Article

3D Object Reconstruction from Imperfect Depth Data Using Extended YOLOv3 Network

Audrius Kulikajevas [1], Rytis Maskeliūnas [1] and Robertas Damaševičius [2,3,*] and Edmond S. L. Ho [4]

[1] Department of Multimedia Engineering, Kaunas University of Technology, 51423 Kaunas, Lithuania; audrius.kulikajevas@ktu.edu (A.K.); rytis.maskeliunas@ktu.lt (R.M.)
[2] Department of Applied Informatics, Vytautas Magnus University, 44404 Kaunas, Lithuania
[3] Faculty of Applied Mathematics, Silesian University of Technology, 44-100 Gliwice, Poland
[4] Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne NE1 8ST, UK; e.ho@northumbria.ac.uk
[*] Correspondence: robertas.damasevicius@vdu.lt

check for updates

Abstract: State-of-the-art intelligent versatile applications provoke the usage of full 3D, depth-based streams, especially in the scenarios of intelligent remote control and communications, where virtual and augmented reality will soon become outdated and are forecasted to be replaced by point cloud streams providing explorable 3D environments of communication and industrial data. One of the most novel approaches employed in modern object reconstruction methods is to use a priori knowledge of the objects that are being reconstructed. Our approach is different as we strive to reconstruct a 3D object within much more difficult scenarios of limited data availability. Data stream is often limited by insufficient depth camera coverage and, as a result, the objects are occluded and data is lost. Our proposed hybrid artificial neural network modifications have improved the reconstruction results by 8.53% which allows us for much more precise filling of occluded object sides and reduction of noise during the process. Furthermore, the addition of object segmentation masks and the individual object instance classification is a leap forward towards a general-purpose scene reconstruction as opposed to a single object reconstruction task due to the ability to mask out overlapping object instances and using only masked object area in the reconstruction process.

Keywords: 3D scanning; 3D shape reconstruction; RGB-D sensors; imperfect data; hybrid neural networks

1. Introduction

One of the pressing issues in computer vision is three-dimensional (3D) object reconstruction, due to it becoming a core technology in numerous high-end industrial applications such as smart manufacturing, industrial automation and Industry 4.0 [1]. Moreover, there exists a wide variety of applications that would benefit from real time computer vision systems that are capable of fully reconstructing scenes, with most notable examples being an interactive medium such as virtual reality (VR) games and simulations [2], augmented reality (AR) applications or even in newest booming technologies such as extended reality (XR) [3]. Further examples for applications of such systems could include gesture [4,5] and posture [6] applications, indoor mapping [7], obstacle detection [8] recreating environments in movies or even digital forensics [9] to allow for crime scene recreation, robotics [10], teleconferencing [11] with the use of holograms and more. Therefore, we can safely assert that there is definitely a need for affordable, commercially viable solutions capable of providing real-time reconstruction capabilities available to the average user with as little complexity and barrier of entry, in terms of both financial investments and knowledge about the field, as possible.

As we cannot expect an average user to have the access to professional filming sets, mounting arrays of laser scanners capable of scanning the entirety of the room, in addition to the computing resources that would be required to stitch the data retrieved from multiple high-fidelity depth sensors, we need a solution that would meet or exceed the previous caveats. Therefore, we need a solution capable of working in real-time on a regular non-enthusiast grade workstation or even on a laptop. Furthermore, while we cannot expect the user to have a modern sensor array setup we can try to minimize the initial setup cost to a single depth sensor available in electronics stores or even in quite a few modern mid-tier and flagship phones. While solutions for scene reconstruction from a single depth sensor already exist, these solutions require incremental building per each frame [12,13]. This is done based on camera localization information and delta frames and in the scene reconstruction algorithms that make use of simultaneous localization and mapping (SLAM) [14]. To reliably fill all the holes in the areas that are occluded by other objects and even because of self-occlusion, we would have to scan the entirety of the object from all sides to have its full profile. Furthermore, incremental methods tend to underperform because of one principal flaw: changes in the scene can disrupt the mesh [15]. Making the applications in non-static real world scenes limited, where instead of the entirety of the view moving some objects can change their localization, or even suddenly pop-in or pop-out of the frame. Other proposed methods, such as space carving [16], would bypass some of the incremental building problems by performing what is essentially a subtractive reconstruction from multiple perspectives. However, these methods assume that you can accurately acquire the mask, which can be impossible in certain lighting conditions.

A majority of current algorithms for performing 3D object reconstruction have limitations: objects must be monitored from a large number of views; or views must follow a small baseline, thus the methods cannot function properly when provided only a small number or a single view. To solve these issues one of the most novel approaches employed for state-of-the-art reconstruction algorithms is to employ a priori knowledge of the objects that are being reconstructed [17,18]. These are generally relying on black-box models such as neural networks (NN). One of the most obvious advantages of using a priori information is for the algorithm to approximate the occluded object information, which we as humans are capable inferring quite easily. These methods have shown success in solving this task. For example, 3D Recurrent Reconstruction Neural Network (3D-R2N2) for multi-view reconstruction on the Sanford Online Products [19] and ShapeNet [20] datasets, has managed to achieve this task with fewer images available with competitive results [21], with the proposed improvement that uses densely connected structure as encoder and utilizing Chamfer Distance as loss function [22]. Additionally, Generative Adversarial Networks (GANs) can be used to generate 3D objects from multiple 2D views [23] or even from a single image [24]. GANs have also been shown to be able to predict former geometry of damaged objects [25]. Other authors have used feedforward NNs to detect valid matches between points in an image using different views with more than 98% accuracy [26]. Additionally it was shown that by adopting Bernstein Basis Function Networks (BBFNs) it is also possible to solve the task of reconstructing a 3D shape [27]. A trilateral convolutional neural network (Tri-CNN) that uses three dilated convolutions in 3D to extend the convolutional receptive field was applied on the ShapeNet and Big Data for Grasp Planning [28] data sets to obtain 3D reconstruction from a single depth image [29].

A majority of methods are using voxel based representations, e.g., PointOutNet [30] has shown the ability to predict and generate plausible 3D object shapes. This allows for the model to perform multiple predictions from a single input and using point cloud distribution modeling to refine the final results. Other approaches include: hierarchical surface predictions (HSPs) [31] for predicting high resolution voxel grids using convolutional neural networks (CNNs); discrete wavelet transform (DWT) and principal component analysis (PCA) can be used to get targeted object models, which can be used as an input to an artificial neural network (ANN) to recognize the 3D shape. Other authors have used geometric adversarial loss (GAL) in order to regularize single-view 3D object for object reconstruction using a global perspective by training the GAN to reconstruct multi-view valid 3D

models [32]. RealPoint3D network composed of an encoder, a 2D-3D fusion module, and a decoder, accepts a single-object image and a nearest-shape retrieved from ShapeNet to generate fine-grained point clouds [33]. Similarly, PGNet [34], a recurrent generative network, uses the original images and partial projection images for fine-grained 3D reconstruction. Finally, it was shown that using ANNs it is possible to produce a fully textured, appropriately proportioned 3D model from a single RGB [35] or RGB-D frame [36], however, this approach was limited to basic volume primitives (rectangular boxes and spheres) .

Even though the black-box methods have shown substantial improvements over existing state-of-art reconstruction algorithms such as incremental reconstruction, they can still be prone to severe mishaps due to poor illumination conditions, and object material interaction with light (mainly reflectivity). Furthermore, due to the fact that these methods rely on the visible light spectrum, they are incapable of working in dark environments. Therefore, they would not be suitable to be used in critical applications such as security.

Starting with the *Microsoft Kinect* released in 2010 [37] to *Intel Realsense* [38], the depth sensors are becoming the norm not only in the flagship mobile phones. As of late, stereoscopic depth is becoming available in newer budget phones with the introduction of multiple back facing cameras on a single device. For these reasons we have almost reached an era of the RGB-Depth (RGB-D) sensors being readily available. Therefore, focusing solely on the RGB cameras is missing the potential that the RGB-D cameras may provide for the object reconstruction tasks. For example, depth data stream from the Kinect camera has been used to generate topologically correct 3D mesh models [39].

Applying additional information provided by the RGB-D senor is the logical next step in the lifecycle of the object reconstruction algorithms as we believe they are less dependent on ambient conditions and could potentially be used in pitch black situations due to modern depth sensors using infrared cameras for object depth calculations on the hardware level. We concede that the depth sensors have their own limitations such as speckling due to surface properties [40,41] and distortions caused by infrared projections [42]. However, we believe that the addition of the depth sensor information in conjunction with readily available color data adds useful information. This information helps ANNs to better generalize input data and increase robustness against different lighting conditions. This includes pitch black environments as the depth information is sufficient to reconstruct the captured scene in most cases.

We present an improved hybrid ANN architecture for reconstructing polygonal meshes using only a single RGB-D frame, and employing a priori knowledge, which allows the neural network to be deployed on low-end RGB-D sensor devices with low frame rates.

2. Materials and Methods

2.1. Proposed Hybrid Neural Network Architecture

Our hybrid NN architecture (Figure 1) consists of two major branches: the preliminary input branch that is used for object instance classification and their mask extraction; secondary input branch, which uses the results of preliminary branch in conjunction with the inputs of preliminary branch to perform individual object reconstruction. However, unlike preliminary branch we do not use generalized branches for reconstruction, instead we have *n* of specialized branches for each of the object categories. This allows us to more easily train additional types of objects in the reconstruction branches without having to re-train for classification, in addition this allows to re-train any of the individual reconstruction branches without losing the existing gradients by performing the training on more models [43]. The modularity of the system also provides the advantage of reduced training times as each branch can specialize onto its own generalization task, which gives the ability to change the network configurations of the reconstruction branches by simplifying for easier objects or having more elaborate ANN structures for more complex objects.
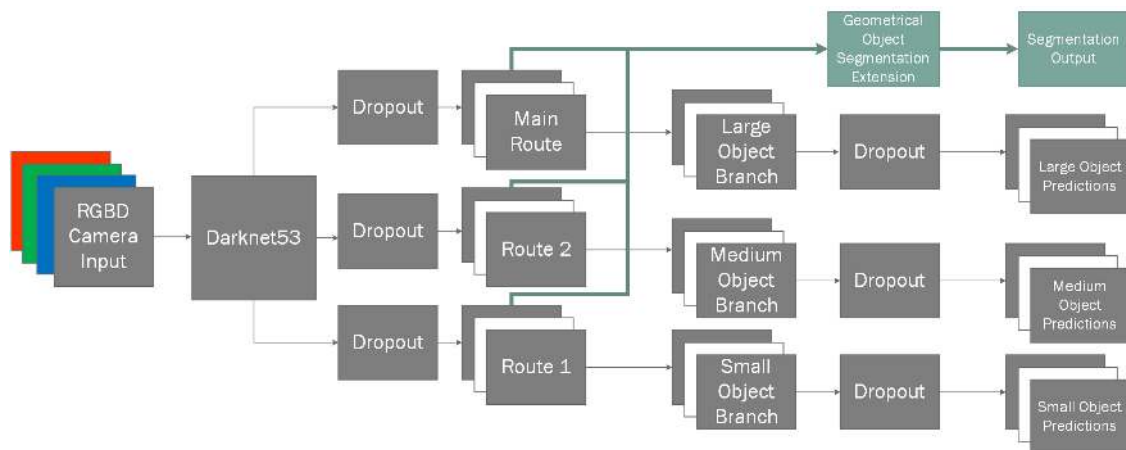
Figure 1. Our extended *YOLOv3* capable of extracting geometric object segmentation along with object bounding boxes.

2.2. Classification and Segmentation Algorithm

Our aim is to detect individual object instances in the scene in order to have a system that is usable in real-world environments. Therefore, we need a classifier that is capable of detecting more than a single object instance for given frame, for example, having two cups and a toy plane on a table would require us to rebuild both of the cups and the toy plane models, respectively. Fortunately, some research has already been performed in the area of individual object instance classification [44–46].

For this reason, to perform our classification task we use one of existing state-of-the-art classifiers as it has shown to produce some of the best results in classification tasks, i.e. *YOLOv3* [47], which we have adapted to our needs to output an additional geometric segmentation mask (Figure 1), while authors have mentioned to be unable to achieve object instance segmentation in their original paper. Additionally, we define the term geometric segmentation as extension to segmentation that allows to discriminate between nearby object instances. This is done by generating a heatmap glow that radiates from the origin of the object. While other more lightweight methods exist, such as *MobileNet* [48], in our paper we try to compare the classification results using three different methods: using only color information; using only depth information; using both color and depth information. Therefore, we have decided to use a slower, but more accurate algorithm to have the most representative results.

Just as the majority of the individual object instance classifying algorithms, *YOLOv3* uses what is know as anchors for object detection. These anchors are used as jumping off bounding boxes when classifying objects, for example, a motor vehicle has a very different profile from a basketball. While the basketball in most cases has close to 1:1 aspect ratio bounding box, meaning that their width is the same, or very close when the image is distorted, to its height, while a motor vehicle like an automobile for the most part has height that is lower than its width. For this reason, one anchor could specialize in detecting automobiles, while the other can specialize in detecting basketballs. Additional feature, albeit a less useful one due to the way our training and testing dataset is generated, is the specification of bounding box scales by the authors of *YOLOv3*. These size specializations group bounding boxes into three groups: small, medium and large. For example small objects may include kitchen utensils, medium objects may include people, large objects may include vehicles. However, these bounding box groups are not exclusionary for these objects unlike anchors as these can vary a lot based on the camera distance from the object. Therefore, as our dataset is completely uniformly generating object scales this grouping loses some of its usefulness.

In our work, we have experimented with three types of inputs into the ANN: color space, front-to-back object depth field and the combination of both. In the case of color space, we use 3 channel inputs for representation of *red*, *green*, *blue* colors; when using depth field, we use a single channel input containing only normalized depth field values and for the combination of both we use

RGBD channels in the same principle. Depth value normalization is performed by dividing each pixel z value using z_{max} of the frame thus landing the depth in range of $z = [0, 1]$. Our input layer is thereafter connected to *DarkNet53* network containing 53 convolutional layers as per specifications, which outputs three routes: *main route* used generally used for larger objects, *route 2* used for medium sized objects and, finally, *route 1* for smaller objects. Due to testing set being uniformly randomly generated, and containing the same object in potentially all size categories, we lose some of the flexibility that is provided by this setup and it impacts classification performance minimally, if removed. However, to stay true to the original algorithm and have an as unbiased result as possible, we have decided to keep all of the branches used in the source material. Additionally, these three routes provide good jumping off points for shortcuts to be used in our segmentation extension (Figure 2).
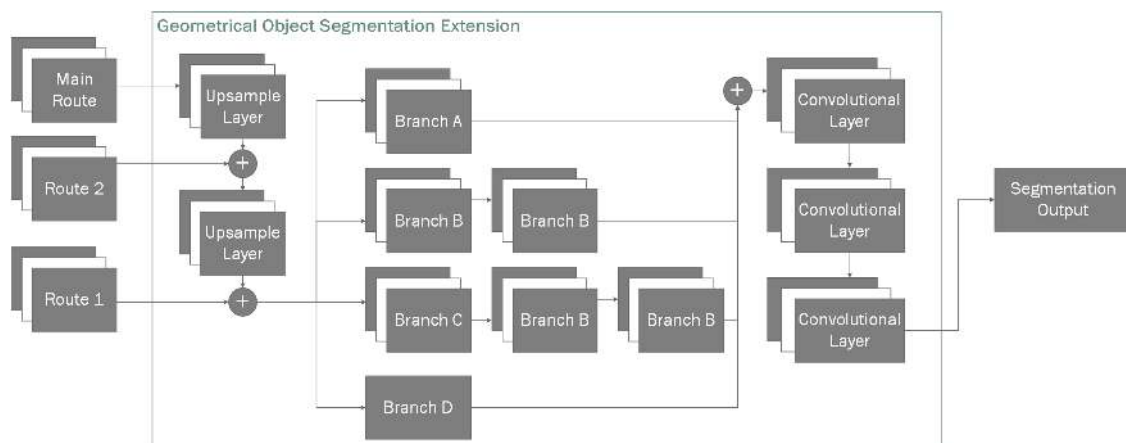

Figure 2. Our proposed geometrical object segmentation extension.

Due to each of the nearby routes being separated by the power-of-two scale, we use transposed convolutional layer [49] to upscale them gradually and then and merge them into desired final shape matrix. We construct our classless geometric segmentation mask by firstly upscaling the *main route* output and merging it with *route 2*, and the resulting layer is then upscaled again and merged with the final *DarkNet* output (*route 1*) which provides us a layer containing latent information of all previous layers that are each specified in learning different sized objects.

Next, we branch out our resulting hidden nodes into four different layers. Each layer contains slightly different network configuration, allowing them to essentially vote on their influence in the final result by extracting different latent feature-maps from the previous layers (Table 1). The first three branches (*A, B, C*) are convolutional branches containing one, two and three convolutional layers, respectively. However, for our final branch (*D*) instead of the convolutional layer, we use a max pool layer to extract the most prominent features. We have selected this parallel stacked approach, because we found it to be more efficient in extracting the object masks than linearly stacked layers when training the segmentation layers independently from the entirety of a model. This decoupling of the segmentation task from the classification task when training gives the additional benefit of allowing us to use transfer learning, which has shown to have very good practical results [50].

Next, we run our concatenated branches through convolutional layers to extract the most viable features and normalize their output in the range of (0, 1) giving us the final segmentation image. In our case the final segmentation output is 80×60 due to it being more than sufficient to extract approximate depth masks as we do not require pixel perfect segment representations. Finally, we use cascading flood-fill (Algorithm 1) to classify the masks pixels-wise. This is done because we found the generated binary masks to be impervious to false positives and false negatives, unlike classification using bounding boxes which can have three types of errors: false positives, false negatives and misclassification. This allows us to remove false positive bounding box detections when they do not intersect the origin of the mask. In our testing set, best cascade parameters were $\epsilon = 0.9, \theta = 0.01$.

Table 1. Geometric Segmentation architecture.

	Type	Filters	Size	Output
	Transposed Convolution	1024	$2 \times 2 \times 2$	20×20
Main route	Concatenate	-	-	20×20
	Convolution	256	1×1	20×20
	Transposed Convolution	256	$2 \times 2 \times 2$	40×40
Route 2	Concatenate	-	-	40×40
	Convolution	256	1×1	40×40
	Upscale	-	-	160×120
Branch A	Convolution	128	$1 \times 1/2$	80×60
Branch B	Convolution	32	1×1	160×120
	Convolution	128	$1 \times 1/2$	80×60
	Convolution	32	1×1	160×120
Branch C	Convolution	128	2×2	160×120
	Convolution	256	$3 \times 3/2$	80×60
Branch D	Max Pool	256	$3 \times 3/2$	80×60
	Concatenate	-	-	80×60
	Convolution	256	1×1	80×60
	Convolution	128	1×1	80×60
	Convolution	1	1×1	80×60
	Clip Values	-	-	80×60

Algorithm 1 Cascading flood-fill

1: **procedure** GET_SEED($box, mask, \epsilon$)　　　　　　　　　　　　　　▷ Seeds initial values.

2:　　$cx, cy \leftarrow box$　　　　　　　　　　　　　　　　　　　　　▷ Get center for *box*.

3:　　$seed \leftarrow \varnothing$

4:　　$seed \leftarrow find\_closest\_max(box, mask)$　　　　▷ Find closest max pixel within bounds.

5:　　**if** $seed \neq \varnothing \wedge seed_{value} \geq \epsilon$ **then**

6:　　　　$seed_{id} \leftarrow box_{id}$　　　　　　　　　　　　　　▷ Set seed *id* to box *id*

7:　　　　**return** $seed$　　　　　　　　　　　　　　▷ Return seed if value greater than ϵ

8:　　**end if**

9:　　**return** $\varnothing$　　　　　　　　　　　　　　　　　　▷ No valid seed was found.

10: **end procedure**

11: **procedure** FILL_NEIGHBOURS($seed, \theta$)　▷ Recursively fill free neightbours with same or lower

　　　values.

12:　　**for each** $n \in seed_{neightbours}$ **do**　　　　　　　▷ For every neighboring mask pixel.

13:　　　　**if** $n_{id} = \varnothing \wedge n_{value} \leq seed_{value} \wedge n_{value} > \theta$ **then**

14:　　　　　$n_{id} \leftarrow seed_{id}$　　　　　　　　　　　　　▷ Set neighbor to same *id* as *seed*.

15:　　　　　$FILL\_NEIGHBOURS(n)$　　　　　　　　　　　　▷ Call recursively.

16:　　　　**end if**

17:　　**end for**

18: **end procedure**

19: $bounding\_boxes \leftarrow sort\_confidence(bounding\_boxes)$　　　▷ Sort bounding boxes by confidence.

20: **for each** $box \in bounding\_boxes$ **do**　　　　　　　　　▷ For each bounding box b

21:　　$seed \leftarrow GET\_SEED(box, \epsilon)$

22:　　**if** $seed \neq \varnothing$ **then**

23:　　　　$FILL\_NEIGHBOURS(seed, \theta)$

24:　　**end if**

25: **end for**

Additionally, we have also modified *YOLOv3* network for we had issues with the network being unable to train by consistently falling into local minima during gradient descent and getting perpetually stuck in them. To solve this issue we introduced periodic hyper parameters [51] during model learning. Specifically, we had changed the learning rate to alternate in specified range of $lr_{min} = 1e^{-6}, lr_{max} = 1e^{-4}$.

$$y(x) = \begin{cases} \frac{x}{w_1} \times (lr_{max} - lr_{min}) + lr_{min}, & if\, x < w_0 \\ \frac{e^{1 + \pi \times \frac{\cos{(x - w_1)mod(w_0 + 1)}}{w_0}}}{\pi e^3} \times (lr_{max} - lr_{min}) + lr_{min}, & otherwise \end{cases} \tag{1}$$

This periodical learning rate (Equation (1)) has vastly improved our models ability to learn the underlying relationships of input date by alternating between low and high training rates, therefore jumping out of potential local minima that it might start orbiting around during stochastic gradient descent. Our function has two stages, the first stage that consists of two training iterations, where $w_1 = 2 \times s$, and the second stage of 4 iterations, where $w_0 = 4 \times s$ where s is the number of steps per batch. We selected the two state learning function because having high learning rates initially may cause the model to diverge. Therefore, during the first stage we linearly increase the learning rate. Once in the second stage we use the cosine function and the modulus operator for the model to alternate between two values. The shape of the alternating function also can have influence in model convergence as some models require to be in different extremum points for different amounts of times. Therefore, having a different dataset may require more fine-tuning of parameters of this equation for different slope shapes, while still maintaining the benefits of having alternating learning rates.

Additionally, as we are training the NN from scratch, we have noticed that our network, despite being able to find better convergence results due to periodical learning rate jumping out of local minima, had a high bias rate. A high bias rate is an indicator that our model is over-fitting on our data set. To solve this additional issue, we modified the *YOLOv3* network by adding additional dropout layers with the dropout rate of $P(x) = 0.5$ after each branch of *DarkNet53* and before each of the final layers predicting the bounding boxes.

Furthermore, we had issues of model overfitting to the training set, to solve this we additionally modified the neural network by adding two additional dropout layers. We trained our model 6 times, each with 50 iterations using mini-batch of size 8 for comparison, because after about 50 iterations the standard *YOLOv3* model starts to overfit and loose precision with our validation dataset. Therefore, for most objective comparison we trained our modified network for same number of epochs. Note that even though our method also starts to overfit, unlike the YOLOv3 network model, the accuracy of our modified model when overfitting remains roughly at the same value from which we can deduce that the changes make the model more stable.

Figure 3 shows the differences in loss function when trained using the RGB, RGB-D and Depth data as input. For the unmodified *YOLOv3* we are using $lr = 1e^{-5}$ as the midpoint between our minimum and maximum learning rates in the periodic learning rate function. As we can see from the graph, the loss function using static learning rate on the RGB and RGB-D datasets reaches a local minimum causing the model to slow down its ability to learn new features, unlike our periodic learning rate which seems to temporarily force the model to overshoot its target which sometimes causes it to fall into a better local minimum. This effect can be seen in the distinct peaks and valleys in the graphs. The outlier in these graphs are depth-only data points. While in both cases the loss function seems lower and has a better downwards trajectory in stochastic descent, however, we have noticed that despite seemingly lower loss when compared to RGB and RGB-D, the actual model accuracy is very unstable on epoch-per-epoch basis. We assert that this is the case due to depth alone providing very unstable data that's very hard to interpret. We make this assumption due to the fact that even when taken an expert to evaluate the depth maps alone, it is usually very hard to discern what type of object it is without knowing its texture; it is only possible to tell that there is in fact an object in the frame.

Finally, we can see that the RGB-D data is a clear winner when training in both cases, which means that depth data can indeed help in model generalization.
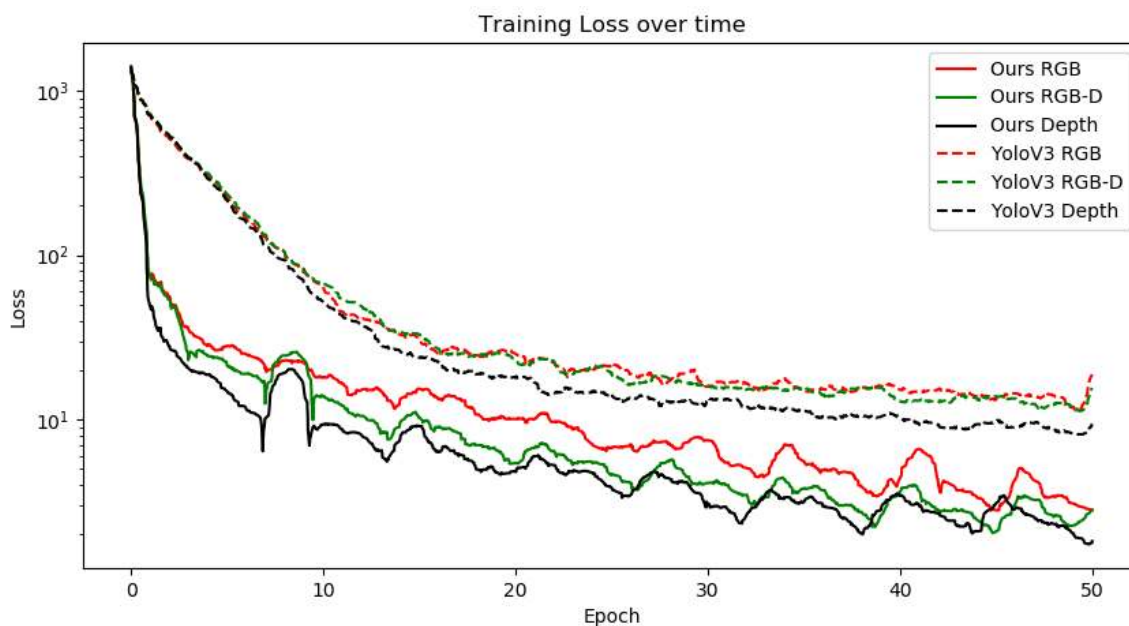

Figure 3. Training loss comparison between baseline *YOLOv3* and our modified version when using RGB, RGB-D and depth data as training. Due to the loss function being inherently noisy for each of the mini-batches, we have used Savitzky-Golay [52] digital filter to perform the smoothing of the overall graph.

2.3. Reconstruction Algorithm

The proposed algorithm for 3D object reconstruction consists of two subsystems: voxel cloud reconstruction and post-processing (Figure 4). In reconstruction step we take the outputs of the 3D classifier mask for the object and in conjunction with the original depth map which we feed into our reconstruction ANN (Figure 5) that performs the object reconstruction task for the given masked input frame. Unlike the classification algorithm we only use the underlying depth input from the classifier as it provides enough information for the specific object reconstruction. This is due to fact that we already know the class of the object, which is required for classification because different objects can have very similar depth representations. However, during reconstruction this is not an issue because our ANN is designed in such a way that each branch is responsible for reconstructing similar object representations.

Once the classifier-segmentation branch has finished its task, for each object instance the appropriately trained reconstruction branch is selected. In our case all the branches are highly specialized on a single type of object that it can reconstruct, which is why object classification is required. However, we believe that there is no roadblock to having more generic object reconstruction branches for example all similar objects may be grouped to a single reconstruction task. This could potentially allow some simplifications in the classification-segmentation as it would no longer be required to classify highly specific object instances thus reducing failure rate caused by object similarities. For example, a cup and a basket can be very similar objects and be misclassified. Additionally, the hybridization allows for fine tuning of the reconstruction branches without having to retrain the entire neural network model potentially losing already existing gradients via on-line training skewing the results towards new data posed. This in turn reduces re-training time if new data points are provided for a specific object as we no longer need to touch the established branches due to modularity.

Inside our reconstruction network branch (Figure 2) for given depth input we use convolutional layers to reduce the dimensionality of the input image during the encoding phase (see Table 2). For a

given input, we create a bottleneck convolution layer which extracts 96 features, afterwards we use a spatial 2D dropout [53] layer before each with $P(x) = 0.1$ to improve generalization. We use spatial dropout as it is shown to improve generalization during training as it reduces the effect of nearby pixels being strongly correlated within the feature maps. Afterwards, we add an additional inception [54] layer (Figure 6) which we will use as a residual block [55] followed by another spatial dropout. Afterwards, we add two additional bottleneck residual layers, each followed by additional dropouts. With final convolution giving us final 256 features with the resolution of 20×15. Our final encoder layer is connected using a fully-connected layer to a variational autoencoder [56] containing 2 latent dimensions, as variational autoencoders have shown great capabilities in generative tasks. Finally, the sampling layer is connected to full-connected layer which is then unpacked into a $4 \times 4 \times 4$ matrix. We use the transposed three-dimensional convolutional layers in order to perform up-sampling. This is done twice, giving us 4 feature maps in $32 \times 32 \times 32$ voxel space. Up to this point we have used Linear Rectified Units [57] (ReLUs) for our activation function, however, for our final 3D convolutional layer we use a softmax function in order to normalize its outputs where each voxel contains two neurons. One neuron indicating the confidence of it being toggled on, the other neuron showing the confidence of the neuron being off. This switches the task from a regression task to a classification task, allowing us to use categorical cross entropy to measure the loss between the predicted value and our ground truth.
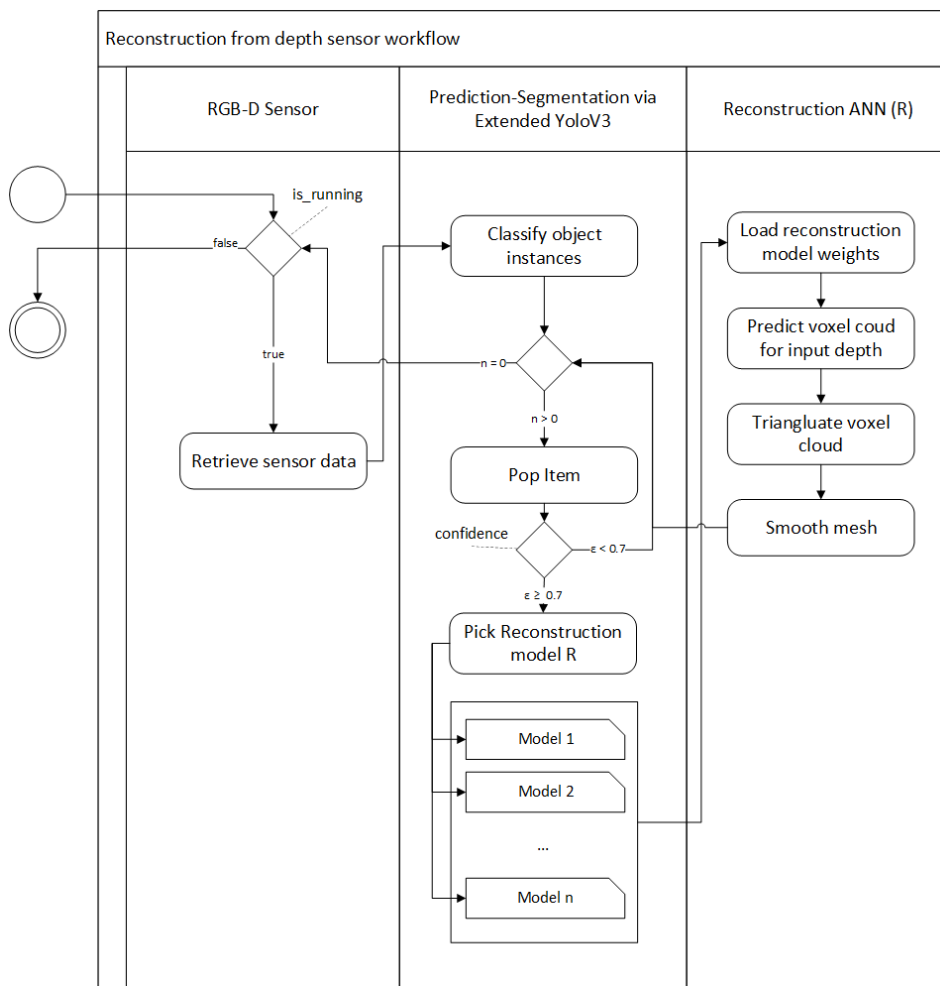

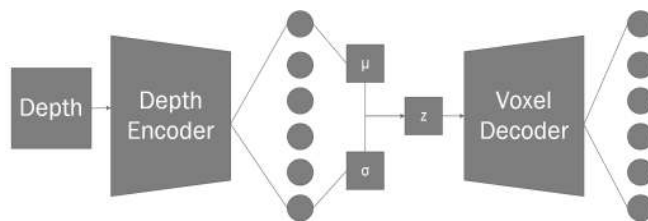**Figure 4.** Workflow of object reconstruction from sensor data.

Figure 5. Diagram of a single object reconstruction network architecture branch. For the given depth frame, the depth encoder creates a bottleneck, which is then directly connected to VAE node, the resulting sampler is connected into voxel decoder. The voxel decoder layer outputs a $32 \times 32 \times 32 \times 2$ matrix which can be explained as $x \times y \times z \times s$, where x, y, z components indicate position in 3D grid, and s component indicates voxel state encoded as one-hot.

Table 2. Architecture of the reconstruction neural network.

|	Type	Filters	Size	Output
	Input	-	-	320×240
Encoder	Convolution	96	$5 \times 5/2$	160×120
	Dropout 2D $P(x) = 0.1$	-		160×120
	Inception	(8, 4)	-	160×120
	Convolution	16	1×1	160×120
	Add	-	-	160×120
	Convolution	128	$5 \times 5/2$	80×60
	Dropout 2D $P(x) = 0.05$	-		80×60
	Inception	(8, 4)	-	80×60
	Inception	(16, 8)	-	80×60
	Convolution	32	1×1	80×60
	Add	-	-	80×60
	Convolution	128	$3 \times 3/2$	40×30
	Dropout 2D $P(x) = 0.025$	-		40×30
	Inception	(8, 4)	-	40×30
	Inception	(16, 8)	-	40×30
	Inception	(32, 16)	-	40×30
	Convolution	64	1×1	40×30
	Add	-	-	40×30
	Convolution	256	$3 \times 3/2$	20×20
VAE	Flatten	-	-	76 800
	Fully-Connected	-	-	512
	Mean	-	-	2
	Standard Deviation	-	-	2
	Sampling	-	-	2
Decoder	Fully-Connected	-	-	64
	Reshape	-	-	$4 \times 4 \times 4$
	Inception 3D	(32, 16)	-	$4 \times 4 \times 4$
	Inception 3D	(16, 8)	-	$4 \times 4 \times 4$
	Inception 3D	(8, 4)	-	$4 \times 4 \times 4$
	Convolution 3D	16	$1 \times 1 \times 1$	$4 \times 4 \times 4$
	Add	-	-	$4 \times 4 \times 4$
	Transposed Conv 3D	64	$3 \times 3 \times 3 \times 2$	$8 \times 8 \times 8$
	Inception 3D	(16, 8)	-	$8 \times 8 \times 8$
	Inception 3D	(8, 4)	-	$8 \times 8 \times 8$
	Convolution 3D	16	$1 \times 1 \times 1$	$8 \times 8 \times 8$
	Add	-	-	$8 \times 8 \times 8$
	Transposed Conv 3D	32	$3 \times 3 \times 3 \times 2$	$16 \times 16 \times 16$
	Inception 3D	(8, 4)	-	$16 \times 16 \times 16$
	Convolution 3D	16	$1 \times 1 \times 1$	$16 \times 16 \times 16$
	Add	-	-	$16 \times 16 \times 16$
	Transposed Conv 3D	4	$5 \times 5 \times 5 \times 2$	$32 \times 32 \times 32$
	Convolution 3D (Softmax)	2	$3 \times 3 \times 3$	$32 \times 32 \times 32$
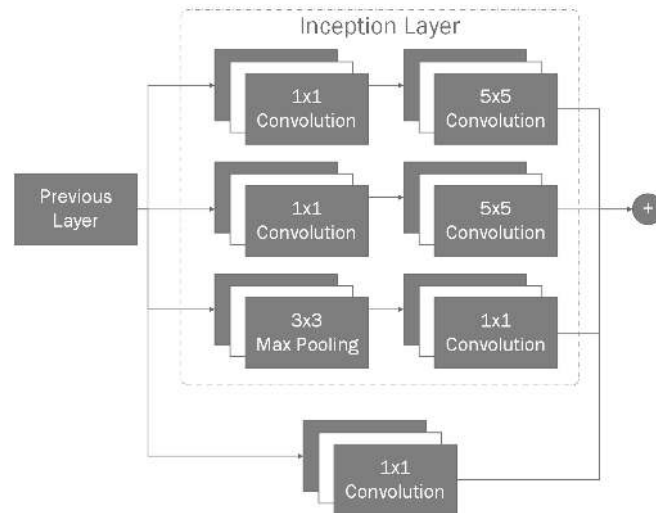
Figure 6. An example of the inception layer. An input layer is connected to three branches in parallel. If multiple inception layers are used inception layers are connected sequentially. Final inception layer outputs and 1×1 convolution are then connected using addition. The result is then used for subsequent layers.

2.4. Proposed Network vs. YOLOv3

Our approach is the hybridization of two ANN architectures: classification-segmentation branch and reconstruction branch (see Figure 7). The classification-segmentation branch as the name suggests performs object instance classification and segmentation. This information is then fed to the object reconstruction branches. Object reconstruction branch contains a fleet of specialized pre-trained autoencoder models where each of the auto-encoders can reconstruct the model's three-dimensional representation while being provided only a single depth frame. The initial classification-segmentation branch is our expanded interpretation of YOLOv3 which adds crucial output to already existing YOLOv3 network output, i.e., the object instance segmentation. This extension adds crucial information which is required for the reconstruction step by extracting the object instance mask that can be applied per each object on the initially captured depth.
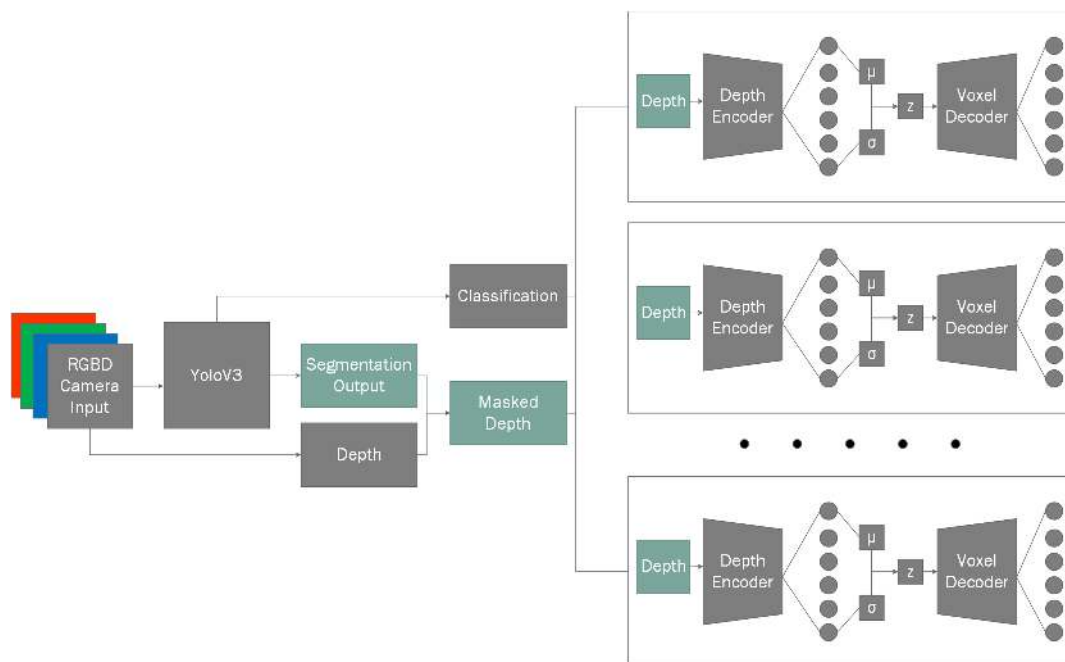

Figure 7. Full view of the proposed network model that extends the YOLOv3 network.

2.5. Dataset

As our method entails the requirement of a priori information for the captured object reconstruction, there is a need for a large well labeled element dataset. However, unlike for object recognition which has multiple datasets, e.g., *COCO* [58] dataset, *Pascal VOC* [59]; there seems to be a lack of any public datasets that provide RGB-D scene representation in addition to it's fully scanned point cloud information viable for our approach. While datasets like *ScanNet* [60] exist, they are missing finer object details due to focusing their scan on full room experience that we are trying to preserve. Therefore, our training data consists exclusively out of synthetically generated datasets, which use the *ShapeNetCore*, a subset of *ShapeNet* dataset that provides 3D object models spanning 55 categories (see an example of a coffee cup model in Figure 8). In addition, we use real-life data acquired by the *Intel Realsense ZR300* and *Intel Realsense D435i* (Intel Corp., Santa Clara, CA, USA) devices for visual validation as it is impossible to measure it objectively without having a 3D artist recreating a 1:1 replica of said objects, which is unfortunately unfeasible option. However, using real world samples as a validation set is not subject to training bias because they are never being use in the training process.

As mentioned, for the training of the black-box model we are using the *ShapeNetCore* dataset that we prepare using *Blender* [61] in order to create the appropriate datasets. Due to the fact that we are training a hybrid neural network, we need two separate training and testing sets, one for each task.

Figure 8. A coffee cup model from the *ShapeNetCore* dataset.

2.5.1. Classification Dataset

To create this subset of data we create random scenes by performing the following procedure. Firstly, we randomly decide how many objects we want to have in the scene in the range of $n_{objects} = [1; 10)$ and pick that many random objects from *ShapeNetCore* dataset to populate the scene. Before applying any external transformations we transform the object geometry so that all objects are of uniform scale and have the same pivot point. To perform the required transformations firstly we calculate the geometry extents. Once we know the object extents we can move all the objects on *Up* axis (in our case this is *z*) and scale down all vertices by the largest axis (Algorithm 2). This gives us a uniformly scaled normalized geometry that we can freely use.

Algorithm 2 Normalize geometry

1: **procedure** EXTENTS(*G*) ▷ Calculates extents for geometry *G*
2:     $min_x, min_y, min_z \leftarrow Infinity$ ▷ Initialize *min* vector
3:     $max_x, max_y, max_z \leftarrow -Infinity$ ▷ Initialize *max* vector
4:     **for each** $v \in G$ **do** ▷ For each vertex *v*
5:        $min_x \leftarrow min(v_x, min_x)$
6:        $min_y \leftarrow min(v_y, min_y)$
7:        $min_z \leftarrow min(v_z, min_z)$
8:        $max_x \leftarrow max(v_x, max_x)$
9:        $max_y \leftarrow max(v_y, max_y)$
10:        $max_z \leftarrow max(v_z, max_z)$
11:     **end for**
12:     **return** min, max
13: **end procedure**
14: $min, max \leftarrow EXTENTS(G)$
15: $bounds \leftarrow max - min$
16: $max\_bound \leftarrow 1/max(bounds_x, bounds_y, bounds_z)$
17: **for each** $v \in G$ **do** ▷ For each vertex *v*
18:     $v_x \leftarrow v_x/max\_bound$
19:     $v_y \leftarrow v_y/max\_bound$
20:     $v_z \leftarrow (v_z - min_z)/max\_bound$ ▷ Offset the vertex on *up* axis before normalizing bounds
21: **end for**

We place the selected objects with random transformation matrices in the scene, making sure sure that the objects would never overlap in space. To generate random local transformation matrix (*L*) (Equation (3)) we need three of it's components: Scale (*S*), Rotation (*R_z*) and with random value; use either capital or lower-case s in both places in the range of $s = [0.7, 2)$; Rotation (*R_z*), where rotation is random value in the range of $\theta = [0, 2\pi)$, we perform rotation only on *z* axis to ensure that randomly generated scenes are realistic and do not require artist intervention; Translation (*T*), where *x* and *y* values are non-intersecting values in the range of $r = [-5, 5]$ and $\alpha = [0, 2\pi)$ (Equation (2)).

$$\begin{cases} x = r \times \cos \alpha \\ y = r \times \sin \alpha \end{cases} \tag{2}$$

$$L = S \times R \times T = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ x & y & z & 1 \end{bmatrix} \tag{3}$$

Once the selection objects are placed we need to apply lighting in order to have real-life like environments. To do this, we use the Lambertian shading model and directional lights. We randomly generate $n_{lights} = [1; 4)$ lights in the scene. We pick a random light rotation, we ignore translation as it does not matter in directional lights; we generate a random color in the range of $Col_{RGB} = [0.7, 1]$, we selected the minimum bound of 0.7 to avoid unrealistic real-world lightning; and random intensity $I = [0.7, 1]$. This light acts as our key light. To avoid hard shadows being created, which wouldn't be the case unless using spotlight in real world, for each key light we create a backlight which is pointing the opposite direction of key light with half the intensity and identical color to the key light.

Once the scene setup is complete, we render the scene in three modes: *color*, *depth* and *mask*. Color mode gives us the scene representation from a regular light spectrum camera. As we are not putting any background objects into the scene the generated background is black. However, later on we use augmentation during training to switch the backgrounds to improve recall rates. Once the color

frame is extracted we extract the mask, in order to extract the mask we assign each object an incremental *ID* starting at 1, this allows us to differentiate between objects in the frame. Finally, we render the depth representation of the scene. Before rendering depth we place a plane on the ground that acts as our ground place, this allows for more realistic depth representations because the objects are no longer *floating* in space. The depth is rendered front-to-back, meaning the closer the object is to the camera the closer to zero depth value is, the front-to-back model was chosen because this is the same as *Intel Realsense* model.

Each of the scenes is rendered in *320 × 240* resolution $n = 25$ times by placing it in random locations (Algorithm 3) and pointing it at the center of the scene, where $r = 10$, $z_{min} = 4$, $z_{max} = 6$.

Algorithm 3 Camera location

1: $step\_size \leftarrow 2\pi/(1-n)$
2: **for** i < n **do**
3:     $\theta \leftarrow random(i, i+1)$ ▷ Random float in the range of [i, i+1]
4:     $x \leftarrow \cos(step\_size \times \theta) \times r$
5:     $y \leftarrow \sin(step\_size \times \theta) \times r$
6:     $z \leftarrow random(z_{min}, z_{max})$
7: **end for**

We save the perspectives as *OpenEXR* format [62] instead of traditional image formats instead of, for example, *PNG*, as *OpenEXR* file format is linear, allowing for retention of all depth range without any loss of information as it is not limited to 32 bits per pixel. The final *EXR* file has these channels in it *R*, *G*, *B* containing red, green and blue color information respectively; *id* channel contains the information about the mask for specific pixel; *Z* information containing the linear depth data.

Once we create the input image, we additionally label the data and extract the segmentation mask that will be used as output when training the artificial neural net. We perform this step after the scene is rendered in order to account for any kind of occlusion that may occur when objects are in front of each other causing them to overlap. We extract the object bounding boxes by finding the most top-left and bottom-right pixel of the mask. The binary mask is extracted based on the pixel square distance from the center of the bounding box. This means that the center pixels for the bounding box are completely white and the closer to the edges it is the darker it gets. We use non-flat segmentation to be able to extrapolate individual object instances in the mask when they overlap, and this is done by interpolating the pixel intensity from the bounding box edge to bounding box center. The mask is then scaled down to *80 × 60* resolution as it is generally sufficient and reduces the required resources.

2.5.2. Anchor Selection

The existing anchors that are being used with *COCO*, *Pascal VOC* and other datasets are not suitable for our dataset, rarely fitting into them. Therefore, we performed class data analysis and selected three most fitting anchors per classifier branch scale. As we can see from Figure 9, our classes generally tend to be biased towards 1:1 aspect ratio due to data set being randomly generated unlike in real world applications.

However, while the classes tend to be biased towards 1:1 for the most part, the assertion that all individual object instances would neatly fit into this aspect ratio would be incorrect as they still retain certain bias. According to previous Single Shot Detection (SSD) research [63], selecting inadequate base anchor boxes can negatively affect the training process and cause the network to overfit. Therefore, we chose to have 3 anchors per anchor size as this seems to sufficiently cover the entire bounding box scale spread by including tall, wide and rectangle objects. We select the anchor points using *K-Means* to split data into 9 distinct groups (Figure 10).

Once we have our cluster points for bounding box detections, we sort them in order to group into small, medium and large anchor sets. Giving us three different anchors, each having the most popular aspect ratios per that scale detection branch as it can be seen in Table 3.
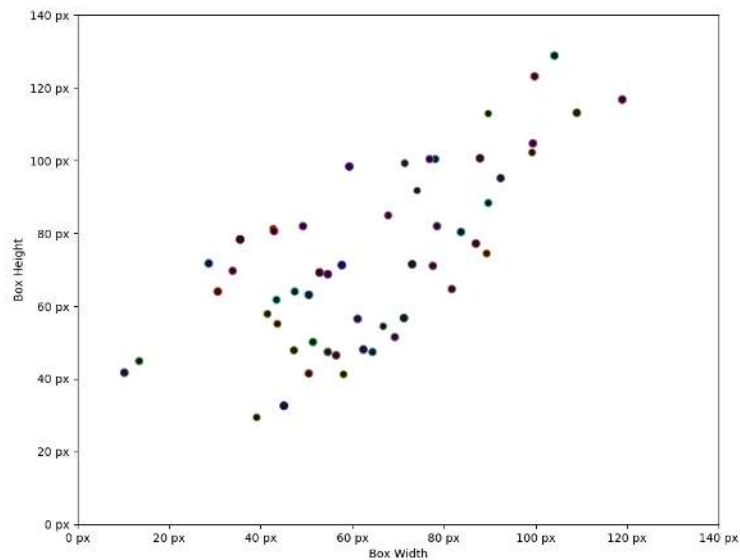

Figure 9. Each individual point denotes the mean object bounding box scale for each class type.
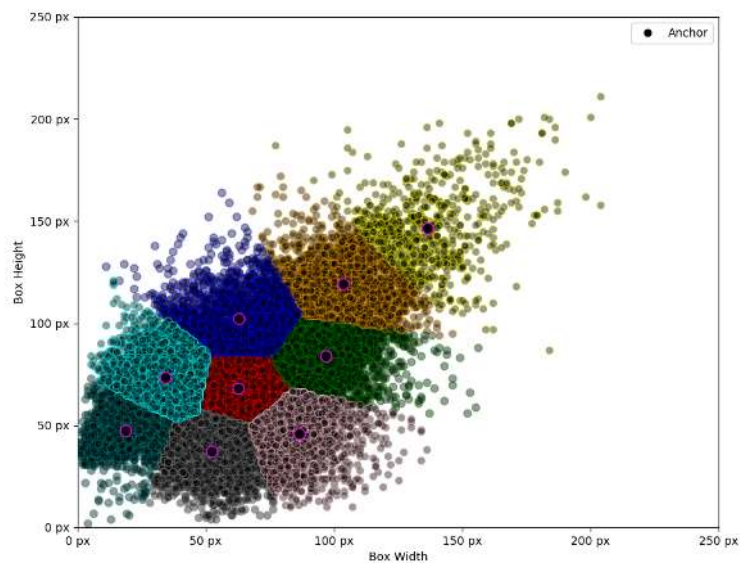

Figure 10. Selected anchors using *K-Means* clustering algorithm. Different colors denote distinct anchor groups responsible for detecting objects in the spread.

Table 3. Anchor scales in pixels calculated using the *K-Means* clustering method.

Anchor Type	Anchor 1	Anchor 2	Anchor 3
Small	18.83, 47.53	52.34, 37.53	34.13, 73.28
Medium	86.35, 46.02	62.74, 68.31	62.75, 102.19
Large	96.69, 84.20	103.66, 119.51	136.34, 146.64

The neural network architecture described in Section 2.2 was trained in three separate modes in order to infer how much the additional depth information improves the classification results.

These three modes consist of RGB, RGB-D and Depth training modes. Where RGB mode implies we train using only the color information that was generated from the dataset, the RGB-D mode uses both depth and color information and finally Depth mode trains the network using only depth information. We do not use any additional data augmentation when training in both RGB and RGB-D modes. We do however, add additional augmentation when training in the RGB-D mode. When training in the RGB-D mode there is a small chance that either RGB or Depth channel will not be included in the testing sample. We perform this augmentation because both RGB camera and Depth sensors may potentially have invalid frames. Therefore, we assert that both of these data points are equally necessary for the classification task, and that they must be generalized separately from each other and should provide equal contributions to the classification task. This is decided randomly when preparing the mini-batch to be sent to the artificial neural network for training. There is $\lambda = 0.1$ chance that the input specific data point will be picked for additional augmentation. If the data point is picked for augmentation then there is equal probability that either RGB or Depth Data will be erased from the input and replaced with zeros. We decided on this augmentation approach because both RGB and Depth frames using real sensors are prone to errors. For example, the RGB camera may fail in bad lighting or even be unavailable when the room is pitch black. Likewise, the depth frames are also prone to errors due to inconsistencies in generating depth map which causes the sensor to create speckling effect in the depth information, additionally cameras being too close to object may be completely unable to extract proper depth information. Therefore, we chose this augmentation approach as it allows for the sensors to work in tandem when both are available, but fill in the gaps, when one of them is failing to provide an accurate information.

2.5.3. Reconstruction Dataset

For the reconstruction training set, we use the same *ShapeNetCore* dataset to generate the corresponding depth images and ground truths for the individual objects voxel cloud. We used *Blender* to generate the training data. However, the generated input data is different. We assert that the object material does not influence the objects shape, therefore we no longer generate the color map unlike when generating classification data. Therefore, we only render the depth information for each object. We render individual objects by placing the cameras in such a way that the specific object would be visible from all angles from 45° to 90° at a distance from 1 to 1.5 m, excluding the bottom. As a result we have 48 perspectives for each of the object models. Once again we save the models as *OpenEXR* file in order to preserve the depth values in this lossless format. Finally, we generate the voxel-cloud representation [64]. Voxelization is performed by partitioning into the equally sized cells, where the cell size is selected based on the largest object dimension axis. Following the space partitioning, we repeat over each of the cells and compute whether the specific cell should be filled by ray-polygon intersection [65].

2.6. Evaluation

In order to evaluate the correctness of our results, we evaluate the results of the proposed algorithm, and additionally we evaluate both of the subsystems individually. To evaluate the classification accuracy, we use the *mAP* metrics to assess the quality of the classifier and it's output bounding boxes. When performing the classification accuracy evaluation, we evaluate all three train models: RGB, RGB-D and Depth. This allows us to determine the quality differences between the addition of depth information in the classification task.

For the reconstruction task we require the output voxel representation of the object to be as close to ground truth as possible. For that, we define our reconstruction quality as the *Intersection-over-Union* metric. Furthermore, we use the *Correctness*, *Completeness*, and *Quality* metrics during evaluation.

3. Results

3.1. Settings

Our experiments have been executed using two computers: (1) a workstation with *Intel i7-4790* CPU with *16 GB of RAM* which achieved 55.76 fps, and *nVidia 1070* graphics card with *8 GB GDDR5 VRAM*; and (2) a laptop computer using *nVidia 960M* graphics chip with *4GB GDDR5 VRAM*, *Intel i5-4210H* CPU and *12 GB of RAM*, which reached 11.503 fps. We consider that these machines should represent the target range of end user devices.

3.2. Quantitative Results

3.2.1. Object Instance Classification Results

In order to evaluate our model in all cases, we have used the mAP metric, which is a widely used method in order to evaluate mean average precision of the predicted bounding boxes with respect to their Intersection-over-Union (IoU), provided that the object classes match. As per suggested *COCO* evaluation we filter out bounding boxes which have an $IoU < 0.5$ in order to compare all of our trained model versions.

As Table 4 suggests, our iterative training approach in addition to dropout layer was substantially better in the object classification task as opposed to the originally suggested variant which would either plateau with too low of a learning rate or get stuck in a constant loop around the local minima due to the initial learning rate being too high. Therefore, we can assert that a periodic learning rate is a useful tool to improve model generalization and the speed at which the network can train by adding additional noise during training time in a form of sudden overshooting. Furthermore, we can see that the addition of depth information as input greatly increases the recall rate in both cases, while the depth information alone has similar recall rate in both cases. This suggests that the depth cameras can not only greatly benefit in the object classification task when used in conjunction with visible light spectrum cameras but it can be used as a fallback when no light source is available, albeit with lower precision.

Table 4. Mean precision values in respect to $IoU > 0.5$ for each of our trained models.

Network Type	mAP (%)
Our RGB-D	60.20%
YoloV3 RGB-D	55.75%
Our RGB	41.27%
YoloV3 RGB	37.96%
Our Depth	26.46%
YoloV3 Depth	20.87%

One of the glaring issues we noticed with the *ShapeNetCore* dataset during our experiments is that, while there are specified a total of 55 classes, a lot of those classes have major overlap in form and function which may dramatically affect the overall mAP value, such as classes that are categorized as distinct (e.g., *pistol* and *handgun*) could still be grouped into the same class as they share key characteristics which may not be viable to differentiate when using relatively low resolution images. Additionally, some groups of objects can be distinct in their use (e.g., *mug* and *ashcan* and *basket*) in many cases have no differentiable features and would require each individual scene to be hand crafted by an artist in order to provide visual queues about the objects in relation to the world, which should potentially allow for differentiation between very similar objects (Figure 11). However, this is currently beyond the scope of our paper.
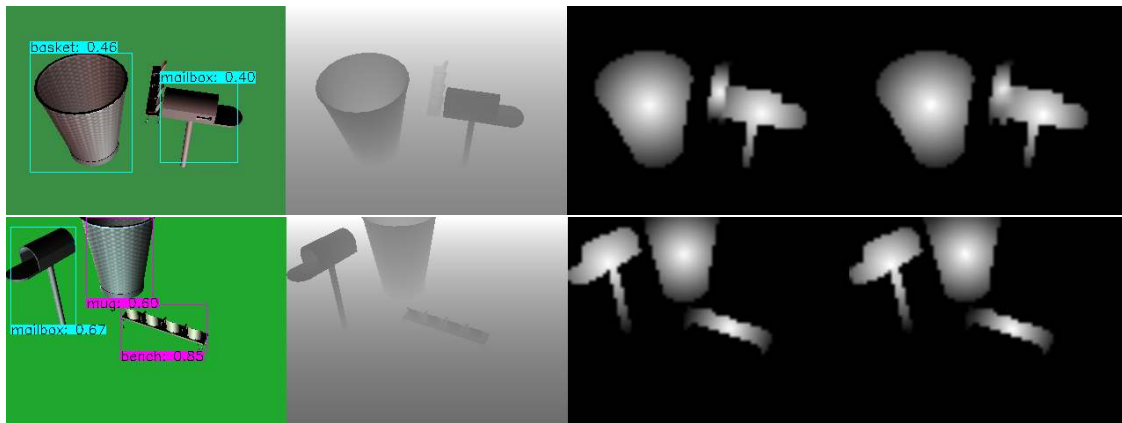
Figure 11. Prediction made with our extended *YOLOv3* network. Left to right: (1) Input color image with predicted object instances; (2) Input depth frame; (3) Upscaled to 320 × 240 ground truth mask; (4) Predicted mask upscaled to 320 × 240. Same object is being treated as two distinct classes due to lack of cues for the artificial neural network of what the specific object may be due to scenes being generated randomly.

3.2.2. Mask Prediction Results

As one of our main goals is to extract individual object instances from the depth map, we extended the *YOLOv3* network architecture to be able to predict object masks. In order to compare the predicted mask similarity with the ground truth we use the structural similarity index metric (SSIM) that measures perceived similarity between two images.

As we can see from Figure 12, in all cases our *YOLOv3* extension for object mask prediction is capable of extracting mask frame not only from the combined RGB-D frames but also from the RGB and Depth frames alone. This shows us that both color and depth information individually is generally enough for this task. However, both of these sensors may fail in different environments so the conjunction of both would most likely procure the most accurate results. Additionally, while in both method cases (static and periodical) the similarity is generally more than enough to extract accurate mask, using periodical approach provides a much lower standard deviation, hence better expected results. Additionally, the higher similarity also signals a tighter mask which may improve reconstruction quality due to reduction in bad data. While in our tests RGB has slight advantage over RGB-D when generating a mask, it is worth noting that Depth adds an additional dimension to the data which makes the dataset slightly harder when compared to RGB alone. This is due to RGB alone being able to drop the randomly generated background, unlike RGB-D which has a non-uniform background due to addition of ground plane. As we can see in Figure 13, our approach is applicable not only for synthetic but for real-world data too. This indicates that the network managed to generalize well and it's result can be used during reconstruction step.

3.3. Reconstruction Results

3.3.1. Quantitative Results

We can observe the achieved results for our proposed method in Figure 14 as they compare to previously achieved results in hybrid neural-network reconstruction [66]. As we can see the mean *IoU* metric value as compared to the results presented in [66] has significantly improved for some of the models, more importantly—even if the the improvement was minimal or if the results were slightly lower the error spread is lower. This indicates that the achieved results are much more stable. Additionally we can see that our reconstruction results are comparable to that of other state-of-art methods like *3D-R2N2* reporting 0.571 mean *IoU*.
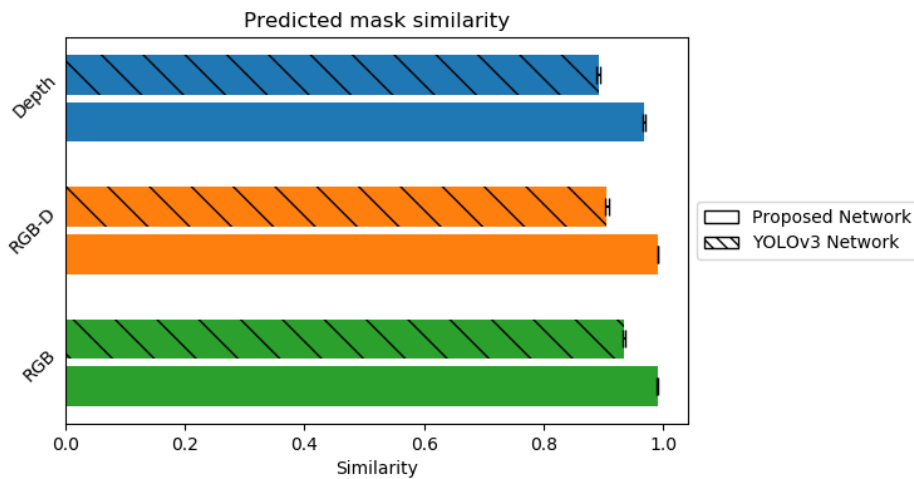
Figure 12. Similarity of created mask to the mask of ground truth. The hashed bar denotes the similarity of masks predicted by the *YOLOv3* network, the solid bar denotes the similarity of masks predicted for Depth, RGB-D and RGB frames by the network model proposed in this paper.

Figure 13. An example of real world object classification using the proposed network model: segmented and classified RGB frame (**left**), depth frame (**middle**), and predicted depth mask (**right**).
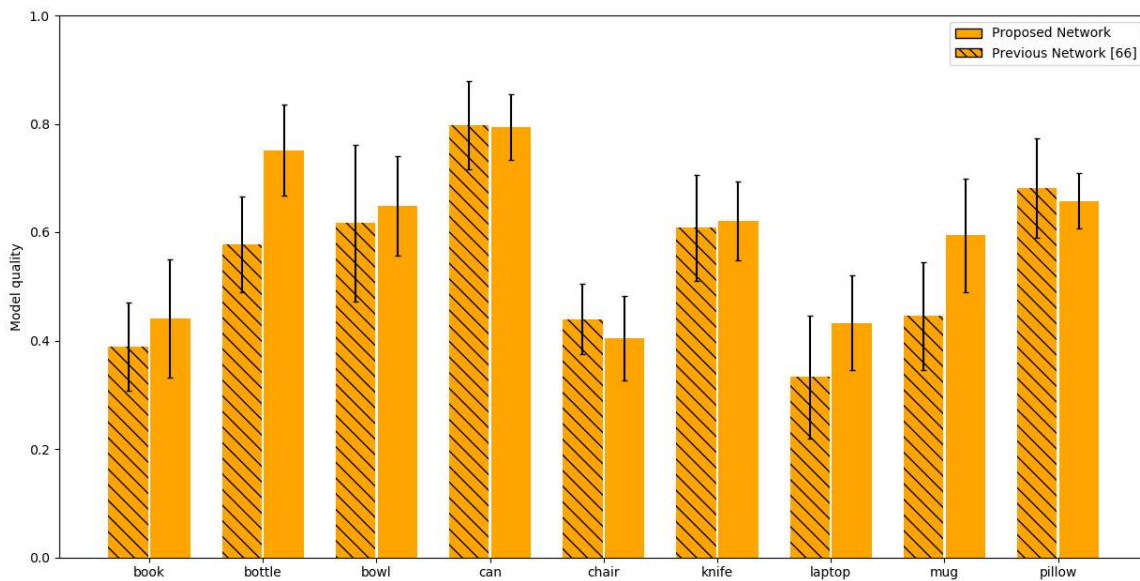

Figure 14. Comparison between the predicted object shape and ground truth using the IoU metric for different objects in the training set. The hashed bars denote the results achieved using the network proposed in [66]. The solid bars denote the results for the proposed network.

3.3.2. Visual Inspection and Result Validation

For every object that we have trained, we had collected real world examples using *Intel Realsense* device in order to compare how well synthetic results transfer into real world data. The results for the given dataset can be seen in the Table 5.

Table 5. Visual evaluation of object reconstruction. Table presents: RGB frame, original depth frame; reconstructed cloud of voxels; triangulated and smoothed surface created using predicted voxel cloud; and a corresponding similar object in the training set.

The reconstructed object shapes are generally recognizable. However, certain object angles cause the network to fail the task, for example, one of the bowls is missing half of it's voxels, while the other bowl may be considered a near perfect reconstruction. While the ANN has managed to reconstruct the *Book* and *Knife* datasets, it has generally only managed to reconstruct their base shape primitives which make the objects somewhat indistinguishable by experts without any prior information of what the objects may be. While the human bias may notice the minute structural differences between the knife handle and blade in terms of it's width, we still consider this a failed reconstruction. *Can* has managed to achieve great results in terms of reconstruction, while the pillow reconstruction could be considered near perfect. *Mug* in our training set is one of the trickiest objects as it contains a handle which should be reconstructed with a hole and additionally the mug cannot be fully filled in with voxels as in our case it is empty. While in all three cases the basic shape of the cup was maintained, there are some issues with two test cases. One of the test cases was missing a hole for the handle, while another is substantially distorted. However, the distortions may be explained by extremely noisy dataset. The *Chair* dataset allowed to reconstruct the shape of the chair although some of the details were missing. The *Laptop* and *Bottle* datasets are the hardest ones in terms of depth sensor capabilities. Depth sensor has issues in retrieving depth information for IR reflective surfaces causing it to distort the images fully. Such surfaces in our case are computer screen and a plastic bottle. However, the *laptop* data has surprisingly managed to account for this error in depth map, albeit containing some distortions.

3.3.3. Reconstruction of Multiple Objects

As a proof of concept, we have performed the experiments to reconstruct multiple objects in the scene (see an example of results in Figure 15). By extracting the individual object masks and performing an object reconstruction individually we have managed to reconstruct the specific objects in the scene. However, we are unable to predict the object's relative position, rotation and scale in relation to camera space. For this reason, we have had to specify the object transformation in relation to camera and other scene objects manually to perform final scene render.

Figure 15. An example of reconstruction of multiple objects in the scene: segmented and classified RGB frame (**left**), depth frame (**middle**), and predicted depth mask (**right**).

4. Discussion and Concluding Remarks

4.1. Discussion

One of the main advantages of our suggested hybrid NN based method is that unlike other non-hybrid approaches, it is relatively easy to include additional objects into the dataset, due to the fact that you can train network branches separately. Unlike other approaches, we do not need to re-train the model with all the previous data as we do not risk losing any of the existing gradients due to network being skewed to the new data points. The modularity of the approach allows us to train the network reconstruction nodes per each object category independently. Additionally, this modularity allows for variance of the model per object class, meaning we can adjust complexity of the ANN depending on the difficulty of the object that is being reconstructed. Furthermore, we believe that our approach is a step forward to generic object reconstruction as we are capable of extracting multiple

objects from the same scene thanks to masking during classification step, which allows to send only the relevant objects depth information into the reconstruction node.

While our approach is capable of extracting the individual object instances and reconstructing them, additional research is required for full scene reconstruction. This feat requires finding the camera space matrices, paving the way for application in Extended Reality systems. One of the standing issues with our current approach in terms of reconstruction is that our ground-truths are perspective-invariant. This makes training the network slightly harder, additionally it may somewhat reduce the quality of the results due to network somewhat trying to adjust to observation angle, therefore making the IoU metric values lower, despite visually being feasible. Solving the perspective invariance may also be a partial solution to the homography [67,68] problem as our reconstructed object would already be rotated with respect to the camera space.

Additionally, the improvements on the dataset may be obtained by creating and incorporating a real-world dataset along with synthetic data for the depth encoding step. Thus, we can potentially improved results when using real depth sensors. Additional improvements to the network architecture may also be found by changing the complexity of the model [69]; pruning dead neurons [70]; using neuro-evolutionary and neuro-genetic algorithms to find a much more fitting solution [71]; enhancing the learning of the artificial neural networks by using metaheuristic control mechanism [72]; or using multiple frames from a video feed instead of the current single frame solution as a large number of depth frames from a single view may reveal some hidden features and improve recall rate [73]. Using multiple frames would allow for exploration of what improvements may be achieved with the use of recurrent neural networks (RNN) for they have shown to be capable of predicting sequential data [74–76]. Finally, using the RGB frames combined with depth frames for reconstruction can potentially add some missing features from the depth due to inherent noisiness of the sensor, therefore improving the recall rate [77,78].

Finally, we have compared the complexity of the proposed network model with the YOLOv3 network as well as with other popular network architectures. The results presented in Table 6 how that the proposed network model is only sightly more complex than YOLOv3 in terms of the number of model parameters and operation, but outperforms other network architectures in terms of operations.

Table 6. Comparison of neural network complexity by the number of parameters, number of operations and model size.

Network Model	No. of Parameters	No. of Operations	Model Size (MB)
YOLOv3 [47]	61.81 M	294.86 M	946
Proposed (extended YOLOv3)	67.45 M	305.61 M	1010
AlexNet [79]	60 M	16.04 G	217
GoogleNet [80]	7 M	16.04 G	40
ResNet152 [81]	60 M	11.3 G	230
VGC16 [82]	138 M	154.7 G	512.24
NIN [83]	7.6 M	11.06 G	29
SimpleNet [84]	5.4 M	652 M	20

4.2. Concluding Remarks

Our proposed hybrid artificial neural network modifications have allowed to improve the reconstruction results with respect to theYOLOv3 network results by 8.53% which allows for much more precise filling of occluded object sides and the reduction of noise during the process. Additionally, the reconstruction results are a lot more stable when compared to previous results. Furthermore, the addition of object segmentation masks and the individual object instance classification is a leap forward towards a general purpose scene reconstruction as opposed to single object reconstruction task due to the ability to mask out overlapping object instances and use only masked object area in the reconstruction process. While further research is needed in order to retrieve object orientation and

position with respect to camera space, we believe our method allows for a much broader application in comparison to previous research due to its focus on single object reconstruction.

Author Contributions: Conceptualization, R.M.; methodology, R.M.; software, A.K.; validation, A.K. and R.M.; formal analysis, R.M.; investigation, A.K., R.M. and R.D.; resources, A.K. and R.M.; data curation, A.K.; writing—original draft preparation, A.K. and R.M.; writing—review and editing, R.D. and E.S.L.H.; visualization, A.K. and R.M.; supervision, R.M. The individual contribution of the authors is as follows: 0.5, A.K.; 0.2, R.M.; 0.2, R.D.; 0.1, E.S.L.H. All authors have read and agreed to the published version of the manuscript.

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

References

1. Bitzidou, M.; Chrysostomou, D.; Gasteratos, A. Multi-camera 3D Object Reconstruction for Industrial Automation. In *IFIP Advances in Information and Communication Technology Advances in Production Management Systems. Competitive Manufacturing for Innovative Products and Services*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 526–533, doi:10.1007/978-3-642-40352-1_66. [CrossRef]

2. 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, doi:10.3390/mti2020020. [CrossRef]

3. Fast-Berglund, Å.; Gong, L.; Li, D. Testing and validating Extended Reality (xR) technologies in manufacturing. *Procedia Manuf.* **2018**, *25*, 31–38, doi:10.1016/j.promfg.2018.06.054. [CrossRef]

4. 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; pp. 84–90, doi:10.1109/icist.2018.8426125. [CrossRef]

5. Vaitkevičius, A.; Taroza, M.; Blažauskas, T.; Damaševičius, R.; Maskeliūnas, R.; Woźniak, M. Recognition of American Sign Language Gestures in a Virtual Reality Using Leap Motion. *Appl. Sci.* **2019**, *9*, 445, doi:10.3390/app9030445. [CrossRef]

6. Zhang, J.; Shum, H.P.H.; McCay, K.; Ho, E.S.L. Prior-less 3D Human Shape Reconstruction with an Earth Mover's Distance Informed CNN. In *Motion, Interaction and Games on-MIG19*; ACM Press: Newcastle upon Tyne, UK, 2019; pp. 1–2, doi:10.1145/3359566.3364694. [CrossRef]

7. 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, doi:10.3390/rs10020328. [CrossRef]

8. Połap, D.; Kęsik, K.; Książek, K.; Woźniak, M. Obstacle Detection as a Safety Alert in Augmented Reality Models by the Use of Deep Learning Techniques. *Sensors* **2017**, *17*, 2803, doi:10.3390/s17122803. [CrossRef] [PubMed]

9. Jusas, V.; Birvinskas, D.; Gahramanov, E. Methods and Tools of Digital Triage in Forensic Context: Survey and Future Directions. *Symmetry* **2017**, *9*, 49, doi:10.3390/sym9040049. [CrossRef]

10. 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, doi:10.3390/s19040893. [CrossRef]

11. Lan, G.; Luo, Z.; Hao, Q. Development of a virtual reality teleconference system using distributed depth sensors. In Proceedings of the 2016 2nd IEEE International Conference on Computer and Communications (ICCC), Cordoba, Spain, 30 June–6 July 2018; pp. 975–978, doi:10.1109/CompComm.2016.7924850. [CrossRef]

12. 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, doi:10.1109/LRA.2018.2852782. [CrossRef]

13. 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, doi:10.1109/LRA.2017.2660769. [CrossRef]

14. Fuentes-Pacheco, J.; Ascencio, J.R.; Rendón-Mancha, J.M. Visual simultaneous localization and mapping: A survey. *Artif. Intell. Rev.* **2012**, *43*, 55–81, doi:10.1007/s10462-012-9365-8. [CrossRef]

15. 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]

16. 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, Corfu, Greece, 20–25 September 1999; Volume 1, pp. 307–314, doi:10.1109/ICCV.1999.791235. [CrossRef]

17. 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, doi:10.1109/CVPR.2017.49. [CrossRef]

18. Yang, B.; Rosa, S.; Markham, A.; Trigoni, N.; Wen, H. Dense 3D Object Reconstruction from a Single Depth View. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *41*, 2820–2834, doi:10.1109/tpami.2018.2868195. [CrossRef]

19. Song, H.O.; Xiang, Y.; Jegelka, S.; Savarese, S. Deep Metric Learning via Lifted Structured Feature Embedding. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016, doi:10.1109/cvpr.2016.434. [CrossRef]

20. 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.

21. Choy, C.B.; Xu, D.; Gwak, J.; Chen, K.; Savarese, S. 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. In *Lecture Notes in Computer Science, Proceedings of the European conference on computer vision, Amsterdam, The Netherlands, 816 October 2016*; SpringerLink: Berlin/Heidelberg, Germany, 2016; pp. 628–644. [CrossRef]

22. Ma, T.; Kuang, P.; Tian, W. An improved recurrent neural networks for 3d object reconstruction. *Appl. Intell.* **2019**, *50*, 905–923, doi:10.1007/s10489-019-01523-3. [CrossRef]

23. Dhondse, A.; Kulkarni, S.; Khadilkar, K.; Kane, I.; Chavan, S.; Barhate, R. Generative Adversarial Networks as an Advancement in 2D to 3D Reconstruction Techniques. *Data Manag. Anal. Innov. Adv. Intell. Syst. Comput.* **2019**, 343–364, doi:10.1007/978-981-13-9364-8_25. [CrossRef]

24. Turhan, C.G.; Bilge, H.S. Fused voxel autoencoder for single image to 3D object reconstruction. *Electron. Lett.* **2020**, *56*, 134–137. [CrossRef]

25. Hermoza, R.; Sipiran, I. 3D Reconstruction of Incomplete Archaeological Objects Using a Generative Adversarial Network. In *Proceedings of Computer Graphics International 2018*; Association for Computing Machinery: New York, NY, USA, 2018. [CrossRef]

26. Elaksher, A.F. 3D object reconstruction from multiple views using neural networks. *Appl. Geomat.* **2013**, *5*, 193–201, doi:10.1007/s12518-013-0110-z. [CrossRef]

27. Espinal, J.; Ornelas, M.; Puga, H.J.; Carpio, J.M.; Munoz, J.A. 3D Object Reconstruction Using Structured Light and Neural Networks. In Proceedings of the 2010 IEEE Electronics, Robotics and Automotive Mechanics Conference, Cuernavaca, Mexico, 28 September–1 October 2010; pp. 74–79, doi:10.1109/cerma.2010.19. [CrossRef]

28. Kappler, D.; Bohg, J.; Schaal, S. Leveraging big data for grasp planning. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 4304–4311, doi:10.1109/ICRA.2015.7139793. [CrossRef]

29. Rivera, P.; Añazco, E.V.; Choi, M.; Kim, T. Trilateral convolutional neural network for 3D shape reconstruction of objects from a single depth view. *IET Image Process.* **2019**, *13*, 2457–2466. [CrossRef]

30. Fan, H.; Su, H.; Guibas, L. 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, doi:10.1109/cvpr.2017.264. [CrossRef]

31. Hane, C.; Tulsiani, S.; Malik, J. Hierarchical Surface Prediction for 3D Object Reconstruction. In Proceedings of the 2017 International Conference on 3D Vision (3DV), Qingdao, Canada, 10–12 October 2017; pp. 5–11, doi:10.1109/3dv.2017.00054. [CrossRef]

32. Jiang, L.; Shi, S.; Qi, X.; Jia, J. GAL: Geometric Adversarial Loss for Single-View 3D-Object Reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*; SpringerLink: Berlin/Heidelberg, Germany, 2018; pp. 820–834. [CrossRef]

33. Zhang, Y.; Liu, Z.; Liu, T.; Peng, B.; Li, X. RealPoint3D: An Efficient Generation Network for 3D Object Reconstruction from a Single Image. *IEEE Access* **2019**, *7*, 57539–57549. [CrossRef]

34. Zhang, Y.; Huo, K.; Liu, Z.; Zang, Y.; Liu, Y.; Li, X.; Zhang, Q.; Wang, C. PGNet: A Part-based Generative Network for 3D object reconstruction. *Knowl.-Based Syst.* **2020**, 105574, doi:10.1016/j.knosys.2020.105574. [CrossRef]

35. Payne, B.R.; Lay, J.F.; Hitz, M.A. Automatic 3D object reconstruction from a single image. In Proceedings of the 2014 ACM Southeast Regional Conference on-ACM SE 14, Kennesaw, GA, USA, 28–29 March 2014; pp. 1–5, doi:10.1145/2638404.2638495. [CrossRef]

36. Li, D.; Shao, T.; Wu, H.; Zhou, K. Shape Completion from a Single RGBD Image. *IEEE Trans. Vis. Comput. Gr.* **2017**, *23*, 1809–1822, doi:10.1109/TVCG.2016.2553102. [CrossRef] [PubMed]

37. Zhang, Z. Microsoft Kinect Sensor and Its Effect. *IEEE Multimed.* **2012**, *19*, 4–10, doi:10.1109/mmul.2012.24. [CrossRef]

38. 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; pp. 1267–1276, doi:10.1109/cvprw.2017.167. [CrossRef]

39. Zhang, C. CuFusion2: Accurate and Denoised Volumetric 3D Object Reconstruction Using Depth Cameras. *IEEE Access* **2019**, *7*, 49882–49893. [CrossRef]

40. Khoshelham, K.; Elberink, S.O. Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. *Sensors* **2012**, *12*, 1437–1454, doi:10.3390/s120201437. [CrossRef] [PubMed]

41. 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, doi:10.1109/JSEN.2017.2703829. [CrossRef]

42. 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, doi:10.1109/TCSVT.2016.2555678. [CrossRef]

43. Du, Y.; Fu, Y.; Wang, L. Representation Learning of Temporal Dynamics for Skeleton-Based Action Recognition. *IEEE Trans. Image Process.* **2016**, *25*, 3010–3022, doi:10.1109/TIP.2016.2552404. [CrossRef]

44. Lin, T.; Goyal, P.; Girshick, R.B.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.

45. Fu, C.; Liu, W.; Ranga, A.; Tyagi, A.; Berg, A.C. DSSD: Deconvolutional Single Shot Detector. *arXiv* **2017**, arXiv:1701.06659.

46. Ren, S.; He, K.; Girshick, R.B.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems*; Neural Information Processing Systems Foundation, Inc.: La Jolla, CA, USA, 2015.

47. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.

48. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.

49. Seyfioğlu, M.S.; Özbayoğlu, A.M.; Gürbüz, S.Z. Deep convolutional autoencoder for radar-based classification of similar aided and unaided human activities. *IEEE Trans. Aerosp. Electron. Syst.* **2018**, *54*, 1709–1723, doi:10.1109/TAES.2018.2799758. [CrossRef]

50. Pan, S.J.; Yang, Q. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359, doi:10.1109/TKDE.2009.191. [CrossRef]

51. Loshchilov, I.; Hutter, F. SGDR: Stochastic Gradient Descent with Restarts. *arXiv* **2016**, arXiv:1608.03983.

52. Savitzky, A.; Golay, M.J.E. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Anal. Chem.* **1964**, *36*, 1627–1639, doi:10.1021/ac60214a047. [CrossRef]

53. Murugan, P.; Durairaj, S. Regularization and Optimization strategies in Deep Convolutional Neural Network. *arXiv* **2017**, arXiv:1712.04711.

54. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.E.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. *arXiv* **2014**, arXiv:1409.4842.

55. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:1512.03385.

56. Kim, Y.; Shin, J.; Park, H.; Paik, J. Real-Time Visual Tracking with Variational Structure Attention Network. *Sensors* **2019**, *19*, 4904, doi:10.3390/s19224904. [CrossRef]

57. Nair, V.; Hinton, G.E. Rectified Linear Units Improve Restricted Boltzmann Machines. In Proceedings of the 27th International Conference on International Conference on Machine Learning, Haifa, Israel, 21–24 June 2010; pp. 807–814.

58. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. *Microsoft COCO: Common Objects in Context*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Computer Vision–ECCV 2014; Springer: Cham, Switzerland, 2014; pp. 740–755.

59. Everingham, M.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338, doi:10.1007/s11263-009-0275-4. [CrossRef]

60. Dai, A.; Chang, A.X.; Savva, M.; Halber, M.; Funkhouser, T.A.; Nießner, M. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. *arXiv* **2017**, 1702.04405.

61. Flaischlen, S.; Wehinger, G.D. Synthetic Packed-Bed Generation for CFD Simulations: Blender vs. STAR-CCM+. *ChemEngineering* **2019**, *3*, 52, doi:10.3390/chemengineering3020052. [CrossRef]

62. Kainz, F.; Bogart, R.R.; Hess, D.K. *The OpenEXR Image File Format*; ACM Press: New York, NY, USA, 2004.

63. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. *Lect. Not. Comput. Sci.* **2016**, 21–37, doi:10.1007/978-3-319-46448-0_2. [CrossRef]

64. Pantaleoni, J. VoxelPipe: A programmable pipeline for 3D voxelization. In Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics-HPG, Vancouver, BC, Canada, 5–7 August 2011; pp. 99–106, doi:10.1145/2018323.2018339. [CrossRef]

65. Baldwin, D.; Weber, M. Fast Ray-Triangle Intersections by Coordinate Transformation. *J. Comput. Gr. Tech. (JCGT)* **2016**, *5*, 39–49.

66. Kulikajevas, A.; Maskeliūnas, R.; Damaševičius, R.; Misra, S. Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from ShapeNetCore Dataset. *Sensors* **2019**, *19*, 1553, doi:10.3390/s19071553. [CrossRef] [PubMed]

67. Cui, Z.; Jiang, K.; Wang, T. Unsupervised Moving Object Segmentation from Stationary or Moving Camera Based on Multi-frame Homography Constraints. *Sensors* **2019**, *19*, 4344, doi:10.3390/s19194344. [CrossRef]

68. Park, K.W.; Shim, Y.J.; Lee, M.j.; Ahn, H. Multi-Frame Based Homography Estimation for Video Stitching in Static Camera Environments. *Sensors* **2019**, *20*, 92, doi:10.3390/s20010092. [CrossRef]

69. 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, doi:10.1109/tnn.2004.836241. [CrossRef]

70. 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, doi:10.1109/TNNLS.2017.2748585. [CrossRef]

71. Arifovic, J.; Gençay, R. Using genetic algorithms to select architecture of a feedforward artificial neural network. *Phys. A Stat. Mech. Appl.* **2001**, *289*, 574–594, doi:10.1016/S0378-4371(00)00479-9. [CrossRef]

72. 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, doi:10.3390/app8020293. [CrossRef]

73. 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, Santa Barbara, CA, USA, 16–19 October 2011; pp. 559–568.

74. 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, doi:10.1109/TNNLS.2017.2677968. [CrossRef] [PubMed]

75. Hawkins, J.; Boden, M. The applicability of recurrent neural networks for biological sequence analysis. *IEEE/ACM Trans. Comput. Biol. Bioinf.* **2005**, *2*, 243–253, doi:10.1109/TCBB.2005.44. [CrossRef] [PubMed]

76. Wang, Y.; Liao, W.; Chang, Y. Gated Recurrent Unit Network-Based Short-Term Photovoltaic Forecasting. *Energies* **2018**, *11*, 2163, doi:10.3390/en11082163. [CrossRef]

77. Liu, Z.; Zhao, C.; Wu, X.; Chen, W. An Effective 3D Shape Descriptor for Object Recognition with RGB-D Sensors. *Sensors* **2017**, *17*, 451, doi:10.3390/s17030451. [CrossRef] [PubMed]

78. 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, doi:10.1109/TIFS.2014.2361028. [CrossRef]

79. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90, doi:10.1145/3065386. [CrossRef]

80. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015, doi:10.1109/cvpr.2015.7298594. [CrossRef]

81. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016, doi:10.1109/cvpr.2016.90. [CrossRef]

82. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.

83. Lin, M.; Chen, Q.; Yan, S. Network In Network. *arXiv* **2013**, arxiv:1312.4400.

84. Hasanpour, S.H.; Rouhani, M.; Fayyaz, M.; Sabokrou, M. Lets keep it simple, Using simple architectures to outperform deeper and more complex architectures. *arXiv* **2016**, arxiv:1608.06037.