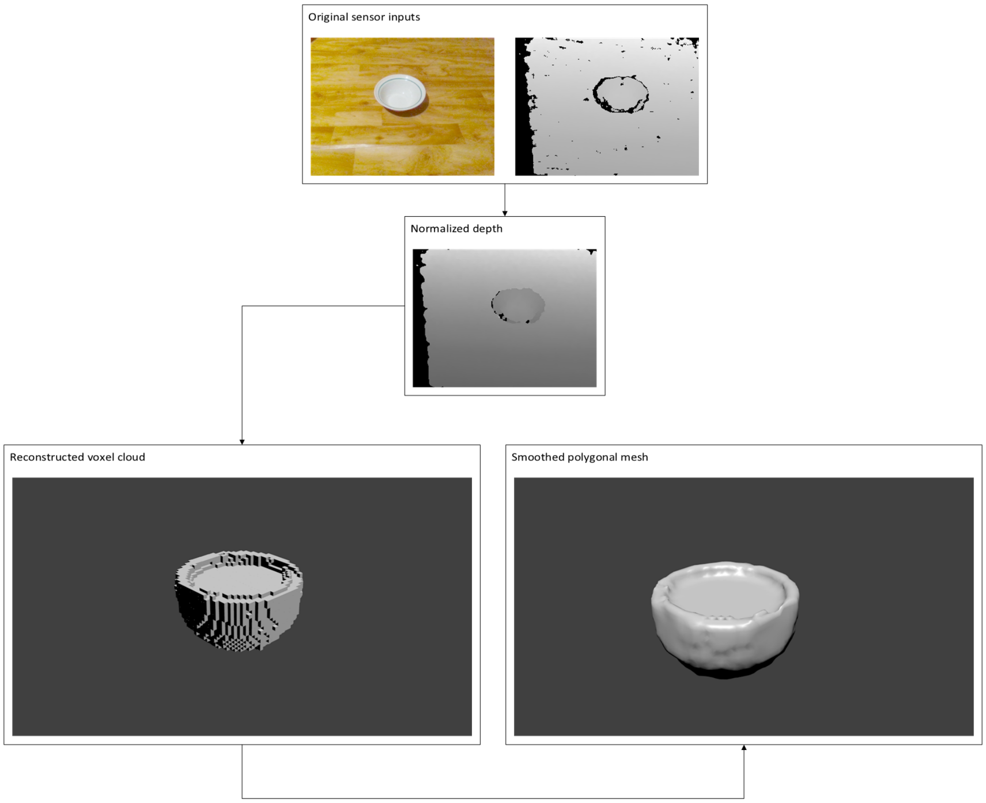


**Figure 11.** Visual comparison between inputs and outputs.

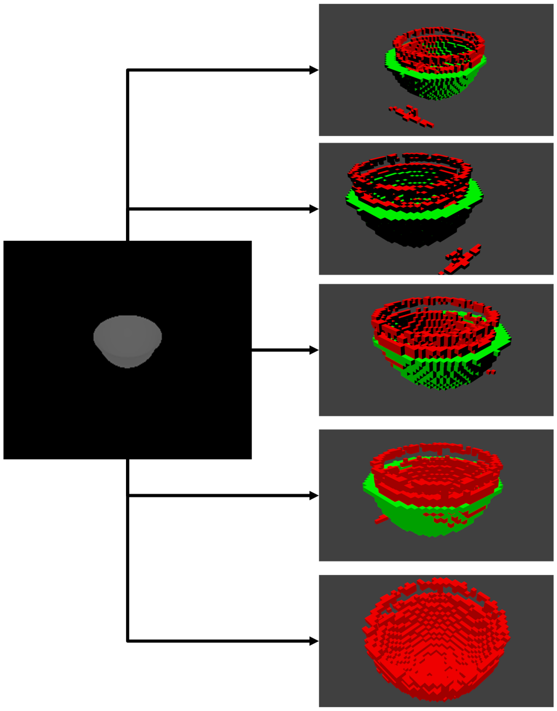
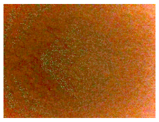
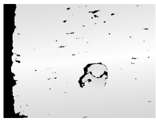
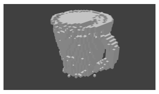
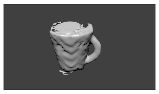
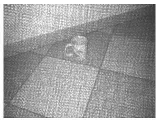
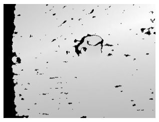
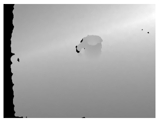
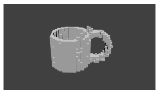
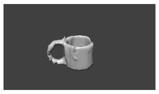


**Figure 12.** An example of visual comparison ground truth and prediction on the validation set of *Bowl* shape. Green color denotes ground truth, and red color shows prediction.

**Table 4.** 3D reconstruction in low illumination conditions: RGB frame, infrared frame, depth frame, normalized depth frame; reconstructed voxel cloud; polygonized and smoothed voxel cloud.

| RGB | Infrared | Depth | Normalized Depth | Voxel Cloud | Mesh |
|---|---|---|---|---|---|
|  | |  | |  |  |
| |  |  |  |  |  |

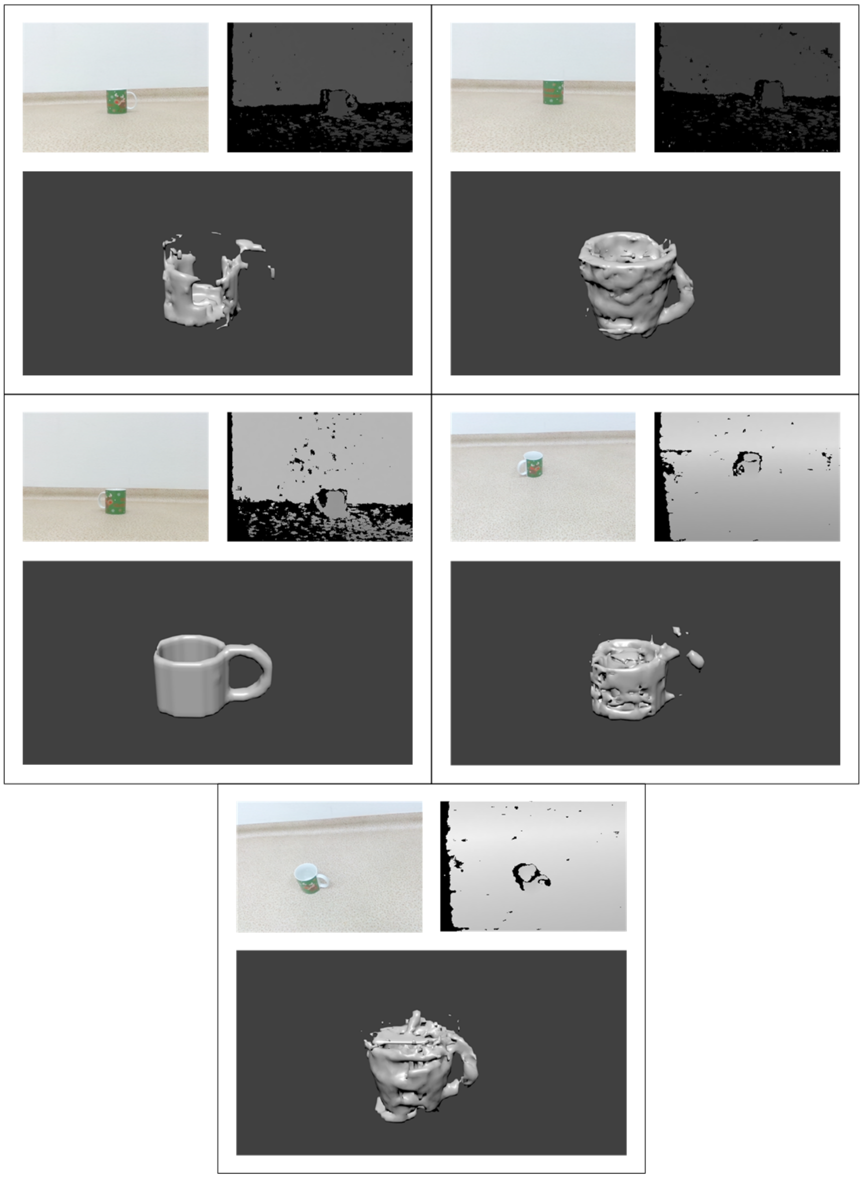


**Figure 13.** Visual comparison of the same *Mug* object from varying perspectives.

## 4. Discussion and Concluding Remarks

### 4.1. Discussion

The advantage of the proposed hybrid neural network (HNN) architecture is that unlike non-hybrid approach, which usually requires to re-train the entire network (off-line) or risk loosing existing gradients (on-line) due to network being skewed towards new data points, the proposed HNN architecture is modular and can easily extend the already trained network by adding additional reconstruction nodes, replace already existing nodes with a better trained model, etc. Although adding additional reconstruction nodes requires to re-train the classifier network, the classifier network still is more light-weight and requires less processing power to train. In addition to this we can have different ANN architectures per node, allowing for a specific reconstruction node to have a more precisely selected reconstruction model. Furthermore, this approach gives us the potential to have variable network complexity contingent upon the complexity of the object we desire to reconstruct, further expanding the applicability of the proposed HNN architecture. Such architecture allowed us to create a system capable of reconstructing polygonal mesh of a self-occluding object by using only a single depth frame on lower-end devices.

We believe that further improvements of network architecture are possible to improve the quantitative performance of 3D recognition. Possible venues of future research may include selecting network architecture that manages to converge well and pruning dead neurons [43]; gradually increasing the complexity of the network until desired reconstruction quality is achieved [44]; using neuro-evolutionary and neuro-genetic algorithms in order to find satisfying network solution [45]; improving learning of networks by using metaheuristic control mechanisms [46]; or using video feed instead of a single frame of an object as multiple depth frames from a single perspective can actually reveal new features [47] thus improving the recall rate, with recurrent neural networks (RNN) being one of the biggest contenders in predicting sequential data [48,49]. Moreover, additional functionality in the method such as solving homography would allow us to extract the transformation matrix of the object, allowing the system to be used for such applications as Virtual Reality in conjunction with Augmented Reality. Finally, using RGB sensor frames in conjunction with depth frames may add some missing features to improve the recall rate even more [50,51].

### 4.2. Threats to Validity

A relatively old *Intel RealSense* device was used for the preparation of real-life training dataset which introduced a limitation as the used device did not provide valid depth information if placed too close to the object, while simultaneously being unable to capture the minute details. This has limited us to relatively large (at least 8 × 10 cm) objects that we can use for reconstruction as putting the camera too close to an object would result in frame corruption (see an example given in Figure 14), while placing the camera far enough (reliable depth capture range is 0.55 to 2.8 m distance) to use the depth sensor would cause the object features to be indistinguishable from the background.

Moreover, the *Intel RealSense* device was unable to properly capture glass surface, e.g., a laptop screen, or translucent PET plastics, which resulted in creating holes and distortions in depth images thus making 3D reconstruction difficult.

### 4.3. Concluding Remarks

We have proposed a hybrid neural network architecture that has managed to reach the goal of reconstructing the shape of 3D objects from different viewing angles using the *Intel RealSense ZR300* device.

The mean IoU value for all objects was in the range of 0.333 to 0.798, obtaining on average 89.5% of good and excellent reconstructions, which is equivalent to the results achieved by other methods, while the reconstructed shapes are easily recognizable by visual inspection.

Furthermore, our proposed architecture allows for an easy extension (requiring very few iterations to train for a new an object type), can work in low illumination environments and has little dependence on ambient lightning, which enables the application of it in more realistic lightning conditions and even where there is no ambient light. This allows our method a broader application spectrum as opposed to other approaches.

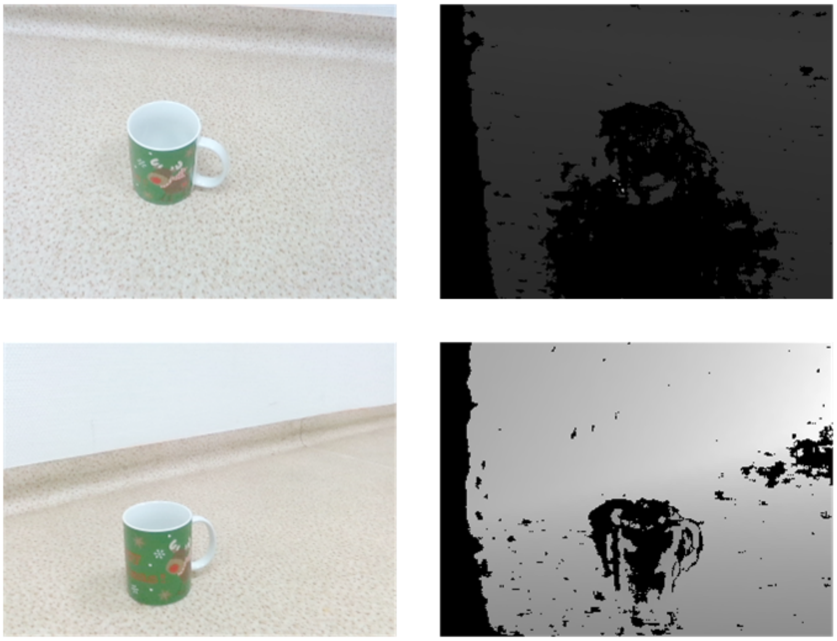


**Figure 14.** An example of corrupted frames.

**Author Contributions:** Conceptualization, R.M.; Formal analysis, S.M.; Investigation, A.K. and R.M.; Methodology, R.M.; Software, A.K.; Supervision, R.M.; Validation, A.K. and R.M.; Visualization, A.K. and R.D.; Writing—original draft, A.K.; Writing–review & editing, R.M., R.D. and S.M.

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Fanini, B.; Pagano, A.; Ferdani, D. A Novel Immersive VR Game Model for Recontextualization in Virtual Environments: The uVRModel. *Multimodal Technol. Interact.* **2018**, *2*, 20. [CrossRef]
2. Liao, B.; Li, J.; Ju, Z.; Ouyang, G. Hand Gesture Recognition with Generalized Hough Transform and DC-CNN Using Realsense. In Proceedings of the 2018 Eighth International Conference on Information Science and Technology (ICIST), Cordoba, Spain, 30 June–6 July 2018. [CrossRef]
3. Chen, C.; Yang, B.; Song, S.; Tian, M.; Li, J.; Dai, W.; Fang, L. Calibrate Multiple Consumer RGB-D Cameras for Low-Cost and Efficient 3D Indoor Mapping. *Remote Sens.* **2018**, *10*, 328. [CrossRef]
4. Jusas, V.; Birvinskas, D.; Gahramanov, E. Methods and Tools of Digital Triage in Forensic Context: Survey and Future Directions. *Symmetry* **2017**, *9*, 49. [CrossRef]
5. Haleem, A.; Javaid, M. 3D scanning applications in medical field: A literature-based review. *Clin. Epidemiol. Glob. Health* **2018**. [CrossRef]

6. Macher, H.; Landes, T.; Grussenmeyer, P. From Point Clouds to Building Information Models: 3D Semi-Automatic Reconstruction of Indoors of Existing Buildings. *Appl. Sci.* **2017**, *7*, 1030. [CrossRef]
7. Wang, L.; Li, R.; Shi, H.; Sun, J.; Zhao, L.; Seah, H.; Quah, C.; Tandianus, B. Multi-Channel Convolutional Neural Network Based 3D Object Detection for Indoor Robot Environmental Perception. *Sensors* **2019**, *19*, 893. [CrossRef] [PubMed]
8. Remondino, F. Heritage Recording and 3D Modeling with Photogrammetry and 3D Scanning. *Remote Sens.* **2011**, *3*, 1104–1138. [CrossRef]
9. Chu, P.M.; Sung, Y.; Cho, K. Generative Adversarial Network-Based Method for Transforming Single RGB Image into 3D Point Cloud. *IEEE Access* **2019**, *7*, 1021–1029. [CrossRef]
10. Wald, J.; Tateno, K.; Sturm, J.; Navab, N.; Tombari, F. Real-Time Fully Incremental Scene Understanding on Mobile Platforms. *IEEE Robot. Autom. Lett.* **2018**, *3*, 3402–3409. [CrossRef]
11. Daudelin, J.; Campbell, M. An Adaptable, Probabilistic, Next-Best View Algorithm for Reconstruction of Unknown 3-D Objects. *IEEE Robot. Autom. Lett.* **2017**, *2*, 1540–1547. [CrossRef]
12. Fuentes-Pacheco, J.; Ascencio, J.R.; Rendón-Mancha, J.M. Visual simultaneous localization and mapping: A survey. *Artif. Intell. Rev.* **2012**, *43*, 55–81. [CrossRef]
13. Zollhöfer, M.; Stotko, P.; Görlitz, A.; Theobalt, C.; Niessner, M.; Klein, R.; Kolb, A. State of the Art on 3D Reconstruction with RGB-D Cameras. *Comput. Graph. Forum* **2018**, *37*, 625–652. [CrossRef]
14. Kutulakos, K.N.; Seitz, S.M. A theory of shape by space carving. In Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 20–27 September 1999; Volume 1, pp. 307–314. [CrossRef]
15. Fan, H.; Su, H.; Guibas, L.J. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2463–2471.
16. Li, C.; Zia, M.Z.; Tran, Q.; Yu, X.; Hager, G.D.; Chandraker, M. Deep Supervision with Shape Concepts for Occlusion-Aware 3D Object Parsing. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 388–397. [CrossRef]
17. Yang, B.; Rosa, S.; Markham, A.; Trigoni, N.; Wen, H. Dense 3D Object Reconstruction from a Single Depth View. *arXiv* **2018**, arXiv:1802.00411 .
18. Zhang, Z. Microsoft Kinect Sensor and Its Effect. *IEEE Multimed.* **2012**, *19*, 4–10. [CrossRef]
19. Keselman, L.; Woodfill, J.I.; Grunnet-Jepsen, A.; Bhowmik, A. Intel(R) RealSense(TM) Stereoscopic Depth Cameras. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 21–26 July 2017. [CrossRef]
20. Tang, L.; Yang, Z.; Jia, K. Canonical Correlation Analysis Regularization: An Effective Deep Multi-View Learning Baseline for RGB-D Object Recognition. *IEEE Trans. Cognit. Dev. Syst.* **2018**, *11*, 107–118. [CrossRef]
21. Zhang, L.; Sun, J.; Zheng, Q. 3D Point Cloud Recognition Based on a Multi-View Convolutional Neural Network. *Sensors* **2018**, *18*, 3681. [CrossRef]
22. Tian, G.; Liu, L.; Ri, J.; Liu, Y.; Sun, Y. ObjectFusion: An object detection and segmentation framework with RGB-D SLAM and convolutional neural networks. *Neurocomputing* **2019**. [CrossRef]
23. Zeng, H.; Yang, B.; Wang, X.; Liu, J.; Fu, D. RGB-D Object Recognition Using Multi-Modal Deep Neural Network and DS Evidence Theory. *Sensors* **2019**, *19*, 529. [CrossRef] [PubMed]
24. Oliveira, F.F.; Souza, A.A.S.; Fernandes, M.A.C.; Gomes, R.B.; Goncalves, L.M.G. Efficient 3D Objects Recognition Using Multifoveated Point Clouds. *Sensors* **2018**, *18* 2302. [CrossRef]
25. Khoshelham, K.; Elberink, S.O. Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. *Sensors* **2012**, *12*, 1437–1454. [CrossRef]
26. Carfagni, M.; Furferi, R.; Governi, L.; Servi, M.; Uccheddu, F.; Volpe, Y. On the Performance of the Intel SR300 Depth Camera: Metrological and Critical Characterization. *IEEE Sens. J.* **2017**, *17*, 4508–4519. [CrossRef]
27. Stutz, D.; Geiger, A. Learning 3D Shape Completion Under Weak Supervision. *Int. J. Comput. Vis.* **2018**. [CrossRef]

28. Wiles, O.; Zisserman, A. Learning to Predict 3D Surfaces of Sculptures from Single and Multiple Views. *Int. J. Comput. Vis.* **2018**. [CrossRef]
29. Cao, Y.; Shen, C.; Shen, H.T. Exploiting Depth From Single Monocular Images for Object Detection and Semantic Segmentation. *IEEE Trans. Image Process.* **2017**, *26*, 836–846. [CrossRef] [PubMed]
30. Hisatomi, K.; Kano, M.; Ikeya, K.; Katayama, M.; Mishina, T.; Iwadate, Y.; Aizawa, K. Depth Estimation Using an Infrared Dot Projector and an Infrared Color Stereo Camera. *IEEE Trans. Circuits Syst. Video Technol.* **2017**, *27*, 2086–2097. [CrossRef]
31. Du, Y.; Fu, Y.; Wang, L. Representation Learning of Temporal Dynamics for Skeleton-Based Action Recognition. *IEEE Trans. Image Process.* **2016**, *25*, 3010–3022. [CrossRef] [PubMed]
32. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2015**, arXiv:1412.6980.
33. Nair, V.; Hinton, G.E. Rectified Linear Units Improve Restricted Boltzmann Machines. In Proceedings of the 27th International Conference on Machine Learning (ICML'10), Haifa, Israel, 21–24 June 2010.
34. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 13–16 December 2015; pp. 1026–1034. [CrossRef]
35. Bartsch, M.; Weiland, T.; Witting, M. Generation of 3D isosurfaces by means of the marching cube algorithm. *IEEE Trans. Magn.* **1996**, *32*, 1469–1472. [CrossRef]
36. Ju, T.; Losasso, F.; Schaefer, S.; Warren, J. Dual Contouring of Hermite Data. *ACM Trans. Graph.* **2002**, *21*, 339–346. [CrossRef]
37. Kainz, F.; Bogart, R.R.; Hess, D.K. The OpenEXR Image File Format. In *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*; Addison-Wesley Professional: Boston, MA, USA, 2004.
38. Pantaleoni, J. VoxelPipe: A programmable pipeline for 3D voxelization. In Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics (HPG '11), Vancouver, BC, Canada, 5–7 August 2011; pp. 99–106. [CrossRef]
39. Baldwin, D.; Weber, M. Fast Ray-Triangle Intersections by Coordinate Transformation. *J. Comput. Graph. Techol.* **2016**, *5*, 39–49.
40. Chang, A.X.; Funkhouser, T.A.; Guibas, L.J.; Hanrahan, P.; Huang, Q.X.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; et al. ShapeNet: An Information-Rich 3D Model Repository. *arXiv* **2015**, arXiv:1512.03012.
41. Marin-Jimenez, M.J.; Zisserman, A.; Eichner, M.; Ferrari, V. Detecting People Looking at Each Other in Videos. *Int. J. Comput. Vis.* **2013**, *106*, 282–296. [CrossRef]
42. Rutzinger, M.; Rottensteiner, F.; Pfeifer, N. A Comparison of Evaluation Techniques for Building Extraction From Airborne Laser Scanning. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2009**, *2*, 11–20. [CrossRef]
43. Wang, J.; Xu, C.; Yang, X.; Zurada, J.M. A Novel Pruning Algorithm for Smoothing Feedforward Neural Networks Based on Group Lasso Method. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 2012–2024. [CrossRef]
44. Huang, G.B.; Saratchandran, P.; Sundararajan, N. A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. *IEEE Trans. Neural Netw.* **2005**, *16*, 57–67. [CrossRef] [PubMed]
45. Arifovic, J.; Gençay, R. Using genetic algorithms to select architecture of a feedforward artificial neural network. *Phys. A Stat. Mech. Its Appl.* **2001**, *289*, 574–594. [CrossRef]
46. Połap, D.; Kęsik, K.; Woźniak, M.; Damaševičius, R. Parallel Technique for the Metaheuristic Algorithms Using Devoted Local Search and Manipulating the Solutions Space. *Appl. Sci.* **2018**, *8*, 293. [CrossRef]
47. Izadi, S.; Kim, D.; Hilliges, O.; Molyneaux, D.; Newcombe, R.; Kohli, P.; Shotton, J.; Hodges, S.; Freeman, D.; Davison, A.; et al. Kinectfusion: Real-time 3D reconstruction and interaction using a moving depth camera. In Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology UIST, Santa Barbara, CA, USA, 16–19 October 2011; pp. 559–568. [CrossRef]
48. Wang, J.; Zhang, L.; Guo, Q.; Yi, Z. Recurrent Neural Networks With Auxiliary Memory Units. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 1652–1661. [CrossRef] [PubMed]

49. Hawkins, J.; Boden, M. The applicability of recurrent neural networks for biological sequence analysis. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2005**, *2*, 243–253. [CrossRef]
50. Liu, Z.; Zhao, C.; Wu, X.; Chen, W. An Effective 3D Shape Descriptor for Object Recognition with RGB-D Sensors. *Sensors* **2017**, *17*, 451. [CrossRef] [PubMed]
51. Hsu, G.J.; Liu, Y.; Peng, H.; Wu, P. RGB-D-Based Face Reconstruction and Recognition. *IEEE Trans. Inf. Forensics Secur.* **2014**, *9*, 2110–2118. [CrossRef]