ktu
1922

AUDRIUS KULIKAJEVAS

RECONSTRUCTION ALGORITHM OF INVISIBLE SIDES OF A 3D OBJECT FOR DEPTH SCANNING SYSTEMS

DOCTORAL DISSERTATION

Kaunas
2022

KAUNAS UNIVERSITY OF TECHNOLOGY

AUDRIUS KULIKAJEVAS

RECONSTRUCTION ALGORITHM OF INVISIBLE SIDES OF A 3D OBJECT FOR DEPTH SCANNING SYSTEMS

Doctoral dissertation
Natural sciences, Informatics (N 009)

Kaunas, 2022

This doctoral dissertation was prepared at Kaunas University of Technology, Faculty of Informatics, Department of Multimedia Engineering during the period of 2017–2021. The doctoral right has been granted to Kaunas University of Technology together with Vilnius Magnus University and Vilnius Gediminas Technical University.

Scientific supervisor
Prof. Dr. Rytis MASKELIŪNAS (Kaunas University of Technology, Technological Sciences, Informatics Engineering, T 007).

Edited by: English language editor dr. Armandas Rumšas (Publishing House *Technologija*), Lithuanian language editor Aurelija Gražina Rukšaitė (Publishing House *Technologija*)

Dissertation Defence Board of Informatics Science Field:
Prof. Hab. Dr. Rimantas BARAUSKAS (Kaunas University of Technology, Natural Sciences, Informatics, N 009) – **chairperson**;
Prof. Hab. Dr. Gintautas DZEMYDA (Vilnius University, Natural Sciences, Informatics, N 009);
Prof. Hab. Dr. Justyna PATALAS-MALISZEWSKA (University of Zielona Góra, Poland, Natural Sciences, Informatics, N 009);
Prof. Dr. Gintaras PALUBECKIS (Kaunas University of Technology, Natural Sciences, Informatics, N 009);
Doc. Dr. Agnė PAULAUSKAITĖ-TARASEVIČIENĖ (Kaunas University of Technology, Natural Sciences, Informatics, N 009).

The official defence of the dissertation will be held at 10:00 a.m. on 22nd April, 2022 at the public meeting of Dissertation Defence Board of Informatics Science Field in Dissertation Defence Hall at Kaunas University of Technology.

Address: Donelaičio 73-403, Kaunas LT-44249, Lithuania.
Phone (+370) 37 300 042; fax. (+370) 37 324 144; e-mail doktorantura@ktu.lt
The doctoral dissertation was sent out on 22 March, 2022.

The doctoral dissertation is available on the internet at http://ktu.edu, at the library of Kaunas University of Technology (Donelaičio 20, Kaunas LT-44239, Lithuania), at the library of Vilnius Magnus University (Donelaičio g. 52, Kaunas LT-44244, Lithuania) and the library of Vilnius Gediminas Technical University (Saulėtekio al. 14, Vilnius LT-10223, Lithuania).

© A. Kulikajevas, 2022

KAUNO TECHNOLOGIJOS UNIVERSITETAS

AUDRIUS KULIKAJEVAS

3D OBJEKTO NEMATOMŲ ZONŲ REKONSTRUKCIJOS ALGORITMAS GYLIO SKENAVIMO SISTEMOMS

Daktaro disertacija
Gamtos mokslai, informatika (N 009)

Kaunas, 2022

Disertacija rengta 2017–2021 m. Kauno technologijos universiteto Informatikos fakultete, Multimedijos inžinerijos katedroje.
Doktorantūros teisė Kauno technologijos universitetui suteikta kartu su Vytauto Didžiojo universitetu ir Vilniaus Gedimino technikos universitetu.

Mokslinis vadovas: prof. dr. Rytis MASKELIŪNAS (Kauno technologijos universitetas, technologijos mokslai, informatikos inžinerija, T 007).

Redagavo: anglų kalbos redaktorius dr. Armandas Rumšas (leidykla „Technologija“), lietuvių kalbos redaktorė Aurelija Gražina Rukšaitė (leidykla „Technologija“)

Informatikos mokslo krypties disertacijos gynimo taryba:
prof. habil. dr. Rimantas BARAUSKAS (Kauno technologijos universitetas, gamtos mokslai, informatika, N 009) – **pirmininkas**;
prof. habil. dr. Gintautas DZEMYDA (Vilniaus universitetas, gamtos mokslai, informatika, N 009);
prof. habil. dr. Justyna PATALAS-MALISZEWSKA (Zielona Guros universitetas, Lenkija, gamtos mokslai, informatika, N 009);
prof. dr. Gintaras Palubeckis (Kauno technologijos universitetas, gamtos mokslai, informatika, N 009);
doc. dr. Agnė PAULAUSKAITĖ-TARASEVICIENĖ (Kauno technologijos universitetas, gamtos mokslai, informatika, N 009).

Disertacija bus ginama viešame Informatikos mokslo krypties disertacijos gynimo tarybos posėdyje 2022 m. balandžio 22 d. 10 val. Kauno technologijos universiteto Disertacijų gynimo salėje.

Adresas: K. Donelaičio g. 73-403, Kaunas LT-44249, Lietuva.
Tel. (+370) 37 30 00 42; faks. (+370) 37 32 41 44; el. paštas doktorantura@ktu.lt

Disertacija išsiųsta 2022 m. kovo 22 d.
Su disertacija galima susipažinti interneto svetainėje http://ktu.edu, Kauno technologijos universiteto (K. Donelaičio g. 20, Kaunas LT-44239), Vytauto Didžiojo universiteto (K. Donelaičio g. 52, Kaunas LT-44244) ir Vilniaus Gedimino technikos universiteto (Saulėtekio al. 14, Vilnius LT-10223) bibliotekose.

© A. Kulikajevas, 2022

Table of Contents

1 Introduction 7
  1.1 Problem Statement 7
  1.2 Research Aim 8
  1.3 Methods and Software 8
  1.4 Practical Application 9
  1.5 Research Novelty 10
  1.6 Dissertation Statements to be Defended 10
2 Scientific literature review 12
3 Submitted paper overview 16
  3.1 General overview 17
  3.2 Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from ShapeNetCore Dataset 19
    3.2.1 Materials and Methods 20
    3.2.2 Dataset 22
    3.2.3 Results 23
  3.3 3D Object Reconstruction from Imperfect Data Using Extended YOLOv3 Network 27
    3.3.1 Materials and Methods 28
    3.3.2 Dataset 33
    3.3.3 Results 34
  3.4 HUMANNET—A Two-Tiered Deep Neural Network Architecture for Self-Occluding Humanoid Pose Reconstruction 37
    3.4.1 Materials and Methods 37
    3.4.2 Dataset 44
    3.4.3 Results 44
  3.5 Auto-Refining Reconstruction Algorithm for Recreation of Limited Angle Humanoid Depth Data 47
    3.5.1 Materials and Methods 47
    3.5.2 Dataset 55
    3.5.3 Results 56
4 Conclusions 59
5 Future Works. 60
6 Summary 61
  6.1 Įvadas 61
    6.1.1 Tiriama problema 61
    6.1.2 Tyrimų tikslas 61
    6.1.3 Darbo uždaviniai 61

6.1.4 Praktinė darbo reikšmė 61
6.1.5 Naujumas 62
6.1.6 Ginamieji teiginiai 62
6.1.7 Darbo rezultatų aprobavimas 62
6.2 Straipsnių apžvalga 65
6.2.1 *Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from ShapeNetCore Dataset* 66
6.2.2 *3D Object Reconstruction from Imperfect Depth Data Using Extended YOLOv3 Network* 69
6.2.3 *HUMANNET—A Two-Tiered Deep Neural Network Architecture for Self-Occluding Humanoid Pose Reconstruction* 73
6.2.4 *Auto-Refining Reconstruction Algorithm for Recreation of Limited Angle Humanoid Depth Data* . . 78
6.3 Rezultatai 85
6.4 Išvados 92
7 References 93
8 Submitted papers 102
8.1 Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from *ShapeNetCore* Dataset 102
8.2 3D Object Reconstruction from Imperfect Depth Data Using Extended YOLOv3 Network 134
8.3 HUMANNET—a Two-Tiered Deep Neural Network Architecture for Self-Occluding Humanoid Pose Reconstruction 175
8.4 Auto-Refining Reconstruction Algorithm for Recreation of Limited Angle Humanoid Depth Data 199
9 Curriculum Vitae 227
10 List of scientific papers and conferences 228
A Abbreviations and Terms 230
B Symbols and notation 232

1. INTRODUCTION

Co-authors and publishers have given their permission to use and include the research papers in the dissertation. The dissertation consists of the following published research papers:

1. *Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from ShapeNetCore Dataset* by **Audrius Kulikajevas**, Rytis Maskeliūnas, Robertas Damaševičius and Sanjay Misra.
2. *3D Object Reconstruction from Imperfect Data Using Extended YOLOv3 Network* by **Audrius Kulikajevas**, Rytis Maskeliūnas, Robertas Damaševičius and Edmond S. L. Ho.
3. *HUMANNET—A Two-Tiered Deep Neural Network Architecture for Self-Occluding Humanoid Pose Reconstruction* by **Audrius Kulikajevas**, Rytis Maskeliūnas, Robertas Damaševičius and Rafal Scherer.
4. *Auto-Refining Reconstruction Algorithm for Recreation of Limited Angle Humanoid Depth Data* by **Audrius Kulikajevas**, Rytis Maskeliūnas, Robertas Damaševičius and Marta Wlodarczyk-Sielicka.

1.1. Problem Statement

The problem to be addressed in this dissertation — full object completion using only a single distorted or otherwise imperfect depth sensor perspective.

Object completion is a critical task in computer vision [1, 2, 3, 4] that is required for various practical applications, e.g., autonomous vehicles or surveillance. The two main types of methods that attempt to solve this issue are: classical algorithms and machine learning models. Classical approaches attempt to solve this issue in different ways — starting with using various assertions about the object itself, e.g., symmetry axes; while other methods use iterative point cloud fusion methods, e.g., *Simultaneous Localization and Mapping* (SLAM) which fuse multiple frames taken over a period of time into a final 3D objects' representation [5, 6, 7, 8, 9]. However, because object completion is such an important problem to solve, in recent time there has been a surge of interest in applying deep learning approaches to solve this task [10, 11, 12].

Different types of sensors are denoted by different pros and cons, e.g., visible light cameras tend to suffer from low light conditions but are more readily available; structured light and infrared depth sensors, while less affected by lightning conditions, are more prone to distortions; laser radar (LiDAR) sensors have an accurate depth field, but distortions can happen from the material absorbing the wavelength that the sensor is emitting [13, 14, 15, 16]. As the world is gradually moving towards full automation, sensor imperfections may cause dangerous situations [17, 18]. Object completion is a useful tool when identifying object volume in robotics [19], for gesture recognition [20], indoor mapping and navigation [21], recreating evidence and crime scenes during forensic analysis [22], etc.

While the existing research already attempts to reconstruct the object, there remains a knowledge gap for, e.g., high fidelity inputs can be slow to reconstruct [23, 24], artifacts can be caused by improper assertions about the reconstructed object [25], certain methods require tailored solutions for the environment [26, 27, 28] or can be expensive due to the specialized sensor requirement [29]. This dissertation attempts to narrow down the knowledge gap left by other research by proposing a solution for object reconstruction from a single perspective.

1.2. Research Aim

The aim of the research is the proposal and implementation of a computer model or models which can reconstruct an object from a single imperfect, noisy or distorted depth sensor perspective. To reach this research goal, three main objectives have been raised:

1. Analyze the already existing classical and machine learning approaches for object completion while using both: single and multiple perspective inputs.
2. Propose and implement a computer model (or models) which can complete probable objects' invisible or occluded regions by using only a single object frame.
3. Conceive and implement a computer model as well as its application strategy for occluded object region reconstruction from a single imperfect depth frame captured either by structured light or laser radar depth sensors.

1.3. Methods and Software

Computer models proposed in this dissertation have been implemented using two neural network frameworks. This allowed for rapid model prototyping and experimentation. The machine learning frameworks used for implementing deep learning models in this dissertation are *Tensorflow* and

PyTorch. The initial experiments were conducted by using *Tensorflow 1.14* and later *Tensorflow 2.3* versions; *CUDA 10.1* and *cuDNN* were used to train the models by utilizing graphic processing units (GPUs). To train the neural network models, initially, a GTX 1070 with 8GB of video memory was used whereas, GTX 960M (4GB VRAM) was used to evaluate the model performance on the lower-end devices. In the later stages of the research, training of the neural networks was dedicated to RTX 3070 (8GB VRAM) GPU. The switch was made due to a new generation of graphical processors having been released. Because of this hardware change, *CUDA* toolkit had to be upgraded to *CUDA 11.1*, whereas the older versions were no longer supported by the manufacturer. Additionally, the move to a newer GPU architecture allowed for half-precision floating point training, and this greatly increased the complexity of the model that can be trained on a limited graphical memory. Finally, later research also moved away from *Tensorflow* to *PyTorch 1.7.1*. This move was made not only due to superior support of the new generation of graphics cards but, during experiments, it was also noticed that the *PyTorch* framework has a lower memory footprint, and this allowed for more complex deep learning models to fit in the video memory with no noticeable impact on training duration.

Synthetic dataset samples for training the deep learning models were created by using *Blender 2.79b*. *Blender* allows for a fully scriptable pipeline where the scene can be created by using the *Python* programming language and rendered in the headless mode. *Python 3.7* was used throughout this dissertation for both *Blender* scriptable pipeline and the implementation of deep learning models.

Real world RGB-D data samples were captured with three different depth sensing devices: *Intel ZR300*, *Intel D435i* and *Intel L515*. *Intel D435i* and *Intel ZR300* are both structured light depth sensors that use light and infrared sensor properties to infer the depth fields; whereas, *Intel L515* is a LiDAR. As *Intel ZR300* was discontinued and no longer supported during the duration of the research, experiments in the later papers do not include it.

1.4. Practical Application

This dissertation overviews four published research papers, each proposing an improved machine learning model for maskless object reconstruction from a single camera perspective. Each of the models can reconstruct a probable object shape alongside other various novelties, e.g., real-time surface reconstruction, multiple object reconstruction, temporally morphing and complex object reconstruction without the need for ground truth for training.

The scientific impact of this dissertation is the publishing of four research papers providing innovative improvements in object completion; the published papers at the time of writing this dissertation have been viewed over 33,000 times and cited 19 times.

In addition to scientific importance, the conducted research has the potential commercial application spanning various fields and disciplines, starting with robotics [30], object reconstruction application for autonomous vehicle collision avoidance [31, 32], as well as medical applications such as posture recognition [33, 34] and magnetic resonance imaging (MRI) [35]. Furthermore, the conducted research has various entertainment industry applications, e.g., obstacle alerts in virtual reality [36, 37], augmented [38] and extended realities [39], film industry [40], etc.

1.5. Research Novelty

Each of the research papers in this dissertation proposed novel machine learning models for object reconstruction while using only a single depth sensor perspective. These novelties can potentially narrow the knowledge gap existing in the field of three-dimensional object completion. The novel suggestions to the field include such proposals as: object reconstruction in real-time by using deep hybrid neural networks for object reconstruction; completion of an object's smooth polygonal mesh; reconstruction of several objects in a single depth frame, whereas previous methods focused on single object reconstruction; maskless object completion; complex and temporally morphing object point cloud completion; detection and clipping of several complex objects in a point cloud for region extraction and individual object reconstruction; real world object point cloud cleanup and completion using only a single distorted or otherwise imperfect structured light or laser sensor frame without a ground truth by applying deep unsupervised adversarial refining neural networks.

1.6. Dissertation Statements to be Defended

1. A computer model was proposed and implemented for object completion from a single noisy, distorted or otherwise imperfect depth perspective, by applying a novel unsupervised deep refining neural network methodology.

2. A training strategy was proposed for object completing machine learning models that can be trained without a ground truth value for the comparison function.

3. Novel machine learning models were proposed for both single and multiple static, complex and temporally morphing object reconstruction

in real time from an imperfect input frame, providing similar reconstruction quality, despite distortions in the depth field, to that of the synthetic (perfect) dataset.

2. SCIENTIFIC LITERATURE REVIEW

The field of research that this dissertation focuses on is the rapidly expanding discipline of computer vision which is developing in part due to the increased research interest in machine learning with deep neural networks in particular. However, while there has been a lot of research and success with the application of machine learning for computer vision tasks starting with such applications as detection of disease in plants [41], detection and diagnosis of faults in industrial devices [42] or even medical applications ranging from gastric cancer segmentation in endoscopic images [43], skin lesion detection [44], recognition of driver fatigue [45] and even recognition of lung disease in an X-ray image [46]. However, there remains a knowledge gap in the field of object completion.

Whereas several classical approaches for object surface reconstruction exist, starting with methods exploiting object surface features such as convex hull for its reconstruction [47] or objects gradient fields' vertex normals [48, 49, 50], these methods can rarely account for occluded or self-occluded object sides and other distorted or missing information. For missing information substitution, several classical approaches exist, and these can range from restoration based on assertions on object symmetry axes or the fusion of multiple point clouds acquired from either multiple cameras or iteratively [51, 52, 53, 54]. However, the application of such methods is flawed and limited, as these methods can fail for temporally shifting and morphing objects [55], e.g., a person performing some exercise, or have high computational complexity thus making high fidelity reconstruction difficult [56]. In addition to these, the requirement of having multiple perspectives for full object occluded region reconstruction can be difficult to meet. Therefore, there is still a need for a solution that could reconstruct a real world object that is not only robust against noise but can also complete an object while using only a single perspective.

There are two primary data structures for three-dimensional object representation — voxel grids and point clouds. One of the first machine learning based object completion methods *3D-R2N2* used deep neural networks [10] trained on *ShapeNet* [57] and *Sandford Online Products* [58] datasets. *3D-R2N2* used a priori knowledge to train recurrent neural networks with *Long short-term memory* (LSTM) layers [59, 60] in order to train deep learning model to recognize and reconstruct an object from either a single or multiple perspectives, by firstly exposing the neural network to the same object shown from multiple sides. While the paper has become a benchmark paper for comparison between state-of-the-art methods due to its progeny status in the field, it suffered from several flaws, one of which — it required objects' masks to perform a reconstruction. Moreover, it could not reconstruct rotations and positions; meanwhile, having only a single perspective resulted in poorer results when compared to reconstruction from multiple perspectives. Further research in the field of object completion has improved on the method by either removing the mask requirement [11] or by proposing the incorporation of *Chamfer* distance as a loss metric for comparison between the prediction and the ground truth, thus substantially improving the reconstruction results [12]. Meanwhile, brand-new approaches attempted to apply generative adversarial neural networks (GANNs) for object reconstruction when given only a few perspectives [61], or even just a single perspective [62]; whereas others attempted to use hierarchical surface reconstruction methods, which allowed for much higher resolution object reconstructions [63]. However, further research attempted to do away with the voxel grid approach completely, while opting for point cloud completion instead. One of the first successful experimental examples is *PointOutNet*; here the authors demonstrated the ability to reconstruct an object's point cloud from an RGB image and its mask input [64]. Similarly to *3D-R2N2*, mask requirement makes the solution not applicable to non-synthetic real-time applications. While further solutions attempted to improve the results for an unstructured point cloud, they also experimented on using flat RGB images as inputs [65]. The primary reason why most authors chose this flat RGB input approach for their machine learning occluded object side reconstruction methods is the ease of application of already well-known two-dimensional convolutional kernels which had been successfully used in the past for computer vision tasks [66, 67]. However, due to monocular cameras losing useful information about objects' shape that can now only be extrapolated from the lighting conditions and material properties, these solutions generally could still be improved upon. Therefore, other researchers proposed an architectural feature extraction solution on how to use unstructured point clouds as an input for machine learning models in the objects point cloud classification and reconstruction tasks, where standard convolutional kernels would have failed [68]. Other research proposed a dense-to-coarse object reconstruction methodology for the improvement of an object's reconstruction [69], where coarse object features are used for primary object feature reconstruction with finer details being reconstructed afterwards. *AtlasNet* authors proposed patch-based approach reconstruction where multiple patches are used for separate object feature reconstruction [70]. Up until this point, research on object reconstruction predominantly used the *Chamfer* distance (CD) as a loss metric and the *Earth Mover's* distance (EMD) only as a quantitative validation metric for the evaluation of the results, despite the *Earth Mover's* distance being a more sensitive metric to objects' defects. This is because EMD is more computationally expensive and has more demanding operative memory requirements. To solve this, researchers proposed an approximation for EMD; this linear approximation allowed for the application of *Earth Mover's* distance for high density unordered point clouds [71]. Despite all these advancements in the field of object completion from a single perspective, state-of-the-art solutions still suffer from some knowledge gaps. These include multiple object reconstruction from a single frame, maskless object reconstruction which would make it applicable to non-synthetic experiments, complex and temporally morphing object reconstruction; lastly, the ability to reconstruct from an imperfect depth sensor input without acquiring ground truth values.

To narrow this knowledge gap, this dissertation provides several machine learning-based computer models, and a comparison between state-of-the-art methods and object completion methods proposed in the research papers overviewed in the dissertation can be seen in the Table 1. The machine learning approach was selected based on the assertion that every person builds a mental model of their world during their lifetime. This a priori mental model allows them to roughly estimate what the occluded or self-occluded parts of an object would look like. From this assertion, a hypothesis was put forward that machine learning approaches, similarly able of pattern matching as a human, is a tool adept of occluded side object reconstruction.

Table 1. Comparison of dissertation methods with existing state-of-the-art object completion methods

Method	**Completion type**	**Input type**	**Synthetic**	**Requires mask**	**Multiple objects**
3D-R2N2 [10]	Voxels	RGB	✗	✓	✗
SiDeNet [11]	Voxels	RGB	✗	✗	✗
Tingsong et al. [12]	Voxels	RGB	✗	✓	✗
OGN [72]	Oct-tree voxels	RGB	✓	✓	✗
PointOutNet [64]	Point cloud	RGB	✓	✓	✗
AtlasNet [70]	Point cloud	Point	✓	✗	✗
MSN [71]	Point cloud	Point	✓	✗	✗
Paper 1 (Section 3.2)	Voxels	Depth	✗	✗	✗
Paper 2 (Section 3.3)	Voxels	RGB-D	✗	✗	✓
Paper 3 (Section 3.4)	Point cloud	Depth	✓	✗	✓
Paper 4 (Section 3.5)	Point cloud	Depth	✗	✗	✗

3. SUBMITTED PAPER OVERVIEW

Based on the previously raised hypothesis in this dissertation, a total of four papers shall be presented:

1. Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from ShapeNetCore Dataset (Section 3.2) [73]
2. 3D Object Reconstruction from Imperfect Data Using Extended YOLOv3 Network (Section 3.3) [74]
3. HUMANNET—A Two-Tiered Deep Neural Network Architecture for Self-Occluding Humanoid Pose Reconstruction (Section 3.4) [75]
4. Auto-Refining Reconstruction Algorithm for Recreation of Limited Angle Humanoid Depth Data (Section 3.5) [76]

3.1. General overview

In this dissertation, a total of four papers have been presented, each incrementally attempting to solve the problem of occluded object region reconstruction from a single perspective. The final computer model iteration in this dissertation can be seen in Figure 1. The presented model can perform single perspective object point cloud completion from an imperfect structured light or a laser depth sensor. To achieve this result, the following steps are taken. Firstly, depth frame frame information is retrieved from a depth scanning sensor. Afterwards, additional processing is performed, where the depth information is clipped to 2.5 meters, thereby giving this approach an effective range of 2.5 meters. Once the candidate depth frame is ready, the depth sensors intrinsic matrix K is used to construct a candidate point cloud. Further processing is applied to filter out points which have a depth of zero; this is important as in the next step the point cloud dimensionality is reduced to 2048 points. Following that, the zero depth points are discarded before resampling in order to prevent meaningless points from ruining the final point cloud. Once the point cloud is ready, the objects refinement computer model is applied. The result of this is the original point cloud feature vector and the refined point cloud. The cleaned-up point cloud is then used to extract the most viable features for course reconstruction and combined using residual connections with the previously obtained input point cloud feature vector. The resulting feature map is used for coarse reconstruction. Finally, the combination of the coarse reconstruction result using residual connections with the combined latent feature vector is performed, thus obtaining the final fine-grained point cloud.

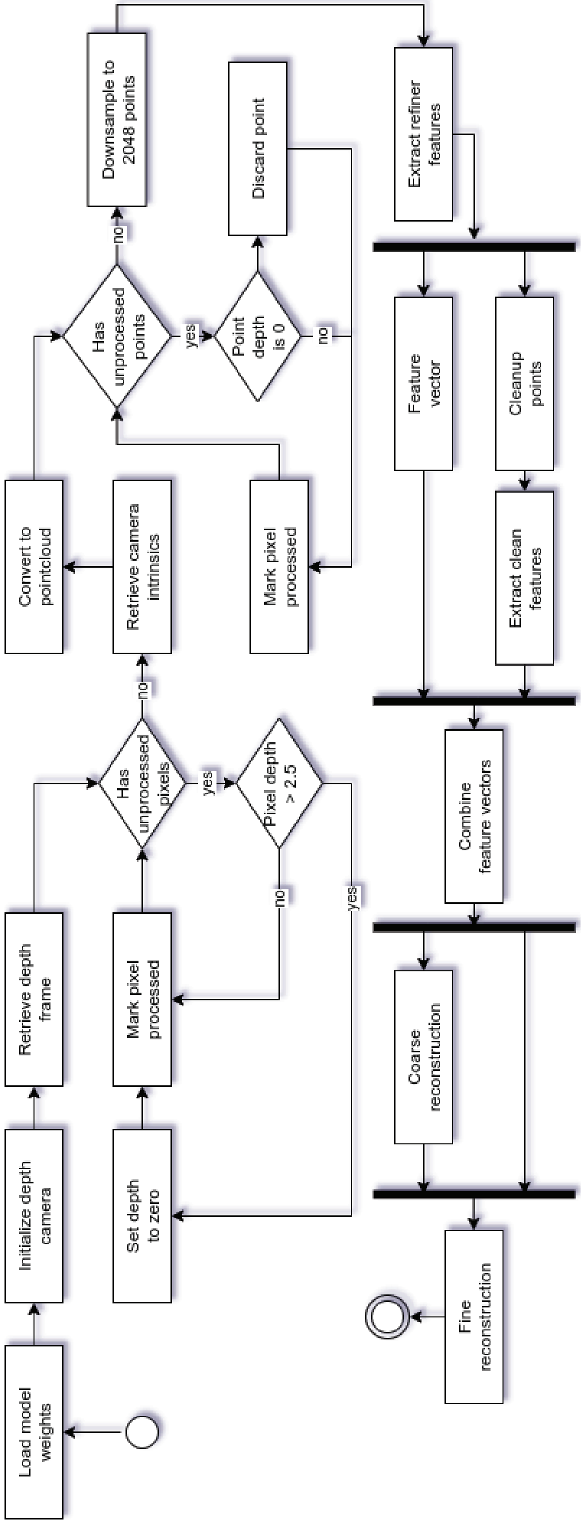


Figure 1. Activity diagram of the finalized computer model

Table 2. Author contributions to the paper

Author	**Contribution**
Rytis Maskeliūnas	Conceptualization of the research direction of the conducted research.
Sanjay Misra	Formal analysis of the conducted research.
Audrius Kulikajevas Rytis Maskeliūnas	Investigation of existing research in the field related to the research paper.
Rytis Maskeliūnas	Proposing the methodology in order to have replicable research results.
Audrius Kulikajevas	Proposing and implementing the computer model used in the research paper.
Rytis Maskeliūnas	Overseeing and supervising the conducted research.
Audrius Kulikajevas Rytis Maskeliūnas	Validation of the experimental results.
Audrius Kulikajevas Robertas Damaševičius	Graphical visualization of the experimental results.
Audrius Kulikajevas	Writing of the original draft, paper modifications based on peer reviewer comments.
Rytis Maskeliūnas Robertas Damaševičius Sanjay Misra	Reviewing and minor editing of the draft for errors and language inconsistencies.

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

The first submitted research paper is *Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from ShapeNetCore Dataset* by **Audrius Kulikajevas**, Rytis Maskeliūnas, Robertas Damaševičius and Sanjay Misra. The research conducted in this paper should provide an insight into the application of modular hybrid neural networks for an object's smooth surface mesh completion from a single imperfect depth sensor frame in real-time applications. The author contribution to the research can be seen in Table 2.

3.2.1. Materials and Methods

The paper presents three novelties in the field of object reconstruction. The first one is the hybrid deep neural network architecture, which allows for faster neural network convergence and real-time performance, and, secondly, smooth surface mesh reconstruction. Thirdly, unlike other similar object reconstruction methods, e.g., *3D-R2N2*, the proposed paper used depth sensor frames, as opposed to flat RGB images, as this allowed for more accurate object retention and maskless object reconstruction. Due to this, the paper's experimental results managed to achieve not only real-time performance, but, when evaluated quantitatively, it also achieved 89.5% of the reconstruction results which were in good ($IoU \in (0.25, 0.75]$) or excellent ($IoU \in (0.75, 1]$) category. The neural network overview can be seen in Figure 2; here **?** denotes selection of the most appropriate reconstruction branch, from one of *n* pretrained feature encoder branches. The proposed computer model is a two-tiered hybrid neural network comprised of classification and reconstruction stages. The classification stage selects the most appropriate reconstruction branch based on the predicted class to which the depth input is passed along to.

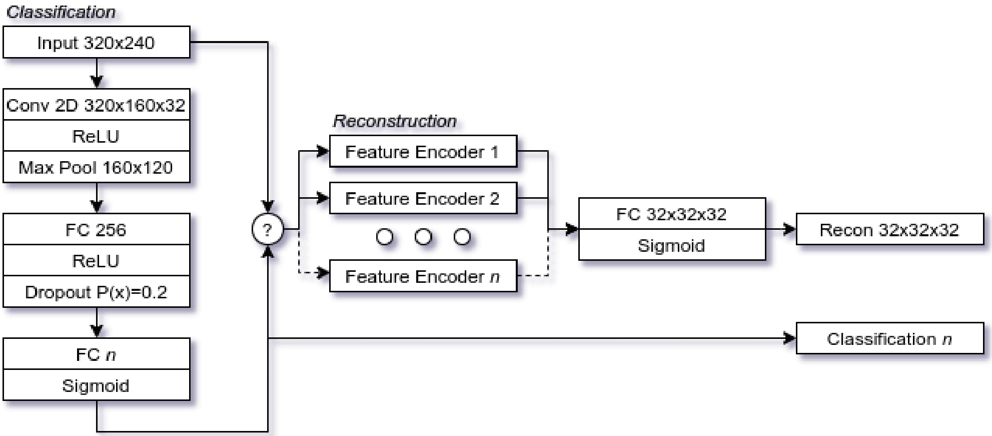


Figure 2. Hybrid-neural network overview

For the neural network input, a 320 × 240 depth frame is used. While the native size of *ZR300* is 640 × 480 recording the depth, due to hardware graphical memory limitations during the training process, it used a downsampled depth image. The image was downsampled without interpolation as not to distort the depth field during the interpolation process. The resulting depth frame is then passed through a 3 × 3 convolutional kernel to extract 32 feature vectors. As the convolutional kernel is a linear operation and this solution is non-linear, a non-linearity function must be introduced. Hence, the rectified linear units (ReLUs) are applied as a non-linearity function [77, 78]. *ReLU* was chosen for its mathematical simplicity which reduces the computational complexity, while, in addition, it has been shown that *ReLU* when used with convolutional neural networks has a better recall and convergence rate when compared to sigmoidal and hyperbolic tangent non-linearity functions [79, 80], the *ReLU* function can be seen in Equation (1) [78].

$$f(x) = max(0, x) \tag{1}$$

Following the non-linearity function, a maximum pooling operation is performed by using a 3 × 3 extraction kernel with a stride of 2; this reduces the feature map dimensionality and picks only the most important features, thus leaving a 160 × 120 latent feature vector. The resulting feature vector is used as an input for a 256 neuron fully-connected layer and its activation function. To increase the sample count and reduce the network bias, a dropout layer is added with the probability of dropping the neuron value set to $P(x) = 0.2$. It has been experimentally shown that when *ReLU* is combined with the dropout layer, generalization and accuracy improve [81, 82, 83]. The latent output is then connected to the fully-connected layer with *n* neurons, where *n* is the trained reconstruction branch count. However, instead of *ReLU* activation, in the final layer, the sigmoid activation function is used as the non-linearity function clamps the neuron saturation range to $[0, 1]$; the sigmoid equation is shown in Equation (2) [84].

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

Once the proper reconstruction branch has been identified by the classification stage, the input is then used in the selected reconstruction branch. Here, the input features are encoded for later reconstruction, and the feature encoder overview is shown in Figure 3. Not unlike the classificator, convolutional neural networks with *ReLU* activation functions throughout the feature extraction layers are used, except for the final layer. Where the fully-connected layer uses the *Leaky ReLU* function, it was found that this increases the network generalization rate because *ReLU* can sometimes suffer from dead neurons, i.e., neurons which are always stuck at zero as their input is negative thus not contributing to the network itself, whereas, *Leaky ReLU* attempts to solve for this [85, 86]. *Leaky ReLU* equation can be seen in Equation (3) [86], where α is the slope constant.

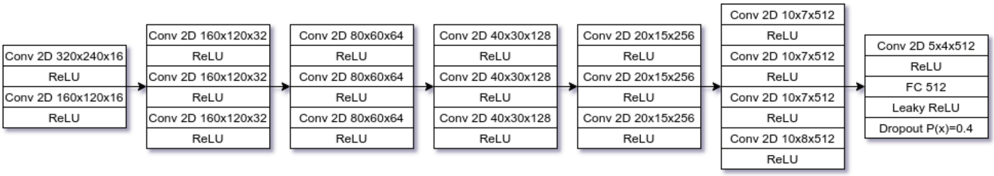


Figure 3. Hybrid-reconstruction encoder branch example

$$f(x) = max(\alpha x, x) \tag{3}$$

Following this, the latent feature encoder output is then reconstructed by using a 32 × 32 × 32 feature reconstruction layer. What is more, due to neural network hybridization, it is also possible to have different network architectures based on the reconstructed object type complexity. To train the hybrid model, a sum of two loss functions have been used, one for classification and one for reconstruction, and the final loss equation can be seen in Equation (4) which includes *softmax cross-entropy* as one of the terms [87]. The resulting model complexity when compared to other state-of-the-art approaches at the time can be seen in Table 3. While the network has a lot of trainable parameters, due to a large input image size and few pooling layers, the operation count remains low, thus the network remains performant in terms of time.

$$f_r = \sum_{i=1}^{N} \frac{\sum_{j=1}^{Q} (\alpha_{ij} - \hat{\alpha}_{ij})^2}{N} - \beta_i \log \frac{e^{\hat{\beta}_i}}{\sum_{j=1}^{N} e^{\hat{\beta}_j}} \tag{4}$$

Here, α and $\hat{\alpha}$ are objects reconstruction prediction and ground truth values, whereas β and $\hat{\beta}$ are objects true and predicted classes, additionally, *N* is the training batch size, and *Q* is the product of grid density.

3.2.2. Dataset

The described object completion method in the paper requires a priori information for the captured object and its occluded region reconstruction. For this reason, a synthetic dataset was created by using the *ShapeNet* dataset as a base. A given 3D model from the *ShapeNet* dataset is placed in the center of the scene and rendered by using only depth information from 1 m and 1.5 m distance, in 45° camera rotation increments thus creating a total of 48 perspectives for a single object in the dataset. As the computer model is only dealing with depth information, RGB information is not rendered. Furthermore, a real world validation dataset is created by using *Intel Realsense ZR300* depth sensor for objects that the computer model is trained

Table 3. Model complexity comparison.

Method	**No. of Parameters (M)**	**No. of Operations (GFLOPs)**	**Model Size (MB)**
3D-LSTM-1	30.92	12.46	174.25
3D-GRU-1	32.21	12.51	179.45
3D-LSTM-3	30.92	35.99	274.91
3D-GRU-3	32.21	37.41	280.44
Res3D-GRU-3	35.97	69.40	539.27
Hybrid	**352.35**	**3.09**	**1458.68**

to complete. Due to the hardware limitations, using the native depth sensor resolution (640 × 480) is not feasible; therefore, a downscaled version (320 × 240) is used for both synthetic and real world datasets. To calculate the reconstruction loss, ground truth voxel grid values are created for each of the objects in the dataset [88]. To perform voxelization, objects' geometry boundaries are partitioned into equally sized cells, where the size of the axes is determined by the axis containing the most cells, thereby creating a uniform voxel grid. Afterwards, the voxel state is set to either filled or empty by performing ray-triangle intersection [89], thus producing a 32 × 32 × 32 voxel grid representing the object's shape. By using the created dataset, the network was trained by using the *Adam* [90] optimizer with the learning rate of $lr = 10^{-3}$ for an average of 50 iterations before the model getting converged. The learning rate and the optimizer were chosen experimentally, for they showed the fastest convergence rates and the best evaluation dataset accuracy.

3.2.3. Results

To quantitatively evaluate the objects' voxel grid reconstruction, three main evaluation metrics are used: *Completeness* (Equation (5)), *Correctness* (Equation (6)) and *Quality* (Equation (7)) as proposed in the literature [91]. Here, *A* is filled with ground truth voxel values and *B* is filled with prediction voxel values. *Completeness* (f_{comp}), also known as *Producer's Accuracy Detection Rate*, is the ratio of voxels in ground truth that are reconstructed.

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

The *Correctness* (f_{corr}) metric shows how well the reconstructed voxels match the ground truth.

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

Meanwhile, *Quality* ($f_{quality}$) gives the combined value that balances both the correctness and completeness metrics.

$$f_{quality} = \frac{f_{comp} \cdot f_{corr}}{f_{comp} + f_{corr} - f_{comp} \cdot f_{corr}} \tag{7}$$

By using these quantitative evaluation metrics, the evaluation of the experimental results is performed for the validation datasets whose results can be seen in Figure 4.

As the results in the figures show, the primary problematic reconstruction results are within the *book* and *laptop* data points. However, the major discrepancies can be easily explained by the fact the two objects in question had very few training samples. Meanwhile, other reconstruction results fit in the good or excellent reconstruction results as defined by the paper being within $0.25 \leq f_{quality} < 0.75$ and $f_{quality} \geq 0.75$, respectively, where the value ranges were selected based on qualitative results. Furthermore, the relative error bar size allows for the evaluation of the reconstruction stability from various perspectives of the same object. In addition to this, the solution managed to achieve 151 frames per second on one of the benchmarking hardware setups with a *GTX 1070* graphics card and 28.88 FPS with a *GTX 960M* graphical chipset, thus making it fit for use in real time applications. As the neural network operation is a deterministic set of arithmetic matrix operations, the performance, in terms of time, remains constant per all object types and is only influenced by network complexity which remains constant during all experiments in this dissertation. The qualitative results can be observed in Figure 5. Row **I** shows the results of input (*a*) for the state-of-the-art object completion model [10] when compared to the proposed hybrid approach **II**. Finally, when evaluating a real dataset, it was noticed that the *ZR300* camera was unable to capture small object features and had an effective reliable range of only 0.55 m to 2.8 m, thus making this type of sensor applicable to only medium sized objects. Additionally, it was observed that this sensor was unable to capture some types of materials, e.g., laptop

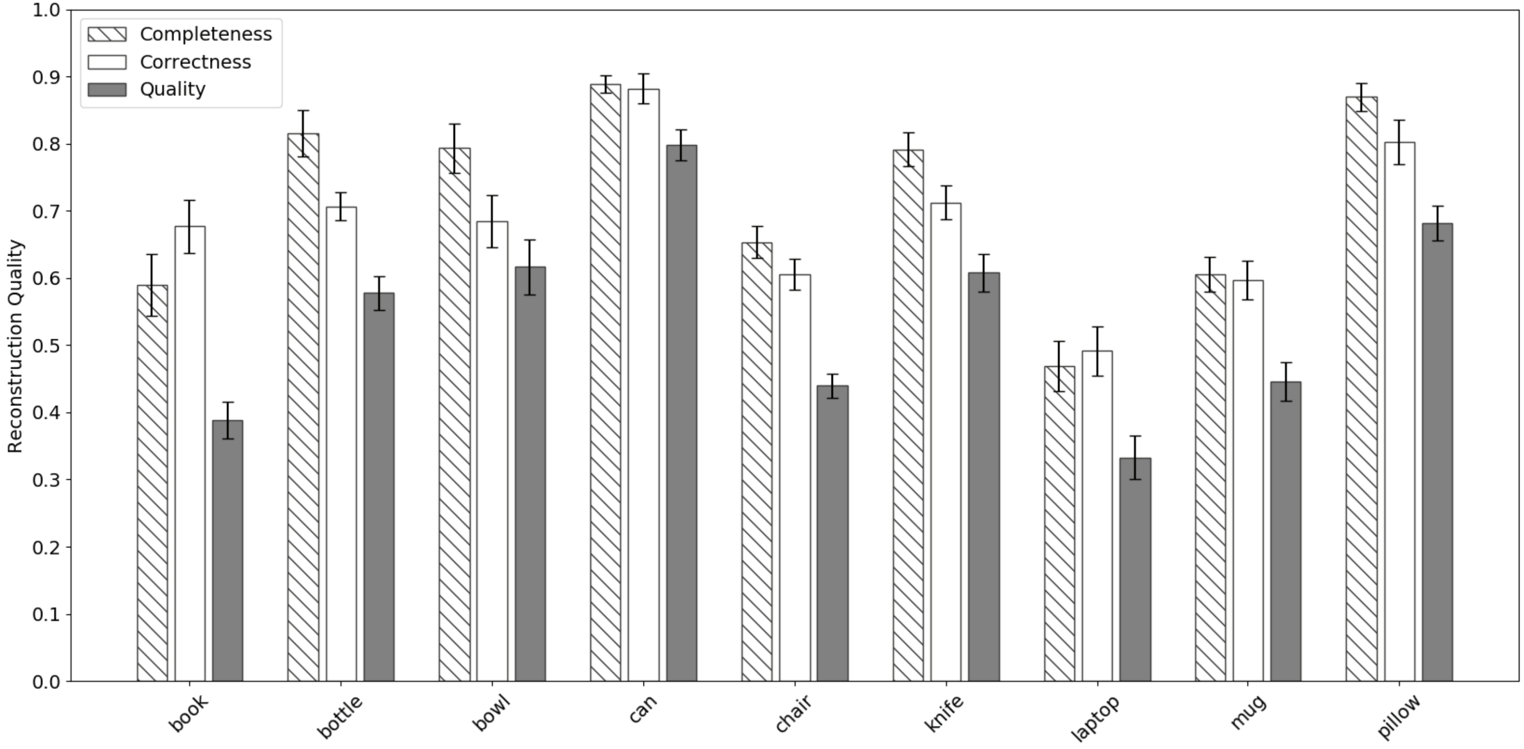


Figure 4. Quantitative reconstruction results using completeness, correctness and quality metrics

screens and plastics, as they created distortions in the depth field that the proposed approach failed to account for.

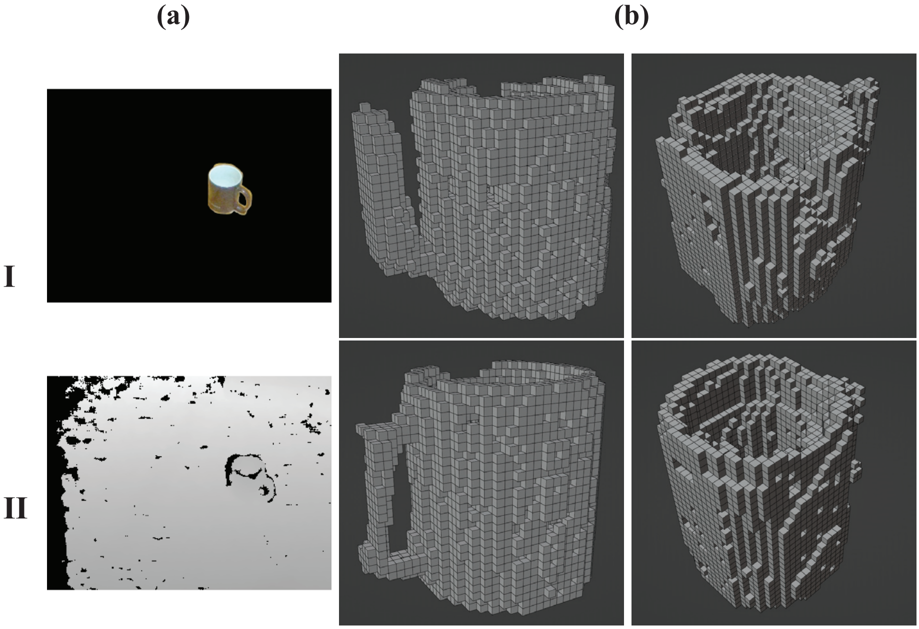


Figure 5. Model result comparison between the state-of-the-art and the proposed versions

Table 4. Author contributions to the paper

Author	**Contribution**
Rytis Maskeliūnas	Conceptualization of the research direction of the conducted research.
Rytis Maskeliūnas	Proposing the methodology in order to have replicable research results.
Audrius Kulikajevas	Proposing and implementing the computer model used in the research paper.
Audrius Kulikajevas Rytis Maskeliūnas	Validation of the experimental results.
Rytis Maskeliūnas	Formal analysis of the conducted research.
Audrius Kulikajevas Rytis Maskeliūnas Robertas Damaševičius	Investigation of existing research in the field related to the research paper.
Audrius Kulikajevas Robertas Damaševičius	Writing of the original draft, paper modifications based on peer reviewer comments.
Robertas Damaševičius Edmond S. L. Ho	Reviewing and minor editing of the draft for errors and language inconsistencies.
Audrius Kulikajevas Rytis Maskeliūnas	Graphical visualization of the experimental results.
Rytis Maskeliūnas	Overseeing and supervising the conducted research.

3.3. 3D Object Reconstruction from Imperfect Data Using Extended YOLOv3 Network

The second submitted research paper is *3D Object Reconstruction from Imperfect Data Using Extended YOLOv3 Network* by **Audrius Kulikajevas**, Rytis Maskeliūnas, Robertas Damaševičius and Edmond S. L. Ho. The research conducted in this paper should provide an insight into the application of deep hybrid neural networks for multiple smooth object surface mesh completion in the scene by using only a single imperfect depth sensor frame in real-time applications. The author contribution to the research is shown in Table 4.

3.3.1. Materials and Methods

The second research paper attempts to solve an issue that was left unaddressed in the previous one — the reconstruction of multiple objects in the frame. During the publishing of paper, in the state-of-the-art research, there was little focus on the multiple object reconstruction, thus making this the main novelty. The approach opted to use *YOLOv3* [92] as the neural networks' backbone model, and this allowed for more effective classification and objects bounding box extraction, which is then used for geometric segmentation allowing for improved object masking and contour clipping. In addition to this, a periodic hyper parameter was suggested, which allowed for better network generalization. This resulted in the relative reconstruction quality increase of 8.53% compared to the previous paper. General neural network architecture overview can be seen in Figure 6; where **?** denotes selection of the pretrained reconstruction model from n reconstruction models, and × is input multiplication with binary mask.

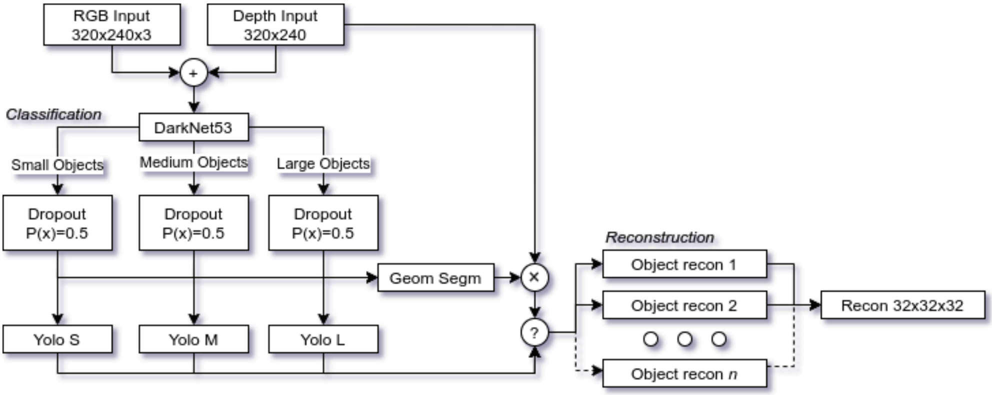


Figure 6. Deep neural network for occluded object side reconstruction overview

Described approach uses a modified version of *DarkNet53* as the backbone for object classification, and this backbone was chosen over others in the same category for it had shown some of the best state-of-the-art accuracy at the time; additionally, unlike some other approaches, it not only detected bounding boxes, but also classified the object in a single pass [93, 94, 95]. The modified *DarkNet53* backbone takes a combined RGB and depth sensor (RGB-D) frame as an input and outputs three main object branches with distinct features, one being for small objects (S), one for medium-sized objects (M) and, lastly, for large objects (L). In order to improve the neural network generalization rate, a dropout layer was added after each of the branches with the probability of $P(x) = 0.5$ to drop the neuron. The result of this stage is the prediction of object classes and their bounding boxes for a given input frame. Additionally, the encoded features are sent to the object segmentation branch which is needed to cut out each of the object features for latter reconstruction, and the segmentation network architecture can be seen in Figure 7; + indicates residual connections. This module uses all three *DarkNet53* outputs as an input. During the first step transposed convolutions are performed on the large object latent feature vector to upscale the features. Unlike interpolation-based upscaling methods, like bilinear and bicubic upscaling, transposed convolutions have trainable parameters which can improve the upscaled edge quality when compared to interpolation [96, 97].

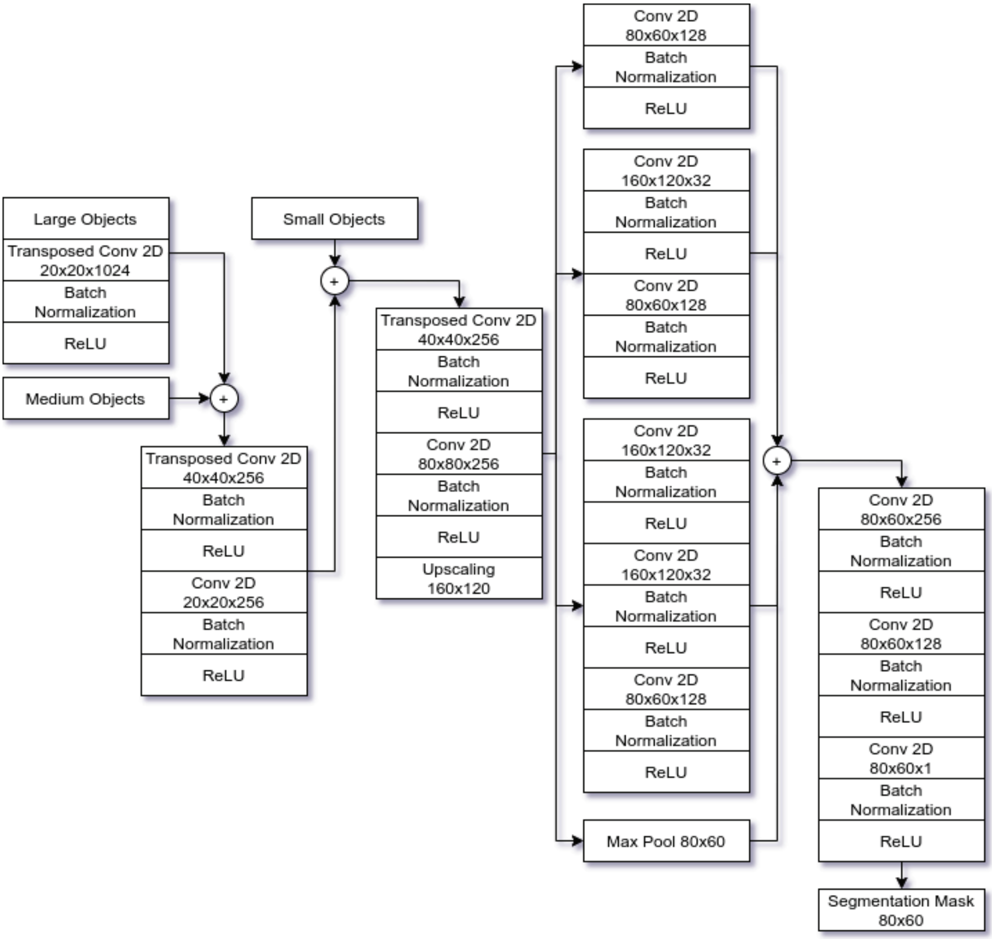


Figure 7. Object geometric segmentation architecture

Batch normalization is applied to the transposed input, for it has been proven to, in many cases, improve the convergence rate and reduce the network bias thus improving overall generalization on validation datasets without adding additional data points during the training [98, 99]. Afterwards, the non-linearity function is applied, and resulting features are combined with medium object features. The process is then repeated with small object latent features to combine them with the previous output. Following this, additional transposition is used, followed by bilinear upscaling to the correct aspect ratio, twice the size of the desired final feature map. This upscaling is performed twice, and, following this, parallel object feature extraction is applied by using the *Inception* model; it has been shown that four-level parallel feature extraction allows for each of the branches to extract separate scale features thus improving object detection [100]. Afterwards, parallel features are then combined by using residual connections, as these have shown to improve the gradient flow by skipping unnecessary connections [101].

Moreover, the paper proposes to use an improved hybridized neural network architecture, where each of the branches is trained to reconstruct different types of objects. A variational auto-encoder node was added to each of the object completion branch bottlenecks. They were chosen for their ability of suggesting non-deterministic predictions for the same input thus improving the results when given an imperfect and noisy input [102, 103]. A single variational auto-encoder branch architecture, used for reconstructing individual objects, can be seen in Figure 8.

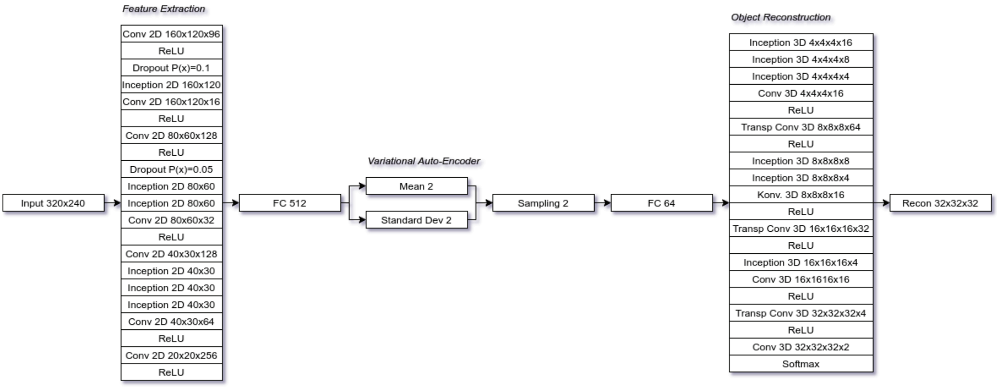


Figure 8. Example of variational auto-encoder branch used for object and its occluded side reconstruction

For object feature extraction, *Inception* architecture is used with two-dimensional convolutional kernels. Whereas, the reconstruction side of the architecture uses three-dimensional *Inception* kernels. In the paper, it was found that symmetrical variational auto-encoder models performed better than their asymmetric counterparts. The variational auto-encoder had two latent vectors and used standard deviation for random value sampling. The resulting model complexity can be seen in Table 5.

Table 5. Model complexity comparison

Method	**No. of Parameters (M)**	**No. of Operations (GFLOPs)**	**Model Size (MB)**
3D-LSTM-1	30.92	12.46	174.25
3D-GRU-1	32.21	12.51	179.45
3D-LSTM-3	30.92	35.99	274.91
3D-GRU-3	32.21	37.41	280.44
Res3D-GRU-3	35.97	69.40	539.27
Hybrid	352.35	3.09	1458.68
YoloExt	**40.14**	**7.50**	**153.15**

To train the neural network, a combination of the classification and reconstruction loss functions was used, to compare the prediction and ground truth results. The final loss function can be seen in Equation (8), which is a sum of these terms: f_d — loss between object bounding box ground truth and prediction; f_p — object confidence loss; $f_{\neg p}$ — confidence that the bounding box does not contain an object; f_k — object class loss; f_s is the object's segmentation mask loss; f_r — object reconstruction loss. Hyperparameter values of $\lambda_a = 5$ and $\lambda_b = 0.5$ were left unchanged as per suggestion in source literature [92].

$$f = \lambda_a f_d + f_p + \lambda_b f_{\neg p} + f_k + f_s + f_r \tag{8}$$

To further break down the terms, f_d is given in Equation (9) [92]; here, (x_i, y_i, w_i, h_i) are object bounding box center coordinates along with its size, S denotes the region count that partitions the image width and height axes, and *B* is the bounding box number.

$$\begin{aligned} f_{dxy} &= (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \\ f_{dwh} &= (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \\ f_d &= \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbb{1}_{ij}^{obj} [f_{dxy} + f_{dwh}] \end{aligned} \tag{9}$$

Moreover, f_p (see Equation (10)) [92] and its inverse $f_{\neg p}$ Equation (see Equation (11)) [92] show confidence loss, where f_p shows the confidence that the given region contains an object, whereas, $f_{\neg p}$ shows the confidence that a given region does not contain an object.

$$f_p = \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 \tag{10}$$

$$f_{\neg p} = \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbb{1}_{ij}^{\neg obj} (C_i - \hat{C}_i)^2 \tag{11}$$

Furthermore, f_k shows the loss between the predicted class and its ground truth, it is shown in Equation (12) [92], where n is the class count that the neural network is trained to differentiate, whereas, $p_i(j)$ and $\hat{p}_i(j)$ are the class true and prediction values, respectively.

$$f_k = \sum_{i=1}^{S^2} \mathbb{1}_{ij}^{obj} \sum_{j \in n} (p_i(j) - \hat{p}_i(j))^2 \tag{12}$$

The next loss functions component is the segmentation term f_s, it can be seen in the Equation (13). Segmentation loss shows the between the ground truth value of the segmentation and the predicted value of the segmentation mask, where W and H are the image width and height respectively, and s and $\hat{s}$ are the true and predicted pixel values.

$$f_s = \sum_{i=1}^{W} \sum_{j=1}^{H} (s_{ij} - \hat{s}_{ij})^2 \tag{13}$$

Finally, the final term f_r as seen in Equation (14), is the reconstruction loss. The reconstruction loss itself is a sum of two sub-terms, first one being KL divergence [104] used for variational auto-encoder loss calculation, and the second one being the cross-entropy loss for object reconstruction [87]. Here, l

is the latent variational auto-encoder neuron count, σ and μ are the probability distribution of the standard deviation.

$$f_r = \frac{\sum_{i=1}^{l} \sigma_i^2 - 1 + \mu_i + e^{\mu_i}}{2l} - \frac{1}{Q} \sum_{i=1}^{Q} \sum_{j=1}^{2} \alpha \log \hat{\alpha} \tag{14}$$

3.3.2. Dataset

As per the first paper, the process for creating a priori information is very similar. However, this computer model has several improvements which not only achieve better accuracy results but also permit the model to reconstruct multiple objects per scene. Therefore, the dataset needs to be modified to have data samples containing multiple objects per scene. This is done by placing $n_{objects} = [1; 10)$ objects from the *ShapeNet* dataset with random transformations within the scene. The random transformations are as follows: objects are given a random scale uniform (on all three axes) in the range of $s = [0.7, 2)$ in addition to a random rotation on the world's Up axis (z) in the range $\theta_z = [0, 2\pi)$. All objects are placed on the same plane; therefore, only the x and y coordinates are given to the object's translation. Objects are placed around it by using the following Equation (15), where $r = [-5, 5]$ and $\alpha = [0, 2\pi)$. Additionally, a check is added to determine whether any objects intersect, and if they do, a new random translation is tried until objects no longer intersect. Moreover, as the *YOLOv3* network now uses RGB information, the object's texture and material are rendered alongside. The real world validation dataset contains all test cases from the first paper in addition to new test cases captured by using a *Realsense D435i* device. When using the created dataset, the neural network was trained by using the *Adam* [90] optimizer for an average of 120 iterations. During training, it was noticed that the neural network may fall into local minimums thus negatively impacting the convergence. Therefore, a periodic learning rate function (seen in Equation (16)) has been proposed, which has allowed the model to potentially jump out of the local minimums. Here, $lr_{min} = 10^{-6}$ and $lr_{max} = 10^{-4}$ were discovered empirically. Additionally, $w_0 = 4 \times s$ and $w_1 = 2 \times s$, where s is the number of mini-batches per epoch. The first function is used to initially train the network with smaller learning rates so that to avoid potentially encountering exploding gradients [105].

$$\begin{cases} x = r \times \cos\alpha \\ y = r \times \sin\alpha \end{cases} \tag{15}$$

$$f_{lr}(x) = \begin{cases} \frac{x}{w_1} \times (lr_{max} - lr_{min}) + lr_{min}, & \text{if } x < w_0 \\ \frac{1}{\pi \mathrm{e}^3} e^{1+\pi \times \frac{\cos{(x-w_1)} mod (w_0+1)}{w_0}} \times & \text{otherwise} \\ \quad (lr_{max} - lr_{min}) + lr_{min}, & \end{cases} \tag{16}$$

3.3.3. Results

Same as per previous paper, computer model results are compared by using the *Completeness*, *Correctness* and *Quality* quantitative metrics as seen in Figure 9.

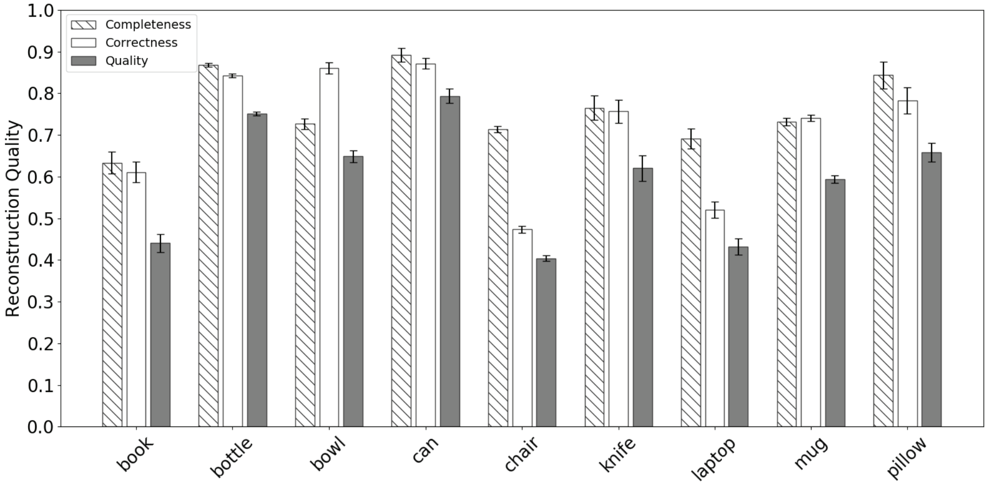


Figure 9. Quantitative reconstruction results using completeness, correctness and quality metrics

Additionally, when the approach is compared to the previous paper in Figure 11, a clear improvement in the results can be observed as most of the reconstructions increased in quality, and the shrunk error bars indicate much more stable results. Additionally, even previously poor results, due to low training dataset samples, showed an improvement. Even though there was a reduction in the speed of the reconstruction to 55.76 and 11.50 frames per second on *GTX 1070* and *GTX 960M* GPUs, respectively, the solution still remains competitive for real-time reconstruction, while improving the reconstruction accuracy and having the ability to reconstruct multiple objects per frame. The qualitative results can be seen in Figure 10, **I** is state-of-the-art [10] results for (*a*) input, **II** is the result of the previous paper, **III** is the result of the proposed new model, it can be observed that the results of the proposed network are a lot more detailed and with fewer distortions. During the research, it was noticed that, unlike *ZR300*, the *Intel Realsense D435i* device had an improved effective range of 0.3 m to 3 m thus allowing for a wider object type capture. Furthermore, there was a noticeable improvement in the clarity and reduction of *dead zones* in the depth field of the *D435i* when compared to its predecessor. However, both depth sensor types seem to suffer when capturing translucent materials, such as PET plastics. Finally, both devices exhibit a wave-like pattern slightly distorting the depth field.

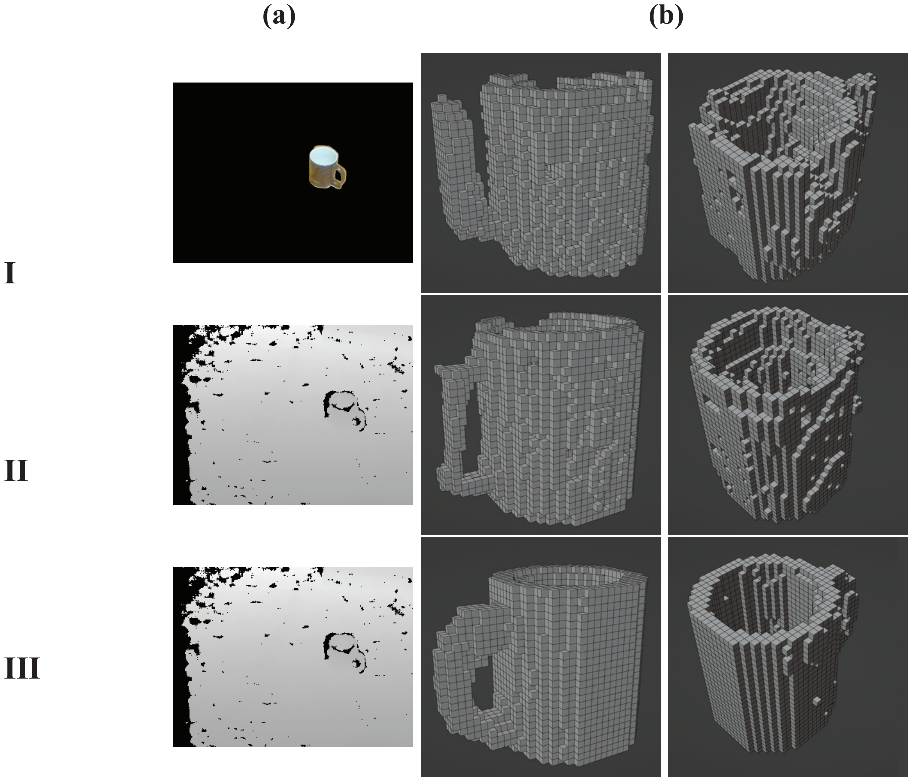


Figure 10. Model result comparison between the state-of-the-art and the proposed versions

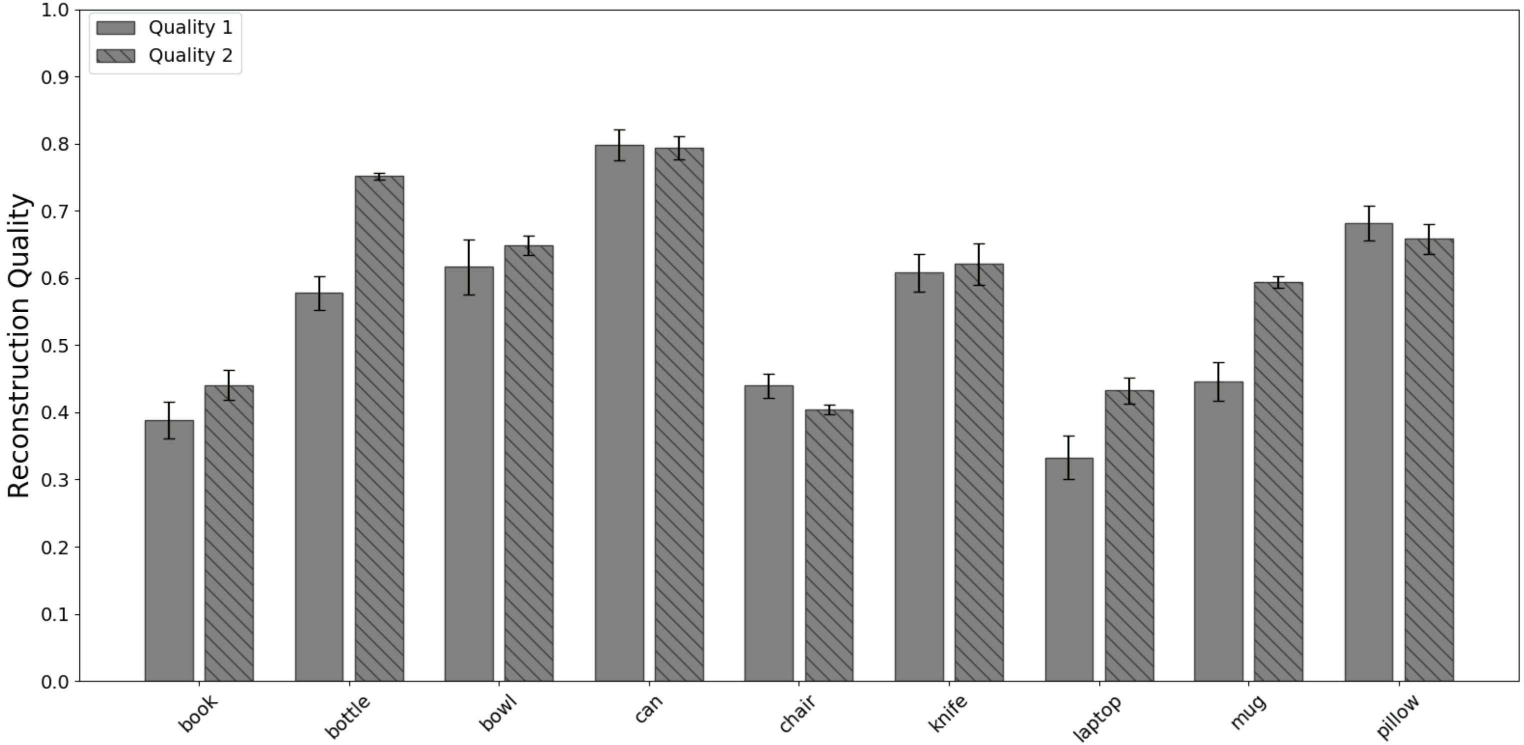


Figure 11. Comparison of the two approaches

3.4. HUMANNET—A Two-Tiered Deep Neural Network Architecture for Self-Occluding Humanoid Pose Reconstruction

The third submitted research paper is *HUMANNET—A Two-Tiered Deep Neural Network Architecture for Self-Occluding Humanoid Pose Reconstruction* by **Audrius Kulikajevas**, Rytis Maskeliūnas, Robertas Damaševičius and Rafal Scherer. The research conducted in this paper should provide a possible strategy for applying point cloud-based object completion deep neural networks for complex and dynamically morphing shapes. The author's contribution to the research is shown in Table 6.

3.4.1. Materials and Methods

The final goal of the object completion research field is the reconstruction of the entire scene with only a single imperfect perspective frame. Previous research showed that this is possible for a simple static object, thus the following research papers in this dissertation shall focus on more complex object completion from a single perspective. Whereas the two initial papers for objects and their occluded side reconstruction used voxel grids for object and its surface mesh reconstruction, the third research paper instead moved away from voxels into point clouds. Previous papers, along with state-of-the-art, used voxel grids as the base object reconstruction due to the simplistic and well-known mathematical comparison of object ground truth Y and the predicted voxel $\hat{Y}$ values. However, voxel grid-based methods suffer from a critical flaw — the inefficiency of data structure that increases the memory requirements in a geometric progression of n^3 when increasing the voxel grid density, which is required for more complex object reconstruction. This issue is a critical one as the modern hardware for training machine learning models (GPUs and TPUs) generally has much stricter operative memory allowances and cannot be sensibly expanded. Although there have been attempts to reduce the memory footprint of voxel grids with more efficient voxel grid representations, such as oct-trees [72, 106], the problems with the voxel grids do not end with the high memory requirements. Another issue that is apparent in voxel-based approaches is the homography problem, where, for an effective reconstruction algorithm, there is also the need to solve for the object transformation matrix which consists of the object's position, rotation and scale. To avoid these issues altogether, in the third paper presented in this dissertation, an unordered point cloud was used instead. Unlike voxel grids, unordered point clouds have very little memory overhead, making them ideal for training deep neural network models. However, while the memory overhead is solved with this data structure, the unstructured nature of the point clouds makes the comparison function an issue. As they

Table 6. Author contributions to the paper

Author	**Contribution**
Rytis Maskeliūnas	Conceptualization of the research direction of the conducted research.
Rytis Maskeliūnas	Proposing the methodology in order to have replicable research results.
Audrius Kulikajevas	Proposing and implementing the computer model used in the research paper.
Audrius Kulikajevas Rytis Maskeliūnas Robertas Damaševičius	Validation of the experimental results.
Robertas Damaševičius Rafal Scherer	Formal analysis of the conducted research.
Audrius Kulikajevas Rytis Maskeliūnas Robertas Damaševičius	Investigation of existing research in the field related to the research paper.
Audrius Kulikajevas Rytis Maskeliūnas	Writing of the original draft, paper modifications based on peer reviewer comments.
Robertas Damaševičius Rafal Scherer	Reviewing and minor editing of the draft for errors and language inconsistencies.
Robertas Damaševičius Rafal Scherer Audrius Kulikajevas	Graphical visualization of the experimental results.
Rytis Maskeliūnas	Overseeing and supervising the conducted research.
Rafal Scherer	Funding of the research.

are an unstructured dataset, regular ground truth comparison methods are not applicable as each of the $\hat{Y}$ values can be assigned to any of the Y values; in addition to that, multiple $\hat{Y}$ can converge on a single Y, and this subsequently creates a complexity of $O(n!)$ possible permutations. For this reason, two main methods for comparing unordered point clouds have been proposed in the scientific literature, with the first the *Chamfer* distance [107] and the other being the *Earth Mover's* distance [108], each of them having slightly different loss characteristics and sensitivity to point cloud errors. These two loss functions allow for the improvement on the mathematical complexity; however, the *Earth Mover's* distance maintains the $O(n^3 \log n)$ memory footprint thus making it not applicable to large point clouds. However, an approximation for it has been proposed, reducing the memory footprint to a linear one [109]. The main focus of the state-of-the-art research is a single simple static object's reconstruction for a given input. Therefore, the main novelty of the proposed solution is the ability to isolate and clip the region of the point cloud that will be used as input for further complex object reconstruction. To do this, similar techniques from a previous paper of the dissertation are borrowed for object isolation, where *YOLOv3* was used to find the object's bounds and to create its segmentation masks, which could then be used to cut out the region-of-interest (RoI) containing the object in question. However, instead of 2D bounding boxes for the isolation of the region in the three-dimensional point cloud, the paper presents clipping bounds. The neural network consists of two stages: clipping proposal and reconstruction. The clipping proposal neural network proposes two clipping boxes which are then used to cut out the RoI from the input point cloud. Meanwhile, the reconstruction network uses the clipped point cloud input for the complex and temporally morphing object reconstruction. This solution has managed to achieve the Jacaards index of 0.7907 for clipping boxes and 0.0256 and 0.0276 for *Chamfer* and *Earth Mover's* distance metrics for reconstruction, respectively, from as little as 50% visible points in the point cloud. The overview of the neural network architecture for the occluded object reconstruction is presented in Figure 12, where × is multiplication with the binary clipping mask and + is residual connections. A depth frame input is combined with the camera intrinsic matrix *K* to produce a point cloud. However, the produced point cloud contains 307200 vertices (640 × 480), which makes it impossible to work with on modern consumer grade parallel computing hardware. For this reason, the input point cloud is downsampled to the desired density (in this case, 2048 vertices) by using the *Farthest Point Sampling* (FPS) algorithm. Whereas previous research used images downsampled to 320 × 240, this paper no longer uses the entire image as an input, and it only uses the resampled point cloud. *Farthest point sampling* was chosen over other options for its ability to select a deterministic number of points in the point cloud where other methods, such as voxel grid-based, may provide a non-concrete number of selected points [110]. Moreover, FPS is also resistant to uneven point distributions in the point cloud, e.g., uniform sampling would be biased towards the highest concentration of points [111, 112]. The resulting downsampled point cloud is then used as an input for feature extraction layers. For the models featuring the extraction structure, the *PointNetFeat* architecture was used, which showed great results when dealing with unstructured data inputs, namely, point clouds, by extracting 64, 128 and 1024 features while using single dimension convolutional kernels for best feature selection [68]. Whereby, following this action, the 1024 neuron feature map is compressed into two by applying the adaptive max pooling function, and this allows the network to filter for the two most prominent latent feature vectors that will act as the clipping boxes [113]. The clipping box is a regression task which can have both positive and negative values; for this reason, the final activation function uses uses the hyperbolic which is provided in Equation (17).

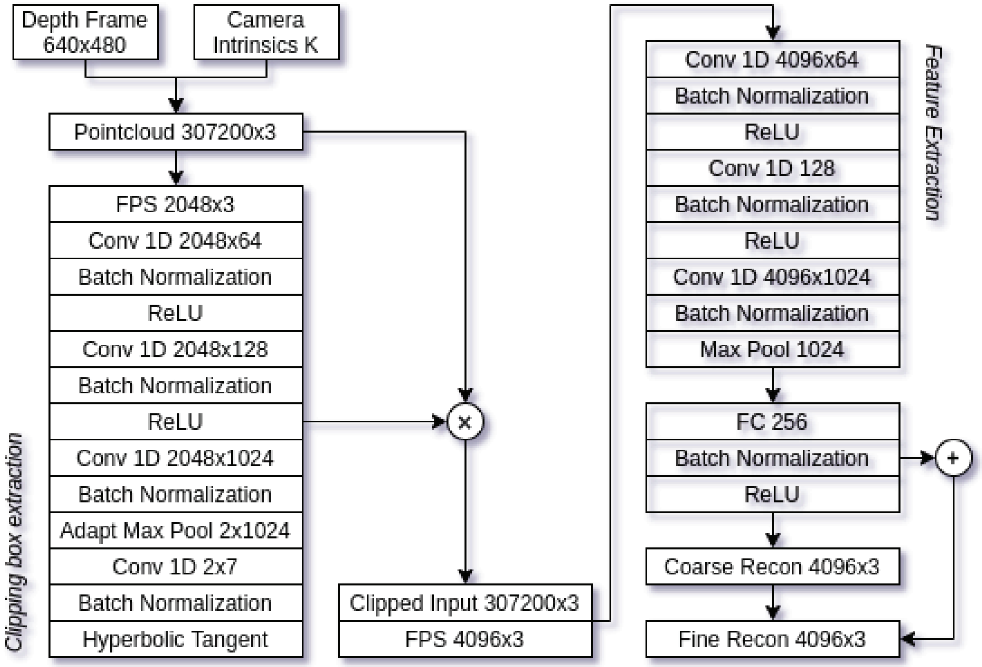


Figure 12. Proposed temporally morphing complex object reconstruction neural network architecture overview

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{17}$$

Afterwards, from the two resulting clipping boxes, the one with the higher confidence is used to filter out points that are within the clipping region. The resulting RoI is then downsampled into the desired density, which is 4096 in the paper. The paper suggests this reclipping methodology to solve for high density point clouds, as, otherwise, it would not be feasible to work with them. However, when resampling the entire input objects which do not belong to the RoI, it would also be counted towards the given vertex allowance. This solution allows the *HumanNet* to retain much higher density inputs isolated from the background noise to be used in reconstruction, thus allowing for complex object reconstruction. The clipped input is then ran through a separate feature extraction bottleneck, again using the *PointNetFeat* bottleneck, after which, coarse reconstruction is applied. Following coarse reconstruction, residual connections and coarse features are used for finer object reconstruction [69, 71]; the reconstruction neural network branch architecture can be seen in Figure 13, yet, for brevity, only one of the *Random Grid Reconstruction* branches is shown. The extracted latent features are then used as an input in random grid patch-based reconstruction where 16 branches are constructed, one for each reconstruction patch. The 16 patches perform a coarse object's point cloud reconstruction, the coarse reconstruction is then combined by using residual connections with the input features. The residual connections act as a guide in creating a finer point cloud reconstruction output, and the complexity of the network is shown in Table 7.

Table 7. Model complexity comparison

Method	**No. of Parameters (M)**	**No. of Operations (GFLOPs)**	**Model Size (MB)**
PointNet w/ FCAE	7.43	1.18	28.36
PCN	6.87	29.5	26.25
AtlasNet	3.31	6.46	12.66
MSN	29.50	12.89	112.89
HumanNet	**29.71**	**11.74**	**112.94**

To train the neural network, as described in the research paper, a two-term loss function is used as described by Equation (18). Here, f_k is the clipping loss and f_r is conditional reconstruction loss, which is only applied when the clipping loss is below threshold $f_k \leq 0.3$.

$$f = f_k + \mathbb{1}\{f_k \leq 0.3\} f_r \tag{18}$$

The three-dimensional region of interest clipping loss f_k is defined by Equation (19). Here, o and $\hat{o}$ are the object's center ground truth and prediction values, meanwhile, s and $\hat{s}$ is its scale, y_c is the confidence that an object exists in the bounding box, n is the allowed clipping box count.

$$\begin{aligned} f_{dp} &= y_c \sum_{i=1}^{n} L1_s(o_i, \hat{o}_i) \\ f_{ds} &= y_c \sum_{i=1}^{n} L1_s(s_i, \hat{s}_i) \\ f_{bce} &= \frac{1}{n} \sum_{i=1}^{n} y_c \cdot \log \hat{y}_c + (1 - y_c) \cdot \log (1 - \hat{y}_c) \\ f_k &= f_{dp} + f_{ds} + f_{bce} \end{aligned} \tag{19}$$

$L1_s$ denotes the smooth loss function, as defined by Equation (20) [114] unlike the least absolute deviation (*L1*), smooth *L1* has lower sensitivity to data outliers, and this can prevent gradient explosion and divergence [114].

$$\begin{aligned} z_i(y_i, \hat{y}_i) &= \begin{cases} \frac{(\hat{y}_i - y_i)^2}{2\beta}, & \text{if } |\hat{y}_i - y_i| < \beta \\ |\hat{y}_i - y_i| - 0.5\beta, & \text{otherwise} \end{cases} \\ L1_s(y, \hat{y}) &= \frac{1}{n} \sum_{i=1}^{n} z_i \end{aligned} \tag{20}$$

The next loss term is the reconstruction, as defined by Equation (21) [71], which contains three sub-terms. The first two terms are f_{emd} (Equation (22)) [71], which compare the reconstruction quality of coarse and fine point clouds against ground truths; meanwhile, f_{exp}, as defined by Equation (23) [71], is a constraint term that prevents points from the same patch from moving too far away from each other, as per the original paper in which the constraint is suggested, and the constraint value is set to $\alpha = 0.1$ [71].

$$f_r = f_{emd}(S, \hat{S}_{fine}) + f_{emd}(S, \hat{S}_{coarse}) + \alpha f_{exp} \tag{21}$$

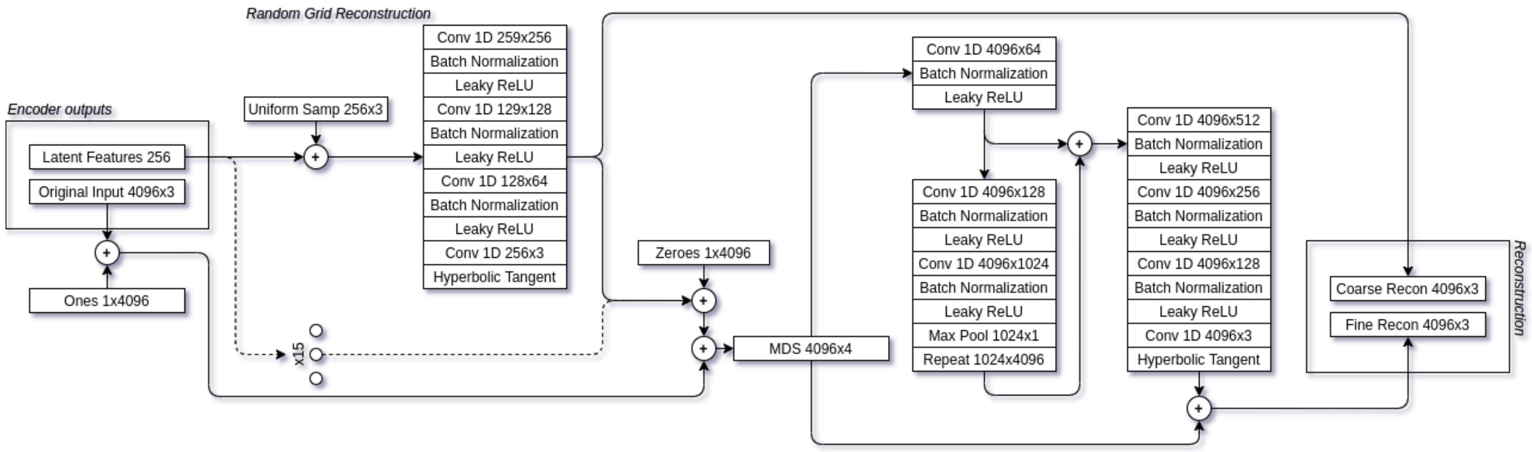


Figure 13. Object and its occluded side reconstruction networks decoder architecture

$$f_{emd} = \min_{\phi:S \to \hat{S}} \frac{1}{|S|} \sum_{x \in S} ||x - \phi(x)||_2 \tag{22}$$

$$f_{exp} = \frac{1}{KN} \sum_{1 \leq i \leq K} \sum_{(u,v \in \tau_i)} \mathbb{1}\{d(u,v) \geq \lambda l_i\} d(u,v) \tag{23}$$

3.4.2. Dataset

In the first two papers, the main focus of reconstruction was simple static object reconstruction because the focus of the research was expanded to more complex objects, e.g., humanoid shapes in addition to improving the reconstruction fidelity by moving away from voxel grid-based approaches to point cloud-based approaches. To achieve this, a new type of the dataset is required. A new synthetic dataset was created by using the *MoVi* [115] dataset which contains a large amount of motion capture data from multiple camera perspectives as a base for the humanoid posture. However, because the approach described in the paper requires depth sensor information, which the *MoVi* dataset lacks, the captured information is skinned onto triangle meshes provided in the *AMASS* [116] dataset. Similarly to the first paper (*Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from ShapeNetCore Dataset*), the dataset is generated by placing an object inside a scene and rendered from various angles by using only depth information. For every motion capture recording, both given male and female models are bound and placed within a scene imitating a room with a wall behind the user. The subject model is being rotated around in the range $\theta_{subject} = [0°, 360°)$ 45° increments, while the camera is rotated around the subject in the range of $\theta_{camera} = [-35°, 35°]$ in 15° increments. Finally, a ground truth point cloud is generated from the polygon mesh; this was done by sampling 4096 surface points while using uniform sampling. To train the network, the *Adam* [90] optimizer on the generated dataset was used with the initial learning rate of $lr = 10^{-3}$ discovered experimentally. Additionally, once the loss plateau [117] has been reached and the loss has not improved in 10 iterations, the learning rate is reduced by a factor of 10. This reduction on the plateau is done until the model converges, which is typically 170 epochs.

3.4.3. Results

Evaluation of the unordered point cloud approaches is not possible by using the same quantitative evaluation metrics as with the voxel grids. For this reason, two main metrics were chosen for the evaluation of the point cloud approach;

these metrics are EMD (Equation (22)) and CD (Equation (24)) [118]. They were chosen for being used as a benchmark in the majority of state-of-the-art point cloud reconstruction evaluations.

$$f_{cd} = \frac{1}{2}\left(\frac{1}{|S|}\sum_{x\in S}\underset{y\in\hat{S}}{min}\|x-y\|_2^2 + \frac{1}{\hat{S}}\sum_{y\in\hat{S}}\underset{x\in S}{min}\|x-y\|_2^2\right) \tag{24}$$

In the experiments, the *MoVi* dataset is used which is comprised of recordings containing subjects performing various actions and poses. These recordings are used as a basis for the training and validation datasets, where every 75th frame is used from the recording in the processed dataset. This is done to reduce the amount of similar, and, in some cases, nearly frames. Even though the *MoVi* dataset contains motion capture information, it is missing depth frames. To solve this, the motion capture poses are bound to the *AMASS* dataset provided male and female meshes generating depth maps for frames. Once the dataset has been prepared, it is split as 80:20 to separate the dataset, where 80% of the frames are used for the network training and 20% for the validation. Once the network has been trained, the validation dataset is used to evaluate the results quantitatively on each of the frames individually. To evaluate the model results, they are separated based on the exercise/pose the subject has performed (see Figure 14) and by the subject's gender (see Figure 15).

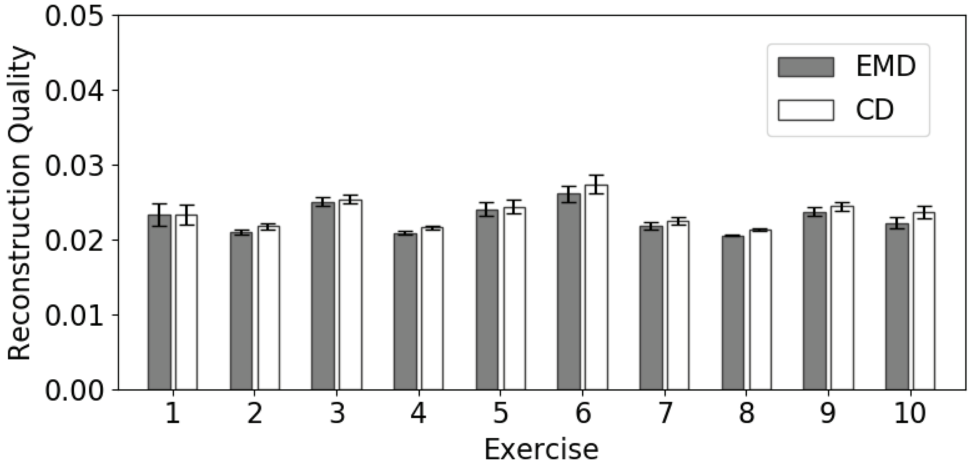


Figure 14. Reconstruction results evaluated based on exercise

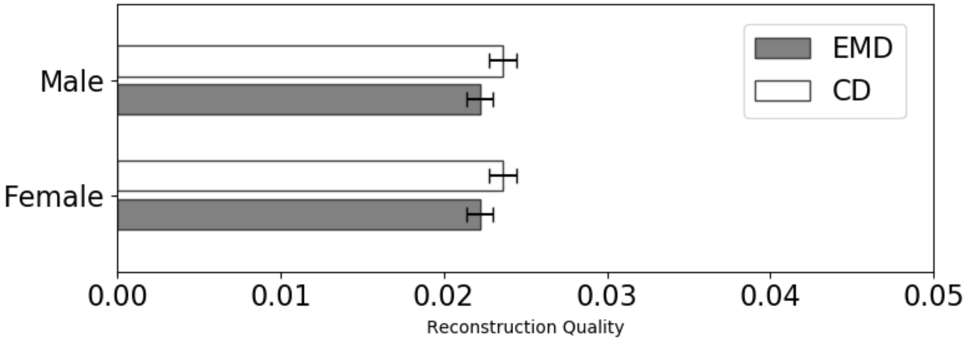


Figure 15. Reconstruction results evaluated based on gender

As the results show, there are no clear disparities between the poses; whereas, small error bars indicate that there are no outliers in the data samples. This indicates that the machine learning model can approximate the human body pattern. Meanwhile, the similarity between the reconstruction results for both genders indicates that not only has the approach managed to generalize the human body pattern by reconstructing such features as limbs, it has also been able to discern the body shape by reconstructing such features as secondary sex characteristics. In addition to this, the achieved quantitative metrics (0.0256 EMD and 0.0276 CD) have shown to be on par or exceed the state-of-the-art research (see Table 8) while providing the novelty of reconstructing complex and temporally morphing objects; whereas, state-of-the-art deals only with single static object reconstruction. *ShapeNet* quantitative metrics are self-reported by authors, *AMASS* metrics were calculated by using the same dataset that was created for this dissertation.

Table 8. Comparison between different methods

Method	**ShapeNet**		**AMASS**	
	EMD	**CD**	**EMD**	**CD**
PCN [119]	0.0734	0.0121	3.0456	4.0955
AtlasNet [120]	0.0653	0.0182	2.0875	6.4343
MSN [109]	0.0378	0.0114	1.1525	0.8016
HumanNet	—	—	0.0256	0.0276

3.5. Auto-Refining Reconstruction Algorithm for Recreation of Limited Angle Humanoid Depth Data

The fourth and final submitted research paper is *Auto-Refining Reconstruction Algorithm for Recreation of Limited Angle Humanoid Depth Data* by **Audrius Kulikajevas**, Rytis Maskeliūnas, Robertas Damaševičius and Marta Wlodarczyk-Sielicka. The research conducted in this paper should improve the field of object and its occluded region completion by applying unsupervised deep adversarial auto-refining neural networks for a complex object point completion from a single imperfect non-synthetic depth frame without known ground truth for network training. The author's contribution to the research can be seen in Table 9.

3.5.1. Materials and Methods

The main knowledge gap that remains even in the state-of-the-art methods is the ability to reconstruct from real structured light and laser depth sensors; whereby, other research focuses on reconstruction from synthetic datasets only. In the fourth paper, *Auto-Refining Reconstruction Algorithm for Recreation of Limited Angle Humanoid Depth Data* that is presented in the dissertation, a method for reconstructing an object and its occluded regions has been proposed by applying unsupervised deep adversarial refining neural networks which can clean-up the noisy sensor input, thus making them similar to synthetically generated data points without losing the original input features. The main novelty of the proposed computer model is the application of adversarial neural networks for the refinement of the input point cloud. Unlike widely known applications for adversarial neural networks where they are used for the generation of non-existing images [121] or even sound and text-to-voice synthesis [122], the paper proposed a way to train the adversarial neural network for cleaning up the input while using fully unsupervised adversarial training for the cleanup of the real world data point input. This is done in order to make it similar to a synthetic dataset thus removing the need for ground truth values during the reconstruction step. This methodology was developed as a solution to the problem of imperfections in depth sensors frames. Both structured light and laser depth sensors provide output of very noisy and distorted frames which even the state-of-the-art methods can fail to account for. When comparing this solution to that of state-of-the-art (see Table 10), it can be observed that the computer model described by the paper outperforms the other by combining such features as point cloud reconstruction for higher resolution reconstructions, *Earth Mover's* distance for better point cloud reconstruction quality, and, finally, unlike most other methods, it can work with real world frames as opposed to only synthetic

Table 9. Author contributions to the paper

Author	**Contribution**
Rytis Maskeliūnas	Conceptualization of the research direction of the conducted research.
Rytis Maskeliūnas	Proposing the methodology in order to have replicable research results.
Audrius Kulikajevas	Proposing and implementing the computer model used in the research paper.
Audrius Kulikajevas Rytis Maskeliūnas Robertas Damaševičius Marta Wlodarczyk-Sielicka	Validation of the experimental results.
Rytis Maskeliūnas Robertas Damaševičius	Formal analysis of the conducted research.
Audrius Kulikajevas Rytis Maskeliūnas	Investigation of existing research in the field related to the research paper.
Audrius Kulikajevas Rytis Maskeliūnas	Writing of the original draft, paper modifications based on peer reviewer comments.
Robertas Damaševičius Marta Wlodarczyk-Sielicka	Reviewing and minor editing of the draft for errors and language inconsistencies.
Audrius Kulikajevas Rytis Maskeliūnas	Graphical visualization of the experimental results.
Rytis Maskeliūnas	Overseeing and supervising the conducted research.
Marta Wlodarczyk-Sielicka	Funding of the research.

ones.

In addition to this, the proposed point cloud refining process solves this issue without having ground truth values for the reconstructed objects with fully unsupervised training. The refining output stage is a latent feature vector along with the cleaned-up point cloud, which is then used for object reconstruction by applying further feature extraction and concatenation with the previously selected latent feature vector. The result of which is then used for the coarse and fine object and its invisible side reconstruction by using residual connections to connect coarse and refined feature vectors for the final reconstruction. After performing quantitative analysis, it was found that the suggested neural network architecture has achieved the *Earth Mover's* and *Chamfer* distance metrics of 0.059 and 0.079, respectively. These values put them on the same level in terms of the reconstruction quality with state-of-the-art, while providing a novel way of reconstructing noisy and distorted real world depth sensor inputs without having ground truth values for training on real objects. The overview of the method is shown in in Figure 16.

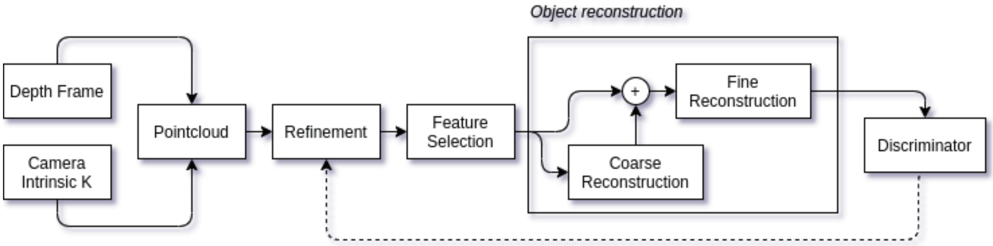


Figure 16. Object refinement and its invisible side reconstruction computer model overview

The input depth field is combined with the depth sensor intrinsic camera matrix K to produce a point cloud which is then used during the refinement step. Following this, similarly to a previous paper (*HUMANNET—A Two-Tiered Deep Neural Network Architecture for Self-Occluding Humanoid Pose Reconstruction*), coarse-to-fine object reconstruction is performed. The output of this process is the fine detail point cloud that is then shown to the discriminator in order to evaluate it by guessing if the reconstruction came from the real world dataset or a synthetic one, the result of which is then subsequently used to update the refiner for it to learn to deceive the discriminator.

For the object cleanup and refinement stage, a very similar architectural solution to the coarse point cloud reconstruction is used. The network branch

Table 10. Comparison of method properties in relation to state-of-the-art

Name	**Voxels**	**Point cloud**	**Input**	**EMD**	**CD**	**Non-Synthetic**
3D-R2N2	✓	✗	RGB	—	—	✗
YoloExt	✓	✗	RGB-D	—	—	✓
PointOutNet	✗	✓	RGB	✓	✓	✗
PCN	✗	✓	Point cloud	✗	✓	✗
AtlasNet	✗	✓	Point cloud	✗	✓	✗
MSN	✗	✓	Point cloud	✓	✗	✗
HumanNet	✗	✓	Depth	✓	✗	✗
Auto-Refine	✗	✓	Depth	✓	✗	✓

structure is shown in Figure 17; for brevity, only one of the parallel branches is displayed. For a given input point cloud, the most influential latent feature vector is extracted, from which, a random grid reconstruction is performed by using 8 patches. The resulting output is a cleaned-up original input and a previously extracted feature vector. Following this, further feature extraction from the reconstructed point cloud is performed. This is done separately because only the refined point cloud is used in the adversarial training. The newly extracted features are then concatenated with the extracted input features and passed on through along with the refined point cloud to the occluded region reconstruction branch seen in Figure 18; for brevity, only, one of the parallel branches is displayed.

The resulting fine point cloud reconstruction is then used as an input in the discriminator branch. The discriminator attempts, as seen in Figure 19, to classify the synthetic data point reconstruction as (1) and the real world reconstruction as (0). Whereas, the refinement neural network branch attempts to deceive the discriminator, by using its inverse loss. This creates a competition between the refiner and the discriminator networks. The resulting model complexity for both the reconstruction and the reconstruction with the discriminator branch is shown in Table 11.

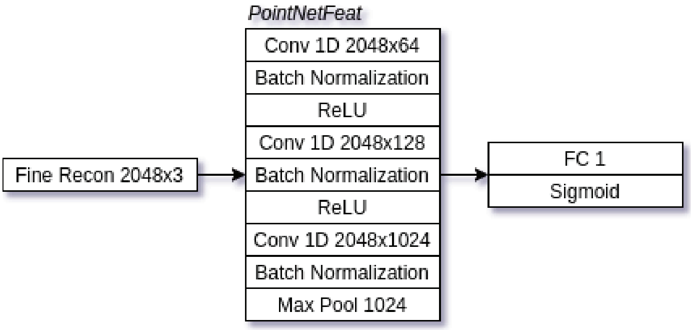


Figure 19. The discriminator that is used to determine if the point cloud is real or synthetic

Due to the complexity of the neural network architecture, it is difficult to train it in a single stage; for this reason, a four-staged training methodology is devised for the unsupervised adversarial network training of the computer model.

Phase I. Auto-encoder During the first phase, the neural network is trained to take an input and return the same input as an output. This is generally referred

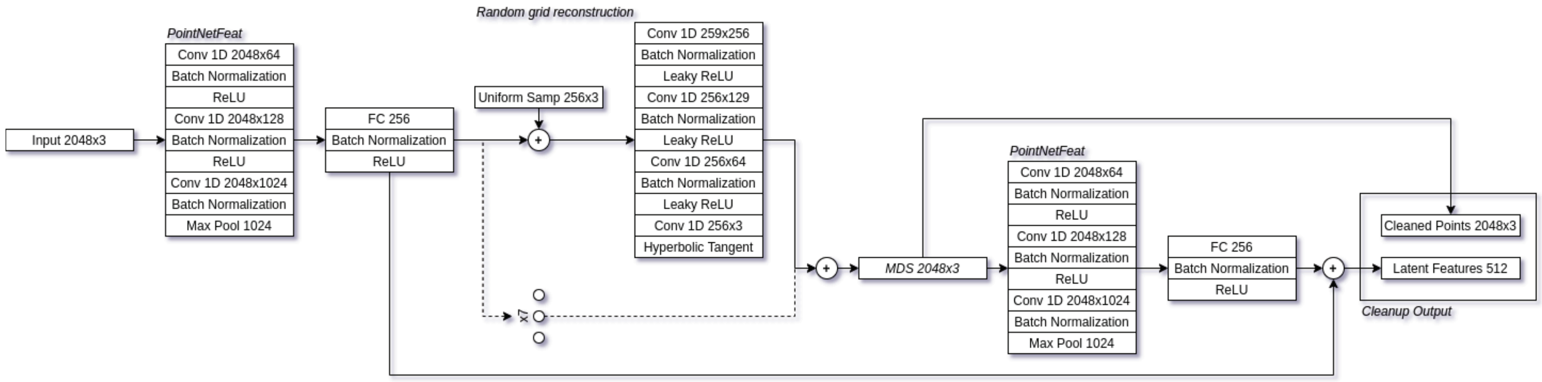


Figure 17. Object input refinement stage

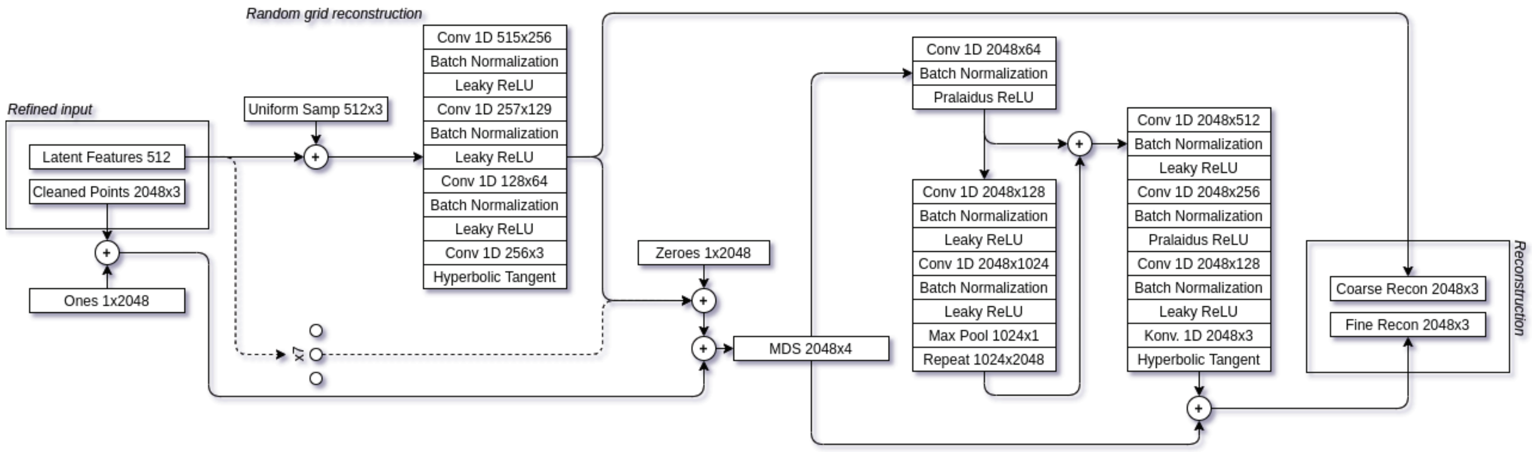


Figure 18. Coarse-to-fine object reconstruction from refined input neural network architecture

Table 11. Model complexity comparison

Method	**No. of Parameters (M)**	**No. of Operations (GFLOPs)**	**Model Size (MB)**
PointNet w/ FCAE	7.43	1.18	28.36
PCN	6.87	29.5	26.25
AtlasNet	3.31	6.46	12.66
MSN	29.50	12.89	112.89
HumanNet	29.71	11.74	112.94
Auto-Refining	**6.93**	**7.51**	**23.12**
Auto-Refining w/ Discriminator	**7.51**	**7.65**	**23.62**

to as auto-encoders, as these are able to reconstruct the output from a reduced dimensionality input. During this phase, Equation (25) is used as the training loss metric which is a derivative of Equations (22) and (23). This phase allows us to increase the training of the refinement phase by training coarse and fine reconstruction branches to act as auto-encoders; whereas, the training refiner alone greatly reduces the network convergence rate.

$$\begin{aligned} f_{\phi 1} = f_{emd}(S_{clean}, \hat{S}_{clean}) + f_{emd}(S_{clean}, \hat{S}_{coarse}) + \\ f_{emd}(S_{clean}, \hat{S}_{fine}) + \\ \alpha(f_{exp}(S_{clean}, \hat{S}_{clean}) + f_{exp}(S_{clean}, \hat{S}_{fine})) \end{aligned} \tag{25}$$

Phase II. Reconstruction The second phase is the coarse and fine reconstruction network branch reconstruction training phase. During it, coarse and fine branches are trained on synthetic input to make a prediction which would result in reducing the loss between the prediction and the ground truth, with the loss being defined by Equation (26).

$$\begin{aligned} f_{\phi 2} = f_{emd}(S_{clean}, \hat{S}_{clean}) + f_{emd}(S, \hat{S}_{fine}) + \\ f_{emd}(S, \hat{S}_{coarse}) + \\ \alpha(f_{exp}(S_{clean}, \hat{S}_{clean}) + f_{exp}(S_{clean}, \hat{S}_{fine})) \end{aligned} \tag{26}$$

Phase III. Discriminator During this phase, the training of the discriminator is performed to classify the fine reconstruction output as either synthetic (1) or real (0). The discriminator is trained in a separate stage, as it was found that training with the initial reconstruction seed values can cause it to have bad gradients or even potentially explode and diverge. As the loss metric binary cross-entropy is used which is defined by Equation (27) [123].

$$f_{\phi 3} = \sum_{i=1}^{N} \hat{y}_i \cdot log(y_i) + (1 - \hat{y}_i) \cdot log(1 - y_i) \tag{27}$$

Phase IV. Adversaries The fourth and final phase combines the training of the refiner, coarse-to-fine reconstruction and discriminator branches. Here, three-step training is applied. During the first step, Equation (28) is used as a loss metric for the reinforcement of the already existing reconstruction weights. The second stage uses the inverse discriminator loss as seen in Equation (29) with the constraint of $\gamma = 0.4$; this allows the real inputs to retain their features. The final step reinforces the discriminator using Equation (27) as a loss metric.

$$f_{\phi 4a} = f_{\phi 2} + f_{\phi 3} \tag{28}$$

$$\begin{gathered} f_{\phi 4b} = \gamma f_{emd}(S_{clean}, \hat{S}_{clean}) + \\ \alpha f_{exp}(S_{clean}, \hat{S}_{clean}) + f_{\phi 3}(1 - y, \hat{y}) \end{gathered} \tag{29}$$

3.5.2. Dataset

In this research, two sets of data are required: synthetic and real world. The synthetic dataset is created in a very similar manner to the previous research paper (*HUMANNET—A Two-Tiered Deep Neural Network Architecture for Self-Occluding Humanoid Pose Reconstruction*). However, the key difference between the two datasets being a synthetic dataset in this research was reduced to 2048 points per point cloud. Any additional points were found to have virtually no effect on the reconstruction accuracy, thus a smaller point cloud allows for a less memory intensive model during the training process. Additionally, a real world dataset was introduced, which was recorded by using *Intel Realsense D435i* and *Intel Realsense L515* depth sensors. The real world dataset consists of 168 different recordings from three subjects performing various exercises. *D435i* was placed to the right side of the subject, meanwhile, the *L515* sensor was placed directly in front of the subject. This recording gives two different depth field errors that the computer model needs to account for due to the different type of the depth sensor characteristics. Each of the subjects has performed the following exercise tasks: 1) shoulder flexion; 2) shoulder flexion and internal rotation; 3) shoulder flexion and internal rotation, elbow flexion; 4) shoulder extension and internal rotation; 5) shoulder flexion, wrist on the back; 6) full shoulder flexion; 7) shoulder adduction. Each of the exercises was performed four times totaling 28 recordings of each subject from each of the cameras. By using this dataset, the network is trained by using the *Adam* optimizer [90] with a learning rate of $lr_{max} = 10^{-4}$ and $lr_{min} = 10^{-5}$ in addition to cosine annealing with warm restarts [117] with a period of 50 epochs; this allows the network to jump out of the potential local minimums that can be caused by real world dataset inputs. The model converges on average in 300 epochs.

3.5.3. Results

Similarly to the previous paper, both EMD and CD metrics are used to evaluate the object reconstruction results quantitatively. Firstly, the findings are evaluated by the performed exercise as shown in Figure 20. Secondly, by gender, the results are presented in Figure 21. When comparing the error bars to those of the previous paper, it can be seen that the error bars are also slightly higher relative to the metric mean, which indicates that certain reconstruction angles may act as outliers.

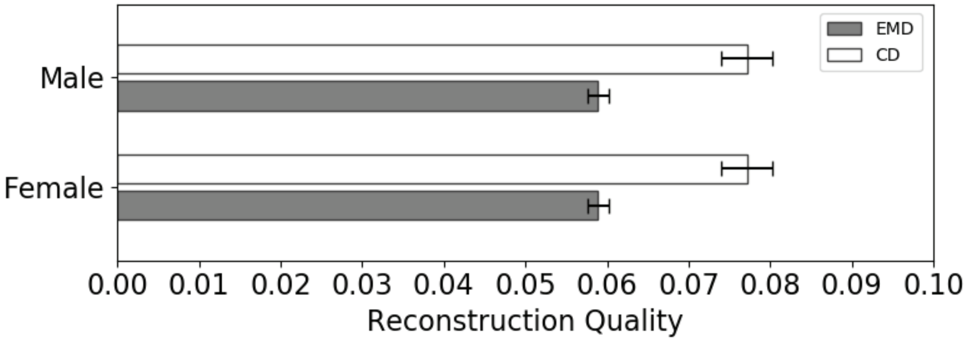


Figure 21. Reconstruction results evaluated based on gender

From the results, assertions can be made that the unsupervised deep adversarial refining neural network can approximate the human pattern as there are no disparities either by exercise or by gender. While the results have slightly lower average EMD and CD values of 0.059 and 0.079, respectively, when compared to the *HUMANNET—A Two-Tiered Deep Neural Network*

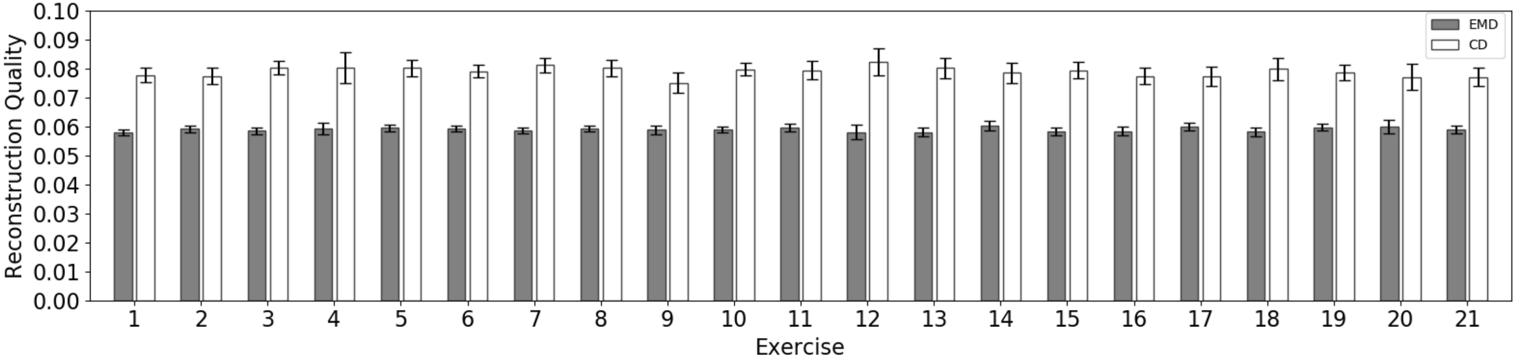


Figure 20. Reconstruction results evaluated based on exercise

Architecture for Self-Occluding Humanoid Pose Reconstruction paper, they are still competitive with those of the state-of-the-art (see Table 12), while additionally fully reconstructing occluded object sides from noisy and distorted ones, structured light or a laser sensor, real world input with very few defects. *ShapeNet* quantitative metrics are self-reported by authors, *AMASS* metrics were calculated by using the same dataset that was created for this dissertation.

Finally, from the results of qualitative experiments, it is possible to draw an assumption that the proposed neural network can reconstruct expert identifiable probable occluded object regions; meanwhile, the quantitative results show that this assumption holds true in the entire validation dataset; from this, it is possible to confirm the hypothesis that machine learning approaches, similarly able of pattern matching as a human, are a tool adept of occluded side object reconstruction. Unlike the first research paper (*Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from ShapeNetCore Dataset*), this paper omitted the use of *Intel Realsense ZR300* for it not only provides depth field results on larger objects, but it was also discontinued, thereby making research on it obsolete. Instead, *D435i* and *L515* depth sensors were used. During the research, it was noticed that, while *L515* usually provides a much cleaner and less distorted depth field, it seemed to suffer from holes in the depth field depending on the fabric the person is wearing, unlike *D435i* which did not have issues with clothing. This is likely because the *L515* laser emitter beam is more likely to be absorbed by the fabric itself, meanwhile, *D435i* uses both visible light and infrared sensors to infer the object depth field, thereby making it more robust against various materials.

Table 12. Comparison between different methods

Method	**ShapeNet**		**AMASS**	
	EMD	**CD**	**EMD**	**CD**
PCN [119]	0.0734	0.0121	3.0456	4.0955
AtlasNet [120]	0.0653	0.0182	2.0875	6.4343
MSN [109]	0.0378	0.0114	1.1525	0.8016
HumanNet	—	—	0.0256	0.0276
Auto-Refining	—	—	0.0590	0.0790

4. CONCLUSIONS

1. This dissertation has proposed and implemented a computer model based on deep hybrid neural networks for smooth polygonal mesh reconstruction achieving 151 frames per second from a single camera perspective thus improving state-of-the-art object reconstruction metrics from a single perspective by 19.3%, without requiring an object segmentation mask for a single real world object reconstruction, which makes the approach applicable for real time problems.

2. This dissertation has proposed a modification to the hybrid neural network model which has improved the reconstruction results by 8.53% relative to the initial hybrid model, in addition to performing maskless multiple object reconstruction from a single perspective frame.

3. This dissertation has proposed and implemented a computer model alongside its application strategy which has used the unordered point data structure as opposed to the voxel grid and can reconstruct multiple complex and temporally morphing objects per scene with the $f_{emd} = 0.0256$ and $f_{cd} = 0.0276$ evaluation metrics, which exceeds or is on par with other state-of-the-art methods, such as PCN, AtlasNet and MSN.

4. This dissertation has proposed and implemented a computer model for reconstructing an object and its invisible sides from a single distorted and noisy perspective input by using unstructured light and laser sensor depth frames as an input at competitive $f_{emd} = 0.059$ and $f_{cd} = 0.079$ quantitative metrics without the need for ground truth for the training of the reconstruction by applying unsupervised deep adversarial refining neural networks.

5. During qualitative evaluation, it was concluded that the models make probable reconstructed object predictions; quantitative evaluation metrics of $f_{emd} = 0.059$ and $f_{cd} = 0.079$ have shown that qualitative results hold true throughout the entire validation dataset; therefore, the hypothesis that machine learning approaches, similarly able of pattern matching as a human, is a tool adept of occluded side object reconstruction has been tested and confirmed.

5. FUTURE WORKS

The proposed human point cloud completion methods are focused on the *naked* subject point clouds, i.e., they are not fit for completing the subject's clothing or accessories. Further research is being conducted in order to reconstruct the worn clothing. Additionally, the point cloud completion method has not proposed a solution for restoring the object's surface mesh. However, such solutions are being developed in the future research. They would allow for full integration into the already existing three-dimensional systems. Finally, the currently available solutions do not yet attempt to complete the object's material; further investigation into this would allow this type of solution to each an even broader real-world audience in terms of application in the daily life, e.g., teleconferencing or the entertainment industry.

6. SUMMARY

6.1. Įvadas

6.1.1. Tiriama problema

Darbo problema — nematomų objekto zonų atkūrimas naudojant vienos kameros perspektyvos vaizdą.

Pagrindiniai šiomis dienomis praktikoje taikomi objekto nematomų zonų atkūrimo metodai — tai klasikiniai algoritmai, naudojantys daugelio kamerų perspektyvų sujungimo metodus tiek naudojant individualių kamerų vaizdus, tiek iteracinius objekto atkūrimo metodus [5, 6, 7, 8, 9]. Tačiau šie metodai dažnai pasižymi prastomis atkūrimo charakteristikomis, pavyzdžiui, atkūrimas ties objektų susijungimo kraštinėmis [25]. Kita kylanti problema — iteracinių bei kitų taškinių debesų atkūrimo metodai gali būti sunkiai prieinami dėl reikalingų kaštų keletui specializuotų jutiklių įsigyti [29]. Galiausiai, iteraciniai metodai, tokie kaip *SLAM*, pasižymi skaičiavimų kompleksiškumu, tai juos daro sunkiai pritaikomus praktikoje realaus laiko sistemose, todėl reikalingos įvairios optimizacijos [26, 27, 28].

Dėl šių priežasčių yra reikalingi atsparūs objektų nematomų zonų atkūrimo metodai, gebantys atkurti objektus naudojant vieną objekto perspektyvą.

6.1.2. Tyrimų tikslas

Pagrindinis tyrimų tikslas — sukurti mašininio mokymo modelį, gebantį iš vieno netobulo gylio jutiklio kadro perspektyvos atkurti objektą bei jo nematomas zonas.

6.1.3. Darbo uždaviniai

Darbui įgyvendinti suformuluoti trys uždaviniai:

1. Išanalizuoti egzistuojančius tiek klasikinius, tiek mašininio mokymo sprendimus, susijusius su objekto bei jo nematomų zonų atkūrimu, esant tiek vienos, tiek keleto perspektyvų kadrams.

2. Pasiūlyti bei realizuoti modelį ar modelius, gebančius atkurti įtikinamas, nematomas ar uždengtas objekto zonas iš vienos objekto perspektyvos.

3. Sukurti gilaus mokymo modelį, gebantį atkurti nematomas objekto zonas esant vienam netobulam gylio kadrui, išgaunamam tiek iš struktūrizuotos šviesos, tiek lazerinių jutiklių šaltinių, bei jo pritaikymo metodologiją.

6.1.4. Praktinė darbo reikšmė

Pasiūlyti bei įgyvendinti modeliai objekto nematomoms zonoms atkurti iš vienos perspektyvos netobulo gylio kadro. Pasiūlyti modeliai gebėjo atkurti vieną arba keletą tiek paprastų, tiek sudėtingų ir laike kintančių objektų, matomų kadre realiuoju laiku. Taip pat buvo pasiūlytas sprendimas išmokyti giliuosius neuroninius tinklus atkurti objektus iš vienos perspektyvos, neturint tikrųjų reikšmių jo mokymo funkcijai. Eksperimentai bei rezultatai buvo paviešinti aukštai vertinamuose atvirai prieinamuose žurnaluose [73, 74, 75, 76].

6.1.5. Naujumas

Tyrimų rezultatai — gilaus mašininio mokymo modeliai, pagrįsti neuroninių tinklų veikimu, gebantys atkurti tiek paprastus, tiek sudėtingus objektus naudojant vieną gylio jutiklio perspektyvos kadrą. Kiekviename straipsnyje buvo siūlomi sprendimai, sumažinantys likusias naujausių sprendimų, susijusių su objektų atkūrimu, spragas. Disertacijos metu atliktų tyrimų indėlis: objektų atkūrimas realiuoju laiku bei greitas atkūrimo tinklo praplėtimas naudojant hibridinius neuroninius tinklus; lygaus objekto paviršiaus tinklelio atkūrimas; keleto objektų atkūrimas vienu metu; objektų atkūrimas nenaudojant kaukės; sudėtingų bei kintančių objektų taškinio debesies atkūrimas naudojant giliuosius neuroninius tinklus; galimybė aptikti bei iškirpti sudėtingus objektus taškiniame debesyje keletui taškinių objektų atkurti; realių struktūrizuotos šviesos bei lazerinių gylio jutiklių taškinių debesų išvalymas naudojant neprižiūrimus besivaržančius neuroninius tinklus objektams atkurti neturint tiesos reikšmių jų mokymui.

6.1.6. Ginamieji teiginiai

1. Pasiūlytas bei įgyvendintas sprendimas, gebantis iš vieno netobulo gylio perspektyvos kadro atkurti nematomas objekto zonas, pritaikant neprižiūrimų besivaržančių giliųjų išvalančiųjų neuroninių tinklų metodologiją.

2. Sukurta strategija giliesiems neuroniniams tinklams, objektams bei jų nematomoms zonoms atkurti iš triukšmingų struktūrizuotos šviesos bei lazerinių gylio jutiklių, mokymui neturint tiesos reikšmių.

3. Pasiūlyti sprendimai tiek vienam, tiek keletui paprastų bei sudėtingų ir laike kintančių objektų bei jų nematomoms zonoms atkurti iš vieno netobulo gylio jutiklio kadro, kurių atkūrimo kokybė, nepaisant jutiklių iškraipymų, prilygsta sintetiniams (tobuliems) duomenims.

6.1.7. Darbo rezultatų aprobavimas

Web of Science ir Scopus duomenų bazių leidiniuose su citavimo rodikliu:

1. **Kulikajevas, Audrius**; Maskeliūnas, Rytis; Damaševičius, Robertas. Detection of sitting posture using hierarchical image composition and deep learning // PeerJ computer science. London : PeerJ. ISSN 2376-5992. 2021, vol. 7, art. no. e442, p. 1-20. DOI: 10.7717/peerj-cs.442. Autoriaus indėlis: 0.334.
2. **Kulikajevas, Audrius**; Maskeliūnas, Rytis; Damaševičius, Robertas; Scherer, Rafal. HUMANNET—a two-tiered deep neural network architecture for self-occluding humanoid pose reconstruction // Sensors. Basel : MDPI. ISSN 1424-8220. 2021, vol. 21, iss. 12, art. no. 3945, p. 1-16. DOI: 10.3390/s21123945. Autoriaus indėlis: 0.250.
3. **Kulikajevas, Audrius**; Maskeliūnas, Rytis; Damaševičius, Robertas; Wlodarczyk-Sielicka, Marta. Auto-refining reconstruction algorithm for recreation of limited angle humanoid depth data // Sensors. Basel : MDPI. ISSN 1424-8220. 2021, vol. 21, iss. 11, art. no. 3702, p. 1-17. DOI: 10.3390/s21113702. Autoriaus indėlis: 0.250.
4. **Kulikajevas, Audrius**; Maskeliūnas, Rytis; Damaševičius, Robertas; S.L. Ho, Edmond. 3D object reconstruction from imperfect depth data using extended YOLOv3 network // Sensors. Basel : MDPI. ISSN 1424-8220. 2020, vol. 20, iss. 7, art. no. 2025, p. 1-28. DOI: 10.3390/s20072025. Autoriaus indėlis: 0.500.
5. **Kulikajevas, Audrius**; Maskeliūnas, Rytis; Damaševičius, Robertas; Misra, Sanjay. Reconstruction of 3D object shape using hybrid modular neural network architecture trained on 3D models from ShapeNetCore dataset // Sensors. Basel : MDPI. ISSN 1424-8220. 2019, vol. 19, iss. 7, art. no. 1553, p. 1-21. DOI: 10.3390/s19071553. Autoriaus indėlis: 0.400.

Tarptautinėse ir nacionalinėse konferencijose:

1. **Kulikajevas, Audrius**; Maskeliūnas, Rytis; Damaševičius, Robertas; Griškevičius, Julius; Daunoravičienė, Kristina; Žižienė, Jurgita; Lukšys, Donatas; Adomavičienė, Aušra; Exercise Abnormality Detection Using BlazePose Skeleton Reconstruction // ICCSA 2021: Computational Science and Its Applications, Cagliari, Italy, 13-16 September 2021. DOI: 10.1007/978-3-030-86976-2_7. p. 90-104. Autoriaus indėlis: 0.125.

2. **Kulikajevas, Audrius**; Maskeliūnas, Rytis; Damaševičius, Robertas; Woźniak, Marcin. Reconstruction algorithm of invisible sides of a 3D object for depth scanning systems of a 3D object for cost effective truncation of point cloud data // ECOS 2019: Proceedings of the 32nd international conference on efficiency, cost, optimization, simulation and environmental impact of energy systems, Wrocław, Poland, 23-28 June 2019 / edited by: Wojciech Stanek, Paweł Gładysz, Sebastian Werle, Wojciech Adamczyk. Gliwice : Institute of Thermal Technology Silesian University of Technology, 2019. ISBN 9788361506515. p. 4505-4507. Autoriaus indėlis: 0.250.
3. Bhandari, Sandeepak; **Kulikajevas, Audrius**. Ontology based image recognition: a review // CEUR workshop proceedings : IVUS 2018: proceedings of the international conference on information technologies, Kaunas, Lithuania, April 27, 2018 / edited by G. Capizzi, R. Damaševičius, A. Lopata, T. Krilavičius, Ch. Napoli, M. Woźniak. Aachen : CEUR-WS. eISSN 1613-0073. 2018, vol. 2145, p. 13-18. Autoriaus indėlis: 0.500.

2018 m. birželio 28 – liepos 6 dienomis dalyvauta tarptautinėje doktorantų vasaros mokykloje Dortmundo universitete, išklausytas kursas, vertinamas 2 ECTS kreditais.

6.2. Straipsnių apžvalga

Disertacijos metu tiriama mokslo sritis — dėl didelio pastarųjų metų susidomėjimo giliaisiais neuroniniais tinklais atgijusi mokslo kryptis. Klasikiniai objektų atkūrimo algoritmai objekto paviršiui atkurti pasitelkdavo tokias esmines savybes, kaip išgaubtasis korpusas [47] arba objekto viršūnių normalės vektorius [48, 49, 50]. O objekto nematomoms zonoms atkurti vyrauja keletas spendimų, tai objekto nufilmavimas iš visų reikiamų perspektyvų bei sujungiant gautus taškinius debesis, taip atkuriant bendrą objekto formą, arba panaudojus objekto simetrijos ašis [51, 52, 53, 54]. Tačiau tokių sprendimų panaudojimas yra ribotas bei sunkiai pritaikomas sudėtingesniems ar laike kintantiems objektams, toliems kaip, pavyzdžiui, žmogus. Siekiant sukurti atsparų objektų pokyčiams bei pritaikomą tikroje aplinkoje objekto atkūrimo modelį iš vienos perspektyvos, buvo atliekami tyrimai, susiję su mašininiu mokymu. Egzistuoja dvi pagrindinės duomenų struktūros trimačiams objektams atvaizduoti erdvėje — tai tūrinio taškų tinklelio bei taškinio debesies. Vienas pirmųjų tyrimų, susijusių su objekto bei jo nematomų zonų atkūrimu naudojant mašininį mokymą *3D-R2N2* [10], naudoja *Sanford Online Products* [58] bei *ShapeNet* [57] duomenų rinkinius kaip *a priori* žinias rekurentiniam neuroniniam tinklui mokyti, kur rekurentiniai ilgos ir trumpos atminties [59, 60] sluoksniai mokomi atkurti objektą parodant tinklui tą patį objektą iš kelių perspektyvų, šiame straipsnyje minimas sprendimas gebėjo atkurti objektą tiek iš vieno, tiek iš keleto kadrų, kur didesnis įvesties perspektyvų kiekis pagerino galutinio atkurto objekto kokybę. Vėlesni sprendimai patobulino gautuosius rezultatus pritaikius *Chamfer* atstumą kaip paklaidos metriką, lyginančią spėjimą bei tiesą [12]. Kiti sprendimai bandė pritaikyti generatyvinius besivaržančius neuroninius tinklus objektui atkurti naudojant tiek keletą skirtingų perspektyvų [61], tiek vieną perspektyvos paveikslą [62]. Kiti autoriai pritaikė hierarchinius paviršiaus atkūrimo metodus, taip užtikrindami aukštesnės raiškos tūrinių taškų tinklelio objektų atkūrimo rezultatus [63]. Vėlesni metodai vietoj tūrinio taškų tinklelio pradėjo naudoti taškinių debesų atkūrimo metodus, pavyzdžiui, *PointOutNet* [64] naudoja tikimybinį objekto paviršiaus taškų pasiskirstymą jo taškiniam debesiui atkurti iš dvimačio paveikslo bei jo kaukės. Vėliau rašiusių autorių pasiūlyti sprendimai bandė pagerinti šiuos metodus, taip pat naudodami nerikiuotiems taškiniams debesims atpažinti bei atkurti iš dvimačių paveikslų [65]. Pagrindinė priežastis, dėl kurios autoriai naudojo dvimačius paveikslus kaip mašininio mokymo įvestį, — jau žinomų konvoliucinių branduolių, kurie yra naudojami panašioms su objektų atpažinimu susijusioms užduotims spręsti, pritaikymas [66, 67], dėl to nukenčia pastarųjų metodų rezultatų kokybė, kadangi monokuliariuose kadruose prarandama svarbi objekto gylio informacija. Kiti autoriai pasiūlė sprendimą, kaip panaudoti nesurikiuotus taškinius debesis kaip įvestį, kadangi standartiniai konvoliuciniai metodai nėra tinkami nerikiuotiems duomenims [68]. Dėl to vėlesni autoriai pasiūlė naudoti tankaus-detalaus tinklelio atkūrimo metodus objekto taškiniam debesiui atkurti [69], o *AtlasNet* autoriai pasiūlė sprendimą pagerinti objektų atkūrimą pritaikius sulopymo metodą, kai naudojama keletas atskirų tinklelių objekto formai atkurti [70]. Kaip pagrindinę metriką tiesai bei spėjimui palyginti pastarieji autoriai naudojo *Chamfer* atstumą, o *Earth Mover's* atstumą tik kaip kiekybinio įvertinimo metriką vertinant gautuosius rezultatus, nors pastarosios metrikos rezultatai yra labiau jautrūs nuokrypiams, tačiau dėl jo sudėtingumo jis yra sunkiai pritaikomas tankesniems taškiniams debesims, kitų autorių pasiūlyta tiesinė aproksimacija išsprendė šią problemą [71]. Nepaisant šios pažangos atkuriant nematomas objektų zonas, tiek Lietuvos, tiek pasaulio mokslinėje literatūroje vis tiek išlieka keletas problemų, susijusių su objektų atkūrimu, tai kelių objektų atkūrimas iš vieno kadro, sudėtingesnių bei laike kintančių objektų nematomų zonų atkūrimas ir objektų atkūrimas ne iš sintetinių duomenų bei nesant papildomos informacijos apie objektą, pavyzdžiui, objekto kaukė kadre.

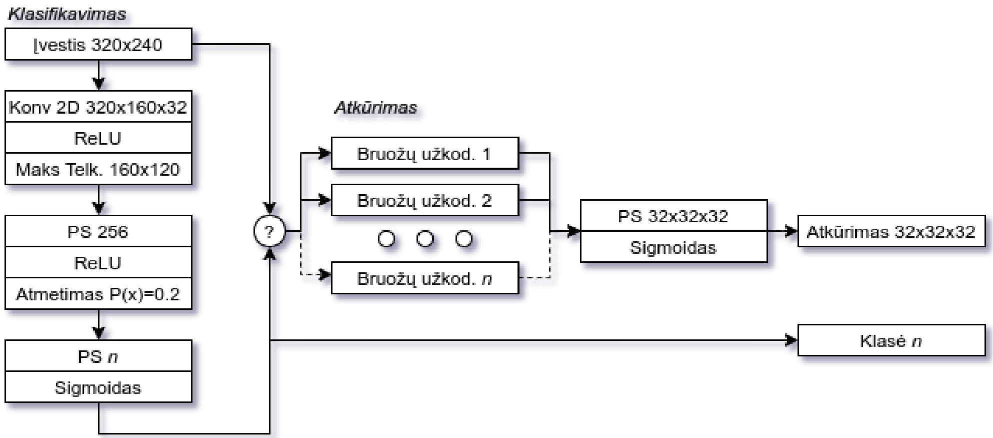
Šioms problemoms spręsti taip pat buvo naudojamas modelis, pagrįstas mašininio mokymo principu. Šis sprendimas buvo pasirinktas remiantis teiginiu, kad kiekvienas žmogus per savo gyvenimą sukuria mintyse esantį modelį, kurį naudodamas gali iš dalies nuspręsti, kaip atrodys nematoma objekto zona, kai šias užstoja tiek pats objektas, esant savęs uždengimo situacijai, tiek jas uždengiant kitiems objektams. Iš to keliame hipotezę, jog mašininio mokymo sprendimai, gebantys aptikti šablonus, yra tinkami šiai problemai spręsti.

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

Šios hipotezės pagrindu buvo išspausdintas pirmasis nematomų objekto zonų atkūrimo straipsnis *Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from ShapeNetCore Dataset*. Pagrindinis šio metodo tikslas — naudojant vieną gylio kameros perspektyvos kadrą atkurti objektą bei jo nematomas zonas. Straipsnyje buvo pasiūlyti du pagrindiniai naujumai, pirmasis tai hibridinė giliųjų neuroninių tinklų architektūra. Ši architektūra leido paspartinti giliųjų neuroninių modelių konvergavimo laiką bei sumažinti kaštus, kai atkūrimui pridedamas naujas objekto tipas, kitaip nei kiti panašūs metodai, atkuriantys trimačius objektus, naudojantys mašininį mokymą, pavyzdžiui *3D-R2N2* [10], kuris naudojo gylio jutiklio duomenis, o ne plokščius paveikslus. Tai leido tiksliau išlaikyti objekto formą bei atkurti objektus nenaudojant papildomų kaukių, o dėl hibridinių savybių suteikiamos galimybės naudoti lengvesnės architektūros tinklus šis modelis galėjo veikti realiuoju laiku. Taip pat kiekybinių tyrimų metu buvo nustatyta, jog sukurto modelio tikslumas atkuriant trimačius objektus buvo 89,5%. Giliojo hibridinio neuroninio tinklo, apibūdinamo straipsnyje, architektūra pateikiama 1 paveiksle. Modelis susideda iš dviejų pakopų, pirmojoje atliekamas objekto tipo atpažinimas. Pastarasis yra reikalingas siekiant atpažinti, kuriai neuroninio tinklo šakai priklauso matomas objektas. Tinklo įvestis yra gylio laukas, perduodamas iš naudojamo jutiklio. Pastarajam yra pritaikomas konvoliucinis 3×3 branduolys, kurio rezultatas yra 32 bruožų žemėlapiai. Kadangi atliekamos užduoties rezultatas nėra tiesinis, reikia įvesti netiesiškumą, šiuo atveju naudojama *ReLU* funkcija [77, 78]. Ši pasirinkta dėl jos matematinės išraiškos paprastumo, tai supaprastina aritmetiką ir sumažina reikalingų kompiuterio resursų kiekį, taip pat *ReLU* beveik visais atvejais pagerina konvoliucinių tinklų veikimą [79, 80], funkcijos išraiška pateikta formulėje (1) [78].

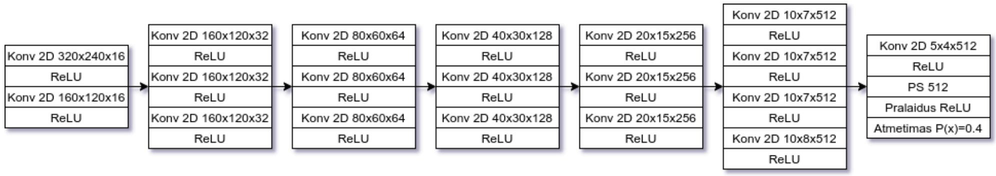
$$f(x) = max(0, x) \tag{1}$$

Kitu žingsniu atliekama telkimo operacija atrenkant tik maksimalias reikšmes. Kaip ir konvoliucinis branduolys, telkimo operacijai naudojamas 3×3 formos branduolys su 2 žingsnių intervalu. Tai sumažina įvesties dydį perpus. Gautas bruožų rinkinys paverčiamas eilute ir sujungiamas su 256-ių pilnai sujungtų neuronų sluoksniu, rezultatui pritaikius netiesiškumo funkciją, papildomai pridedamas atmetimas su 20% tikimybe atmesti neurono reikšmę. Atmetimo strategija buvo pasirinkta siekiant pagerinti modelio generalizavimą bei sumažinti šališkumą, kadangi įrodyta, jog atmetimas gali pagerinti neuroninių tinklų, naudojančių *ReLU* netiesiškumo funkciją, įsiminimą [81, 82, 83]. Išvestis jungiama su iki galo sujungtu sluoksniu, turinčiu *n* neuronų, kur *n* nurodo, kiek yra išmokyta antrojo tipo pakopų, čia kiekviena pakopa yra atskira hibridinio tinklo rekonstravimo šaka.



1 pav. Bendroji giliojo neuroninio tinklo struktūra

Nustačius tinkamą šaką, gylio žemėlapis perduodamas į užkodavimo šaką. Užkodavimo šakos pavyzdys matomas 2 paveiksle, hibridinės savybės leidžia kiekvienos užkodavimo šakos architektūrai skirtis. Hibridinė architektūra leidžia keisti parametrų kiekį pagal atkuriamo objekto sudėtingumą. Tai sutrumpina mokymo laiką bei pagreitina naujų objektų pridėjimo procesą, kadangi nereikia iš naujo mokyti viso tinklo, užtenka išmokyti tik klasifikatorių atpažinti naująjį objektą bei šaką, kuriai priskiriamas objektas.



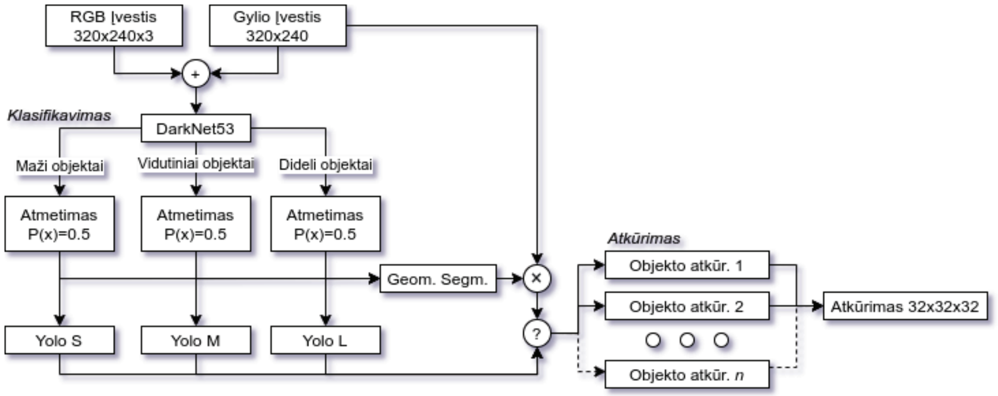
2 pav. Giliojo neuroninio tinklo objekto bruožų vektoriaus šaka, skirta vieno tipo objekto esybėms atkurti

Neuroniniam tinklui mokyti naudojama dviejų palyginimo funkcijų suma (žiūrėti formulę (2)), kur α ir $\hat{\alpha}$ yra objekto rekonstruoto tūrinio taško tikroji bei spėjama vertės, β ir $\hat{\beta}$ objekto klasės tikroji bei spėjama reikšmė [87], N yra partijos dydis, o Q yra tinklelio tankio ašių sandauga.

$$f_r = \sum_{i=1}^{N} \frac{\sum_{j=1}^{Q} (\alpha_{ij} - \hat{\alpha}_{ij})^2}{N} - \beta_i \log \frac{e^{\hat{\beta}_i}}{\sum_{j=1}^{N} e^{\hat{\beta}_j}} \tag{2}$$

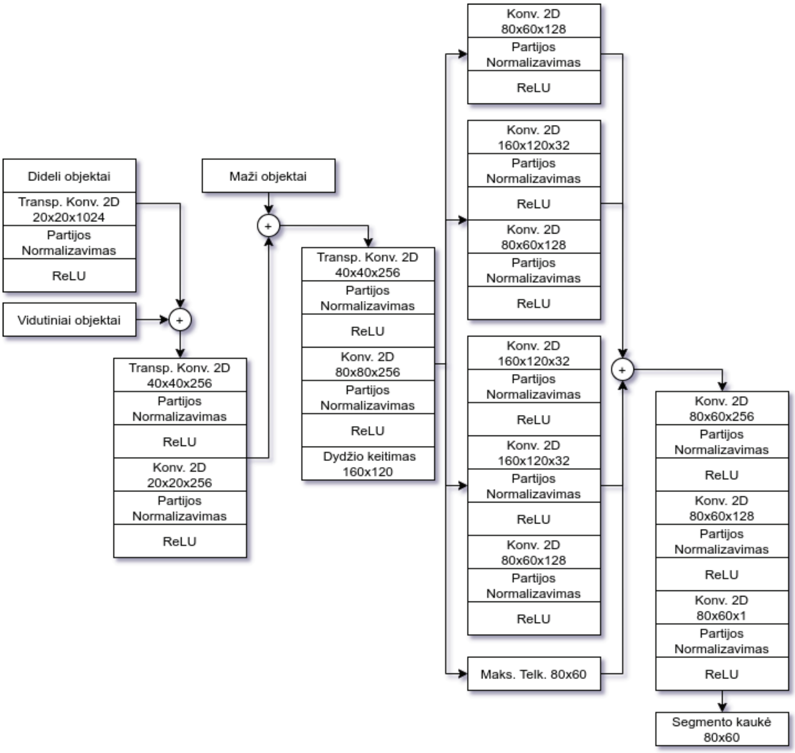
6.2.2. *3D Object Reconstruction from Imperfect Depth Data Using Extended YOLOv3 Network*

Tolimesniu tyrimų etapu buvo sprendžiama kelių objektų atkūrimo problema, kurios pagrindu buvo išleistas straipsnis *3D Object Reconstruction from Imperfect Depth Data Using Extended YOLOv3 Network*, aprašantis rastus sprendimus. Pastarojo tyrimo metu buvo naudojamas modifikuotas *YOLOv3* [92] stuburinis modelis, kuris leido efektyviai rasti bei identifikuoti individualius objektus scenoje. Vienas iš *YOLOv3* privalumų yra tas, jog, be to, kad klasifikuojamas objektas, naudojama objektus ribojanti dėžutė. Panaudojus dėžutę buvo randama geometrinė kaukė objektų kontūrams aptikti. Naudojant geometrinę kaukę galima rasti bei išfiltruoti potencialias objektų instancijas, tai leido atlikti individualių rekonstrukcijas, perduodant tik reikiamo objekto gylio žemėlapius į tam išmokytas hibridinio neuroninio tinklo dalis. Taip pat buvo pasiūlytas periodinis hiperparametras neuroninio tinklo mokymo žingsniui, šie priimti sprendimai ne tik pagerino ankstesnius tyrimus atsiradusia galimybe atkurti keletą objektų iš vieno kadro, bet taip pat ir pagerino objektų atkūrimo kokybę 8,53%. Bendrą neuroninio tinklo vaizdą galima matyti 3 paveiksle, kaip ir *YOLOv3* straipsnyje, objektui atpažinti naudojamas modifikuotas *DarkNet53* [92] neuroninio tinklo stuburas, kuriam perduodamas RGB-D (matomo šviesos spektro bei gylio) kardas. Pastarojo išvestis yra trys tinklo atšakos, kurios yra skirstomos į mažų (S), vidutinių (M) bei didelių objektų (L) grupes pagal tai, kokio dydžio objektus šaka yra specializuota atpažinti bei atkurti. Siekiant pagerinti neuroninio tinklo abstrahavimo lygį, prie kiekvienos šakos yra prijungiamas atmetimo sluoksnis su tikimybe atmesti 50% įvesties neuronų, kaip ir originaliajame *YOLOv3* modelyje, toliau seka objekto dėžučių atkūrimas kiekvienam iš objektų tipų, kur randama objekto klasė bei jį ribojanti dėžutė.



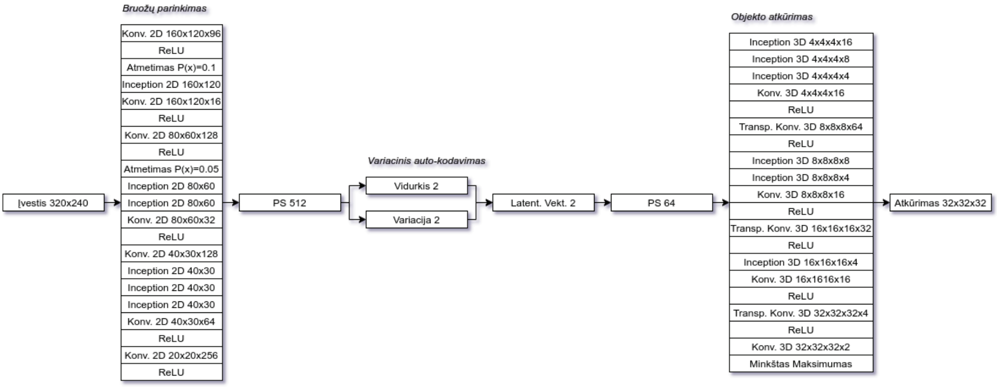
3 pav. Giliojo neuroninio tinklo architektūra daugeliui objektų atkurti. Architektūra naudoja *DarkNet53* kaip stuburinį tinklą bruožams ištraukti, atlieka objektų atpažinimą, objektų regiono aptikimą, automatinį segmentavimą bei daugelio objektų atkūrimą

Tačiau *DarkNet53* atrinkti bruožai yra taip pat perduodami į objekto geometrinio segmentavimo giliojo neuroninio tinklo šaką, kurioje yra atkuriamos objektų kaukės, reikalingos keletui objektų atrinkti jų atkūrimo stadijoje. Geometrinio segmentavimo šaka matoma 4 paveiksle. Ši kaukė atkuriama naudojant *DarkNet53* tinklo šakas, pirmuoju žingsniu didžiųjų objektų šakai yra pritaikomas transponuotas konvoliucinis branduolys, kitaip nei interpoliaciniai bruožų žemėlapių bei paveikslėlių padidinimo metodai transponuoti neuroniniai konvoliuciniai branduoliai, dar žinomi kaip dekonvoliuciniai branduoliai, geba išmokti, ši mašininio mokymosi savybė dažnai pagerina rezultato kontūrų ryškumą, palyginti su interpoliaciniais metodais [96, 97]. Transponuotam rezultatui pritaikomas partijos normalizavimas, kuris standartizuoja partijos rezultatus. Įrodyta, jog šis sprendimas daugeliu atvejų pagreitina neuroninio tinklo mokymosi procesą bei pagerina jo tikslumą su validavimo duomenų rinkiniais [98, 99]. Pritaikius netiesiškumo funkciją, rezultatas sujungiamas su vidutinio dydžio objektų bruožų rinkiniu. Šis procesas yra kartojamas su gautu rezultatu, kuris yra sujungiamas su mažaisiais objektais. Tolesniu etapu atliekamas objekto bruožų apdorojimas, kurio paskutiniu žingsniu naudojama dvinarių interpoliacija tam, kad 80×80 dydžio bruožų žemėlapis būtų padidintas į tinkamo kraštinių santykio paveikslą. Toliau naudojant *Inception* paralelinio objekto bruožų ištraukimo strategiją, yra atrenkami keturių lygių bruožai, kurie leidžia kiekvienai neuroninio tinklo šakai išsirinkti skirtingo tipo bei dydžio bruožus, šis paralelizmas pagerina objektų aptikimą [100], paraleliniai bruožai sujungiami liekamosiomis jungtimis, kurios pagerina gradiento perdavimą ankstesniems sluoksniams, praleidžiant nereikalingas neuronų jungtis [101].



4 pav. Objektų geometrinio segmentavimo modulis

Naudojant hibridinių neuroninių tinklų metodologiją, yra mokomos skirtingos objektų atkūrimo šakos, kiekviena specializuojasi atkurti tam tikro tipo objektus. Objektui atkurti naudojami variaciniai autokoduotuojai, kurie pasirinkti dėl gebėjimo pasiūlyti kintančius, nedeterministinius įvesties spėjimus, tai leidžia pagerinti atkūrimo rezultatus esant triukšmingiems ar kitokių trūkumų turintiems duomenims [102, 103], neuroninio tinklo šakos architektūra matoma 5 paveiksle. Objekto bruožams atrinkti naudojami *Inception* architektūros moduliai, naudojantys dvimatės konvoliucijos branduolius, o objektui atkurti naudojami trimatės konvoliucijos branduoliai. Tai leidžia iš dalies išlaikyti tinklo architektūros simetriškumą, šiuo atveju kuo didesnė asimetrija tarp užkodavimo bei atkodavimo sluoksnių, tuo tinklui buvo sudėtingiau išmokti tinkamus bruožus. Variacinio autokodavimo sluoksnyje pasirinkti du latentiniai vektoriai.



5 pav. Viena objektų rekonstravimo šaka, naudojamas variacinis autokodavimo modelis latentiniams vektoriams atrinkti

Neuroniniam tinklui mokyti naudota sudėtinė funkcija, siekiant palyginti klasifikavimo tikslumą, ribojančios dėžutės tikslumą, geometrijos segmentacijos nuokrypį nuo tiesos bei atkūrimo tikslumą. Galutinė palyginimo funkcija matoma formulėje (3), kuri susideda iš šių termų: f_d — objektą gaubiančios dėžutės paklaida; f_p — objekto aptikimo pasitikėjimo paklaida; $f_{\neg p}$ — pasitikėjimo, kad regione nėra objekto, paklaida; f_k — objekto klasės paklaida; f_s — objekto segmentacijos paklaida; f_r — nurodo objekto rekonstravimo paklaidą, $\lambda_a = 5$, $\lambda_b = 0,5$, šios konstantų vertės buvo parinktos pagal literatūroje siūlomas reikšmes [92].

$$f = \lambda_a f_d + f_p + \lambda_b f_{\neg p} + f_k + f_s + f_r \tag{3}$$

Funkcijos f_d išraiška pateikiama (4) [92] lygtyse, čia (x_i, y_i, w_i, h_i) yra objekto dėžutės centro koordinatės bei objektą gaubiančios dėžutės ilgis ir plotis, S yra regionų, dalijančių kiekvieną iš objekto ašių, skaičius, o B — gaubiamųjų dėžučių skaičius.

$$\begin{gathered} f_{dxy} = (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \\ f_{dwh} = (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \\ f_d = \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbb{1}_{ij}^{obj} [f_{dxy} + f_{dwh}] \end{gathered} \tag{4}$$

Funkcijos f_p (žiūrėti (5) lygtį [92]) bei $f_{\neg p}$ ((6) lygtis [92]) išraiškos labai panašios. Pagrindinis skirtumas tai, kad f_p yra pasitikėjimo paklaida, jog nurodytame regione yra objektas, o $f_{\neg p}$ yra atvirkštinė jai — tikimybės paklaida, jog objekto nėra.

$$f_p = \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 \tag{5}$$

$$f_{\neg p} = \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbb{1}_{ij}^{\neg obj} (C_i - \hat{C}_i)^2 \tag{6}$$

Funkcijos f_k nurodo objekto klasės paklaidą, išraiška matoma (7) [92] lygtyje, čia n yra klasių, kuriomis mokomas objektas, skaičius, o $p_i(j)$ bei $\hat{p}_i(j)$ yra regionų tikroji bei spėjamoji klasės tikimybės reikšmė.

$$f_k = \sum_{i=1}^{S^2} \mathbb{1}_{ij}^{obj} \sum_{j \in n} (p_i(j) - \hat{p}_i(j))^2 \tag{7}$$

Tolimesnis termas f_s — tai segmentavimo paklaidos termas, matomas (8) lygtyje, jis nurodo paklaidą tarp tikrosios bei spėjamosios geometrinės segmentacijos reikšmės. Čia W ir H yra segmentacijos kaukės ilgio ir pločio dimensijos, o s ir $\hat{s}$ yra segmentacijos tikroji bei spėjama reikšmės.

$$f_s = \sum_{i=1}^{W} \sum_{j=1}^{H} (s_{ij} - \hat{s}_{ij})^2 \tag{8}$$

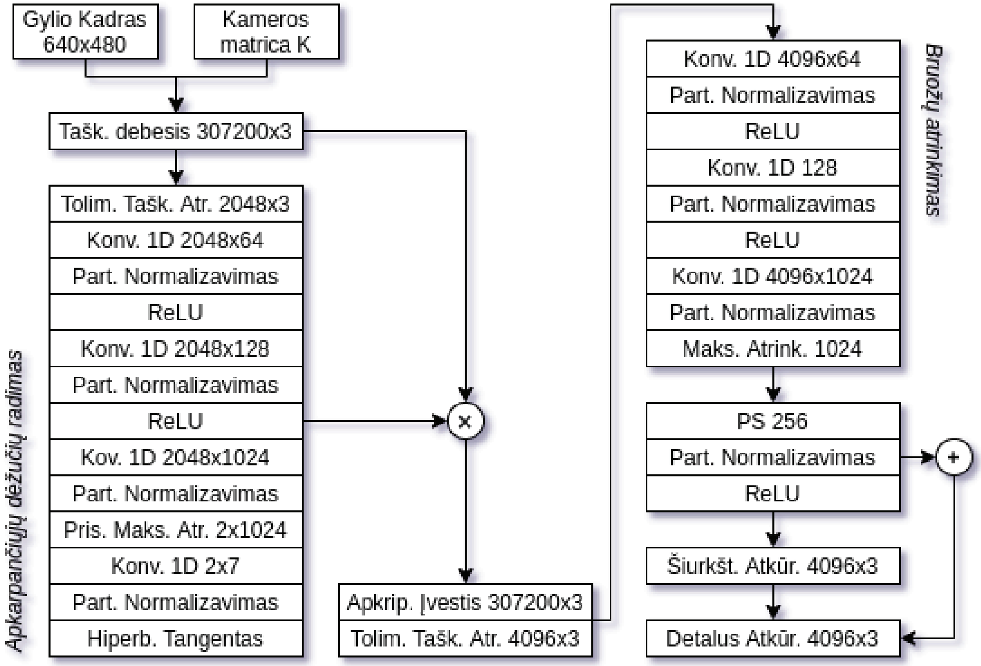
Paskutinysis termas f_r yra tūrinių taškų rekonstravimo paklaida, jos išraiška matoma (9) lygtyje, ši susideda iš KL divergacijos [104] bei rekonstrukcijos paklaidos sumos [87]. Čia l yra latentinių variacinio autokoduotojo neuronų kiekis, σ ir μ tikimybių pasiskirstymas normaliuoju skirstiniu.

$$f_r = \frac{\sum_{i=1}^{l} \sigma_i^2 - 1 + \mu_i + e^{\mu_i}}{2l} - \frac{1}{Q} \sum_{i=1}^{Q} \sum_{j=1}^{2} \alpha \log \hat{\alpha} \tag{9}$$

6.2.3. *HUMANNET—A Two-Tiered Deep Neural Network Architecture for Self-Occluding Humanoid Pose Reconstruction*

Ankstesni du metodai objekto rekonstravimo tematika naudojo objektų atkūrimą tūrinių taškų principu. Nors ši duomenų struktūra supaprastina tikrojo objekto Y bei tinklo spėjimo $\hat{Y}$ matematinę išraišką, dėl ko yra mažesni tinklo mokymo kaštai tiek laiko, tiek reikiamų resursų atžvilgiu, tačiau tūrinių taškų sprendimai turi kritinį trūkumą: siekiant atkurti nematomas zonas sudėtingesniems objektams, tūrinių taškų tinklelis turi būti ganėtinai tankus, kitaip prarandamos svarbios detalės. Tačiau tūrinis taškų tinklelis yra neefektyvi duomenų struktūra reikalingos operatyviosios atminties atžvilgiu, kadangi reikalingi kaštai jiems atvaizduoti seka geometrine priklausomybes kreive n^3. Dėl šios priežasties efektyviai išmokyti tokius tinklus tampa progresyviai sudėtinga, kadangi viena iš priežasčių, kodėl gilieji neuroniniai tinklai taip išpopuliarėjo, tai galimybė pritaikyti paralelinio skaičiavimo įrenginius — grafinius procesorius, turinčius ribotą operatyviosios atminties kiekį. Net ir iš dalies išsprendus šią problemą ir efektyviau atvaizduojant tūrinių taškų tinklelius pasinaudojus tokias struktūras, kaip oktaninis medis [72, 106], išlieka kitos problemos, susijusios su tūriniu taškų tinkleliu, pavyzdžiui, homografijos problema, kai, norint atkurti objekto poziciją erdvėje, kuri reikalinga efektyviam atkūrimui, taip pat reikia atkurti objekto transformacijas, t. y. poziciją, pasukimą bei dydį. Siekiant išvengti šių problemų, tolimesniems tyrimams buvo naudojami nesurikiuoti taškiniai debesys ir buvo išspausdintas straipsnis *HUMANNET—A Two-Tiered Deep Neural Network Architecture for Self-Occluding Humanoid Pose Reconstruction*, trimačiam objektui atkurti naudojant vienos perspektyvos vaizdą. Kitaip nei tūrinių taškų tinkleliai, taškiniai debesys neturi perteklinės informacijos, tai juos padaro viena efektyviausių trimačius objektus atvaizduojančių struktūrų. Tačiau, išsprendus duomenų dydžio problemą, kyla skaičiavimo sudėtingumo problema. Kadangi taškiniai debesys yra nesurikiuota duomenų struktūra, kiekvienas iš $\hat{Y}$ taškų gali būti priskiriamas bet kuriam kitam Y taškui, taip pat bet koks kiekis $\hat{Y}$ taškų gali būti priskiriamas vienam Y taškui, tai padaro tokio palyginimo sudėtingumą $O(n!)$, dėl šios priežasties įprastos giliųjų neuronininių tinklų mokymo nuostolio funkcijos yra netinkamos. Mokslinėje literatūroje egzistuoja dvi pagrindinės nuostolio funkcijos dviem nerikiuotiems taškiniams debesims palyginti — tai *Chamfer* [107] bei *Earth Mover's* [108] atstumai, šie leidžia supaprastinti skaičiavimo sudėtingumą. Abu sprendimai turi skirtingas taškų pasiskirstymo charakteristikas, tačiau *Earth Mover's* rezultatai yra tolygesni. Vis dėlto pastarojo skaičiavimo sudėtingumas yra $O(n^3 \log n)$, dėl šios priežasties tyrime buvo naudojama tiesinio laiko bei operatyviosios atminties aproksimacija [109]. Kadangi kiti pažangiausi modeliai objektų rekonstravimo srityje fokusuojami ties vieno statinio objekto atkūrimu, pagrindinis pasiūlytas naujumas — apkarpymo dėžutės. Šis naujumas iš dalies pasinaudoja *3D Object Reconstruction from Imperfect Depth Data Using Extended YOLOv3 Network* idėja, tačiau vietoj dvimačių ribojimo dėžučių bei objektų klasifikavimo buvo naudojamos trimatės apkarpomosios dėžutės objektui izoliuoti. Šis gilaus mokymo modelis — tai dviejų lygių gilusis neuroninis tinklas. Pirmasis neuroninis tinklas pasiūlo dvi apkarpomąsias dėžutes bei atrenka geriausias nepersidengiančias iš jų tolimesniam atkūrimui. Karpomoji dėžutė iš įvesties neuroninio tinklo išfiltruoja tik jos regionui priklausančius taškus, kitu etapu regiono taškų imtis yra sumažinama, kadangi įvesties taškinis debesis yra per tankus naudoti. Sumažintas taškinis debesis yra perduodamas į antrąją giliojo neuroninio tinklo pakopą, kur atliekamas nematomų objekto zonų atkūrimas. Atlikus kiekybinį įvertinimą, buvo nustatyta, jog apkarpomųjų dėžučių tikslumas pagal Džakardo panašumo indeksą yra 0,7907, o objekto atkūrimo panašumo indeksas pagal *Earth Mover's* bei *Chamfer* atstumą atitinkamai yra 0,0256 ir 0,0276. Dėl šios priežasties pasiūlyto modelio tikslumas yra konkurencingas su kitais sprendimais, tačiau taip pat, kitaip nei kiti pažangiausi sprendimai, sugeba atkurti keletą objektų iš vieno taškinio debesies įvesties. Be to, kitaip nei egzistuojantys atkūrimo sprendimai, pasiūlytas gilaus neuroninio tinklo modelis sugebėjo atkurti ne tik statinius objektus, bet ir sudėtingesnius, laike deformuojamus objektus, šiuo atveju žmogaus modelis buvo atkuriamas iš apytiksliai 50% matomų taškų. Bendras tinklo vaizdas matomas 6 paveiksle, naudojamo gylio jutiklio kadras bei kameros vidinių parametrų matrica K sujungiama — taip atkuriamas taškinis debesis, tačiau pastarojo tankis yra per didelis dirbti su šiuolaikine technine įranga, dėl šios priežasties yra reikalingas mastelio sumažinimas, todėl naudojamas tolimiausio taško atrinkimo algoritmas dėl jo savybės išlaikyti objekto paviršiaus formą netolygiai pasiskirsčiusiuose taškiniuose debesyse bei deterministinio atrenkamų taškų kiekio [112]. Pirmojo etapo metu atliekamas gylio kadro trimačių objektus apgaubiančių dėžučių atrinkimas, įvesties taškinį debesį sumažinus iki 2048 taškų bei naudojant vienmačius konvoliucinius tinklus reikšmingiausiems bruožams atrinkti. Bruožams atrinkti naudojama *PointNetFeat* struktūra, kai atrenkami 64, 128 bei 1024 svarbiausi bruožai [68]. Atrinktiesiems bruožams yra pritaikomas prisitaikančio maksimalaus atrinkimo branduolys, atrenkantis du rezultatus, darančius didžiausią įtaką bruožų žemėlapiams [113], dėl ko yra ištraukiami branduolio rezultatai — dvi objektus gaubiančios trimatės dėžutės.

Atrinkus dėžutę su didžiausiu spėjimo įsitikinimu, įvesties debesis yra apkarpomas pagal naudotos dėžutės matmenis, taip atrenkamos tik tam objektui priklausančios taškinio debesies viršūnės, kurių tankis sumažinamas iki 4096. Šis sprendimas buvo pasirinktas dėl to, kad jis leidžia atliekant debesies taškinio debesies sumažinimo operaciją turėti daug didesnės raiškos įvestį, kadangi priešingu atveju kiti objektai, pavyzdžiui, sienos, ne tik sumažintų įvesties tikslumą, pridėdami savo informaciją į ribotą viršūnių skaičių, bet ir pasunkintų tinklo mokymą. Be to, naudotas sprendimas suteikia galimybę daugelio objektų atkūrimui naudoti tą pačią įvestį. Tolesniame etape apkarpytas taškinis debesis yra apdorojamas bruožų atrinkimo šakoje. Bruožams apdoroti taip pat naudojama *PointNetFeat* struktūra dėl jos



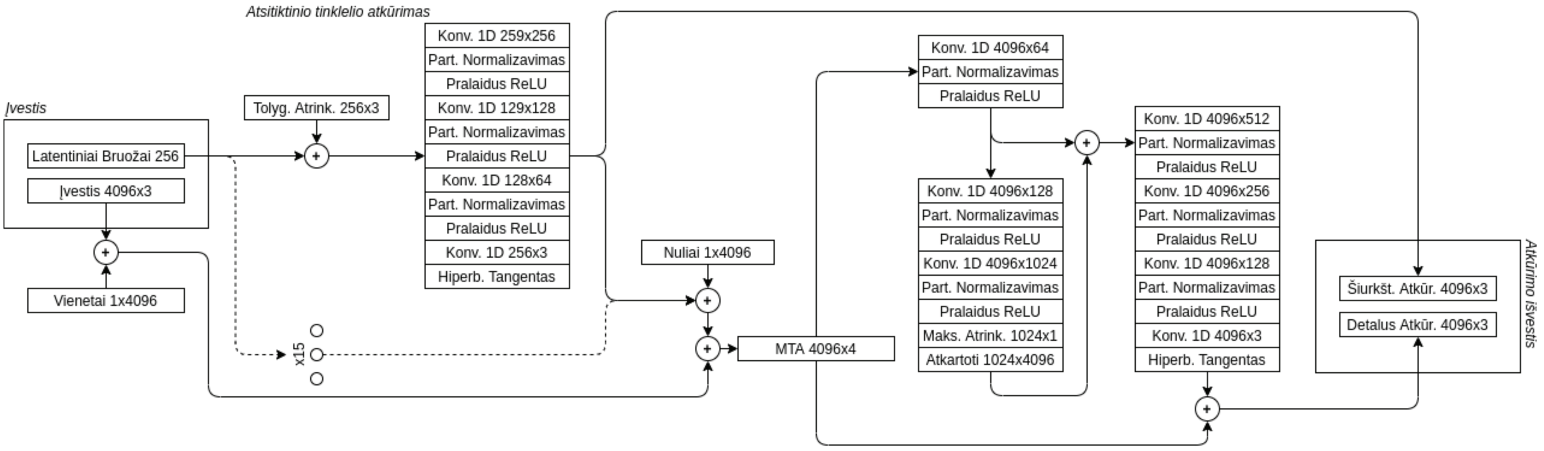
6 pav. Straipsnyje pasiūlyto tinklo apžvalga, pastarasis susideda iš apkirpimo, šiurkštaus atkūrimo bei detalaus atkūrimo stadijų

gebėjimo dirbti su netvarkingais taškiniais debesimis. Toliau naudojama dviejų žingsnių objektų taškinio debesies atkūrimo strategija, kai pirmojo etapo metu yra atkuriami šiurkštūs objekto bruožai, o kito žingsnio metu atkuriami detalesni objekto bruožai naudojant išliekamąsias jungtis bei jau atkurtus šiurkščiuosius objekto bruožus [69, 71], tinklo šakos struktūra matoma 7 paveiksle. Praėjusiame žingsnyje užkoduoti latentiniai bruožai yra sujungiami kartu su 16 atšakų, šios atšakos sudaromos naudojant atsitiktines tolyginio skirstinio reikšmes. Taip sukonstruojama 16 skirtingų tarpusavyje susijusių paviršiaus grupių atšakų, kuriomis padengiamas objekto paviršius, taip atkuriant šiurkščius objekto bruožus. Šiurkščių objektų bruožų bei įvesties liekamosios sąjungos bruožai, atrinkti minimalaus tankio atrinkimo būdu, yra atkuriami į detalų objekto taškinį debesį.

Straipsnyje minimam tinklui mokyti naudojama sudėtinė funkcija iš dviejų pagrindinių termų, matomų (10) lygtyje.

$$f = f_k + \mathbb{1}\{f_k \leq 0,3\} f_r \tag{10}$$

Čia f_k (žiūrėti (11) lygtis) — iškirpimo paklaida, kur o yra objektą gaubiančiosios dėžutės centras, s — jos dydis, y_c — tikimybė, jog egzistuoja



7 pav. Objekto atkūrimo tinklo šaka

objektas joje, o *n* — galimų dėžučių skaičius.

$$\begin{aligned} f_{dp} &= y_c \sum_{i=1}^{n} L1_s(o_i, \hat{o}_i) \\ f_{ds} &= y_c \sum_{i=1}^{n} L1_s(s_i, \hat{s}_i) \\ f_{bce} = \frac{1}{n} \sum_{i=1}^{n} y_c \cdot \log \hat{y}_c &+ (1 - y_c) \cdot \log (1 - \hat{y}_c) \\ f_k &= f_{dp} + f_{ds} + f_{bce} \end{aligned} \tag{11}$$

$L1_s$, matoma (12) [114] lygtyje, tai tolygios absoliutinės paklaidos funkcija, turinti mažesnį jautrumą ekstremumo reikšmėms bei padedanti išvengti diverguojančių gradientų [114].

$$\begin{aligned} z_i(y_i, \hat{y}_i) &= \begin{cases} \frac{(\hat{y}_i - y_i)^2}{2\beta}, & \text{jei } |\hat{y}_i - y_i| < \beta \\ |\hat{y}_i - y_i| - 0.5\beta, & \text{kitaip} \end{cases} \\ L1_s(y, \hat{y}) &= \frac{1}{n} \sum_{i=1}^{n} z_i \end{aligned} \tag{12}$$

Kitas paklaidos termas — tai objekto atkūrimo f_r (žiūrėti lygtį (13) [71]), čia atliekamas palyginimas naudojant *Earth Mover's* f_{emd} (matomas (14) [71] lygtyje) atstumą tiek šiurkštaus $\hat{S}_{\check{s}iurk\check{s}tus}$, tiek detalaus $\hat{S}_{detalus}$ atkūrimo rezultatas kartu su tikrąja taškinio debesies reikšme *S*. Taip pat pridedamas išsiplėtimo suvaržymas f_{exp} (žiūrėti (15) [71] lygtį, neleidžiantis tos pačios šakos taškams nutolti per toli viena kitos, kur $\alpha = 0,1$ [71].

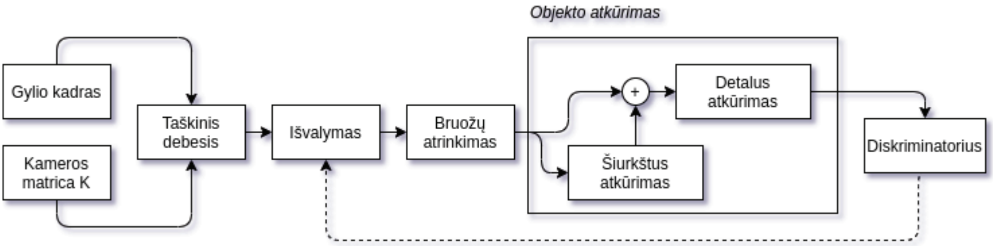
$$f_r = f_{emd}(S, \hat{S}_{detalus}) + f_{emd}(S, \hat{S}_{\check{s}iurk\check{s}tus}) + \alpha f_{exp} \tag{13}$$

$$f_{emd} = \underset{\phi: S \rightarrow \hat{S}}{min} \frac{1}{|S|} \sum_{x \in S} ||x - \phi(x)||_2 \tag{14}$$

$$f_{exp} = \frac{1}{KN} \sum_{1 \leq i \leq K} \sum_{(u, v \in \tau_i)} \mathbb{1}\{d(u, v) \geq \lambda l_i\} d(u, v) \tag{15}$$

6.2.4. *Auto-Refining Reconstruction Algorithm for Recreation of Limited Angle Humanoid Depth Data*

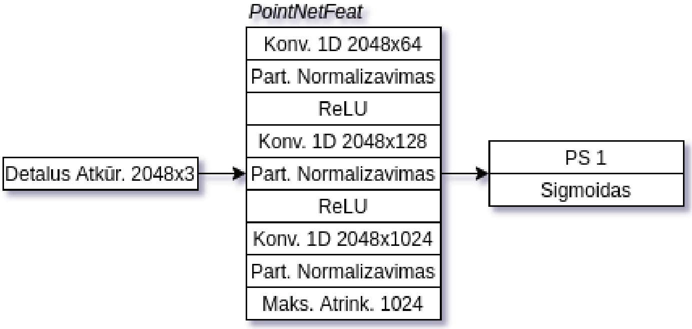
Viena iš pagrindinių ankstesnio metodo problemų — galimybė dirbti tik su sintetiniais duomenimis. Tai pagrindinė problema, su kuria susiduria ir kiti pažangiausi objektų atkūrimo sprendimai, naudojantys taškinius debesis kaip įvestį. Tačiau ketvirtame straipsnyje *Auto-Refining Reconstruction Algorithm for Recreation of Limited Angle Humanoid Depth Data* buvo pasiūlytas sprendimas naudoti neprižiūrimus besivaržančius išvalančiuosius giliuosius neuroninius tinklus, gebančius išvalyti tikro gylio jutiklio kadro įvestį, taip juos supanašinant su sintetiniais duomenimis, bet nesugadinant jų originalios formos dėl pritaikomų suvaržymo kriterijų. Pagrindinis šio modelio naujumas — atvirkštiniam procesui besivaržančių giliųjų neuroninių tinklų pritaikymas. Kitaip nei įprastai, kur atsitiktinio skirstinio duomenys yra panaudojami dar neegzituojančiam vaizdui [121] ar garsui [122] sukurti, straipsnyje buvo pasiūlytas jų pritaikymas kartu su neprižiūrimu mokymu, tokiu kaip realaus pasaulio duomenų išvalymas siekiant juos padaryti panašesnius į sintetinius, taip atsikratant priklausomybės nuo tiesos reikšmių. Šis sprendimas buvo reikalingas, kadangi tiek struktūrizuotos šviesos, tiek lazeriniai gylio jutikliai grąžina labai triukšmingus duomenis, dėl ko net ir pažangiausi atkūrimo sprendimai tampa neveiksmingi atkuriant iš realiųjų duomenų. Pasiūlytasis rafinavimo procesas išsprendžia šią problemą neturint tikrųjų tiesos Y reikšmių iki galo neprižiūrimo mokymo procesui. Išvalymo proceso rezultatas yra tikrosios įvesties latentiniai bruožai bei išvalytas taškinis debesis. Tolesniame etape atliekamas išvalyto taškinio debesies užkodavimas, kurio metu atrenkami aktualiausi objekto latentiniai bruožai. Pastarieji yra sujungiami su įvesties bruožais ir tolesniame žingsnyje atrinktieji bruožai yra naudojami objekto šiurkščiam atkūrimui, šio etapo metu grąžinamas šiurkštus objekto taškinis debesis kartu su atkurtomis nematomomis zonomis. Taškinis debesis gryninimo etape yra sujungiamas su išvalyto taškinio debesies liekamosiomis jungtimis, taip išgaunamas išgrynintas objekto ir jo nematomų zonų taškinis debesis. Šis rezultatas yra perduodamas diskriminatoriui, kuris klasifikuoja tikrus bei sintetinius duomenis, šis rezultatas bei paklaida yra naudojami išvalymo etapui atnaujinti. Atlikus kiekybinę rezultatų analizę nustatyta, kad pasiekta atkūrimo kokybė pagal *Earth Mover's* atstumo metriką 0,059, o pagal *Chamfer* atstumą — 0,079. Šie rezultatai leidžia daryti išvadą, jog modelio atkūrimo kokybė yra tarp pažangiausių, tačiau taip pat turi galimybę atkurti realaus pasaulio duomenis. 8 paveiksle pateiktas objektų nematomų zonų atkūrimo modelio bendras vaizdas. Objekto taškinis debesis yra apdorojamas išvalymo žingsnyje, todėl, kaip ir *HUMANNET—A Two-Tiered Deep Neural Network Architecture for Self-Occluding Humanoid Pose Reconstruction* modelyje, vyksta bruožų atrinkimas bei šiurkštus ir detalus atkūrimai. Detalaus atkūrimo rezultatas yra naudojamas diskriminatoriaus mokymui, kuriuo siekiama atskirti tikrus bei sintetinius duomenis, taip mokant išvalymo šaką.



8 pav. Bendras objekto nematomų zonų atkūrimo modelio vaizdas

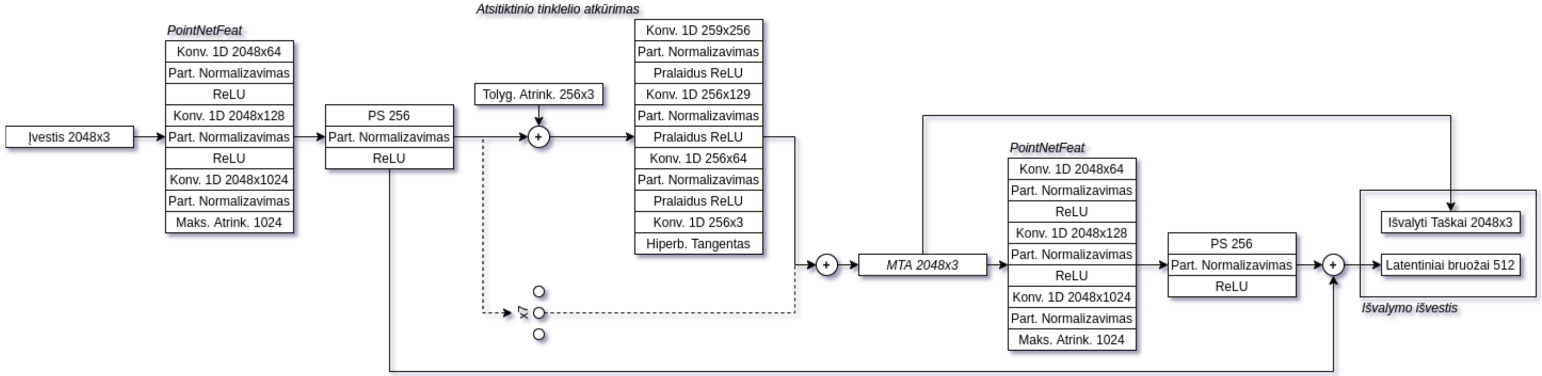
Objektams išvalyti naudojama panaši giliojo neuroninio tinklo struktūra, kaip ir šiurkštiems bruožams atkurti, ši matoma 9 paveiksle. Taškinio debesies įvesčiai yra randamas tinkamiausių latentinių bruožų vektorius ir atliekamas atsitiktinio tinklelio atkūrimas naudojant aštuonis primityvus. Primityvų atkūrimo rezultatas yra išvalytas nuo triukšmo neuroninio tinklo rezultatas. Tolesniame etape išrenkami atkurto neuroninio tinklo latentiniai bruožai, kurie yra sujungiami su prieš tai rastais įvesties latentiniais bruožais naudojant išliekamąsias aksonų jungtis. Galiausiai latentinių bruožų vektorius bei išvalytos taškinio debesies viršūnės yra perduodamos atkūrimo objekto nematomų zonų atkūrimo tinklui, matomam 10 paveiksle.

Atkurtasis detalusis rezultatas yra naudojamas kaip diskriminatoriaus įvestis, šis matomas 11 paveiksle. Diskriminatorius — tai atskirai išmokomas neuroninio tinklo modulis, kuris klasifikuoja tarp sintetinių (1) bei tikrų (0) duomenų, stengdamasis teisingai atspėti, iš kokio duomenų rinkinio yra atkuriamasis objektas. O išvalymo tinklas naudoja atvirkštinę diskriminatoriaus paklaidos funkcijos reikšmę, taip besivaržydamas su diskriminatorium, siekdamas jį apgauti.

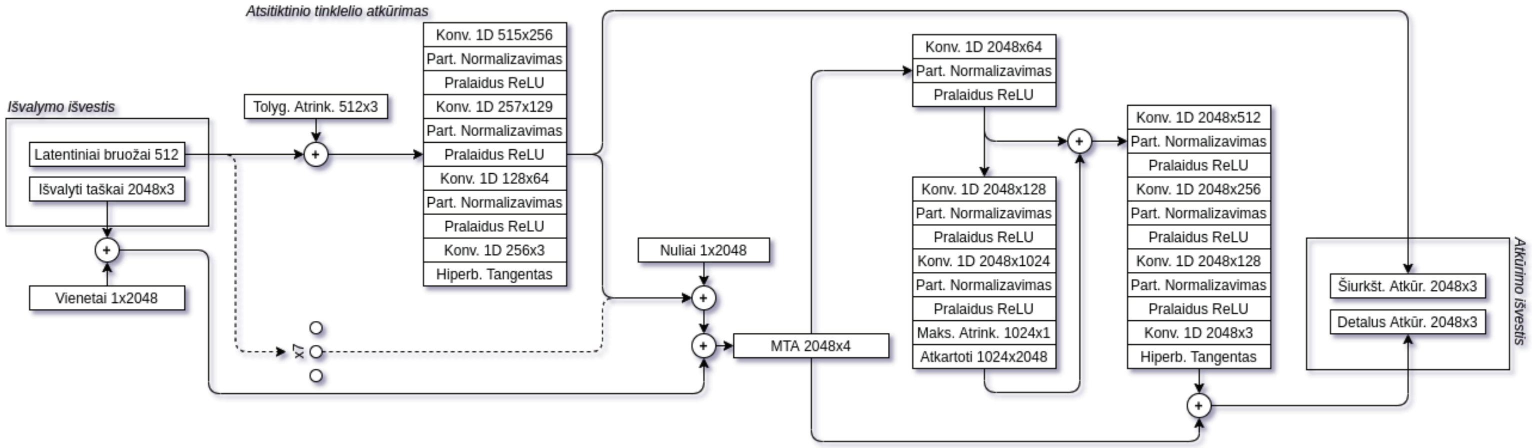


11 pav. Diskriminatorius, naudojamas atkurtam objektui klasifikuoti į tikrą bei sintetinį

Dėl neuroninio tinklo sudėtingumo jį išmokyti vienu žingsniu buvo per



9 pav. Objekto išvalymo stadija



10 pav. Išvalyto objekto nematomų zonų atkūrimo stadija daug sudėtinga užduotis, dėl šios priežasties buvo sukurta keturių fazių mokymo strategija.

I fazė. Autokodavimas Pirmos fazės metu yra mokoma tik neuroninio tinklo atkūrimo šaka. Jos metu visos išvalymo, šiurkštaus atkūrimo bei detalaus atkūrimo šakos yra mokomos tik atkurti matomą įvestį. Autokodavimo fazės metu naudojama paklaidos funkcija matoma (16) lygtyje, kuri išvedama iš (14) ir (15) lygčių kombinacijos. Šis sprendimas leidžia paspartinti išvalymo neuroninio tinklo mokymą, mat kitos šakos paspartina gradientų perdavimą į ją.

$$\begin{aligned} f_{\phi 1} = f_{emd}(S_{\check{s}varus}, \hat{S}_{\check{s}varus}) + f_{emd}(S_{\check{s}varus}, \hat{S}_{\check{s}iurk\check{s}tus}) + \\ f_{emd}(S_{\check{s}varus}, \hat{S}_{detalus}) + \\ \alpha(f_{exp}(S_{\check{s}varus}, \hat{S}_{\check{s}varus}) + f_{exp}(S_{\check{s}varus}, \hat{S}_{detalus})) \end{aligned} \tag{16}$$

II fazė. Atkūrimas Šios fazės metu mokomos tiek šiurkštaus atkūrimo, tiek detalaus atkūrimo tinklo šakos, taip pat sutvirtinamas išvalymo šakos veikimas. Atkūrimo stadijos metu abi nematomų zonų atkūrimo šakos mokomos su tikrosiomis objektų reikšmėmis, todėl, kitaip nei ankstesniosios fazės metu, šioje fazėje yra naudojami tik sintetiniai duomenys. Fazės paklaidos funkcija matoma (17) lygtyje.

$$\begin{aligned} f_{\phi 2} = f_{emd}(S_{\check{s}varus}, \hat{S}_{\check{s}varus}) + f_{emd}(S, \hat{S}_{detalus}) + \\ f_{emd}(S, \hat{S}_{\check{s}iurk\check{s}tus}) + \\ \alpha(f_{exp}(S_{\check{s}varus}, \hat{S}_{\check{s}varus}) + f_{exp}(S_{\check{s}varus}, \hat{S}_{detalus})) \end{aligned} \tag{17}$$

III fazė. Diskriminatorius Šios fazės metu yra mokomas tik diskriminatorius atpažinti, ar įvesties rezultatas yra sintetinis (1), ar tikras (0). Diskriminatorius mokomas atskiroje fazėje, kadangi ankstyvas jo mokymas gali potencialiai diverguoti, taip sugadindamas gradientus. Šios fazės mokymo paklaida matoma (18) lygtyje.

$$f_{\phi 3} = \sum_{i=1}^{N} \hat{y}_i \cdot log(y_i) + (1 - \hat{y}_i) \cdot log(1 - y_i) \tag{18}$$

IV fazė. Varžymasis Ketvirtosios ir paskutinės fazės metu yra vykdomas giliųjų besivaržančių tinklų mokymas. Šioje fazėje mokomas tiek diskriminatorius, tiek atkūrimo šakos. Tai atliekama trimis žingsniais, pirmojo žingsnio metu sustiprinami egzistuojantys svoriai naudojant paklaidos funkciją, matomą lygtyje (19). Kito žingsnio metu atnaujinami išvalymo tinklo svoriai naudojant atvirkštinę diskriminatoriaus funkciją. Mokymo paklaidos funkcija matoma (20) lygtyje, taip pat įvedamas $\gamma = 0,4$ suvaržymas, kuris naudojamas tam, kad realūs įvesties duomenys neprarastų jų bendros formos. Paskutinio žingsnio metu sustiprinami diskriminatoriaus svoriai naudojant funkciją, nurodomą (18) lygtyje.

$$f_{\phi 4a} = f_{\phi 2} + f_{\phi 3} \tag{19}$$

$$\begin{gathered} f_{\phi 4b} = \gamma f_{emd}(S_{\text{š}varus}, \hat{S}_{\text{š}varus}) + \\ \alpha f_{exp}(S_{\text{š}varus}, \hat{S}_{\text{š}varus})) + f_{\phi 3}(1 - y, \hat{y}) \end{gathered} \tag{20}$$

6.3. Rezultatai

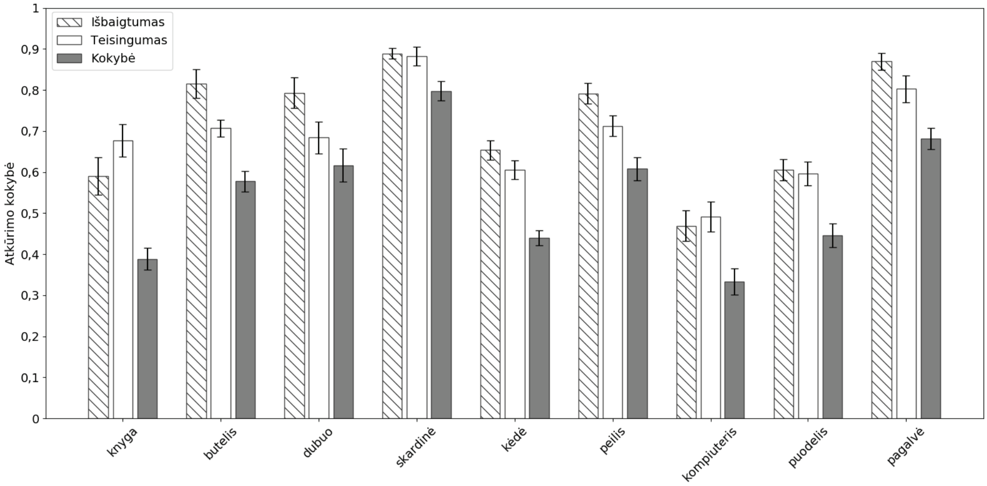
Tūrinių taškų tinklelio objekto nematomų zonų atkūrimo tikslumo kiekybiniam palyginimui atlikti buvo naudojamos trys pagrindinės metrikos — tai *išbaigtumas* ((21) lygtis [91]), *teisingumas* ((22) lygtis [91]) bei *kokybė* ((23) lygtis [91]). Čia A yra užpildytos tiesos tūrinio taškų tinklelio reikšmės, B yra užpildytos spėjimo reikšmės, o P — sąlyginė tikimybė. *Išbaigtumas* nurodo tiesos bei spėjimo santykį taškams, kurie turėjo būti įjungti. *Teisingumas* nurodo, kaip tiksliai atkuriamas objektas, o *kokybė* parodo *teisingų* bei *išbaigtų* metrikų balansą.

$$f_{\text{išb}} = \frac{P(B|A)}{P(B|A) + P(B|\neg A)} \tag{21}$$

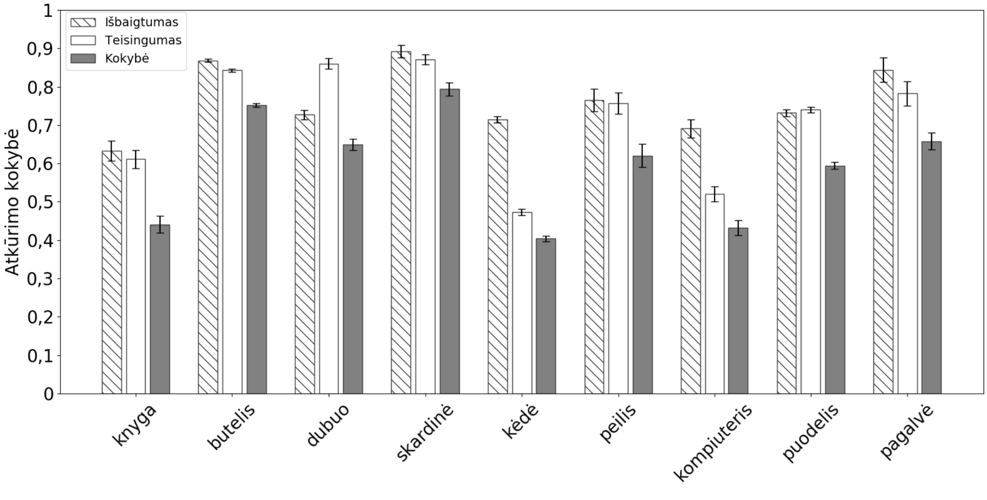
$$f_{teis} = \frac{P(B|A)}{P(B|A) + P(\neg B|A)} \tag{22}$$

$$f_{koky\text{bė}} = \frac{f_{\text{išb}} \cdot f_{teis}}{f_{\text{išb}} + f_{teis} - f_{\text{išb}} \cdot f_{teis}} \tag{23}$$

Naudojant šias metrikas įvertinta atkūrimo kokybė straipsniuose *Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from ShapeNetCore Dataset* (12 pav.) ir *3D Object Reconstruction from Imperfect Depth Data Using Extended YOLOv3 Network* (13 pav.), atkūrimai, kurių $0,25 \leq f_{koky\text{bė}} < 0,75$, buvo vertinami kaip patenkinami spėjimai, rėžis pasirinktas kokybinio vertinimo metu, gebant atskirti atkuriamą objektą. Atkūrimai, kurių $f_{koky\text{bė}} \geq 0,75$, buvo vertinami kaip puikiai atkurti. Kaip matoma 12 paveiksle, problemiškiausi objektai buvo kompiuteris bei knyga, tačiau tai galima paaiškinti tuo, jog šių objektų kiekis duomenų rinkinyje buvo neproporcingai mažas, palyginti su kitais.

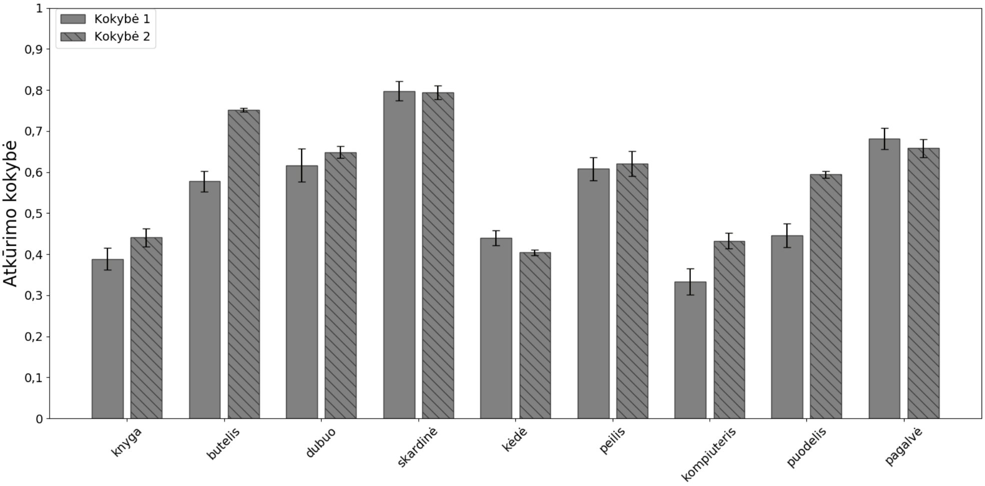


12 pav. Straipsnio *Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from ShapeNetCore Dataset* objekto nematomų zonų atkūrimo kokybės kiekybinio palyginimo metrikos



13 pav. Straipsnio *3D Object Reconstruction from Imperfect Depth Data Using Extended YOLOv3 Network* objekto nematomų zonų atkūrimo kokybės kiekybinio palyginimo metrikos

Taip pat, palyginus abiejų tyrimų metu atliktų eksperimentų rezultatus (14 pav.), matoma, kad, pritaikius naująją architektūrą, beveik visais atvejais buvo pasiekti daug geresni atkūrimo rezultatai. Be to, dėl sumažėjusių klaidos stulpelių galima teigti, jog atkūrimai kad tapo daug stabilesni. Galiausiai, net ir objektai, turintys mažesnį duomenų rinkinio pavyzdžių skaičių, buvo atkuriami kokybiškiau.



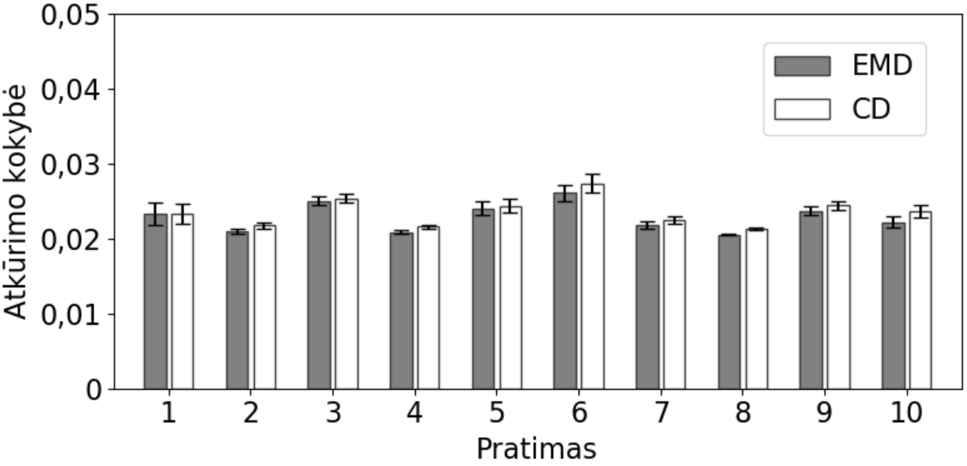
14 pav. Atkūrimo kokybės metrikos palyginimas tarp *Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from ShapeNetCore Dataset* (1) ir *3D Object Reconstruction from Imperfect Depth Data Using Extended YOLOv3 Network* (2) gautųjų rezultatų

Tolimesnių tyrimų metu buvo atliekami eksperimentai naudojant taškinius debesis, dėl šios priežasties negalima naudoti anksčiau naudotų kiekybinių metrikų, o taškinių debesų atkūrimo kokybei palyginti bus naudojamas *Earth Mover's* atstumas (EMD), kurio lygtis matoma (14), bei *Chamfer* atstumas (CD), šio lygtis matoma (24) [118].

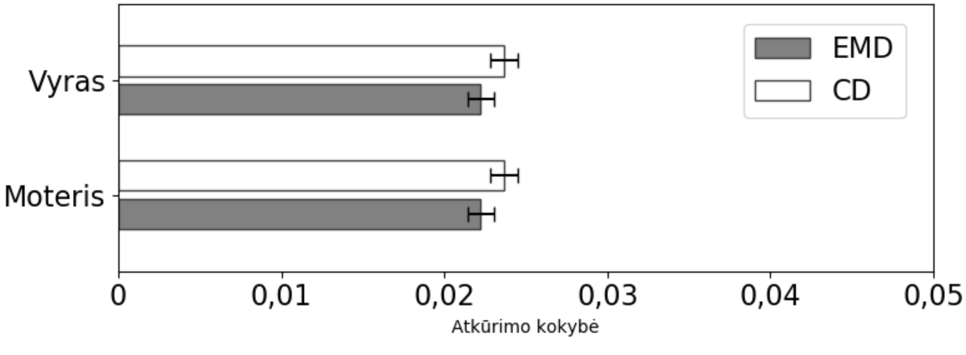
$$f_{cd} = \frac{1}{2}\left(\frac{1}{|S|}\sum_{x\in S}\underset{y\in\hat{S}}{min}\|x-y\|_2^2 + \frac{1}{\hat{S}}\sum_{y\in\hat{S}}\underset{x\in S}{min}\|x-y\|_2^2\right) \tag{24}$$

Tyrimuose naudojamas duomenų rinkinys (*MoVi*) susideda iš įvairių atliekamų pratimų bei veiksmų įrašų. Įrašų kadrai yra naudojami norint sukurti tyrimuose naudojamą duomenų rinkinį, o tam, kad būtų sumažintas panašių duomenų kiekis rinkinyje, yra atrenkamas tik kas 75 kadras. Be to, *MoVi* duomenų rinkinys neturi gylio žemėlapių, tačiau turi judesių fiksavimo informaciją. Pritaikydami judesių fiksavimo duomenis *AMASS* duomenų rinkinyje pateikiamam vyro bei moters modeliams, sukuriami atskiri duomenų taškai kiekvienai lyčiai bei pratimui. Sukurtą duomenų rinkinį suskaldžius pagal 80:20 taisyklę, 80% kadrų priskiriama neuroninio tinklo mokymui, 20% jo validavimui. Su validavimo rinkiniu atliekami kiekybiniai tinklo įvertinimo eksperimentai individualiems kadrams. Išskaidžius eksperimento metu gautus kiekybinius rezultatus pagal atliekamą pratimą (15 pav.) bei subjekto lytį (16 pav.), matyti, kad atkuriamųjų objektų kokybė *HUMANNET—A Two-Tiered Deep Neural Network Architecture for*

Self-Occluding Humanoid Pose Reconstruction straipsnyje neturi didelių nuokrypių. Nesant didelių kokybės metrikų nukrypimų pozų atžvilgiu galima teigti, kad šis gilaus mokymo modelis gali aproksimuoti žmogaus figūrą bei įsiminti žmogaus figūros šabloną. Tolimesnis išskaidymas pagal lytį parodo, kad gilusis neuroninis tinklas ne tik prisitaiko tik prie subjekto kūno formos, bet ir geba suprasti ir atkurti švarias kūno savybes, pavyzdžiui, antrines lyties charakteristikas. Gautieji rezultatai prilygsta ir daugeliu atvejų pranoksta iki šiol literatūroje minimus atkūrimo kokybės rezultatus naudojant tas pačias metrikas (1 lentelė).



15 pav. Straipsnio *HUMANNET—A Two-Tiered Deep Neural Network Architecture for Self-Occluding Humanoid Pose Reconstruction* objekto nematomų zonų atkūrimo kokybės kiekybinių metrikų išskaidymas pagal pratimą



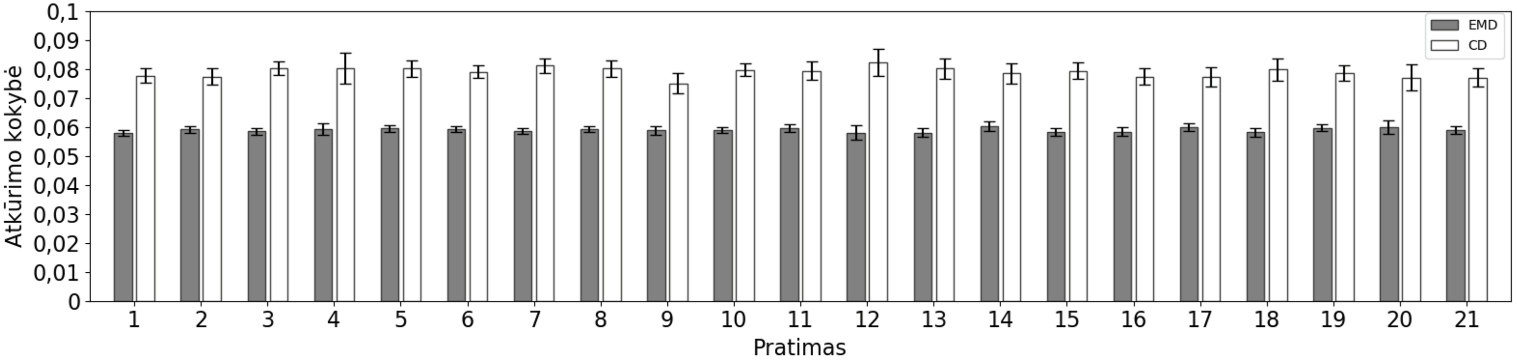
16 pav. Straipsnio *HUMANNET—A Two-Tiered Deep Neural Network Architecture for Self-Occluding Humanoid Pose Reconstruction* objekto nematomų zonų atkūrimo kokybės kiekybinių metrikų išskaidymas pagal lytį

1 lentelė. Skirtingų metodų kiekybinių atkūrimo metrikų palyginimas

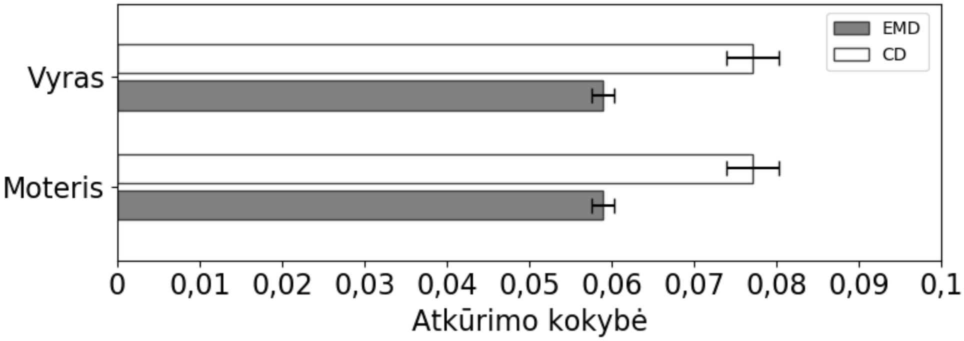
Metodas	**ShapeNet**		**AMASS**	
	EMD	**CD**	**EMD**	**CD**
PCN [119]	0,0734	0,0121	3,0456	4,0955
AtlasNet [120]	0,0653	0,0182	2,0875	6,4343
MSN [109]	0,0378	0,0114	1,1525	0,8016
HumanNet	—	—	0,0256	0,0276

Paskutiniojo mokslinio straipsnio *Auto-Refining Reconstruction Algorithm for Recreation of Limited Angle Humanoid Depth Data* modelis yra pagrįstas taškiniais debesimis, todėl EMD bei CD metrikos buvo naudojamos kiekybiniam įvertinimui. Eksperimentinius atkūrimo rezultatus išskaidžius pagal subjekto atliekamą pratimą (17 pav.) bei lytį (18 pav.), kaip ir ankstesnio straipsnio rezultatuose, nematyti didesnių nukrypimų tarp subjekto kūno formos ar atliekamo pratimo, tai indikatorius, kad neuroninis tinklas tinkamai aproksimuoja žmogaus figūrą bei jos formą. Nors gautieji rezultatai kiekybiniu atžvilgiu yra šiek tiek prastesni (2 lentelė), tačiau šie rezultatai vis tiek prilygsta iki šiol publikuotiems literatūroje. Be to, kitaip nei literatūroje minimi pažangiausi sprendimai, pasiūlytas mašininio mokymo modelis gali dirbti ne tik su sintetiniais duomenimis, bet ir su realaus pasaulio duomenimis. Iki šiol literatūroje minimi nematomų objektų zonų atkūrimo metodai sprendimo šiai problemai spręsti neturėjo, kadangi struktūrizuotos šviesos bei lazeriniai gylio jutikliai grąžina labai triukšmingus rezultatus.

Įvertinus kokybinius eksperimentų rezultatus galima daryti prielaidą, jog modelis geba atkurti įtikinamas nematomas objektų zonas, o kokybiniai tyrimų rezultatai su validavimo duomenų rinkiniu šią prielaidą patvirtina, todėl patvirtinama hipotezė, kad mašininio mokymo sprendimai, gebantys aptikti šablonus, yra tinkami iškeltai problemai spręsti.



17 pav. Straipsnio *Auto-Refining Reconstruction Algorithm for Recreation of Limited Angle Humanoid Depth Data* objekto nematomų zonų atkūrimo kokybės kiekybinių metrikų išskaidymas pagal pratimą



18 pav. Straipsnio *Auto-Refining Reconstruction Algorithm for Recreation of Limited Angle Humanoid Depth Data* objekto nematomų zonų atkūrimo kokybės kiekybinių metrikų išskaidymas pagal lytį

2 lentelė. Skirtingų metodų kiekybinių atkūrimo metrikų palyginimas

Metodas	**ShapeNet**		**AMASS**	
	EMD	**CD**	**EMD**	**CD**
PCN [119]	0,0734	0,0121	3,0456	4,0955
AtlasNet [120]	0,0653	0,0182	2,0875	6,4343
MSN [109]	0,0378	0,0114	1,1525	0,8016
HumanNet	—	—	0,0256	0,0276
Auto-Refining	—	—	0,0590	0,0790

6.4. Išvados

1. Pasiūlytas gilaus mokymo modelis, naudojantis hibridinius giliuosius neuroninius tinklus tūrinių taškų pagrindu, galintis atkurti nematomas objekto zonas ir jo paviršiaus formą 151 kartą per sekundę bei pranokstantis ankstesnius pažangiausius sprendimus 19,3%; taip pat, skirtingai nei kiti sprendimai, pasiūlytam modeliui nereikėjo segmentavimo kaukės vienam objektui atkurti, tai leidžia jį panaudoti realaus laiko sprendimams.

2. Pasiūlytas modelio praplėtimas, atkuriantis keletą objektų tame pačiame gylio jutiklio kadre bei pagerinantis pirmuoju modeliu pasiektus rezultatus 8,53%, naudojant tas pačias kiekybinio vertinimo metrikas.

3. Pasiūlytas mašininio mokymo modelis, naudojantis giliuosius neuroninius tinklus, taškinio debesies pagrindu atkuriantis keletą sudėtingų laike kintančių žmogaus figūrų, kurio kiekybinės $f_{emd} = 0,0256$ bei $f_{cd} = 0,0276$ metrikos prilygsta arba pranoksta kitus pažangiausius objektų atkūrimo sprendimus, tokius kaip PCN, AtlasNet bei MSN, taikomus vieno paprasto objekto nematomų zonų atkūrimo sprendimuose.

4. Pasiūlytas modelis bei jo panaudojimo strategija, naudojant neprižiūrimus giliuosius išvalančiuosius neuroninius tinklus taškinio debesies pagrindu, kitaip nei ankščiau publikuoti pažangiausi sprendimai, atkuria sudėtingus objektus iš triukšmingų struktūrizuotos šviesos bei lazerinių gylio jutiklių duomenų, jo kiekybinės $f_{emd} = 0,059$ bei $f_{cd} = 0,079$ prilygsta kitiems sprendimams, atkuriantiems nematomas objekto zonas tik iš sintetinių duomenų.

5. Atlikti kokybiniai eksperimentai rodo, kad pasiūlyti modeliai geba atkurti objekto nematomas zonas; įvertinus kiekybines metrikas ($f_{emd} = 0,059$ bei $f_{cd} = 0,079$), galima teigti, kad gautieji kokybinio vertinimo rezultatai yra pastovūs visame validavimo duomenų rinkinyje; gilieji neuroniniai tinklai, taip pat kaip ir žmogus, geba atkurti trimatį objektą bei jo nematomas zonas net ir sudėtingomis sąlygomis, t. y. esant netobulam gylio kadrui; tuo patvirtinta iškelta hipotezė, kad mašininio mokymo sprendimai, gebantys aptikti šablonus, yra tinkami iškeltai problemai spręsti.

7. REFERENCES

1. Anju Jose Tom and Sudhish N. George. “Video Completion and Simultaneous Moving Object Detection for Extreme Surveillance Environments”. In: *IEEE Signal Processing Letters* 26.4 (2019), pp. 577–581. DOI: `10.1109/LSP.2019.2900126`.
2. Ibrahim Kajo, Nidal Kamel, and Yassine Ruichek. “Incremental Tensor-Based Completion Method for Detection of Stationary Foreground Objects”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 29.5 (2019), pp. 1325–1338. DOI: `10.1109/TCSVT.2018.2841825`.
3. Jingzhi Tu, Gang Mei, and Francesco Piccialli. “An Efficient Deep Learning Approach Using Improved Generative Adversarial Networks for Incomplete Information Completion of Self-driving”. In: *ArXiv* abs/2109.02629 (2021).
4. Takahiro Hayashi and Motoki Sasaki. “Contour Completion of Partly Occluded Skew-Symmetry Objects”. In: *2014 IEEE International Symposium on Multimedia*. 2014, pp. 90–93. DOI: `10.1109/ISM.2014.15`.
5. Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. “Visual simultaneous localization and mapping: a survey”. In: *Artificial Intelligence Review* 43.1 (2012), pp. 55–81. DOI: `10.1007/s10462-012-9365-8`.
6. Jing Li, Hongtao Huo, Chenhong Sui, Chenchen Jiang, and Chang Li. “Poisson Reconstruction-Based Fusion of Infrared and Visible Images via Saliency Detection”. In: *IEEE Access* 7 (2019), pp. 20676–20688. DOI: `10.1109/ACCESS.2019.2897320`.
7. Yu Zhang, Mao Ye, Dinesh Manocha, and Ruigang Yang. “3D Reconstruction in the Presence of Glass and Mirrors by Acoustic and Visual Fusion”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.8 (2018), pp. 1785–1798. DOI: `10.1109/TPAMI.2017.2723883`.
8. Chen Zhang. “CuFusion2: Accurate and Denoised Volumetric 3D Object Reconstruction Using Depth Cameras”. In: *IEEE Access* 7 (2019), pp. 49882–49893. DOI: `10.1109/ACCESS.2019.2911119`.
9. Xu Chen, Qingfeng Wu, and Shengzhe Wang. “Research on 3D Reconstruction Based on Multiple Views”. In: *2018 13th International Conference on Computer Science Education (ICCSE)*. 2018, pp. 1–5. DOI: `10.1109/ICCSE.2018.8468705`.
10. Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. “3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction”. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Cham: Springer International Publishing, 2016, pp. 628–644. ISBN: 978-3-319-46484-8.
11. Olivia Wiles and Andrew Zisserman. “Learning to Predict 3D Surfaces of Sculptures from Single and Multiple Views”. In: *International Journal of Computer Vision* 127.11-12 (2018), pp. 1780–1800. DOI: `10.1007/s11263-018-1124-0`.
12. Tingsong Ma, Ping Kuang, and Wenhong Tian. “An improved recurrent neural networks for 3d object reconstruction”. In: *Applied Intelligence* 50 (2019), pp. 905–923.
13. Jacky C. K. Chow and Derek D. Lichti. “Photogrammetric Bundle Adjustment With Self-Calibration of the PrimeSense 3D Camera Technology: Microsoft Kinect”. In: *IEEE Access* 1 (2013), pp. 465–474. DOI: `10.1109/ACCESS.2013.2271860`.
14. Lin Yang, Longyu Zhang, Haiwei Dong, Abdulhameed Alelaiwi, and Abdulmotaleb El Saddik. “Evaluating and Improving the Depth Accuracy of Kinect for Windows v2”. In: *IEEE Sensors Journal* 15.8 (2015), pp. 4275–4285. DOI: `10.1109/JSEN.2015.2416651`.
15. Ji-Min Cho, Soon-Yong Park, and Sung-Il Chien. “Hole-Filling of RealSense Depth Images Using a Color Edge Map”. In: *IEEE Access* 8 (2020), pp. 53901–53914. DOI: `10.1109/ACCESS.2020.2981378`.

16. Jingfang Yin, Dengming Zhu, Min Shi, and Zhaoqi Wang. “Depth Maps Restoration for Human Using RealSense”. In: *IEEE Access* 7 (2019), pp. 112544–112553. DOI: `10.1109/ACCESS.2019.2934863`.
17. Hossein Eshraghi, Babak Majidi, and Ali Movaghar. “Anomaly modelling in machine learning based navigation system of autonomous vehicles”. In: *2020 6th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*. 2020, pp. 1–6. DOI: `10.1109/ICSPIS51611.2020.9349606`.
18. Takashi Yasuno, Daiki Tanaka, and Akinobu Kuwahara. “Autonomous navigation system based on collision danger-degree for unmanned ground vehicle”. In: *2014 International Power Electronics Conference (IPEC-Hiroshima 2014 - ECCE ASIA)*. 2014, pp. 3179–3184. DOI: `10.1109/IPEC.2014.6870141`.
19. Maria João Sousa, Alexandra Moutinho, and Miguel Almeida. “Thermal Infrared Sensing for Near Real-Time Data-Driven Fire Detection and Monitoring Systems”. In: *Sensors* 20.23 (2020). ISSN: 1424-8220. DOI: `10.3390/s20236803`. Online: `https://www.mdpi.com/1424-8220/20/23/6803`.
20. Bo Liao, Jing Li, Zhaojie Ju, and Gaoxiang Ouyang. “Hand Gesture Recognition with Generalized Hough Transform and DC-CNN Using Realsense”. In: *2018 Eighth International Conference on Information Science and Technology (ICIST)*. IEEE, June 2018. DOI: `10.1109/icist.2018.8426125`. Online: `https://doi.org/10.1109/icist.2018.8426125`.
21. Chi Chen, Bisheng Yang, Shuang Song, Mao Tian, Jianping Li, Wenxia Dai, and Lina Fang. “Calibrate Multiple Consumer RGB-D Cameras for Low-Cost and Efficient 3D Indoor Mapping”. In: *Remote Sensing* 10.2 (Feb. 2018), p. 328. DOI: `10.3390/rs10020328`. Online: `https://doi.org/10.3390/rs10020328`.
22. Madalina Maria Diac, Kamel Earar, Simona Irina Damian, Anton Knieling, Tatiana Iov, Sarah Shrimpton, Maria Castaneyra-Ruiz, Caroline Wilkinson, and Diana Bulgaru Iliescu. “Facial Reconstruction: Anthropometric Studies Regarding the Morphology of the Nose for Romanian Adult Population I: Nose Width”. In: *Applied Sciences* 10.18 (2020). ISSN: 2076-3417. DOI: `10.3390/app10186479`. Online: `https://www.mdpi.com/2076-3417/10/18/6479`.
23. Mohammad Rostami, Oleg V. Michailovich, and Zhou Wang. “Surface Reconstruction in Gradient-Field Domain Using Compressed Sensing”. In: *IEEE Transactions on Image Processing* 24.5 (2015), pp. 1628–1638. DOI: `10.1109/TIP.2015.2409565`.
24. Chen Yin, Dang Gang, Cheng Zhi-quan, Li Hong-hua, Li Jun, and Jin Shi-yao. “An algorithm of CUDA-based Poisson surface reconstruction”. In: *2010 International Conference on Audio, Language and Image Processing*. 2010, pp. 203–207. DOI: `10.1109/ICALIP.2010.5684972`.
25. F. Pedersini, A. Sarti, and S. Tubaro. “Visible surface reconstruction with accurate localization of object boundaries”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 10.2 (2000), pp. 278–292. DOI: `10.1109/76.825727`.
26. Yubao Liu and Jun Miura. “RDS-SLAM: Real-Time Dynamic SLAM Using Semantic Segmentation Methods”. In: *IEEE Access* 9 (2021), pp. 23772–23785. DOI: `10.1109/ACCESS.2021.3050617`.
27. Hriday Bavle, Paloma De La Puente, Jonathan P. How, and Pascual Campoy. “VPS-SLAM: Visual Planar Semantic SLAM for Aerial Robotic Systems”. In: *IEEE Access* 8 (2020), pp. 60704–60718. DOI: `10.1109/ACCESS.2020.2983121`.
28. Maxime Ferrera, Alexandre Eudes, Julien Moras, Martial Sanfourche, and Guy Le Besnerais. “OV^{2}SLAM: A Fully Online and Versatile Visual SLAM for Real-Time Applications”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 1399–1406. DOI: `10.1109/LRA.2021.3058069`.
29. Seung-Mok Lee, Jongdae Jung, Shin Kim, In-Joo Kim, and Hyun Myung. “DV-SLAM (Dual-Sensor-Based Vector-Field SLAM) and Observability Analysis”.

In: *IEEE Transactions on Industrial Electronics* 62.2 (2015), pp. 1101–1112. DOI: `10.1109/TIE.2014.2341595`.

30. Javier Pérez, Mitch Bryson, Stefan B. Williams, and Pedro J. Sanz. “Recovering Depth from Still Images for Underwater Dehazing Using Deep Learning”. In: *Sensors* 20.16 (2020). ISSN: 1424-8220. DOI: `10.3390/s20164580`. Online: `https://www.mdpi.com/1424-8220/20/16/4580`.
31. Alberto Díaz-Álvarez, Miguel Clavijo, Felipe Jiménez, and Francisco Serradilla. “Inferring the Driver’s Lane Change Intention through LiDAR-Based Environment Analysis Using Convolutional Neural Networks”. In: *Sensors* 21.2 (2021). ISSN: 1424-8220. DOI: `10.3390/s21020475`. Online: `https://www.mdpi.com/1424-8220/21/2/475`.
32. Melissa Latella, Fabio Sola, and Carlo Camporeale. “A Density-Based Algorithm for the Detection of Individual Trees from LiDAR Data”. In: *Remote Sensing* 13.2 (2021). ISSN: 2072-4292. DOI: `10.3390/rs13020322`. Online: `https://www.mdpi.com/2072-4292/13/2/322`.
33. W. Ren, O. Ma, H. Ji, and X. Liu. “Human Posture Recognition Using a Hybrid of Fuzzy Logic and Machine Learning Approaches”. In: *IEEE Access* 8 (2020), pp. 135628–135639. DOI: `10.1109/ACCESS.2020.3011697`.
34. Audrius Kulikajevas, Rytis Maskeliunas, and Robertas Damaševičius. “Detection of sitting posture using hierarchical image composition and deep learning”. In: *PeerJ Computer Science* 7 (2021). DOI: `10.7717/peerj-cs.442`.
35. F. F. Ting, K. S. Sim, and Y. Lee. “Three-dimensional model reconstruction using surface interpolation with the interfacing of Hermite surface for breast cancer MRI imaging system”. In: *2016 International Conference on Robotics, Automation and Sciences (ICORAS)*. 2016, pp. 1–5. DOI: `10.1109/ICORAS.2016.7872625`.
36. Bert Coolen, Peter J. Beek, Daphne J. Geerse, and Melvyn Roerdink. “Avoiding 3D Obstacles in Mixed Reality: Does It Differ from Negotiating Real Obstacles?” In: *Sensors* 20.4 (2020), p. 1095. DOI: `10.3390/s20041095`.
37. Bruno Fanini, Alfonsina Pagano, and Daniele Ferdani. “A Novel Immersive VR Game Model for Recontextualization in Virtual Environments: The uVRModel”. In: *Multimodal Technologies and Interaction* 2.2 (Apr. 2018), p. 20. DOI: `10.3390/mti2020020`. Online: `https://doi.org/10.3390/mti2020020`.
38. Alex Ibañez-Etxeberria, Cosme J. Gómez-Carrasco, Olaia Fontal, and Silvia García-Ceballos. “Virtual Environments and Augmented Reality Applied to Heritage Education. An Evaluative Study”. In: *Applied Sciences* 10.7 (2020). ISSN: 2076-3417. DOI: `10.3390/app10072352`. Online: `https://www.mdpi.com/2076-3417/10/7/2352`.
39. Åsa Fast-Berglund, Liang Gong, and Dan Li. “Testing and validating Extended Reality (xR) technologies in manufacturing”. In: *Procedia Manufacturing* 25 (2018). Proceedings of the 8th Swedish Production Symposium (SPS 2018), pp. 31–38. ISSN: 2351-9789. DOI: `https://doi.org/10.1016/j.promfg.2018.06.054`. Online: `http://www.sciencedirect.com/science/article/pii/S2351978918305730`.
40. Hansung Kim, Jean-Yves Guillemaut, Takeshi Takai, Muhammad Sarim, and Adrian Hilton. “Outdoor Dynamic 3-D Scene Reconstruction”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.11 (2012), pp. 1611–1622. DOI: `10.1109/TCSVT.2012.2202185`.
41. A. Kamilaris and F. X. Prenafeta-Boldú. “Deep learning in agriculture: A survey”. In: *Computers and Electronics in Agriculture* 147 (2018), pp. 70–90.
42. L. Wen, X. Li, L. Gao, and Y. Zhang. “A New Convolutional Neural Network-Based Data-Driven Fault Diagnosis Method”. In: *IEEE Transactions on Industrial Electronics* 65.7 (2018), pp. 5990–5998.

43. T. Hirasawa, K. Aoyama, T. Tanimoto, S. Ishihara, S. Shichijo, T. Ozawa, T. Ohnishi, M. Fujishiro, K. Matsuo, J. Fujisaki, and T. Tada. “Application of artificial intelligence using a convolutional neural network for detecting gastric cancer in endoscopic images”. In: *Gastric Cancer* 21.4 (2018), pp. 653–660.
44. Y. Li and L. Shen. “Skin lesion analysis towards melanoma detection using deep learning network”. In: *Sensors (Switzerland)* 18.2 (2018).
45. R. Chai, S. H. Ling, P. P. San, G. R. Naik, T. N. Nguyen, Y. Tran, A. Craig, and H. T. Nguyen. “Improving EEG-based driver fatigue classification using sparse-deep belief networks”. In: *Frontiers in Neuroscience* 11.MAR (2017).
46. Q. Ke, J. Zhang, W. Wei, D. Połap, M. Woźniak, L. Kośmider, and R. Damaševĭcius. “A neuro-heuristic approach for recognition of lung diseases from X-ray images”. In: *Expert Systems with Applications* 126 (2019), pp. 218–232.
47. Herbert Edelsbrunner and Ernst P. Mücke. “Three-Dimensional Alpha Shapes”. In: *ACM Trans. Graph.* 13.1 (Jan. 1994), pp. 43–72. ISSN: 0730-0301. DOI: `10.1145/174462.156635`. Online: `https://doi.org/10.1145/174462.156635`.
48. F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. “The ball-pivoting algorithm for surface reconstruction”. In: *IEEE Transactions on Visualization and Computer Graphics* 5.4 (1999), pp. 349–359. DOI: `10.1109/2945.817351`.
49. Julie Digne. “An Analysis and Implementation of a Parallel Ball Pivoting Algorithm”. In: *Image Processing On Line* 4 (2014). `https://doi.org/10.5201/ipol.2014.81`, pp. 149–168.
50. Michael Kazhdan and Hugues Hoppe. “Screened Poisson Surface Reconstruction”. In: *ACM Trans. Graph.* 32.3 (July 2013). ISSN: 0730-0301. DOI: `10.1145/2487228.2487237`. Online: `https://doi.org/10.1145/2487228.2487237`.
51. I. Shimshoni, Y. Moses, and M. Lindenbaumlpr. “Shape reconstruction of 3D bilaterally symmetric surfaces”. In: *Proceedings 10th International Conference on Image Analysis and Processing*. 1999, pp. 76–81. DOI: `10.1109/ICIAP.1999.797574`.
52. Demetri Terzopoulos, Andrew Witkin, and Michael Kass. “Symmetry-seeking models and 3D object reconstruction”. In: *International Journal of Computer Vision* 1.3 (1988), pp. 211–221. DOI: `10.1007/bf00127821`.
53. Allen Y. Yang, Kun Huang, Shankar Rao, Wei Hong, and Yi Ma. “Symmetry-Based 3-D Reconstruction from Perspective Images”. In: *Comput. Vis. Image Underst.* 99.2 (Aug. 2005), pp. 210–240. ISSN: 1077-3142.
54. Tianfan Xue, Jianzhuang Liu, and Xiaoou Tang. “Symmetric piecewise planar object reconstruction from a single image”. In: *CVPR 2011*. 2011, pp. 2577–2584. DOI: `10.1109/CVPR.2011.5995405`.
55. Marc Forstenhäusler, Nico Engel, and Klaus Dietmayer. “Temporal Filtering to Stabilize Features for SLAM”. In: *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2019, pp. 1592–1597. DOI: `10.1109/AIM.2019.8868846`.
56. Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. “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 ’11. Santa Barbara, California, USA: Association for Computing Machinery, 2011, pp. 559–568. ISBN: 9781450307161. DOI: `10.1145/2047196.2047270`. Online: `https://doi.org/10.1145/2047196.2047270`.
57. Angel X. Chang, Thomas A. Funkhouser, Leonidas J Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. “ShapeNet: An Information-Rich 3D Model Repository”. In: *CoRR* abs/1512.03012 (2015).

58. Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. "Deep Metric Learning via Lifted Structured Feature Embedding". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

59. K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. "LSTM: A Search Space Odyssey". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (2017), pp. 2222–2232. DOI: `10.1109/TNNLS.2016.2582924`.

60. W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang. "Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network". In: *IEEE Transactions on Smart Grid* 10.1 (2019), pp. 841–851. DOI: `10.1109/TSG.2017.2753802`.

61. Amol Dhondse, Siddhivinayak Kulkarni, Kunal Khadilkar, Indrajeet Kane, Sumit Chavan, and Rahul Barhate. "Generative Adversarial Networks as an Advancement in 2D to 3D Reconstruction Techniques". In: *Data Management, Analytics and Innovation*. Ed. by Neha Sharma, Amlan Chakrabarti, and Valentina Emilia Balas. Singapore: Springer Singapore, 2020, pp. 343–364. ISBN: 978-981-13-9364-8.

62. C. Guzel Turhan and H.S. Bilge. "Fused voxel autoencoder for single image to 3D object reconstruction". In: *Electronics Letters* 56.3 (2020), pp. 134–137. DOI: `https://doi.org/10.1049/el.2019.3293`. eprint: `https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/el.2019.3293`. Online: `https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/el.2019.3293`.

63. Christian Häne, Shubham Tulsiani, and Jitendra Malik. "Hierarchical Surface Prediction for 3D Object Reconstruction". In: *2017 International Conference on 3D Vision (3DV)* (2017), pp. 412–420.

64. Haoqiang Fan, Hao Su, and L. Guibas. "A Point Set Generation Network for 3D Object Reconstruction from a Single Image". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 2463–2471.

65. Wenxuan Wu, Zhongang Qi, and Fuxin Li. "PointConv: Deep Convolutional Networks on 3D Point Clouds". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 9613–9622.

66. S. Targ, Diogo Almeida, and Kevin Lyman. "Resnet in Resnet: Generalizing Residual Architectures". In: *ArXiv* abs/1603.08029 (2016).

67. Andrew G. Howard, Menglong Zhu, Bo Chen, D. Kalenichenko, Weijun Wang, Tobias Weyand, M. Andreetto, and Hartwig Adam. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *ArXiv* abs/1704.04861 (2017).

68. R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 77–85. DOI: `10.1109/CVPR.2017.16`.

69. Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. "PCN: Point Completion Network". In: *2018 International Conference on 3D Vision (3DV)*. 2018, pp. 728–737. DOI: `10.1109/3DV.2018.00088`.

70. Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. "A Papier-Mache Approach to Learning 3D Surface Generation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 216–224.

71. Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. "Morphing and Sampling Network for Dense Point Cloud Completion". In: *arXiv preprint arXiv:1912.00280* (2019).

72. M. Tatarchenko, A. Dosovitskiy, and T. Brox. "Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2107–2115. DOI: `10.1109/ICCV.2017.230`.

73. Audrius Kulikajevas, Rytis Maskeliūnas, Robertas Damaševičius, and Sanjay Misra. "Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from ShapeNetCore Dataset". In: *Sensors* 19.7 (2019). ISSN: 1424-8220. DOI: `10.3390/s19071553`. Online: `https://www.mdpi.com/1424-8220/19/7/1553`.
74. Audrius Kulikajevas, Rytis Maskeliūnas, Robertas Damaševičius, and Edmond S. L. Ho. "3D Object Reconstruction from Imperfect Depth Data Using Extended YOLOv3 Network". In: *Sensors* 20.7 (2020). ISSN: 1424-8220. DOI: `10.3390/s20072025`. Online: `https://www.mdpi.com/1424-8220/20/7/2025`.
75. Audrius Kulikajevas, Rytis Maskeliunas, Robertas Damasevicius, and Rafal Scherer. "HUMANNET—A Two-Tiered Deep Neural Network Architecture for Self-Occluding Humanoid Pose Reconstruction". In: *Sensors* 21.12 (2021). ISSN: 1424-8220. DOI: `10.3390/s21123945`. Online: `https://www.mdpi.com/1424-8220/21/12/3945`.
76. Audrius Kulikajevas, Rytis Maskeliūnas, Robertas Damaševičius, and Marta Wlodarczyk-Sielicka. "Auto-Refining Reconstruction Algorithm for Recreation of Limited Angle Humanoid Depth Data". In: *Sensors* 21.11 (2021). ISSN: 1424-8220. DOI: `10.3390/s21113702`. Online: `https://www.mdpi.com/1424-8220/21/11/3702`.
77. Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. "What is the best multi-stage architecture for object recognition?" In: *2009 IEEE 12th International Conference on Computer Vision*. 2009, pp. 2146–2153. DOI: `10.1109/ICCV.2009.5459469`.
78. Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 9781605589077.
79. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034. DOI: `10.1109/ICCV.2015.123`.
80. Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. "A Simple Way to Initialize Recurrent Networks of Rectified Linear Units". In: *CoRR* abs/1504.00941 (2015). arXiv: `1504.00941`. Online: `http://arxiv.org/abs/1504.00941`.
81. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. Online: `http://jmlr.org/papers/v15/srivastava14a.html`.
82. George E. Dahl, Tara N. Sainath, and Geoffrey E. Hinton. "Improving deep neural networks for LVCSR using rectified linear units and dropout". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 8609–8613. DOI: `10.1109/ICASSP.2013.6639346`.
83. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Commun. ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782. DOI: `10.1145/3065386`. Online: `https://doi.org/10.1145/3065386`.
84. Jun Han and Claudio Moraga. "The influence of the sigmoid function parameters on the speed of backpropagation learning". In: *From Natural to Artificial Neural Computation*. Ed. by José Mira and Francisco Sandoval. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 195–201. ISBN: 978-3-540-49288-7.
85. Gang Wang, Georgios B. Giannakis, and Jie Chen. "Learning ReLU Networks on Linearly Separable Data: Algorithm, Optimality, and Generalization". In: *IEEE Transactions on Signal Processing* 67.9 (2019), pp. 2357–2370. DOI: `10.1109/TSP.2019.2904921`.

86. Khaled Mabrouk Amer Adweb, Nadire Cavus, and Boran Sekeroglu. “Cervical Cancer Diagnosis Using Very Deep Networks Over Different Activation Functions”. In: *IEEE Access* 9 (2021), pp. 46612–46625. DOI: `10.1109/ACCESS.2021.3067195`.
87. Kamaledin Ghiasi-Shirazi. “Competitive Cross-Entropy Loss: A Study on Training Single-Layer Neural Networks for Solving Nonlinearly Separable Classification Problems”. In: *Neural Processing Letters* 50.2 (2018), pp. 1115–1122. DOI: `10.1007/s11063-018-9906-5`.
88. Jacopo Pantaleoni. “VoxelPipe”. In: *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics - HPG*. ACM Press, 2011. DOI: `10.1145/2018323.2018339`. Online: `https://doi.org/10.1145/2018323.2018339`.
89. Doug Baldwin and Michael Weber. “Fast Ray-Triangle Intersections by Coordinate Transformation”. In: *Journal of Computer Graphics Techniques (JCGT)* 5.3 (Sept. 2016), pp. 39–49. ISSN: 2331-7418. Online: `http://jcgt.org/published/0005/03/03/`.
90. Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2015).
91. Martin Rutzinger, Franz Rottensteiner, and Norbert Pfeifer. “A Comparison of Evaluation Techniques for Building Extraction From Airborne Laser Scanning”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 2.1 (2009), pp. 11–20. DOI: `10.1109/JSTARS.2009.2012488`.
92. Joseph Redmon and Ali Farhadi. “YOLOv3: An Incremental Improvement”. In: *CoRR* abs/1804.02767 (2018). Online: `http://arxiv.org/abs/1804.02767`.
93. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. “SSD: Single Shot MultiBox Detector”. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Cham: Springer International Publishing, 2016, pp. 21–37. ISBN: 978-3-319-46448-0.
94. Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587. DOI: `10.1109/CVPR.2014.81`.
95. Debjyoti Sinha and Mohamed El-Sharkawy. “Thin MobileNet: An Enhanced MobileNet Architecture”. In: *2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*. 2019, pp. 0280–0285. DOI: `10.1109/UEMCON47517.2019.8993089`.
96. Dongseok Im, Donghyeon Han, Sungpill Choi, Sanghoon Kang, and Hoi-Jun Yoo. “DT-CNN: An Energy-Efficient Dilated and Transposed Convolutional Neural Network Processor for Region of Interest Based Image Segmentation”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 67.10 (2020), pp. 3471–3483. DOI: `10.1109/TCSI.2020.2991189`.
97. Mehmet Saygın Seyfioğlu, Ahmet Murat Özbayoğlu, and Sevgi Zubeyde Gürbüz. “Deep convolutional autoencoder for radar-based classification of similar aided and unaided human activities”. In: *IEEE Transactions on Aerospace and Electronic Systems* 54.4 (2018), pp. 1709–1723. DOI: `10.1109/TAES.2018.2799758`.
98. K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. “Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising”. In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3142–3155. DOI: `10.1109/TIP.2017.2662206`.
99. S. Wu, G. Li, L. Deng, L. Liu, D. Wu, Y. Xie, and L. Shi. “$L1$ -Norm Batch Normalization for Efficient Training of Deep Neural Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* 30.7 (2019), pp. 2043–2051. DOI: `10.1109/TNNLS.2018.2876179`.
100. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. “Going deeper with convolutions”. In: *2015 IEEE Conference on Computer Vision*

and Pattern Recognition (CVPR). 2015, pp. 1–9. DOI: `10.1109/CVPR.2015.7298594`.

101. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: `10.1109/CVPR.2016.90`.
102. Yeongbin Kim, Joongchol Shin, Hasil Park, and Joonki Paik. “Real-Time Visual Tracking with Variational Structure Attention Network”. In: *Sensors* 19.22 (2019). ISSN: 1424-8220. DOI: `10.3390/s19224904`. Online: `https://www.mdpi.com/1424-8220/19/22/4904`.
103. Kouhei Sekiguchi, Yoshiaki Bando, Aditya Arie Nugraha, Kazuyoshi Yoshii, and Tatsuya Kawahara. “Semi-Supervised Multichannel Speech Enhancement With a Deep Speech Prior”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27.12 (2019), pp. 2197–2212. DOI: `10.1109/TASLP.2019.2944348`.
104. S. Kullback and R. A. Leibler. “On Information and Sufficiency”. In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. DOI: `10.1214/aoms/1177729694`.
105. George Philipp, Dawn Xiaodong Song, and Jaime G. Carbonell. “The exploding gradient problem demystified - definition, prevalence, impact, origin, tradeoffs, and solutions”. In: *arXiv: Learning* (2017).
106. Z. Mi, Y. Luo, and W. Tao. “SSRNet: Scalable 3D Surface Reconstruction Network”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 967–976. DOI: `10.1109/CVPR42600.2020.00105`.
107. G. Borgefors. “Hierarchical chamfer matching: a parametric edge matching algorithm”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10.6 (1988), pp. 849–865. DOI: `10.1109/34.9107`.
108. Yossi Rubner, Carlo Tomasi, and Leonidas Guibas. “The Earth Mover’s Distance as a metric for image retrieval”. In: *International Journal of Computer Vision* 40 (Jan. 2000), pp. 99–121.
109. Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. “Morphing and Sampling Network for Dense Point Cloud Completion”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.07 (Apr. 2020), pp. 11596–11603. DOI: `10.1609/aaai.v34i07.6827`. Online: `https://ojs.aaai.org/index.php/AAAI/article/view/6827`.
110. Marius Miknis, Ross Davies, Peter Plassmann, and Andrew Ware. “Near real-time point cloud processing using the PCL”. In: *2015 International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2015, pp. 153–156. DOI: `10.1109/IWSSIP.2015.7314200`.
111. Carsten Moenning and Neil A. Dodgson. “Fast Marching farthest point sampling”. In: *Eurographics 2003 - Posters*. Eurographics Association, 2003. DOI: `10.2312/egp.20031024`.
112. Pegah Kamousi, Sylvain Lazard, Anil Maheshwari, and Stefanie Wuhrer. “Analysis of farthest point sampling for approximating geodesics in a graph”. In: *Computational Geometry* 57 (2016), pp. 1–7. ISSN: 0925-7721. DOI: `https://doi.org/10.1016/j.comgeo.2016.05.005`. Online: `https://www.sciencedirect.com/science/article/pii/S0925772116300487`.
113. Dong Liu, Yizhou Zhou, Xiaoyan Sun, Zhengjun Zha, and Wenjun Zeng. “Adaptive Pooling in Multi-instance Learning for Web Video Annotation”. In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. 2017, pp. 318–327. DOI: `10.1109/ICCVW.2017.46`.
114. Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *CoRR* abs/1506.01497 (2015). arXiv: `1506.01497`. Online: `http://arxiv.org/abs/1506.01497`.

115. Saeed Ghorbani, Kimia Mahdaviani, Anne Thaler, Konrad Kording, Douglas James Cook, Gunnar Blohm, and Nikolaus F. Troje. *MoVi: A Large Multipurpose Motion and Video Dataset*. 2020. arXiv: `2003.01888 [cs.CV]`.

116. Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. “AMASS: Archive of Motion Capture as Surface Shapes”. In: *International Conference on Computer Vision*. Oct. 2019, pp. 5442–5451.

117. Koyel Mukherjee, Alind Khare, and Ashish Verma. “A Simple Dynamic Learning Rate Tuning Algorithm For Automated Training of DNNs”. In: *ArXiv* abs/1910.11605 (2019).

118. Andras Hajdu, Lajos Hajdu, and Robert Tijdeman. *Approximations of the Euclidean distance by chamfer distances*. 2012. arXiv: `1201.0876 [cs.IT]`.

119. Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. “PCN: Point Completion Network”. In: *CoRR* abs/1808.00671 (2018). arXiv: `1808.00671`. Online: `http://arxiv.org/abs/1808.00671`.

120. Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. “AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation”. In: *CoRR* abs/1802.05384 (2018). arXiv: `1802.05384`. Online: `http://arxiv.org/abs/1802.05384`.

121. Yanghua Jin, Jiakai Zhang, Minjun Li, Yingtao Tian, Huachun Zhu, and Zhihao Fang. “Towards the Automatic Anime Characters Creation with Generative Adversarial Networks”. In: *ArXiv* abs/1708.05509 (2017).

122. Yuki Saito, Shinnosuke Takamichi, and Hiroshi Saruwatari. “Statistical Parametric Speech Synthesis Incorporating Generative Adversarial Networks”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.1 (2018), pp. 84–96. DOI: `10.1109/TASLP.2017.2761547`.

123. Yan Huang, Wei Wang, Liang Wang, and Tieniu Tan. “Multi-task deep neural network for multi-label learning”. In: *2013 IEEE International Conference on Image Processing*. 2013, pp. 2897–2900. DOI: `10.1109/ICIP.2013.6738596`.

8. SUBMITTED PAPERS

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

Authors 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

Sensors 2019, 19(7), 1553; https://doi.org/10.3390/s19071553

Received: 28 February 2019 / Revised: 27 March 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

8.1.1. 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, 16, 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 8.1.2 describes the proposed modular neural network, reconstruction algorithm for 3D object recognition, and network training; Section 8.1.3 presents experimental results of the proposed network architecture; and finally, Section 8.1.4 discusses the results and concludes the article.

8.1.2. Materials and Methods

8.1.2.1. 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 reconstruction nodes, where each reconstruction 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

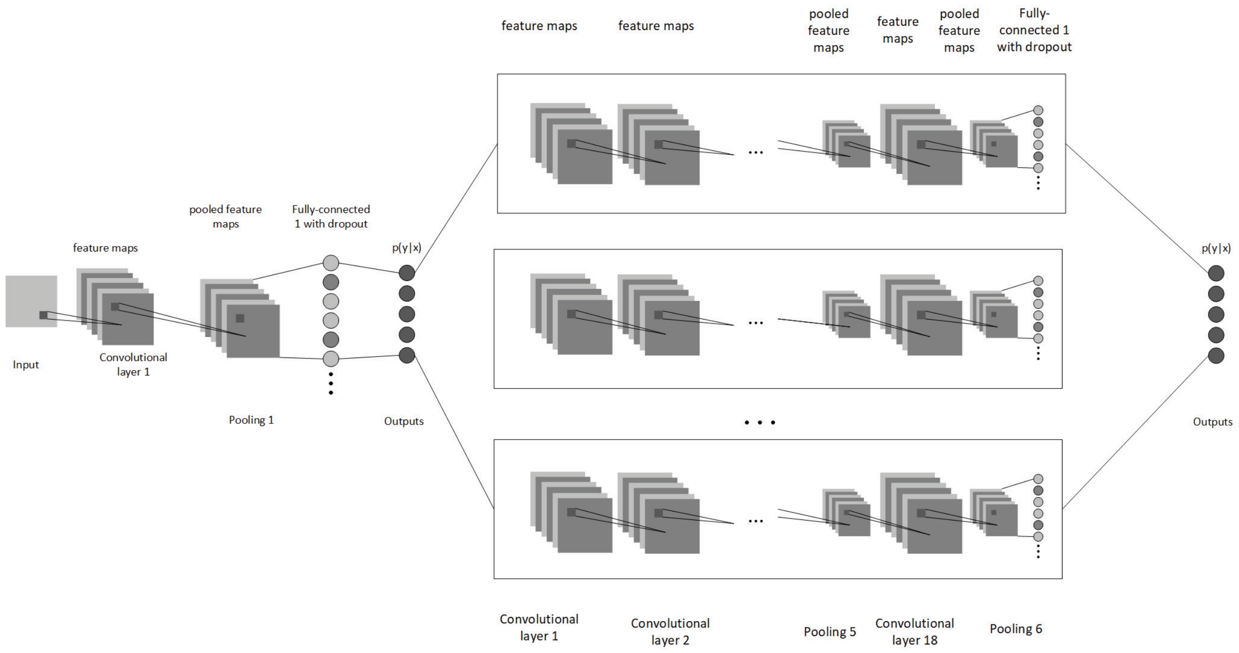


Figure 1. Architecture of hybrid neural network consisting of discriminator and reconstruction. The node accepts depth data as an input, applies a 3 × 3 max-pool convolution layer with a stride of 2 and 32 samples. Following the convolution layer a dropout layer of $P(x) = 0.2$ is applied to avoid overfitting. 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.

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.

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

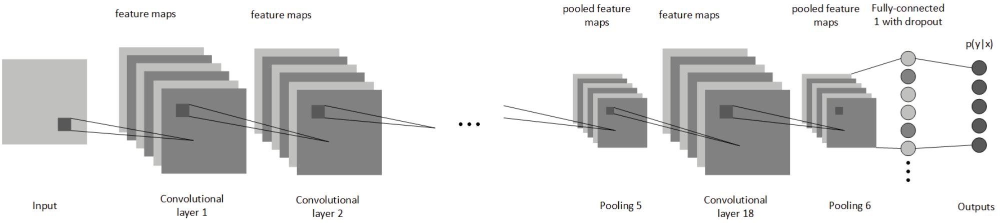


Figure 2. A single branch of ANN used for reconstruction of voxel space.

8.1.2.2. 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.

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

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

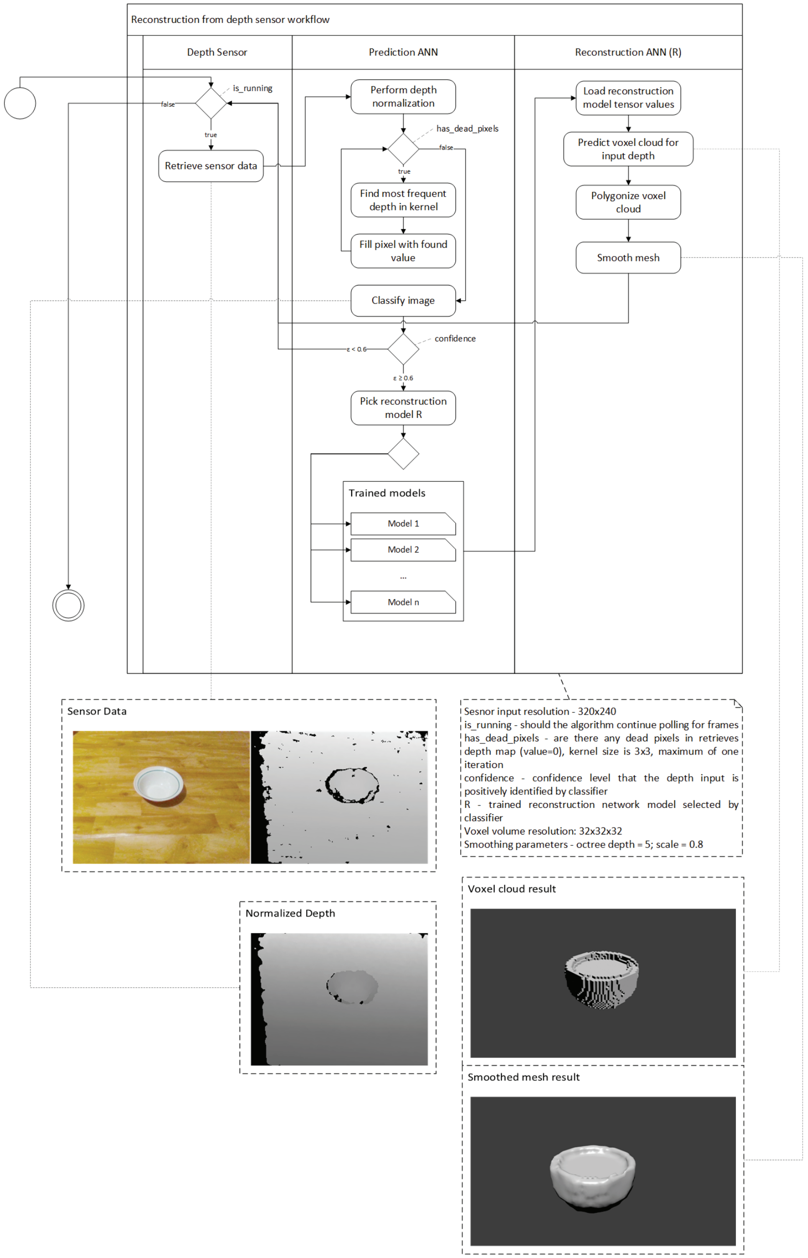
8.1.2.3. Network Training

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.

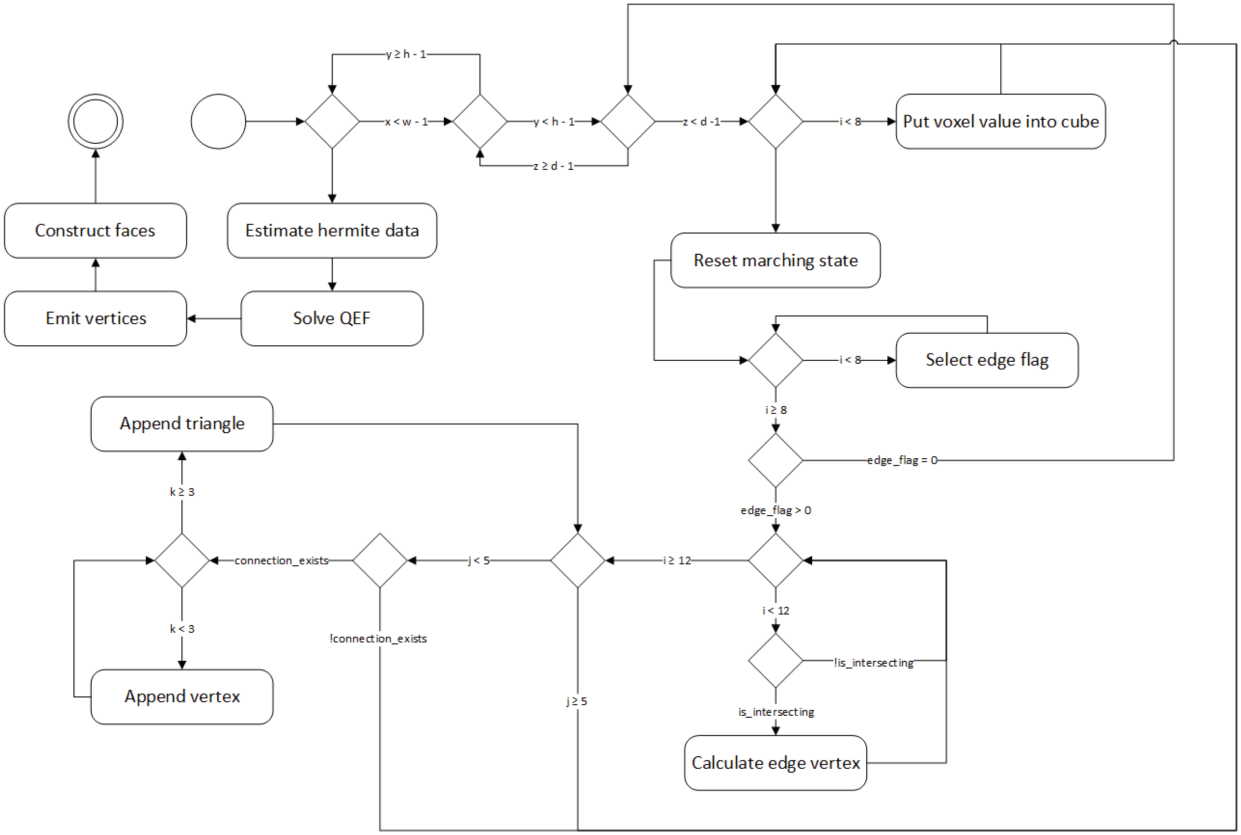
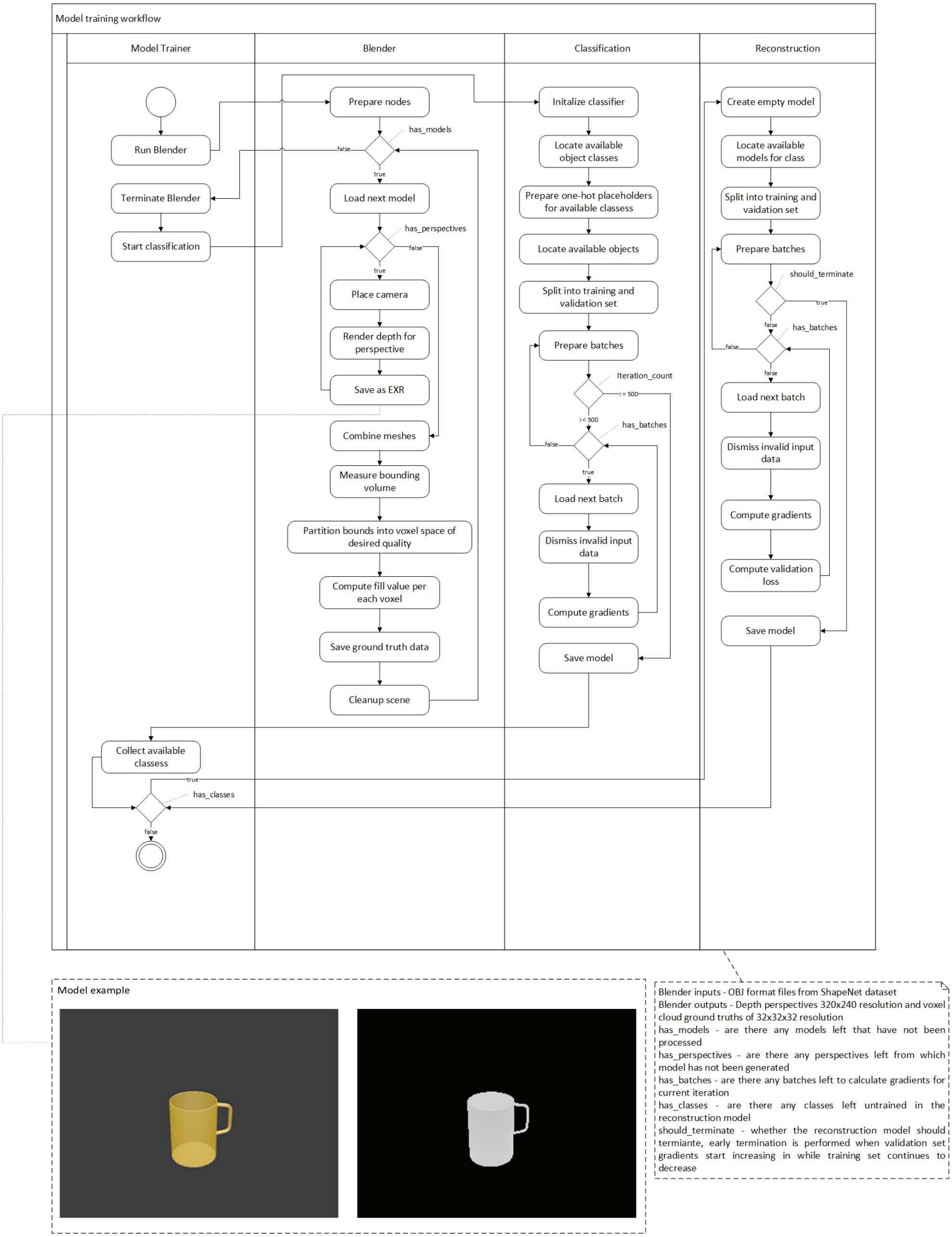


Figure 4. Mesh conversion into polygons via marching cubes.

8.1.2.4. 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 8, 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.

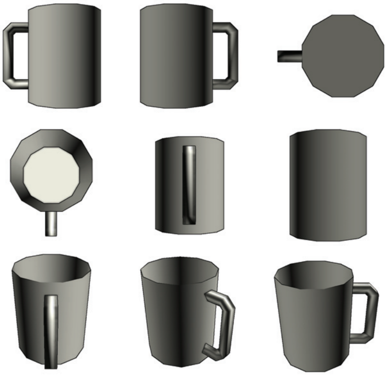


Figure 6. An example of a model from *ShapeNetCore* dataset.

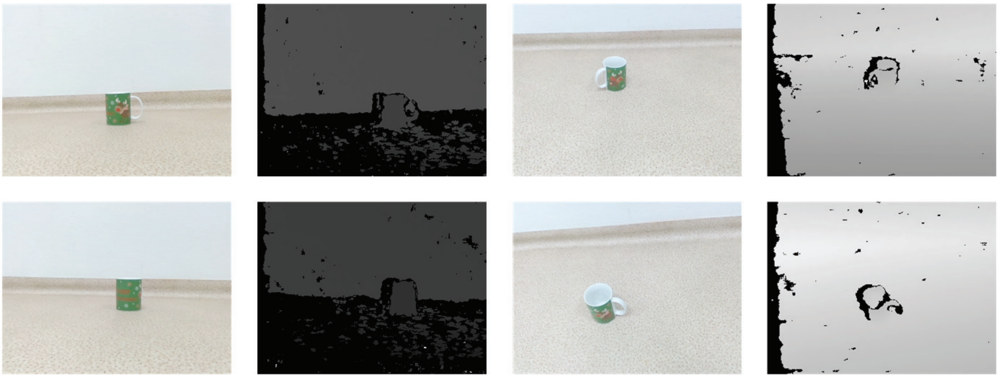


Figure 7. An example of real depth sensor data set captured by *Intel RealSense ZR300* captured from different vantage points.

8.1.2.5. 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:

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

8.1.3. Results

8.1.3.1. 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.

8.1.3.2. 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.

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

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,

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 Training 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

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

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.

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.

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.

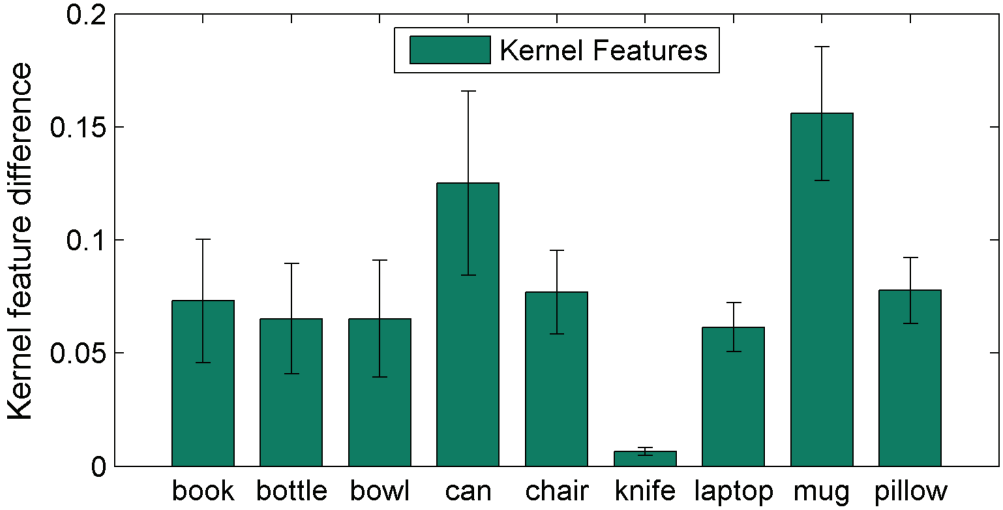


Figure 10. Comparison of kernel features when using 3×3 kernel to identify similarities between objects. Zero indicates that kernel features are identical to those of ground truth.

8.1.3.3. 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

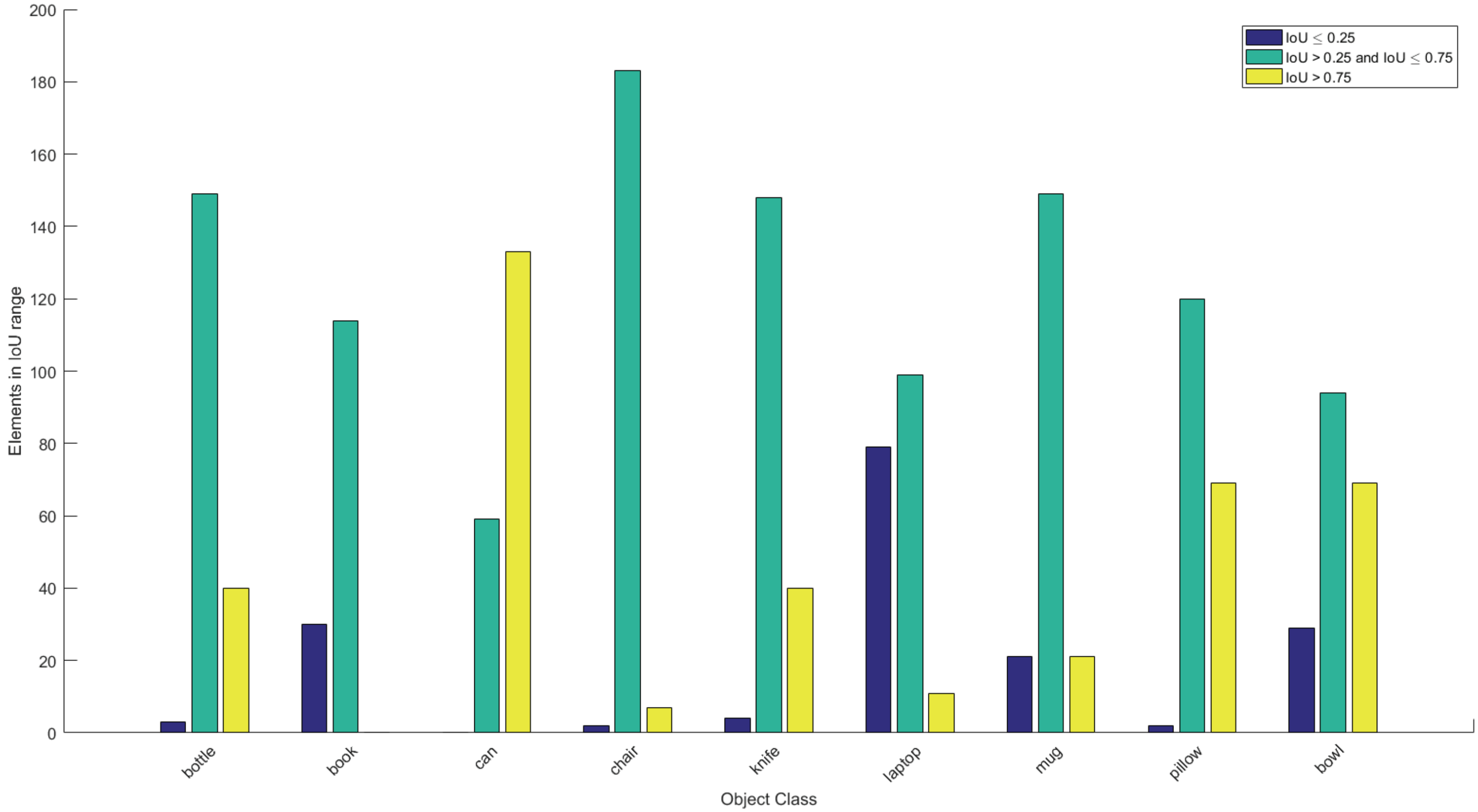


Figure 8. Histogram of the IoU values for heuristic comparison of reconstruction results.

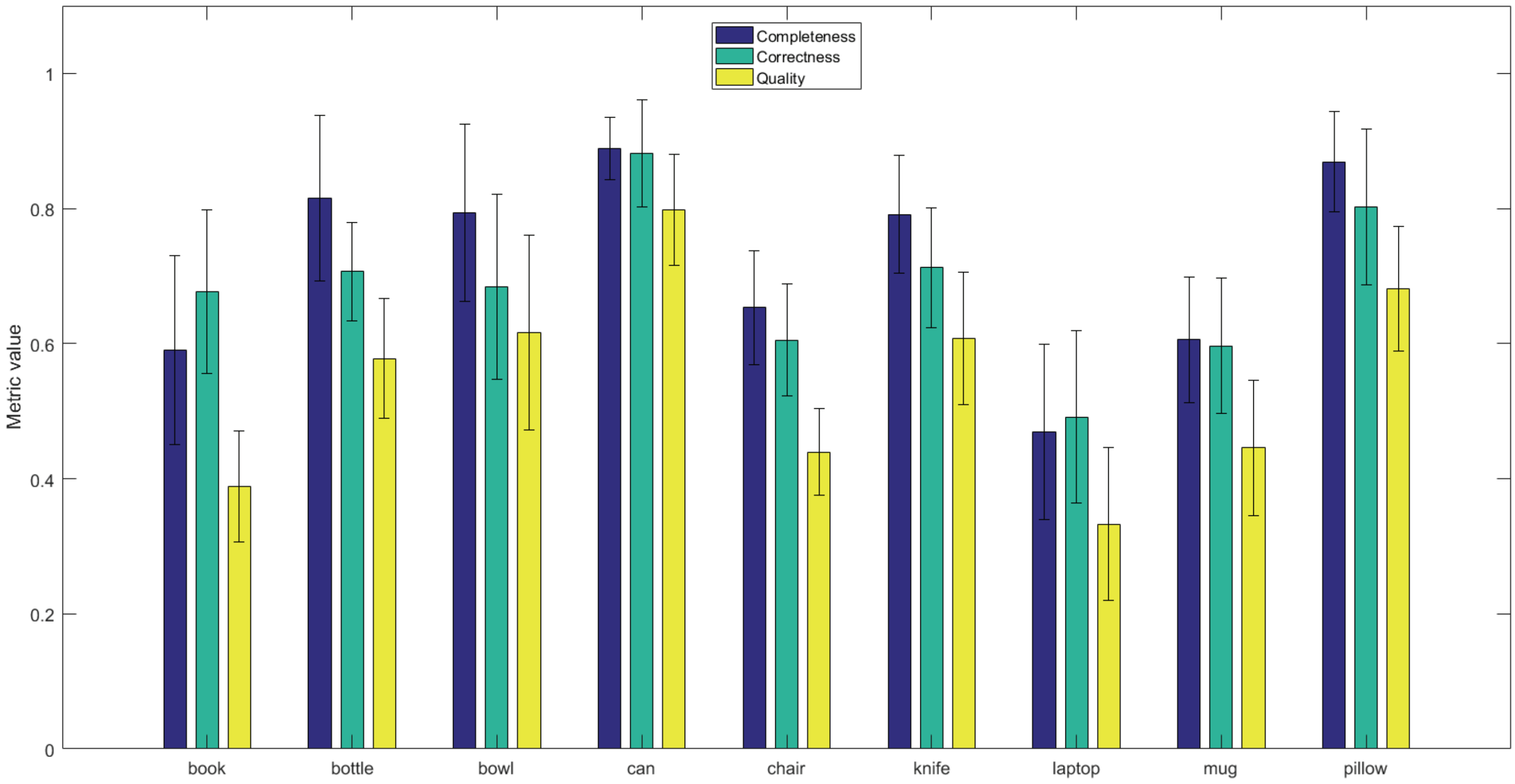


Figure 9. Comparison of reconstruction results using *Completeness*, *Correctness*, and *Quality*) metric values.

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 5 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 5), the reconstructed bowl is very shallow. In the second reconstruction (see the 2nd row of Table 5), 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 × 32 × 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.

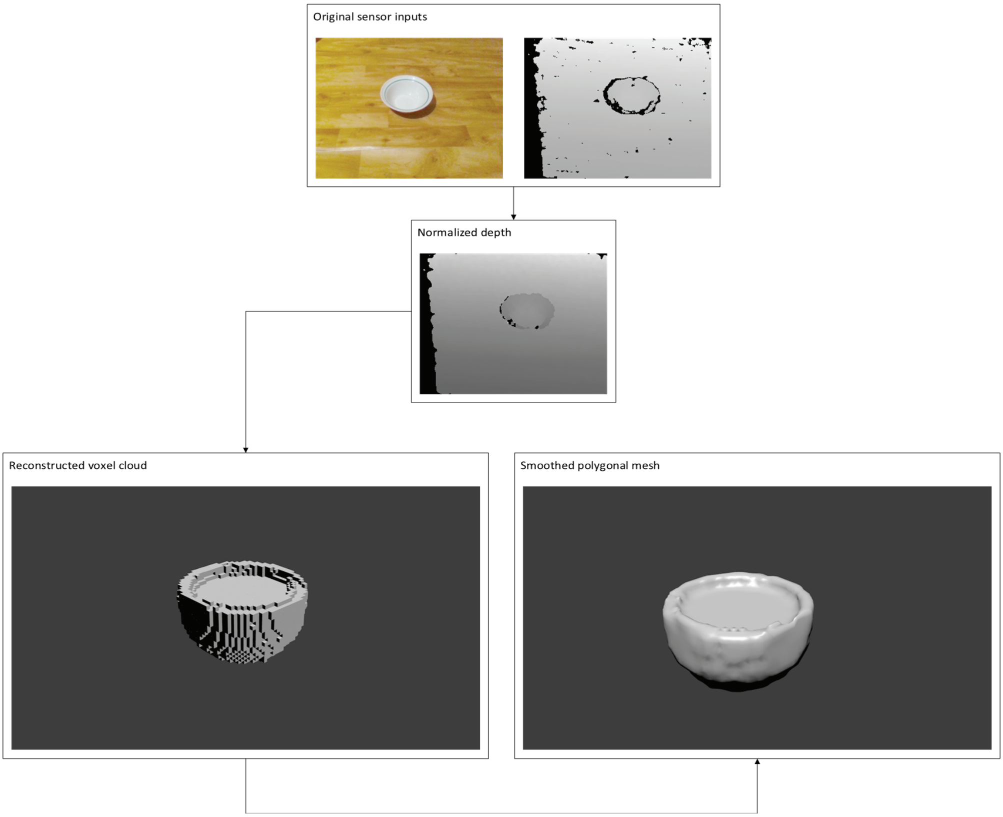
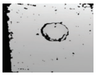
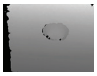
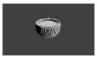
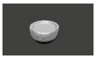
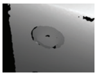
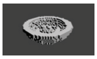
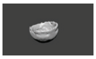
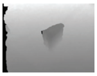
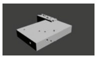
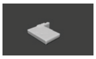
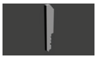
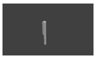
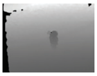
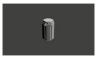
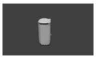
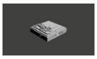
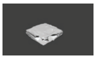
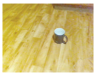
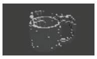
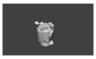
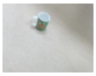
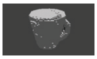
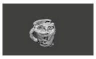
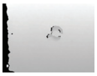
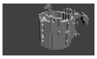
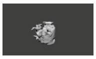
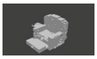
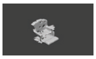
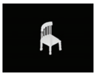
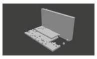
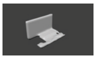
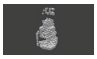
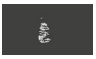


Figure 11. Visual comparison between inputs and outputs.

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
					
					
					
					
					
					
					
					
					
					
					
					

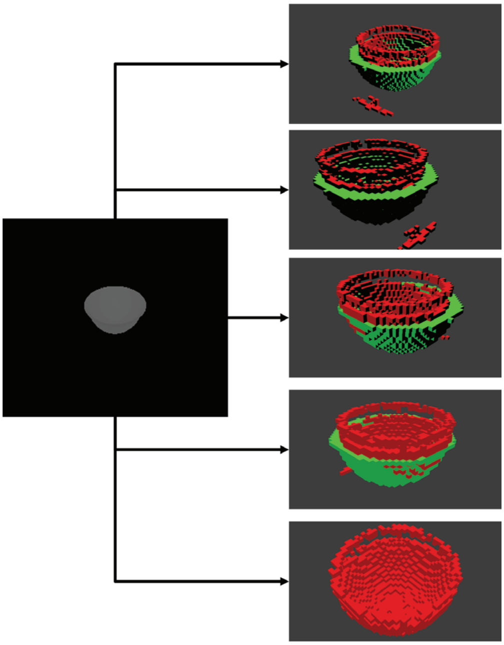
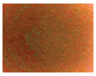
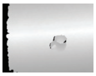
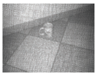


Figure 12. An example of visual comparison ground truth and prediction on the validation set of *Bowl* shape. Green color denotes ground truth, and red color shows prediction.

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	Normalized	Normalized Depth	Voxel Cloud	Mesh
					
					

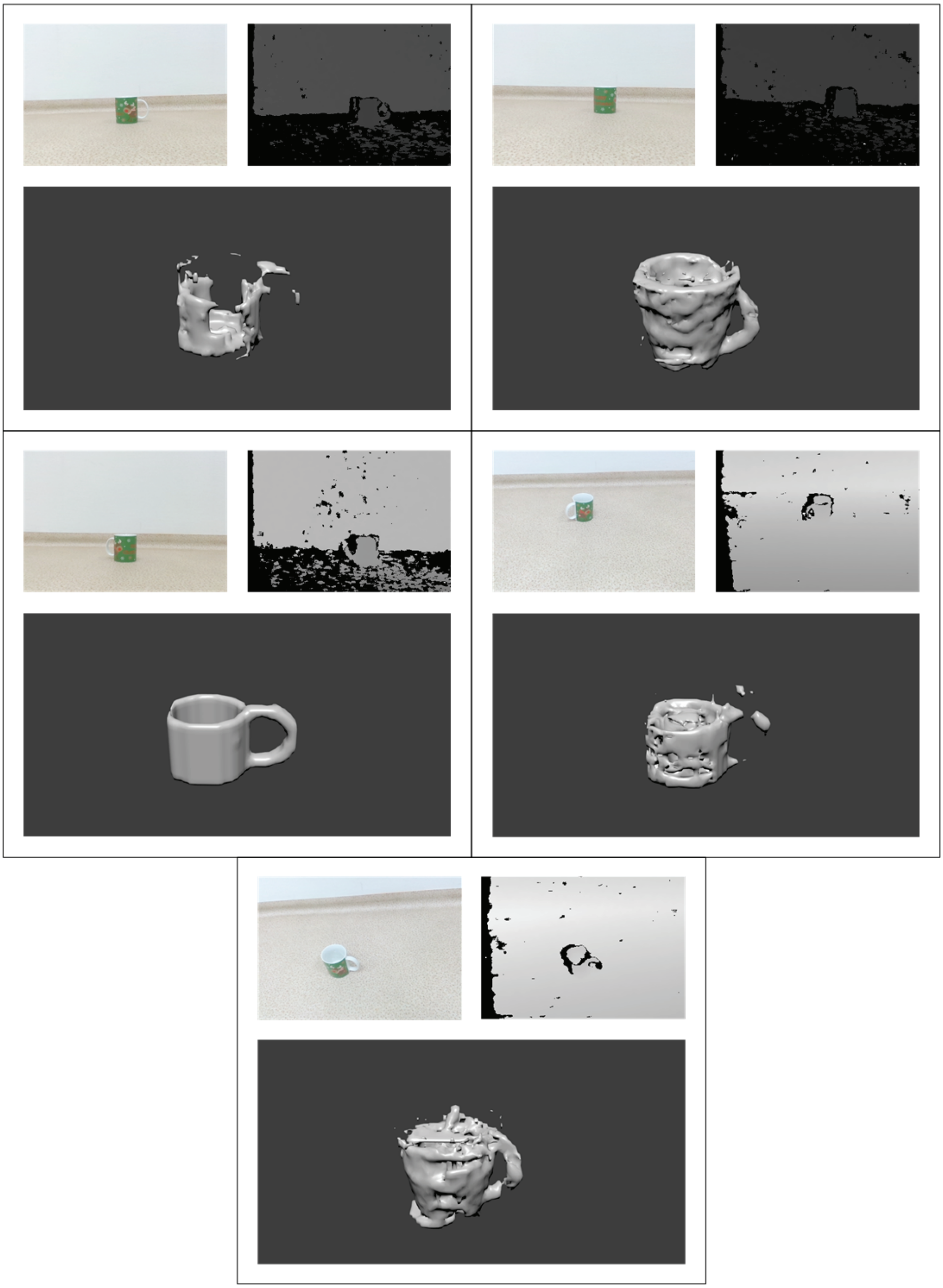


Figure 13. Visual comparison of the same *Mug* object from varying perspectives.

8.1.4. Discussion and Concluding Remarks

8.1.4.1. 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].

8.1.4.2. 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.

8.1.4.3. 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.

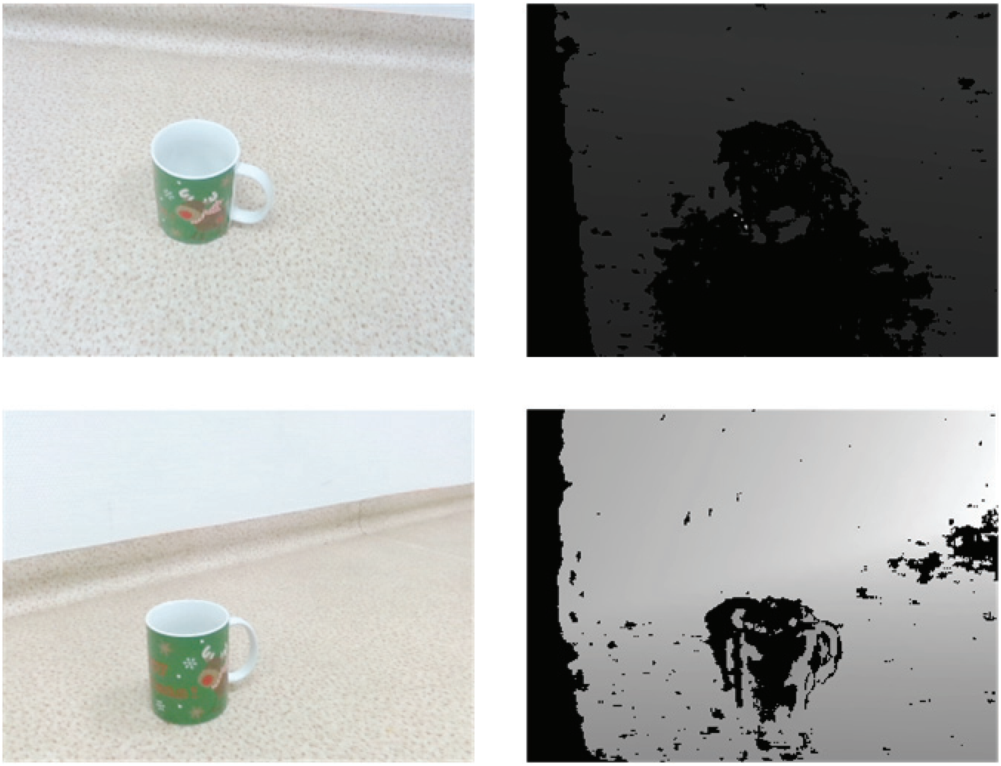


Figure 14. An example of corrupted frames.

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. Bruno Fanini, Alfonsina Pagano, and Daniele Ferdani. "A Novel Immersive VR Game Model for Recontextualization in Virtual Environments: The uVRModel". In: *Multimodal Technologies and Interaction* 2.2 (Apr. 2018), p. 20. DOI: `10.3390/mti2020020`. Online: `https://doi.org/10.3390/mti2020020`.

2. Bo Liao, Jing Li, Zhaojie Ju, and Gaoxiang Ouyang. “Hand Gesture Recognition with Generalized Hough Transform and DC-CNN Using Realsense”. In: *2018 Eighth International Conference on Information Science and Technology (ICIST)*. IEEE, June 2018. DOI: `10.1109/icist.2018.8426125`. Online: `https://doi.org/10.1109/icist.2018.8426125`.
3. Chi Chen, Bisheng Yang, Shuang Song, Mao Tian, Jianping Li, Wenxia Dai, and Lina Fang. “Calibrate Multiple Consumer RGB-D Cameras for Low-Cost and Efficient 3D Indoor Mapping”. In: *Remote Sensing* 10.2 (Feb. 2018), p. 328. DOI: `10.3390/rs10020328`. Online: `https://doi.org/10.3390/rs10020328`.
4. Vacius Jusas, Darius Birvinskas, and Elvar Gahramanov. “Methods and Tools of Digital Triage in Forensic Context: Survey and Future Directions”. In: *Symmetry* 9.4 (Mar. 2017), p. 49. DOI: `10.3390/sym9040049`. Online: `https://doi.org/10.3390/sym9040049`.
5. Abid Haleem and Mohd. Javaid. “3D scanning applications in medical field: A literature-based review”. In: *Clinical Epidemiology and Global Health* (May 2018). DOI: `10.1016/j.cegh.2018.05.006`. Online: `https://doi.org/10.1016/j.cegh.2018.05.006`.
6. Hélène Macher, Tania Landes, and Pierre Grussenmeyer. “From Point Clouds to Building Information Models: 3D Semi-Automatic Reconstruction of Indoors of Existing Buildings”. In: *Applied Sciences* 7.10 (Oct. 2017), p. 1030. DOI: `10.3390/app7101030`. Online: `https://doi.org/10.3390/app7101030`.
7. Li Wang, Ruifeng Li, Hezi Shi, Jingwen Sun, Lijun Zhao, Hock Seah, Chee Quah, and Budianto Tandianus. “Multi-Channel Convolutional Neural Network Based 3D Object Detection for Indoor Robot Environmental Perception”. In: *Sensors* 19.4 (Feb. 2019), p. 893. DOI: `10.3390/s19040893`.
8. Fabio Remondino. “Heritage Recording and 3D Modeling with Photogrammetry and 3D Scanning”. In: *Remote Sensing* 3.6 (May 2011), pp. 1104–1138. DOI: `10.3390/rs3061104`. Online: `https://doi.org/10.3390/rs3061104`.
9. P. M. Chu, Y. Sung, and K. Cho. “Generative Adversarial Network-Based Method for Transforming Single RGB Image Into 3D Point Cloud”. In: *IEEE Access* 7 (2019), pp. 1021–1029. ISSN: 2169-3536. DOI: `10.1109/ACCESS.2018.2886213`.
10. J. Wald, K. Tateno, J. Sturm, N. Navab, and F. Tombari. “Real-Time Fully Incremental Scene Understanding on Mobile Platforms”. In: *IEEE Robotics and Automation Letters* 3.4 (Oct. 2018), pp. 3402–3409. ISSN: 2377-3766. DOI: `10.1109/LRA.2018.2852782`.
11. J. Daudelin and M. Campbell. “An Adaptable, Probabilistic, Next-Best View Algorithm for Reconstruction of Unknown 3-D Objects”. In: *IEEE Robotics and Automation Letters* 2.3 (July 2017), pp. 1540–1547. ISSN: 2377-3766. DOI: `10.1109/LRA.2017.2660769`.
12. Jorge Fuentes-Pacheco, José Ruíz Ascencio, and Juan M. Rendón-Mancha. “Visual simultaneous localization and mapping: a survey”. In: *Artificial Intelligence Review* 43 (2012), pp. 55–81. DOI: `10.1007/s10462-012-9365-8`.
13. Michael Zollhöfer, Patrick Stotko, Andreas Görlitz, Christian Theobalt, Matthias Niessner, Reinhard Klein, and Andreas Kolb. “State of the Art on 3D Reconstruction with RGB-D Cameras”. In: *Comput. Graph. Forum* 37 (2018), pp. 625–652.
14. K. N. Kutulakos and S. M. Seitz. “A theory of shape by space carving”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 1. Sept. 1999, 307–314 vol.1. DOI: `10.1109/ICCV.1999.791235`.
15. Haoqiang Fan, Hao Su, and L. Guibas. “A Point Set Generation Network for 3D Object Reconstruction from a Single Image”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 2463–2471.

16. C. Li, M. Z. Zia, Q. Tran, X. Yu, G. D. Hager, and M. Chandraker. “Deep Supervision with Shape Concepts for Occlusion-Aware 3D Object Parsing”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017, pp. 388–397. DOI: `10.1109/CVPR.2017.49`.
17. Bo Yang, Stefano Rosa, Andrew Markham, Niki Trigoni, and Hongkai Wen. “Dense 3D Object Reconstruction from a Single Depth View”. In: *TPAMI*. 2018.
18. Zhengyou Zhang. “Microsoft Kinect Sensor and Its Effect”. In: *IEEE Multimedia* 19.2 (Feb. 2012), pp. 4–10. DOI: `10.1109/mmul.2012.24`. Online: `https://doi.org/10.1109/mmul.2012.24`.
19. Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. “Intel(R) RealSense(TM) Stereoscopic Depth Cameras”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, July 2017. DOI: `10.1109/cvprw.2017.167`. Online: `https://doi.org/10.1109/cvprw.2017.167`.
20. L. Tang, Z. Yang, and K. Jia. “Canonical Correlation Analysis Regularization: An Effective Deep Multi-View Learning Baseline for RGB-D Object Recognition”. In: *IEEE Transactions on Cognitive and Developmental Systems* (2018), pp. 1–1. ISSN: 2379-8920. DOI: `10.1109/TCDS.2018.2866587`.
21. Le Zhang, Jian Sun, and Qiang Zheng. “3D Point Cloud Recognition Based on a Multi-View Convolutional Neural Network”. In: *Sensors* 18.11 (2018). ISSN: 1424-8220. DOI: `10.3390/s18113681`. Online: `http://www.mdpi.com/1424-8220/18/11/3681`.
22. Guanzhong Tian, Liang Liu, JongHyok Ri, Yong Liu, and Yiran Sun. “ObjectFusion: An object detection and segmentation framework with RGB-D SLAM and convolutional neural networks”. In: *Neurocomputing* (2019). ISSN: 0925-2312. DOI: `https://doi.org/10.1016/j.neucom.2019.01.088`. Online: `http://www.sciencedirect.com/science/article/pii/S0925231219301377`.
23. Hui Zeng, Bin Yang, Xiuqing Wang, Jiwei Liu, and Dongmei Fu. “RGB-D Object Recognition Using Multi-Modal Deep Neural Network and DS Evidence Theory”. In: *Sensors* 19.3 (2019). ISSN: 1424-8220. DOI: `10.3390/s19030529`. Online: `http://www.mdpi.com/1424-8220/19/3/529`.
24. Fabio F. Oliveira, Anderson A. S. Souza, Marcelo A. C. Fernandes, Rafael B. Gomes, and Luiz M. G. Goncalves. “Efficient 3D Objects Recognition Using Multifoveated Point Clouds”. In: *Sensors* 18.7 (2018). ISSN: 1424-8220. DOI: `10.3390/s18072302`. Online: `http://www.mdpi.com/1424-8220/18/7/2302`.
25. Kourosh Khoshelham and Sander Oude Elberink. “Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications”. In: *Sensors* 12.2 (Feb. 2012), pp. 1437–1454. DOI: `10.3390/s120201437`. Online: `https://doi.org/10.3390/s120201437`.
26. M. Carfagni, R. Furferi, L. Governi, M. Servi, F. Uccheddu, and Y. Volpe. “On the Performance of the Intel SR300 Depth Camera: Metrological and Critical Characterization”. In: *IEEE Sensors Journal* 17.14 (July 2017), pp. 4508–4519. ISSN: 1530-437X. DOI: `10.1109/JSEN.2017.2703829`.
27. David Stutz and Andreas Geiger. “Learning 3D Shape Completion Under Weak Supervision”. In: *International Journal of Computer Vision* (Oct. 2018). ISSN: 1573-1405. DOI: `10.1007/s11263-018-1126-y`. Online: `https://doi.org/10.1007/s11263-018-1126-y`.
28. Olivia Wiles and Andrew Zisserman. “Learning to Predict 3D Surfaces of Sculptures from Single and Multiple Views”. In: *International Journal of Computer Vision* (Oct. 2018). ISSN: 1573-1405. DOI: `10.1007/s11263-018-1124-0`. Online: `https://doi.org/10.1007/s11263-018-1124-0`.
29. Y. Cao, C. Shen, and H. T. Shen. “Exploiting Depth From Single Monocular Images for Object Detection and Semantic Segmentation”. In: *IEEE Transactions on Image*

Processing 26.2 (Feb. 2017), pp. 836–846. ISSN: 1057-7149. DOI: `10.1109/TIP.2016.2621673`.

30. K. Hisatomi, M. Kano, K. Ikeya, M. Katayama, T. Mishina, Y. Iwadate, and K. Aizawa. “Depth Estimation Using an Infrared Dot Projector and an Infrared Color Stereo Camera”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.10 (Oct. 2017), pp. 2086–2097. ISSN: 1051-8215. DOI: `10.1109/TCSVT.2016.2555678`.
31. Y. Du, Y. Fu, and L. Wang. “Representation Learning of Temporal Dynamics for Skeleton-Based Action Recognition”. In: *IEEE Transactions on Image Processing* 25.7 (July 2016), pp. 3010–3022. ISSN: 1057-7149. DOI: `10.1109/TIP.2016.2552404`.
32. Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2015).
33. Vinod Nair and Geoffrey E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *ICML*. 2010.
34. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034. DOI: `10.1109/ICCV.2015.123`.
35. M. Bartsch, T. Weiland, and M. Witting. “Generation of 3D isosurfaces by means of the marching cube algorithm”. In: *IEEE Transactions on Magnetics* 32.3 (May 1996), pp. 1469–1472. ISSN: 0018-9464. DOI: `10.1109/20.497526`.
36. Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. “Dual Contouring of Hermite Data”. In: *ACM Trans. Graph.* 21.3 (July 2002), pp. 339–346. ISSN: 0730-0301. DOI: `10.1145/566654.566586`. Online: `http://doi.acm.org/10.1145/566654.566586`.
37. Franz Kainz, Rebecca R. Bogart, and David K. Hess. “The OpenEXR image file format”. In: 2004.
38. Jacopo Pantaleoni. “VoxelPipe”. In: *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics - HPG*. ACM Press, 2011. DOI: `10.1145/2018323.2018339`. Online: `https://doi.org/10.1145/2018323.2018339`.
39. Doug Baldwin and Michael Weber. “Fast Ray-Triangle Intersections by Coordinate Transformation”. In: *Journal of Computer Graphics Techniques (JCGT)* 5.3 (Sept. 2016), pp. 39–49. ISSN: 2331-7418. Online: `http://jcgt.org/published/0005/03/03/`.
40. Angel X. Chang, Thomas A. Funkhouser, Leonidas J Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. “ShapeNet: An Information-Rich 3D Model Repository”. In: *CoRR* abs/1512.03012 (2015).
41. M. J. Marin-Jimenez, A. Zisserman, M. Eichner, and V. Ferrari. “Detecting People Looking at Each Other in Videos”. In: *International Journal of Computer Vision* 106.3 (Sept. 2013), pp. 282–296. DOI: `10.1007/s11263-013-0655-7`. Online: `https://doi.org/10.1007/s11263-013-0655-7`.
42. M. Rutzinger, F. Rottensteiner, and N. Pfeifer. “A Comparison of Evaluation Techniques for Building Extraction From Airborne Laser Scanning”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 2.1 (Mar. 2009), pp. 11–20. DOI: `10.1109/jstars.2009.2012488`. Online: `https://doi.org/10.1109/jstars.2009.2012488`.
43. J. Wang, C. Xu, X. Yang, and J. M. Zurada. “A Novel Pruning Algorithm for Smoothing Feedforward Neural Networks Based on Group Lasso Method”. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.5 (May 2018), pp. 2012–2024. ISSN: 2162-237X. DOI: `10.1109/TNNLS.2017.2748585`.

44. and P. Saratchandran and N. Sundararajan. “A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation”. In: *IEEE Transactions on Neural Networks* 16.1 (Jan. 2005), pp. 57–67. ISSN: 1045-9227. DOI: `10.1109/TNN.2004.836241`.
45. Jasmina Arifovic and Ramazan Gençay. “Using genetic algorithms to select architecture of a feedforward artificial neural network”. In: *Physica A: Statistical Mechanics and its Applications* 289.3 (2001), pp. 574–594. ISSN: 0378-4371. DOI: `https://doi.org/10.1016/S0378-4371(00)00479-9`. Online: `http://www.sciencedirect.com/science/article/pii/S0378437100004799`.
46. Dawid Połap, Karolina Kęsik, Marcin Woźniak, and Robertas Damaševičius. “Parallel Technique for the Metaheuristic Algorithms Using Devoted Local Search and Manipulating the Solutions Space”. In: *Applied Sciences* 8.2 (Feb. 2018), p. 293. DOI: `10.3390/app8020293`. Online: `https://doi.org/10.3390/app8020293`.
47. Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. “Kinectfusion: real-time 3D reconstruction and interaction using a moving depth camera”. In: *In Proc. UIST*. 2011, pp. 559–568.
48. J. Wang, L. Zhang, Q. Guo, and Z. Yi. “Recurrent Neural Networks With Auxiliary Memory Units”. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.5 (May 2018), pp. 1652–1661. ISSN: 2162-237X. DOI: `10.1109/TNNLS.2017.2677968`.
49. J. Hawkins and M. Boden. “The applicability of recurrent neural networks for biological sequence analysis”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2.3 (July 2005), pp. 243–253. ISSN: 1545-5963. DOI: `10.1109/TCBB.2005.44`.
50. Zhong Liu, Changchen Zhao, Xingming Wu, and Weihai Chen. “An Effective 3D Shape Descriptor for Object Recognition with RGB-D Sensors”. In: *Sensors* 17.3 (2017). ISSN: 1424-8220. DOI: `10.3390/s17030451`. Online: `http://www.mdpi.com/1424-8220/17/3/451`.
51. G. J. Hsu, Y. Liu, H. Peng, and P. Wu. “RGB-D-Based Face Reconstruction and Recognition”. In: *IEEE Transactions on Information Forensics and Security* 9.12 (Dec. 2014), pp. 2110–2118. ISSN: 1556-6013. DOI: `10.1109/TIFS.2014.2361028`.

8.2. 3D Object Reconstruction from Imperfect Depth Data Using Extended YOLOv3 Network

Authors 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

Sensors 2020, 20(7), 2025; https://doi.org/10.3390/s20072025

Received: 21 February 2020 / Revised: 29 March 2020 / Accepted: 2 April 2020 / Published: 3 April 2020

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

8.2.1. 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.

8.2.2. Materials and Methods

8.2.2.1. 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.

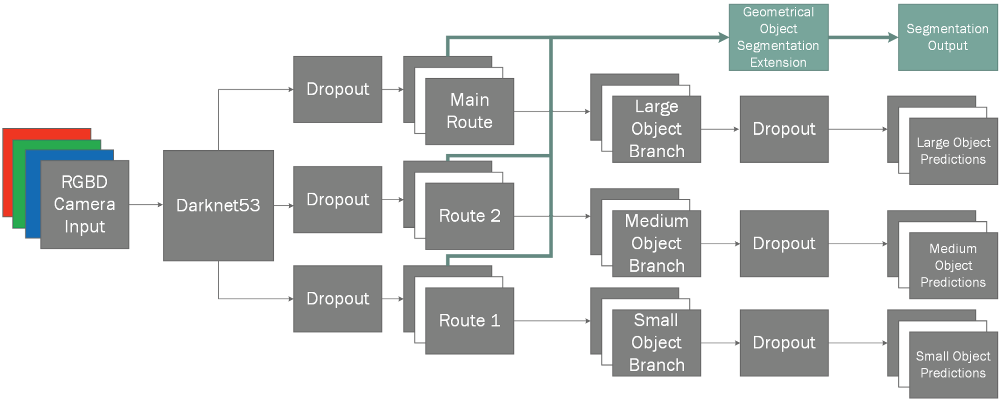


Figure 1. Our extended *YOLOv3* capable of extracting geometric object segmentation along with object bounding boxes.

8.2.2.2. 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, 45, 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).

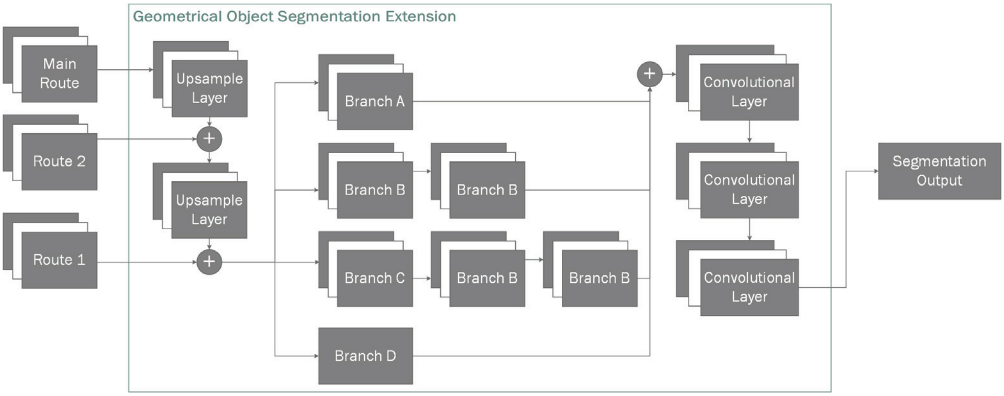


Figure 2. Our proposed geometrical object segmentation extension.

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**
Main route	Transposed Conv	1024	$2 \times 2 \times 2$	20×20
	Concatenate	-	-	20×20
	Convolution	256	1×1	20×20
Route 2	Transposed Conv	256	$2 \times 2 \times 2$	40×40
	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
Branch C	Convolution	32	1×1	160×120
	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

```
procedure GET_SEED($box, mask, \epsilon$)    ▷ Seeds initial values.
    $cx, cy \leftarrow box$    ▷ Get center for $box$.
    $seed \leftarrow \varnothing$
    $seed \leftarrow find\_closest\_max(box, mask)$    ▷ Find closest max pixel within bounds.
    if $seed \neq \varnothing \wedge seed_{value} \geq \epsilon$ then
        $seed_{id} \leftarrow box_{id}$    ▷ Set seed $id$ to box $id$
        return $seed$    ▷ Return seed if value greater than $\epsilon$
    end if
    return $\varnothing$    ▷ No valid seed was found.
end procedure
procedure FILL_NEIGHBOURS($seed, \theta$)    ▷ Recursively fill free neightbours with same or lower values.
    for each $n \in seed_{neightbours}$ do    ▷ For every neighboring mask pixel.
        if $n_{id} = \varnothing \wedge n_{value} \leq seed_{value} \wedge n_{value} > \theta$ then
            $n_{id} \leftarrow seed_{id}$    ▷ Set neighbor to same $id$ as $seed$.
            $FILL\_NEIGHBOURS(n)$    ▷ Call recursively.
        end if
    end for
end procedure
$bounding\_boxes \leftarrow sort\_confidence(bounding\_boxes)$    ▷ Sort bounding boxes by confidence.
for each $box \in bounding\_boxes$ do    ▷ For each bounding box $b$
    $seed \leftarrow GET\_SEED(box, \epsilon)$
```

```
    if $seed \neq \varnothing$ then
        $FILL\_NEIGHBOURS(seed, \theta)$
    end if
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 $lr_{min} = 1e^{-6}$, $lr_{max} = 1e^{-4}$.

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

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.

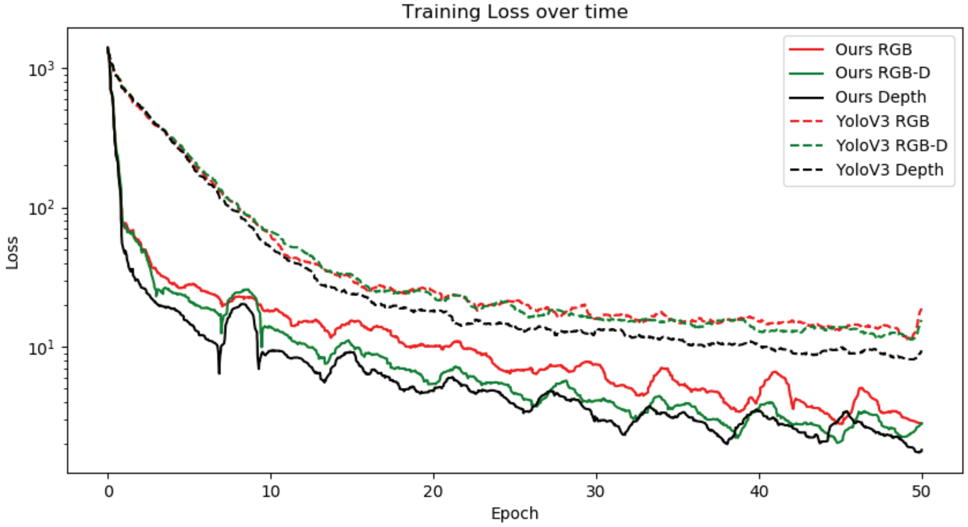


Figure 3. Training loss comparison between baseline *YOLOv3* and our modified version when using RGB, RGB-D and depth data as training. Due to the loss function being inherently noisy for each of the mini-batches, we have used Savitzky-Golay [52] digital filter to perform the smoothing of the overall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

8.2.2.3. 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.

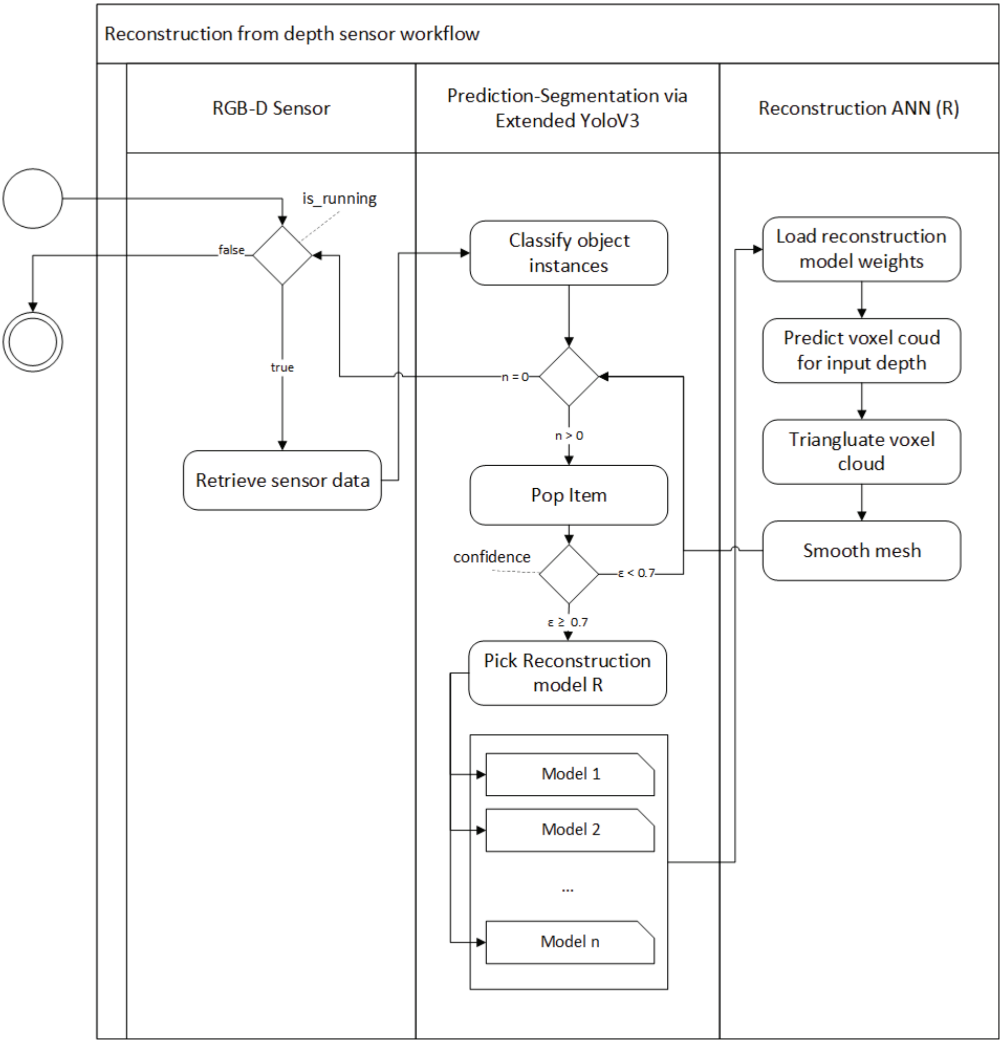

Reconstruction from depth sensor workflow
RGB-D Sensor
Prediction-Segmentation via Extended YoloV3
Reconstruction ANN (R)
is_running
false
true
Retrieve sensor data
Classify object instances
n = 0
n > 0
Pop Item
confidence
ε < 0.7
ε ≥ 0.7
Pick Reconstruction model R
Model 1
Model 2
...
Model n
Load reconstruction model weights
Predict voxel coud for input depth
Triangluate voxel cloud
Smooth mesh

Figure 4. Workflow of object reconstruction from sensor data.

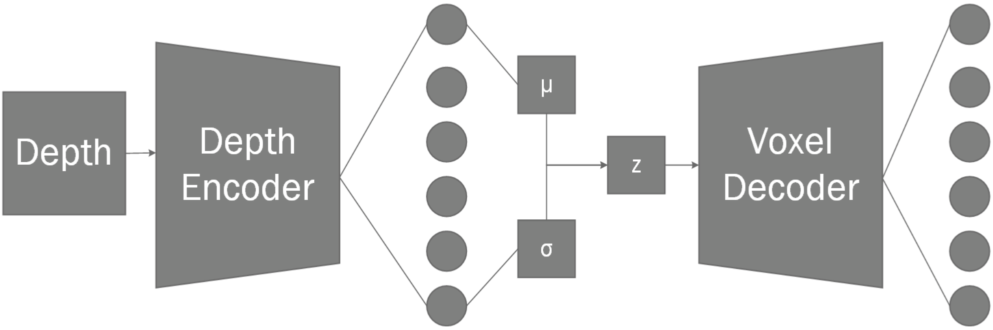


Figure 5. Diagram of a single object reconstruction network architecture branch. For the given depth frame, the depth encoder creates a bottleneck, which is then directly connected to VAE node, the resulting sampler is connected into voxel decoder. The voxel decoder layer outputs a $32 \times 32 \times 32 \times 2$ 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

	Type	**Filters**	**Size**	**Output**
	Input	-	-	320×240
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)	-	$4 \times 4 \times 4$
	Conv 3D	16	$1 \times 1 \times 1$	$4 \times 4 \times 4$
	Add	-	-	$4 \times 4 \times 4$
	Transposed Convolution 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)	-	$8 \times 8 \times 8$
	Conv 3D	16	$1 \times 1 \times 1$	$8 \times 8 \times 8$
	Add	-	-	$8 \times 8 \times 8$
	Transposed Convolution 3D	32	$3 \times 3 \times 3 \times 2$	$16 \times 16 \times 16$
	Inception 3D	(8, 4)	-	$16 \times 16 \times 16$
	Conv 3D	16	$1 \times 1 \times 1$	$16 \times 16 \times 16$
	Add	-	-	$16 \times 16 \times 16$
	Transposed Convolution 3D	4	$5 \times 5 \times 5 \times 2$	$32 \times 32 \times 32$
	Conv 3D (Softmax)	2	$3 \times 3 \times 3$	$32 \times 32 \times 32$

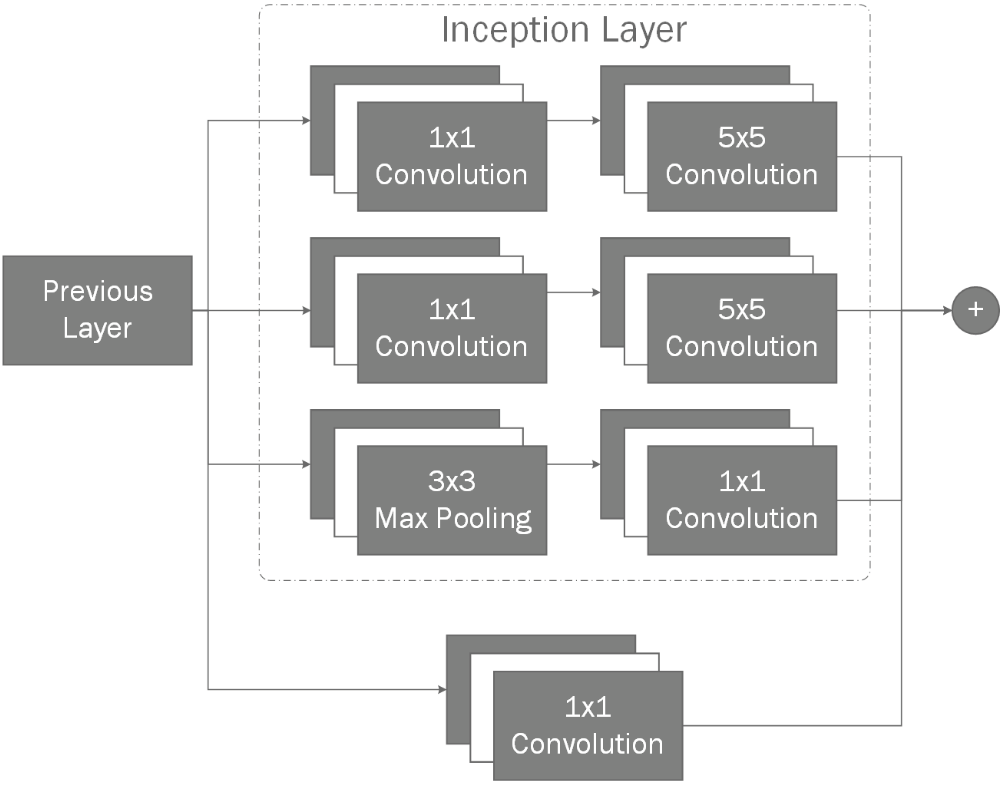


Figure 6. An example of the inception layer. An input layer is connected to three branches in parallel. If multiple inception layers are used inception layers are connected sequentially. Final inception layer outputs and 1 × 1 convolution are then connected using addition. The result is then used for subsequent layers.

8.2.2.4. 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.

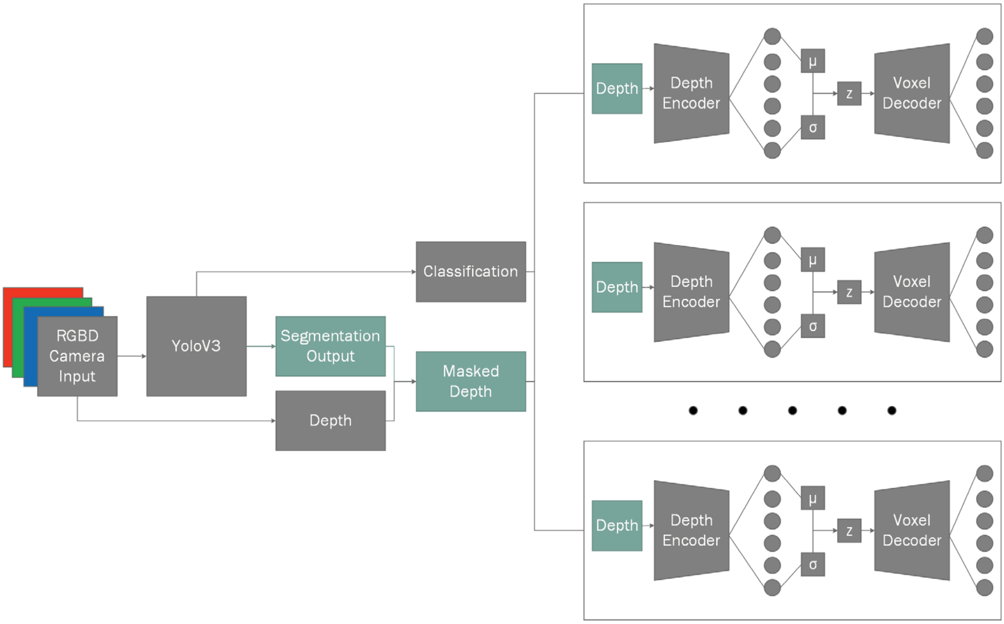


Figure 7. Full view of the proposed network model that extends the YOLOv3 network.

8.2.2.5. 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.

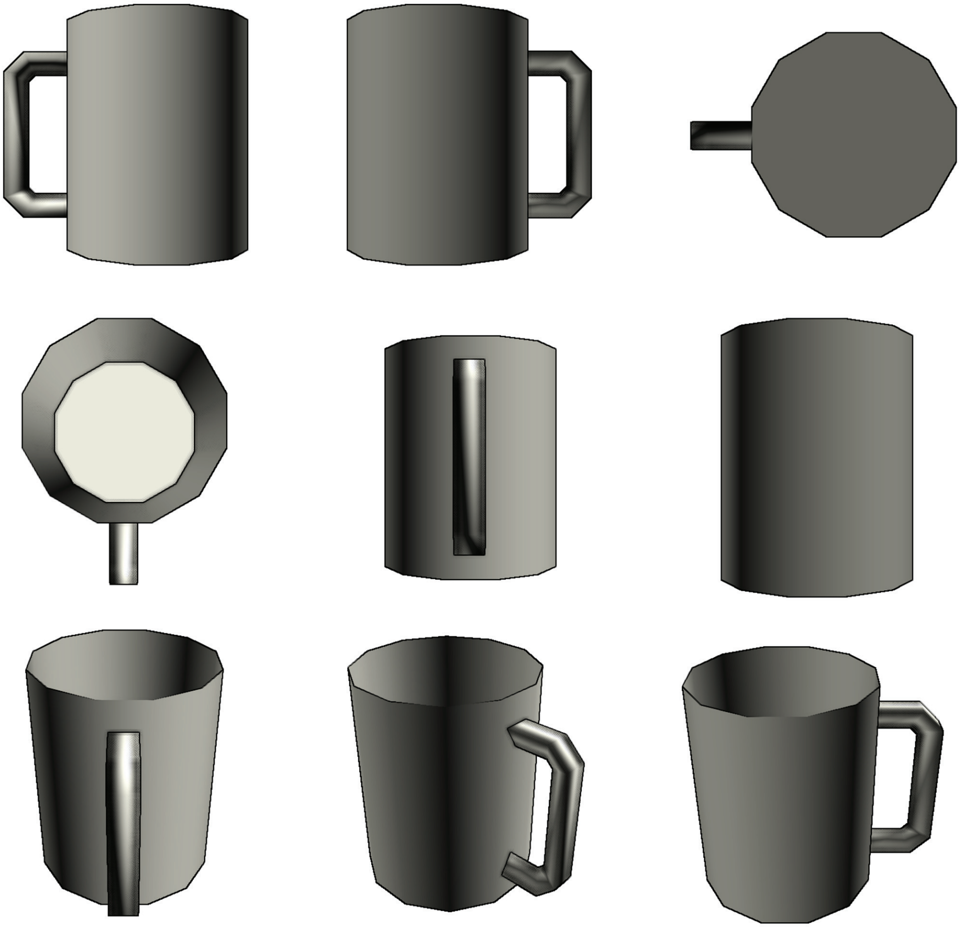


Figure 8. A coffee cup model from the *ShapeNetCore* dataset.

8.2.2.5.1. 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

```
procedure Extents($G$)                              ▷ Calculates extents for geometry $G$
    $min_x, min_y, min_z \leftarrow Infinity$                      ▷ Initialize $min$ vector
    $max_x, max_y, max_z \leftarrow -Infinity$                    ▷ Initialize $max$ vector
    for each $v \in G$ do                                      ▷ For each vertex $v$
        $min_x \leftarrow min(v_x, min_x)$
        $min_y \leftarrow min(v_y, min_y)$
        $min_z \leftarrow min(v_z, min_z)$
        $max_x \leftarrow max(v_x, max_x)$
        $max_y \leftarrow max(v_y, max_y)$
        $max_z \leftarrow max(v_z, max_z)$
    end for
    return $min, max$
end procedure
$min, max \leftarrow EXTENTS(G)$
$bounds \leftarrow max - min$
$max\_bound \leftarrow 1/max(bounds_x, bounds_y, bounds_z)$
for each $v \in G$ do                                          ▷ For each vertex $v$
    $v_x \leftarrow v_x/max\_bound$
    $v_y \leftarrow v_y/max\_bound$
    $v_z \leftarrow (v_z - min_z)/max\_bound$  ▷ Offset the vertex on up axis before
    normalizing bounds
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

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

8.2.2.5.2. 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.

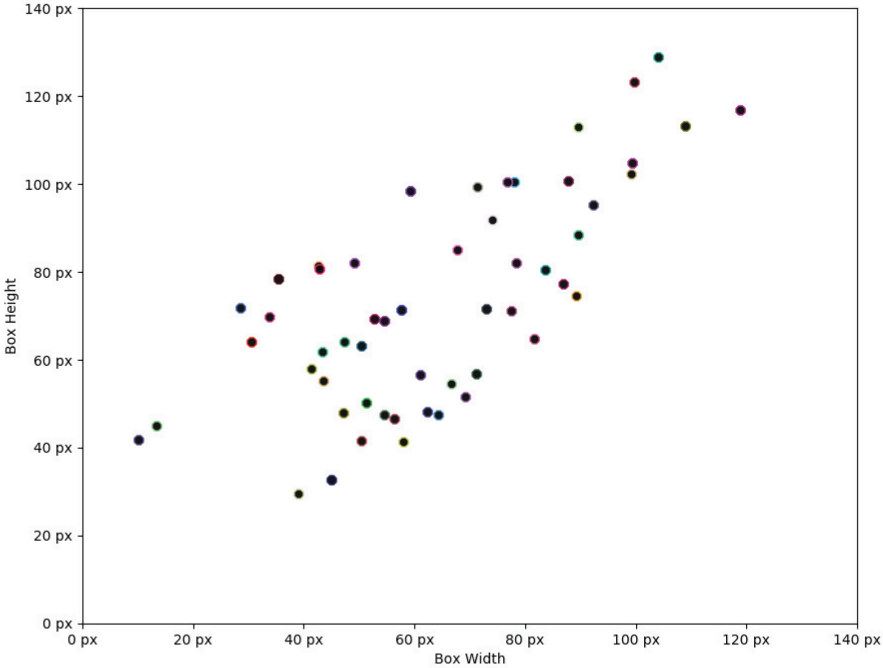


Figure 9. Each individual point denotes the mean object bounding box scale for each class type.

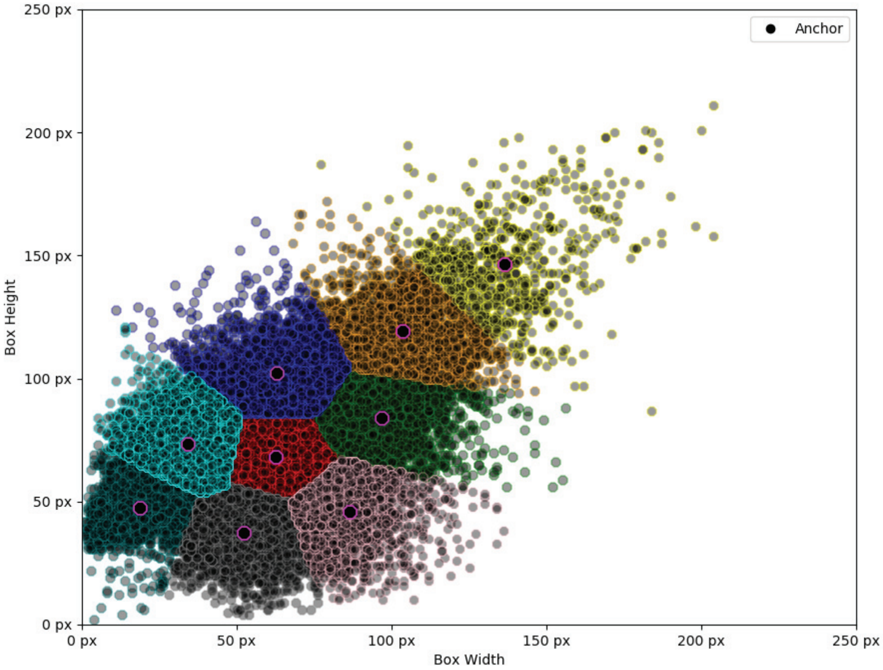


Figure 10. Selected anchors using *K-Means* clustering algorithm. Different colors denote distinct anchor groups responsible for detecting objects in the spread.

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 8.2.2.2. 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.

8.2.2.5.3. 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].

8.2.2.6. 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.

8.2.3. Results

8.2.3.1. 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.

8.2.3.2. Quantitative Results

8.2.3.2.1. 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.

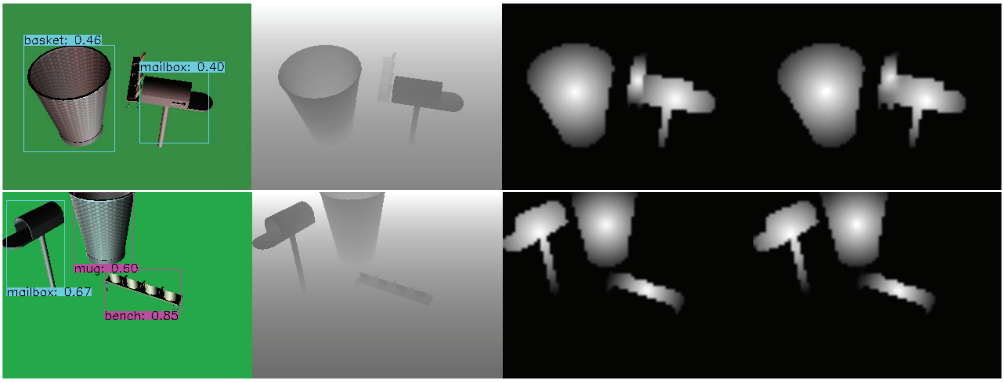


Figure 11. Prediction made with our extended *YOLOv3* network. Left to right: (1) Input color image with predicted object instances; (2) Input depth frame; (3) Upscaled to 320 × 240 ground truth mask; (4) Predicted mask upscaled to 320 × 240. Same object is being treated as two distinct classes due to lack of cues for the artificial neural network of what the specific object may be due to scenes being generated randomly.

8.2.3.2.2. 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.

8.2.3.3. Reconstruction Results

8.2.3.3.1. 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*.

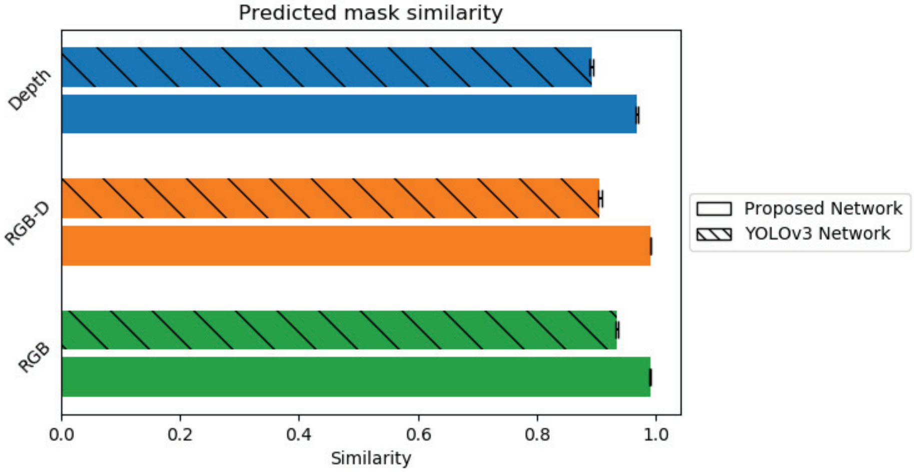


Figure 12. Similarity of created mask to the mask of ground truth. The hashed bar denotes the similarity of masks predicted by the *YOLOv3* network, the solid bar denotes the similarity of masks predicted for Depth, RGB-D and RGB frames by the network model proposed in this paper.

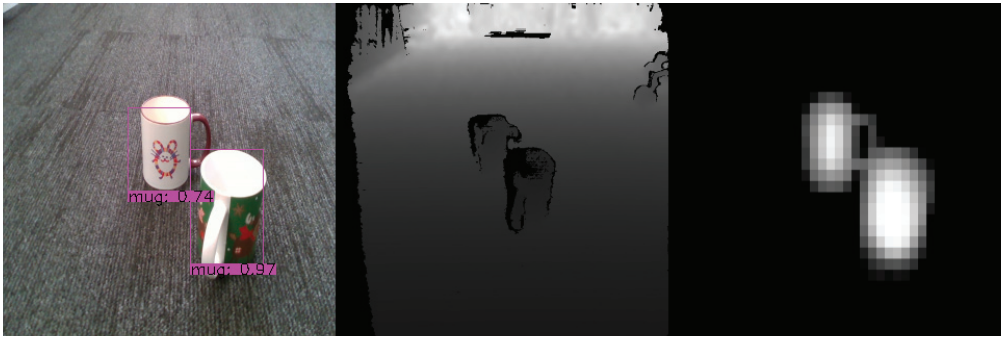


Figure 13. An example of real world object classification using the proposed network model: segmented and classified RGB frame (**left**), depth frame (**middle**), and predicted depth mask (**right**).

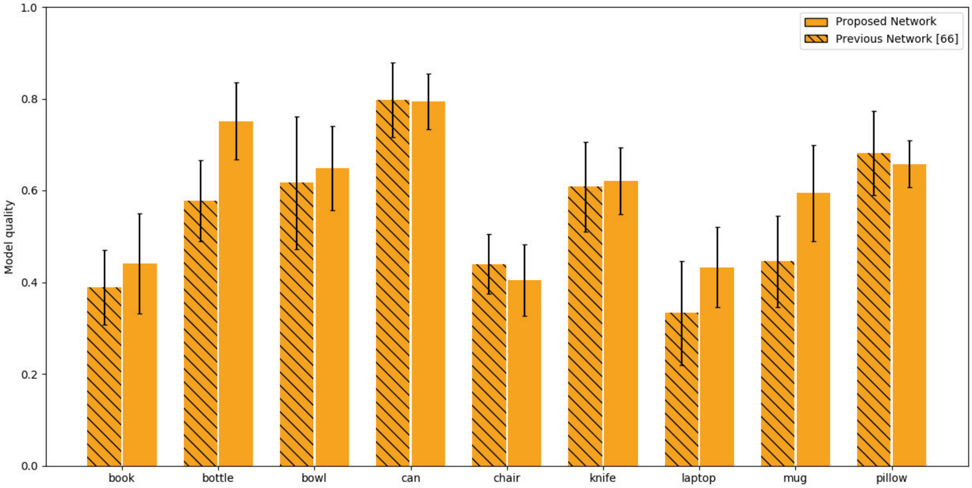


Figure 14. Comparison between the predicted object shape and ground truth using the IoU metric for different objects in the training set. The hashed bars denote the results achieved using the network proposed in [66]. The solid bars denote the results for the proposed network.

8.2.3.3.2. 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.

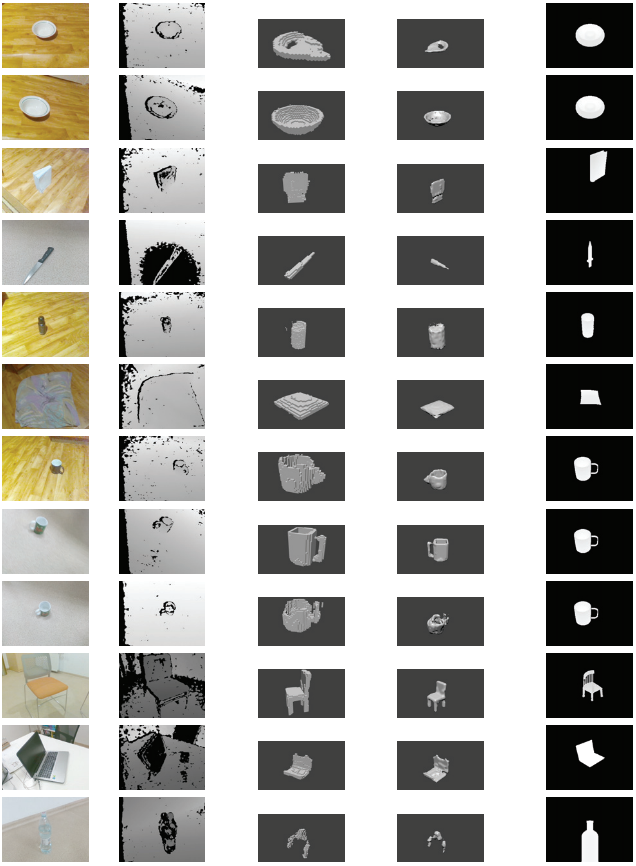
RGB

Depth

Voxel Cloud

Mesh

Training Data



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.

8.2.3.3.3. 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.

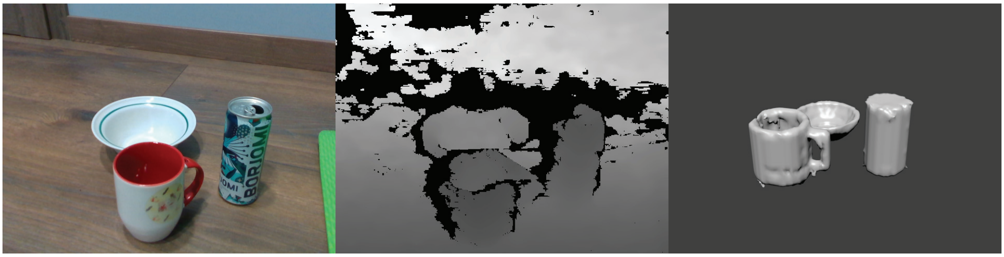


Figure 15. An example of reconstruction of multiple objects in the scene: segmented and classified RGB frame (**left**), depth frame (**middle**), and predicted depth mask (**right**).

8.2.4. Discussion and Concluding Remarks

8.2.4.1. 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, 75, 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 [54]	7 M	16.04 G	40
ResNet152 [80]	60 M	11.3 G	230
VGC16 [80]	138 M	154.7 G	512.24
NIN [81]	7.6 M	11.06 G	29
SimpleNet [82]	5.4 M	652 M	20

8.2.4.2. 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.

Funding This research received no external funding

References

1. Malamati Bitzidou, Dimitrios Chrysostomou, and Antonios Gasteratos. "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* (2013), pp. 526–533. doi: `10.1007/978-3-642-40352-1_66`.
2. Bruno Fanini, Alfonsina Pagano, and Daniele Ferdani. "A Novel Immersive VR Game Model for Recontextualization in Virtual Environments: The uVRModel". In: *Multimodal Technologies and Interaction* 2.2 (Apr. 2018), p. 20. doi: `10.3390/mti2020020`. online: `https://doi.org/10.3390/mti2020020`.
3. Åsa Fast-Berglund, Liang Gong, and Dan Li. "Testing and validating Extended Reality (xR) technologies in manufacturing". In: *Procedia Manufacturing* 25 (2018), pp. 31–38. doi: `10.1016/j.promfg.2018.06.054`. online: `https://doi.org/10.1016/j.promfg.2018.06.054`.
4. Bo Liao, Jing Li, Zhaojie Ju, and Gaoxiang Ouyang. "Hand Gesture Recognition with Generalized Hough Transform and DC-CNN Using Realsense". In: *2018 Eighth International Conference on Information Science and Technology (ICIST)*. IEEE, June 2018. doi: `10.1109/icist.2018.8426125`. online: `https://doi.org/10.1109/icist.2018.8426125`.
5. Aurelijus Vaitkevičius, Mantas Taroza, Tomas Blažauskas, Robertas Damaševičius, Rytis Maskeliūnas, and Marcin Woźniak. "Recognition of American Sign Language Gestures in a Virtual Reality Using Leap Motion". In: *Applied Sciences* 9.3 (Jan. 2019), p. 445. doi: `10.3390/app9030445`. online: `https://doi.org/10.3390/app9030445`.
6. Jingtian Zhang, Hubert P. H. Shum, Kevin McCay, and Edmond S. L. Ho. "Prior-less 3D Human Shape Reconstruction with an Earth Mover's Distance Informed CNN". In: *Motion, Interaction and Games on - MIG19*. ACM Press, 2019. doi: `10.1145/3359566.3364694`.
7. Chi Chen, Bisheng Yang, Shuang Song, Mao Tian, Jianping Li, Wenxia Dai, and Lina Fang. "Calibrate Multiple Consumer RGB-D Cameras for Low-Cost and Efficient 3D Indoor Mapping". In: *Remote Sensing* 10.2 (Feb. 2018), p. 328. doi: `10.3390/rs10020328`. online: `https://doi.org/10.3390/rs10020328`.
8. Dawid Połap, Karolina Kęsik, Kamil Książek, and Marcin Woźniak. "Obstacle Detection as a Safety Alert in Augmented Reality Models by the Use of Deep Learning Techniques". In: *Sensors* 17.12 (Dec. 2017), p. 2803. doi: `10.3390/s17122803`. online: `https://doi.org/10.3390/s17122803`.
9. Vacius Jusas, Darius Birvinskas, and Elvar Gahramanov. "Methods and Tools of Digital Triage in Forensic Context: Survey and Future Directions". In: *Symmetry* 9.4 (Mar. 2017), p. 49. doi: `10.3390/sym9040049`. online: `https://doi.org/10.3390/sym9040049`.

10. Li Wang, Ruifeng Li, Hezi Shi, Jingwen Sun, Lijun Zhao, Hock Seah, Chee Quah, and Budianto Tandianus. “Multi-Channel Convolutional Neural Network Based 3D Object Detection for Indoor Robot Environmental Perception”. In: *Sensors* 19.4 (Feb. 2019), p. 893. doi: `10.3390/s19040893`.
11. Gongjin Lan, Ziyun Luo, and Qi Hao. “Development of a virtual reality teleconference system using distributed depth sensors”. In: *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. Oct. 2016, pp. 975–978. doi: `10.1109/CompComm.2016.7924850`.
12. J. Wald, K. Tateno, J. Sturm, N. Navab, and F. Tombari. “Real-Time Fully Incremental Scene Understanding on Mobile Platforms”. In: *IEEE Robotics and Automation Letters* 3.4 (Oct. 2018), pp. 3402–3409. issn: 2377-3766. doi: `10.1109/LRA.2018.2852782`.
13. J. Daudelin and M. Campbell. “An Adaptable, Probabilistic, Next-Best View Algorithm for Reconstruction of Unknown 3-D Objects”. In: *IEEE Robotics and Automation Letters* 2.3 (July 2017), pp. 1540–1547. issn: 2377-3766. doi: `10.1109/LRA.2017.2660769`.
14. Jorge Fuentes-Pacheco, José Ruíz Ascencio, and Juan M. Rendón-Mancha. “Visual simultaneous localization and mapping: a survey”. In: *Artificial Intelligence Review* 43 (2012), pp. 55–81. doi: `10.1007/s10462-012-9365-8`.
15. Michael Zollhöfer, Patrick Stotko, Andreas Görlitz, Christian Theobalt, Matthias Niessner, Reinhard Klein, and Andreas Kolb. “State of the Art on 3D Reconstruction with RGB-D Cameras”. In: *Comput. Graph. Forum* 37 (2018), pp. 625–652.
16. K. N. Kutulakos and S. M. Seitz. “A theory of shape by space carving”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 1. Sept. 1999, 307–314 vol.1. doi: `10.1109/ICCV.1999.791235`.
17. C. Li, M. Z. Zia, Q. Tran, X. Yu, G. D. Hager, and M. Chandraker. “Deep Supervision with Shape Concepts for Occlusion-Aware 3D Object Parsing”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017, pp. 388–397. doi: `10.1109/CVPR.2017.49`.
18. Bo Yang, Stefano Rosa, Andrew Markham, Niki Trigoni, and Hongkai Wen. “Dense 3D Object Reconstruction from a Single Depth View”. In: *TPAMI*. 2018.
19. Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. “Deep Metric Learning via Lifted Structured Feature Embedding”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
20. Angel X. Chang, Thomas A. Funkhouser, Leonidas J Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. “ShapeNet: An Information-Rich 3D Model Repository”. In: *CoRR* abs/1512.03012 (2015).
21. Christopher B. Choy, Danfei Xu, Junyoung Gwak, Kevin Chen, and Silvio Savarese. “3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction”. In: *Computer Vision – ECCV 2016 Lecture Notes in Computer Science* (2016), pp. 628–644. doi: `10.1007/978-3-319-46484-8_38`.
22. Tingsong Ma, Ping Kuang, and Wenhong Tian. “An improved recurrent neural networks for 3d object reconstruction”. In: *Applied Intelligence* (2019). doi: `10.1007/s10489-019-01523-3`.
23. Amol Dhondse, Siddhivinayak Kulkarni, Kunal Khadilkar, Indrajeet Kane, Sumit Chavan, and Rahul Barhate. “Generative Adversarial Networks as an Advancement in 2D to 3D Reconstruction Techniques”. In: *Data Management, Analytics and Innovation Advances in Intelligent Systems and Computing* (2019), pp. 343–364. doi: `10.1007/978-981-13-9364-8_25`.
24. C. G. Turhan and H. S. Bilge. “Fused voxel autoencoder for single image to 3D object reconstruction”. In: *Electronics Letters* 56.3 (2020), pp. 134–137.

25. Renato Hermoza and Ivan Sipiran. “3D Reconstruction of Incomplete Archaeological Objects Using a Generative Adversarial Network”. In: *Proceedings of Computer Graphics International 2018 on - CGI 2018* (2018). doi: `10.1145/3208159.3208173`.
26. Ahmed F. Elaksher. “3D object reconstruction from multiple views using neural networks”. In: *Applied Geomatics* 5.3 (2013), pp. 193–201. doi: `10.1007/s12518-013-0110-z`.
27. Juan Espinal, Manuel Ornelas, Hector J. Puga, Juan M. Carpio, and J. Apolinar Munoz. “3D Object Reconstruction Using Structured Light and Neural Networks”. In: *2010 IEEE Electronics, Robotics and Automotive Mechanics Conference* (2010). doi: `10.1109/cerma.2010.19`.
28. D. Kappler, J. Bohg, and S. Schaal. “Leveraging big data for grasp planning”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. May 2015, pp. 4304–4311. doi: `10.1109/ICRA.2015.7139793`.
29. P. Rivera, E. V. Añazco, M. -. Choi, and T. -. Kim. “Trilateral convolutional neural network for 3D shape reconstruction of objects from a single depth view”. In: *IET Image Processing* 13.13 (2019), pp. 2457–2466.
30. Haoqiang Fan, Hao Su, and Leonidas Guibas. “A Point Set Generation Network for 3D Object Reconstruction from a Single Image”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017). doi: `10.1109/cvpr.2017.264`.
31. Christian Hane, Shubham Tulsiani, and Jitendra Malik. “Hierarchical Surface Prediction for 3D Object Reconstruction”. In: *2017 International Conference on 3D Vision (3DV)* (2017). doi: `10.1109/3dv.2017.00054`.
32. Li Jiang, Shaoshuai Shi, Xiaojuan Qi, and Jiaya Jia. “GAL: Geometric Adversarial Loss for Single-View 3D-Object Reconstruction”. In: *Computer Vision – ECCV 2018 Lecture Notes in Computer Science* (2018), pp. 820–834. doi: `10.1007/978-3-030-01237-3_49`.
33. Y. Zhang, Z. Liu, T. Liu, B. Peng, and X. Li. “RealPoint3D: An Efficient Generation Network for 3D Object Reconstruction from a Single Image”. In: *IEEE Access* 7 (2019), pp. 57539–57549.
34. Y. Zhang, K. Huo, Z. Liu, Y. Zang, Y. Liu, X. Li, Q. Zhang, and C. Wang. “PGNet: A Part-based Generative Network for 3D object reconstruction”. In: *Knowledge-Based Systems* (2020).
35. Bryson R. Payne, James F. Lay, and Markus A. Hitz. “Automatic 3D object reconstruction from a single image”. In: *Proceedings of the 2014 ACM Southeast Regional Conference on - ACM SE 14* (2014). doi: `10.1145/2638404.2638495`.
36. D. Li, T. Shao, H. Wu, and K. Zhou. “Shape Completion from a Single RGBD Image”. In: *IEEE Transactions on Visualization and Computer Graphics* 23.7 (July 2017), pp. 1809–1822. issn: 2160-9306. doi: `10.1109/TVCG.2016.2553102`.
37. Zhengyou Zhang. “Microsoft Kinect Sensor and Its Effect”. In: *IEEE Multimedia* 19.2 (Feb. 2012), pp. 4–10. doi: `10.1109/mmul.2012.24`. online: `https://doi.org/10.1109/mmul.2012.24`.
38. Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. “Intel(R) RealSense(TM) Stereoscopic Depth Cameras”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, July 2017. doi: `10.1109/cvprw.2017.167`. online: `https://doi.org/10.1109/cvprw.2017.167`.
39. C. Zhang. “CuFusion2: Accurate and Denoised Volumetric 3D Object Reconstruction Using Depth Cameras”. In: *IEEE Access* 7 (2019), pp. 49882–49893.
40. Kourosh Khoshelham and Sander Oude Elberink. “Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications”. In: *Sensors* 12.2 (Feb. 2012), pp. 1437–1454. doi: `10.3390/s120201437`. online: `https://doi.org/10.3390/s120201437`.

41. M. Carfagni, R. Furferi, L. Governi, M. Servi, F. Uccheddu, and Y. Volpe. “On the Performance of the Intel SR300 Depth Camera: Metrological and Critical Characterization”. In: *IEEE Sensors Journal* 17.14 (July 2017), pp. 4508–4519. issn: 1530-437X. doi: `10.1109/JSEN.2017.2703829`.
42. K. Hisatomi, M. Kano, K. Ikeya, M. Katayama, T. Mishina, Y. Iwadate, and K. Aizawa. “Depth Estimation Using an Infrared Dot Projector and an Infrared Color Stereo Camera”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.10 (Oct. 2017), pp. 2086–2097. issn: 1051-8215. doi: `10.1109/TCSVT.2016.2555678`.
43. Y. Du, Y. Fu, and L. Wang. “Representation Learning of Temporal Dynamics for Skeleton-Based Action Recognition”. In: *IEEE Transactions on Image Processing* 25.7 (July 2016), pp. 3010–3022. issn: 1057-7149. doi: `10.1109/TIP.2016.2552404`.
44. Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. “Focal Loss for Dense Object Detection”. In: *CoRR* abs/1708.02002 (2017). online: `http://arxiv.org/abs/1708.02002`.
45. Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C. Berg. “DSSD : Deconvolutional Single Shot Detector”. In: *CoRR* abs/1701.06659 (2017). online: `http://arxiv.org/abs/1701.06659`.
46. Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *CoRR* abs/1506.01497 (2015). arXiv: `1506.01497`. online: `http://arxiv.org/abs/1506.01497`.
47. Joseph Redmon and Ali Farhadi. “YOLOv3: An Incremental Improvement”. In: *CoRR* abs/1804.02767 (2018). online: `http://arxiv.org/abs/1804.02767`.
48. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”. In: *CoRR* abs/1704.04861 (2017). online: `http://arxiv.org/abs/1704.04861`.
49. Mehmet Saygın Seyfioğlu, Ahmet Murat Özbayoğlu, and Sevgi Zubeyde Gürbüz. “Deep convolutional autoencoder for radar-based classification of similar aided and unaided human activities”. In: *IEEE Transactions on Aerospace and Electronic Systems* 54.4 (2018), pp. 1709–1723. doi: `10.1109/TAES.2018.2799758`.
50. S. J. Pan and Q. Yang. “A Survey on Transfer Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (Oct. 2010), pp. 1345–1359. issn: 2326-3865. doi: `10.1109/TKDE.2009.191`.
51. Ilya Loshchilov and Frank Hutter. “SGDR: Stochastic Gradient Descent with Restarts”. In: *CoRR* abs/1608.03983 (2016). online: `http://arxiv.org/abs/1608.03983`.
52. Abraham. Savitzky and M. J. E. Golay. “Smoothing and Differentiation of Data by Simplified Least Squares Procedures.” In: *Analytical Chemistry* 36.8 (1964), pp. 1627–1639. doi: `10.1021/ac60214a047`. eprint: `https://doi.org/10.1021/ac60214a047`. online: `https://doi.org/10.1021/ac60214a047`.
53. Pushparaja Murugan and Shanmugasundaram Durairaj. “Regularization and Optimization strategies in Deep Convolutional Neural Network”. In: *CoRR* abs/1712.04711 (2017). online: `http://arxiv.org/abs/1712.04711`.
54. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. “Going Deeper with Convolutions”. In: *CoRR* abs/1409.4842 (2014). online: `http://arxiv.org/abs/1409.4842`.
55. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). online: `http://arxiv.org/abs/1512.03385`.

56. Yeongbin Kim, Joongchol Shin, Hasil Park, and Joonki Paik. “Real-Time Visual Tracking with Variational Structure Attention Network”. In: *Sensors* 19.22 (2019). issn: 1424-8220. doi: `10.3390/s19224904`. online: `https://www.mdpi.com/1424-8220/19/22/4904`.
57. Vinod Nair and Geoffrey E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *ICML*. 2010.
58. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. “Microsoft COCO: Common Objects in Context”. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars. Cham: Springer International Publishing, 2014, pp. 740–755. isbn: 978-3-319-10602-1.
59. Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. “The Pascal Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338. issn: 1573-1405. doi: `10.1007/s11263-009-0275-4`. online: `https://doi.org/10.1007/s11263-009-0275-4`.
60. Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. “ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes”. In: *CoRR* abs/1702.04405 (2017). arXiv: `1702.04405`. online: `http://arxiv.org/abs/1702.04405`.
61. Steffen Flaischlen and Gregor D. Wehinger. “Synthetic Packed-Bed Generation for CFD Simulations: Blender vs. STAR-CCM+”. In: *ChemEngineering* 3.2 (2019). issn: 2305-7084. doi: `10.3390/chemengineering3020052`. online: `https://www.mdpi.com/2305-7084/3/2/52`.
62. Franz Kainz, Rebecca R. Bogart, and David K. Hess. “The OpenEXR image file format”. In: 2004.
63. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. “SSD: Single Shot MultiBox Detector”. In: *Lecture Notes in Computer Science* (2016), pp. 21–37. issn: 1611-3349. doi: `10.1007/978-3-319-46448-0_2`. online: `http://dx.doi.org/10.1007/978-3-319-46448-0_2`.
64. Jacopo Pantaleoni. “VoxelPipe”. In: *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics - HPG*. ACM Press, 2011. doi: `10.1145/2018323.2018339`. online: `https://doi.org/10.1145/2018323.2018339`.
65. Doug Baldwin and Michael Weber. “Fast Ray-Triangle Intersections by Coordinate Transformation”. In: *Journal of Computer Graphics Techniques (JCGT)* 5.3 (Sept. 2016), pp. 39–49. issn: 2331-7418. online: `http://jcgt.org/published/0005/03/03/`.
66. Audrius Kulikajevas, Rytis Maskeliūnas, Robertas Damaševičius, and Sanjay Misra. “Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from ShapeNetCore Dataset”. In: *Sensors* 19.7 (2019). issn: 1424-8220. doi: `10.3390/s19071553`. online: `https://www.mdpi.com/1424-8220/19/7/1553`.
67. Zhigao Cui, Ke Jiang, and Tao Wang. “Unsupervised Moving Object Segmentation from Stationary or Moving Camera Based on Multi-frame Homography Constraints”. In: *Sensors* 19.19 (2019). issn: 1424-8220. doi: `10.3390/s19194344`. online: `https://www.mdpi.com/1424-8220/19/19/4344`.
68. Keon-woo Park, Yoo-Jeong Shim, Myeong-jin Lee, and Heejune Ahn. “Multi-Frame Based Homography Estimation for Video Stitching in Static Camera Environments”. In: *Sensors* 20.1 (2019). issn: 1424-8220. doi: `10.3390/s20010092`. online: `https://www.mdpi.com/1424-8220/20/1/92`.
69. and P. Saratchandran and N. Sundararajan. “A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation”. In: *IEEE Transactions on*

Neural Networks 16.1 (Jan. 2005), pp. 57–67. issn: 1045-9227. doi: `10.1109/TNN.2004.836241`.

70. J. Wang, C. Xu, X. Yang, and J. M. Zurada. “A Novel Pruning Algorithm for Smoothing Feedforward Neural Networks Based on Group Lasso Method”. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.5 (May 2018), pp. 2012–2024. issn: 2162-237X. doi: `10.1109/TNNLS.2017.2748585`.
71. Jasmina Arifovic and Ramazan Gençay. “Using genetic algorithms to select architecture of a feedforward artificial neural network”. In: *Physica A: Statistical Mechanics and its Applications* 289.3 (2001), pp. 574–594. issn: 0378-4371. doi: `https://doi.org/10.1016/S0378-4371(00)00479-9`. online: `http://www.sciencedirect.com/science/article/pii/S0378437100004799`.
72. Dawid Połap, Karolina Kęsik, Marcin Woźniak, and Robertas Damaševičius. “Parallel Technique for the Metaheuristic Algorithms Using Devoted Local Search and Manipulating the Solutions Space”. In: *Applied Sciences* 8.2 (Feb. 2018), p. 293. doi: `10.3390/app8020293`. online: `https://doi.org/10.3390/app8020293`.
73. Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. “Kinectfusion: real-time 3D reconstruction and interaction using a moving depth camera”. In: *In Proc. UIST*. 2011, pp. 559–568.
74. J. Wang, L. Zhang, Q. Guo, and Z. Yi. “Recurrent Neural Networks With Auxiliary Memory Units”. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.5 (May 2018), pp. 1652–1661. issn: 2162-237X. doi: `10.1109/TNNLS.2017.2677968`.
75. J. Hawkins and M. Boden. “The applicability of recurrent neural networks for biological sequence analysis”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2.3 (July 2005), pp. 243–253. issn: 1545-5963. doi: `10.1109/TCBB.2005.44`.
76. Yusen Wang, Wenlong Liao, and Yuqing Chang. “Gated Recurrent Unit Network-Based Short-Term Photovoltaic Forecasting”. In: *Energies* 11.8 (2018). issn: 1996-1073. doi: `10.3390/en11082163`. online: `https://www.mdpi.com/1996-1073/11/8/2163`.
77. Zhong Liu, Changchen Zhao, Xingming Wu, and Weihai Chen. “An Effective 3D Shape Descriptor for Object Recognition with RGB-D Sensors”. In: *Sensors* 17.3 (2017). issn: 1424-8220. doi: `10.3390/s17030451`. online: `http://www.mdpi.com/1424-8220/17/3/451`.
78. G. J. Hsu, Y. Liu, H. Peng, and P. Wu. “RGB-D-Based Face Reconstruction and Recognition”. In: *IEEE Transactions on Information Forensics and Security* 9.12 (Dec. 2014), pp. 2110–2118. issn: 1556-6013. doi: `10.1109/TIFS.2014.2361028`.
79. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Commun. ACM* 60.6 (May 2017), pp. 84–90. issn: 0001-0782. doi: `10.1145/3065386`. online: `https://doi.org/10.1145/3065386`.
80. K. Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *CoRR* abs/1409.1556 (2015).
81. Min Lin, Qiang Chen, and Shuicheng Yan. *Network In Network*. 2014. arXiv: `1312.4400 [cs.NE]`.
82. Seyyed Hossein Hasanpour, Mohammad Rouhani, Mohsen Fayyaz, and Mohammad Sabokrou. *Lets keep it simple, Using simple architectures to outperform deeper and more complex architectures*. 2018. arXiv: `1608.06037 [cs.CV]`.

8.3. HUMANNET—a Two-Tiered Deep Neural Network Architecture for Self-Occluding Humanoid Pose Reconstruction

Authors Audrius Kulikajevas[1], Rytis Maskeliūnas[2], Robertas Damaševičius[3] and Rafal Scherer[3]

[1] Department of Multimedia Engineering, Kaunas University of Technology, 51368 Kaunas, Lithuania; audrius.kulikajevas@ktu.lt (A.K.); rytis.maskeliunas@ktu.lt (R.M.)

[2] Faculty of Applied Mathematics, Silesian University of Technology, 44-100 Gliwice, Poland

[3] Department of Intelligent Computer Systems, Częstochowa University of Technology, 42-200 Częstochowa, Poland; rafal.scherer@pcz.pl

Academic Editors: Chen Chen and Marco Diani

Sensors 2021, 21(12), 3945; https://doi.org/10.3390/s21123945

Received: 9 February 2021 / Revised: 29 April 2021 / Accepted: 6 June 2021 / Published: 8 June 2021

Abstract Majority of current research focuses on a single static object reconstruction from a given pointcloud. However, the existing approaches are not applicable to real world applications such as dynamic and morphing scene reconstruction. To solve this, we propose a novel two-tiered deep neural network architecture, which is capable of reconstructing self-obstructed human-like morphing shapes from a depth frame in conjunction with cameras intrinsic parameters. The tests were performed using on custom dataset generated using a combination of AMASS and MoVi datasets. The proposed network achieved Jaccards' Index of 0.7907 for the first tier, which is used to extract region of interest from the point cloud. The second tier of the network has achieved Earth Mover's distance of 0.0256 and Chamfer distance of 0.0276, indicating good experimental results. Further, subjective reconstruction results inspection shows strong predictive capabilities of the network, with the solution being able to reconstruct limb positions from very few object details.

Keywords 3D shape recognition; 3D depth scanning; pointcloud reconstruction; human shape reconstruction

8.3.1. Introduction

Computer vision is a quickly expanding field because of the success of deep neural networks [1]. The RGB camera frames have already been adopted in various industries for environment recognition [2] and object detection [3] tasks. Depth information is however, is less likely to be used due to generally requiring special sensors or monocular camera setups. For this reason computer vision field has a lot of open questions regarding the application of depth information. One of important computer vision research fields, related to application of depth information, is three-dimensional object reconstruction [4].

A lot of applications that would benefit from real-time object reconstruction such as self-driving cars [5, 6], interactive medium particularly virtual reality [7] (VR) and video games, augmented reality [8] (AR) and extended reality [9] (XR). Furthermore, depth sensor information can improve gesture [10, 11] and posture recognition [12] technologies as these tasks generally have a lot of important depth information embedded into them. Additional uses for object reconstruction from depth sensor information could include recreating environments in film industry and teleconferencing with the use of holograms, indoor mapping [13] or robotics [14, 15]. Unfortunately, while this object reconstruction gives a lot of value to various fields, generally such applications require intricate camera setups to scan the entire object from all sides or to move camera in order to gradually build the object depth profile. This makes the reconstruction technology have a high barrier of entry.

Users cannot be forced to have professional filming setups containing laser sensor arrays that would scan entire object from all perspectives in a single shot, or expect user to bother scanning the object from all sides to reconstruct it each time they add additionally obstacles to the scene. In addition, it potentially requires a lot of technical know-how and computing power to perform high fidelity pointcloud fusion, this reduces the end-user experience. For this reason, there is a need for different type solution which is capable of performing such task using only a single view. Some novel state-of-the-art methods already attempt to solve this problem using a priori knowledge. Such methods generally involve using black-box models such as deep neural networks as it gives the approaches ability to approximate the occluded object information that is generally quite easy for a person to infer based on the mental model each of us builds over our lifespans. Initial successful research in the object reconstruction field has focused in the voxel based reconstruction [16]. The proposed approach dubbed 3D-R2N2 has used Sanford Online Products [17] and ShapeNet [18] datasets as a priori knowledge to guess object shape using multi-view reconstruction. Other research has improved the results with the addition of Chamfer Distance as a loss function [19] thus increasing the reconstruction accuracy. Other attempts have attempted improving the reconstruction by using network hybridization where each network branch is trained on different group of objects thus allowing for faster model convergence and real-time reconstruction [20]. While all the mentioned methods focus on single object per scene reconstruction, there have been attempts in improving this with the use of object segmentation layer [21]. By segmenting only necessary depth information and using that as reconstruction it allows for multiple object per scene.

While the majority of methods focus on voxel based mesh representation [22, 23, 24, 25, 26, 27], for object reconstruction due to their representation simplicity, voxels have one major flaw—exponentially increasing requirements to train them with increasing fidelity. Some papers tried to solve this ever-increasing memory requirements using smarter data representation styles like octrees [28, 29]. These allow for more details to be preserved, however, they still are not as detailed as pointclouds. There already exists some solutions that attempt to do this such as PointOutNet [30] that has shown the ability to predict and generate plausible 3D shapes of objects. While this solution has shown generally good prediction results, it relies on user segmentation mask for reconstruction. While PointOutNet is capable of leveraging 2D convolutions in order to reconstruct 3D object, there is some information that is missing for this approach to be stable. Even though 3D convolutions can be easily applied to voxel clouds both 2D and 3D convolutions are not very useful when dealing with pointclouds as they have fundamentally different structure. Some approaches configurations have shown the ability to generalize pointcloud information [31]. Further modifications to PointNet have been shown to be able to reconstruct shapes using pointcloud inputs [32].

We propose a novel two tiered approach capable of full human body pointcloud reconstruction using a single realistic imperfect (self-occluding) depth view, where the first rank network clips the initial depth cloud and the second rank uses prime output to reconstruct the captured object. Our contribution to the field of object reconstruction is the addition of the clipping-resampling node which gives our approach the ability to extract three-dimensional Regions of Interest (RoIs) that can be then used for reconstruction. Unlike previous existing approaches which rely on user-defined masks to extract regions of interest, ours is completely independent and provides a complete solution sensor-to-screen object reconstruction.

Generally, reconstruction focuses on static single object per scene reconstruction. However, we attempt to reach new a frontier in this field. Our approach attempts to take one step further, reconstruction of full human shape using single imperfect depth frame information in order to reconstruct missing scene information. Our method involves two tiered reconstruction networks and a priori knowledge of the human body to make the predictions of the reconstructed pose.

8.3.2. Related Work

Object reconstruction is a rapidly expanding computer vision field. Most of the new solutions that relate to this topic benefit from the advancements in the artificial intelligence. Two main approaches for three-dimensional object reconstruction are: voxel based and pointcloud based. One such voxel based solution is 3D-R2N2. It uses Long Short Term Memory [33, 34] (LSTM) in order to learn the object features from multiple views and later reconstruct them. This approach is afterwards capable of reconstructing voxel grid using only a single RGB view based on a priori knowledge obtained during training. The method requires additional masks provided separately in order to reconstruct the results. Another solution attempted to use an extended YoloV3 [21] (YoloExt) has attempted to get rid of this dependency by merging YoloV3 [35] with the reconstruction task. Unlike prior solution the YoloExt was capable of detecting and then segmenting the RoIs itself and passing them mask and depth to the reconstruction branches. This allowed for the solution to be independent of additional user input and could work with real world data. However, the voxel based solutions while being simple to train suffer from two major flaws: exponential memory requirements to train and requiring high granularity grid in order to preserve small features. To resolve high memory requirements while maintaining high fidelity another competing reconstruction approach exists, i.e., pointcloud reconstruction. Unlike previous approaches it has a much lower memory impact, therefore potentially allowing for much higher fidelity reconstruction. However, the pointcloud solutions are notoriously hard to train due to a more complex loss function being required.

One of first such solutions was PointOutNet. Just like 3D-R2N2 it requires an external mask provided to the network and reconstructs the shape using RGB frames. However, unlike 3D-R2N2 it reconstructs the shape using unstructured pointcloud. Thus obtaining higher efficiency than the competing voxel approaches. The approach suggests both Chamfer and Earth Mover's distance as loss metrics.

Further research in pointcloud reconstruction in PointNet [36] has attempted to instead of using RGB frame as input using a pointcloud. However, such pointcloud methods are unable to use the traditional 2D convolutions due to pointclouds being unstructured dataset. To solve for this problem, PointNet attempts to learn symmetric functions and learn local features. The addition of fully-connected auto-encoders to the PointNet has shown the ability to fill in missing chunks of the malformed pointcloud.

PCN [37] proposes a fine-grained pointcloud completion method while maintaining a small number of training parameters due to its coarse-to-fine approach. AtlasNet [38] proposes a patches based approach capable of mapping 2D information into parametric 3D objects. Due to high complexity of $O(n^2)$ required for the calculation of Earth Mover's distance the majority of solutions tend to use Chamfer distance as loss metric. However, the latter is less sensitive to density distribution. For this reason, MSN [39] proposes an Earth Mover's approximation which can be applied to pointclouds and a sampling algorithm for obtaining evenly distributed subset of pointcloud. However, all prior approaches all revolve around reconstructing quite static objects and not dynamically morphing meshes such as human body. Some approaches dealing with human body prediction using depth information exist [40, 41, 42, 43] however their body predictions do not deal with full body reconstruction and only pose estimation.

The comparison of existing methods versus ours can be seen in Table 1, as we can see our solution is capable reconstructing sensor-to-screen pointclouds using only sensor provided information, while maintaining sensitivity to high density distributions due to the use of EMD as loss metric.

8.3.3. Materials and Methods

8.3.3.1. Proposed Deep Neural Network Architecture

Our synthetic dataset attempts to create real-world like dataset that other approaches were incapable of generalizing. For this reason our proposed black-box model (artificial neural network) consists of two tier network structure (see Figure 1). First network rank deals with extracting the required features of the pointcloud and downsampling. The second rank uses the clipped and resampled pointcloud in order to learn the required features for full human body reconstruction.

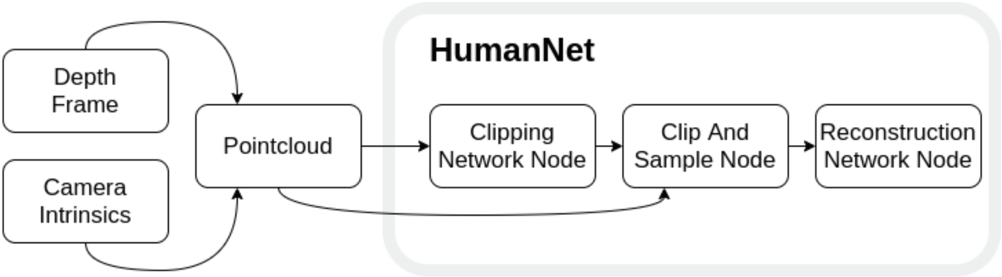


Figure 1. Proposed two-tiered network overview. Intrinsic camera matrix is applied to depth information in order to generate pointcloud. Pointcloud is then passed onto Clipping Network Node which finds predict the bounding box. Bounding box is then used along with initial point cloud to clip the Region of Interest and downsample. The result is then used to reconstruct the human shape.

Table 1. Table comparing different existing implementations. Standalone refers to sensor-to-screen solutions where for any given sensor input a fully reconstructed model can be expected without inputting external information that the sensor itself cannot provide, such as masks.

Name	**Voxels**	**Pointcloud**	**Input**	**EMD**	**CD**	**Standalone**
3D-R2N2	✓	✗	RGB	—	—	✗
YoloExt	✓	✗	RGB-D	—	—	✓
PointOutNet	✗	✓	RGB	✓	✓	✗
PointNet w/ FCAE	✗	✓	Pointcloud	✗	✓	✗
PCN	✗	✓	Pointcloud	✗	✓	✗
AtlasNet	✗	✓	Pointcloud	✗	✓	✗
MSN	✗	✓	Pointcloud	✓	✗	✗
Ours	✗	✓	Depth	✓	✗	✓

8.3.3.2. Clipping Network Architecture

Our dataset involves two inputs: pinhole depth image and camera intrinsic matrix K (see Equation 1). By applying camera intrinsics to each of depth points we create undistorted pointcloud that we can use for training. The first rank network (see Table 2) is responsible for filtering as much unnecessary information that the pointcloud contains as possible. This is done to avoid *poisoning* the initial neural network training states as they are tightly dependent on the input frame during training. Having too much unnecessary information makes the reconstruction network very difficult to train. For this reason the main purpose of the first rank is to detect the desired feature bounding box.

One of the approaches to mask out only interesting data is to try and predict the 2D mask by using segmentation techniques capable of segmenting objects in the frame [44, 45, 46]. While such approaches can easily exploit 2D convolutions they lack one very important feature — third dimension. Therefore, we would be unable to filter out objects that are in front of the object. Additionally, 2D convolutions are much slower than the approach we have chosen that deals with pointclouds directly. Because our input depth resolution is $640x480$ pixels once convert it into pointcloud (see Equation 2) we get a total of 307200 vertices in the cloud. While it is possible to use this entire pointcloud as the neural network input it would make it unusable in real-time applications. For this reason we use *Farthest Point Sampling* [47] (FPS) operation to collect 2048 points. We found that this amount of vertices is more than enough to extract all necessary features from frames. The downsampled input is then used as input for the network.

While the network was capable of learning most of the feature bounding boxes it was heavily biased by the imbalances of dataset. Our dataset contains two primary types of bounding boxes tall-thin and short-wide due to two main human poses being either standing or crouching. For this reason we had borrowed widely used approach in Single Shot Detection methods where anchor boxes are used to help neural network learn the 2D object bounding boxes [48, 49, 50]. However, if we only had two anchor boxes our dataset would become very imbalanced, for that reason we have increased the anchor count to four anchors, this gives us a more even pose distribution. The predicted three-dimensional bounding box acts as six clipping planes that allows us to filter out all vertices that do not belong to that object.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$p_{(x,y,z)} = \begin{bmatrix} x - \frac{c_x \cdot z_i}{f_x} \\ y - \frac{c_y \cdot z_i}{f_y} \\ z \end{bmatrix} \tag{2}$$

Due to the fact that our approach has four potential bounding box anchors we get four potential bounding boxes. However, our network also outputs the confidence level of the bounding box. The bounding box with the highest confidence level is used for clipping. Once the highest confidence bounding box is acquired we may perform clipping and resampling operation using the initial 307200 vertex pointcloud. As our initial downsampling included points that do not belong to *Region of Interest* the resulting point cloud has a much lower density, hence less information that could be used for reconstruction. For this reason we clip the original pointcloud and downsample to 4096 points. While it may seem counter-productive to resample twice instead of having initial resampling much higher density however, *FPS* is a cheaper operation than working with much higher pointcloud resolution.

$$\begin{gathered} \epsilon_{clip}(y, \hat{y}) = \sum L1_s(y_{pos}, \hat{y}_{pos}) \cdot y_{conf} + \\ \sum L1_s(y_{scl}, \hat{y}_{scl}) \cdot y_{conf} + \epsilon_{bce}(y_{conf}, \hat{y}_{conf}) \end{gathered} \tag{3}$$

When training our neural network we calculate three different loss functions: position loss, scale loss and confidence loss (see Equation 3). $L1_s$ in Equation 3 refers to smooth L1 loss (see Equation 5) [51], while BCE refers to binary cross entropy loss (see Equation 4),

$$\epsilon_{bce}(y, \hat{y}) = -\frac{1}{n} \cdot \sum_{i}^{n} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log(1 - \hat{y}_i) \tag{4}$$

$$L1_s(y, \hat{y}) = \frac{1}{n} \sum_{i}^{n} z_i \tag{5}$$

where z_i is Equation 6 with $\beta = 0.1$.

$$z_i(y_i, \hat{y}_i) = \begin{cases} \frac{0.5 \cdot (\hat{y}_i - y_i)^2}{\beta}, & if |\hat{y}_i - y_i| < \beta \\ |\hat{y}_i - y_i| - 0.5 \cdot \beta, & otherwise \end{cases} \tag{6}$$

Table 2. Architecture of clipping network. Last convolutional layer does not contain activation function because finding bounding boxes is a regression task.

Type	Filters	Size	Output
Depth	-	-	640×480
Pointcloud	-	-	307200×3
Resample	-	-	2048×3
Convolution 1D	64	1	2048×64
Convolution 1D	128	1	2048×128
Convolution 1D	1024	1	2048×1024
Adaptive Max Pool 1D	-	2	2×1024
Convolution 1D	512	1	2×512
Linear Convolution 1D	7	1	2×7
Clip Inputs	-	-	307200×3
Resample	-	-	4096×3

8.3.3.3. Reconstruction Network Architecture

Our second rank network (see Table 3) was heavily inspired by Morphing and Sampling Network (MSN) which shows state-of-the-art reconstruction results for pointcloud reconstruction. However, the proposed network got easily poisoned by excess information that did not belong to the object which was being reconstructed, as it was heavily influenced by the initial pointcloud used as input.

Table 3. Architecture of the reconstruction neural network. We use 16 Morph-Based-Decoders for 16 potential surfaces for the network to be able to predict.

Label	Type	Filters	Size	Output
	Input	-	-	4096×3
Encoder	Conv 1D	64	1	4096×64
	Conv 1D	128	1	4096×128
	Linear Conv 1D	1024	1	4096×1024
	Max Pool 1D	-	-	1024
	Fully Connected	256	-	256
$16 \times$ Coarse Decoder	Conv 1D	256	1	$16 \times 256 \times 258$
	Conv 1D	129	1	$16 \times 256 \times 129$
	Conv 1D	64	1	$16 \times 256 \times 64$
	Conv 1D	3	1	$16 \times 256 \times 3$
	Conv	-	-	4096×3
Final Decoder	Conv 1D	64	1	4096×64
	Conv 1D	128	1	4096×128
	Conv 1D	1024	1	4096×1024
	Max Pool 1D	-	-	1024
	Residual	-	-	1088
	Conv 1D	512	1	4096×512
	Conv 1D	256	1	4096×256
	Conv 1D	128	1	4096×128
	Conv 1D	3	1	4096×3

As we can see from the Table 4, the modifications we made to the deep neural network architecture, had an overall negligible impact in terms of trainable parameters our neural network had to learn weights for and the model size, while slightly reducing the overall number of operations for the network to process due to the addition of resampling after clipping the objects RoIs.

Table 4. Comparison of neural network complexity by number of parameters, number of operations and model size.

Method	**No. of Parameters (M)**	**No. of Operations (GFLOPs)**	**Model Size (MB)**
PointNet w/ FCAE	7.43	1.18	28.36
PCN	6.87	29.5	26.25
AtlasNet	3.31	6.46	12.66
MSN	29.50	12.89	112.89
Ours	29.71	11.74	112.94

Because the reconstruction network could easily get poisoned by bad input data due to its dependence on initial point positions, clipping loss had to reach $\epsilon < 0.3$ before reconstruction starts weights got updated. This approach kept randomized initial weight values in stable positions, easing the training process. The reconstruction training process requires a metric in order to compare ground truth S and prediction $\hat{S}$ values. While one of the most popular metrics when comparing pointclouds is Chamfer Distance [52] due to its low memory impact and fast computation. The metric measures mean distance between two pointclouds. However, we found that for our task it was not able to learn the features properly causing vertices to congregate together instead of spreading uniformly around the object shape. For this reason, we chose to use Earth Mover's Distance (see Equation (3)) with expansion penalty (see Equation (4)), as per suggested penalization criteria for surface regularization proposed in MSN, where $d(u, v)$ is Euclidean distance between two vertices in three-dimensional space and ϕ is the bijection of pointclouds. $\mathbb{1}$ is the indicator function used to filter which shorter than λl_i with $\lambda = 1.5$ as per suggested value, giving us a final combined reconstruction loss as final Equation (9) with $\alpha = 0.1$, $\hat{S}_{coarse}$ is coarse decoder output and $\hat{S}_{final}$ is final decoder output.

$$\epsilon_{emd}(S, \hat{S}) = \min_{\phi: S \to \hat{S}} \frac{1}{|S|} \sum_{x \in S} ||x - \phi(x)||_2 \tag{7}$$

$$\epsilon_{exp} = \frac{1}{KN} \sum_{1 \leq i \leq K} \sum_{(u,v \in \tau_i)} \mathbb{1}\{d(u,v) \geq \lambda l_i\} d(u,v) \tag{8}$$

$$\begin{gathered} \epsilon = \epsilon_{clip} + \\ \mathbb{1}\{\epsilon_{clip} < 0.3\}(\epsilon_{emd}(S, \hat{S}_{final}) + \epsilon_{emd}(S, \hat{S}_{coarse}) + \alpha\epsilon_{exp}) \end{gathered} \tag{9}$$

8.3.3.4. Dataset

There are various existing datasets for object detection that contain labeled image data such as COCO [53] and Pascal VOC [54], 3D object datasets such as ShapeNet and even labeled voxel data [55]. However, our task required a very specific dataset: it required human meshes that could be used as ground truth, and it needed to contain depth camera information matching the mesh positions. As far as we are aware there exists no publicly available dataset matching this description. For this reason we generated a synthetic dataset using Blender [56]. The MoVi [57] dataset contains a vast amounts of motion capture data and multiple camera perspective video. However, videos contain no depth information, therefore it does not fully match our criteria. For this reason we used motion capture data bound to the AMASS [58] triangle meshes. An example of AMASS dataset can be seen in Figure 1.

To create the dataset we placed the motion captured model into it and capture depth frames from various angles by rotating the camera and the person model itself. Rotating the camera simulated multiple cameras seeing same event, while rotating the model emulated the person doing same poses from different angles (see Figure 2). The person was rotated from 0°to 360°in the increments of 45°, while the camera was rotated from -35°to 35°in the increments of 15°. The camera was placed 4.5 m away from the person. The rendered depth frame was saved using OpenEXR [59] file format as unlike other general purpose image formats, such as JPEG, it is linear and lossless therefore it does not lose any depth information and is not limited to 8 bits per channel. Additionally, the frame itself was rendered using mesh and our ground-truth demands for pointcloud, to generate it we used uniform random sampling.

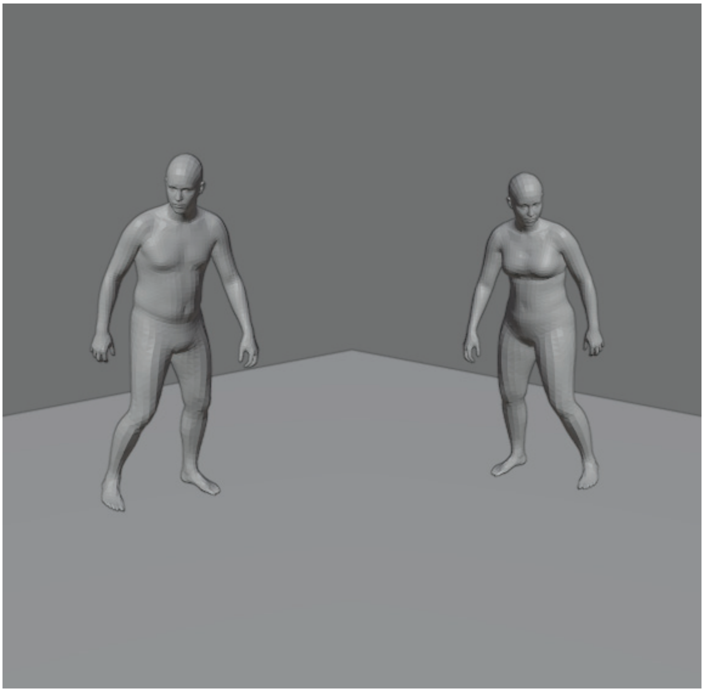


Figure 2. An example MoVi dataset motion capture pose applied to models provided by AMASS. The same pose is applied to female and male body type.

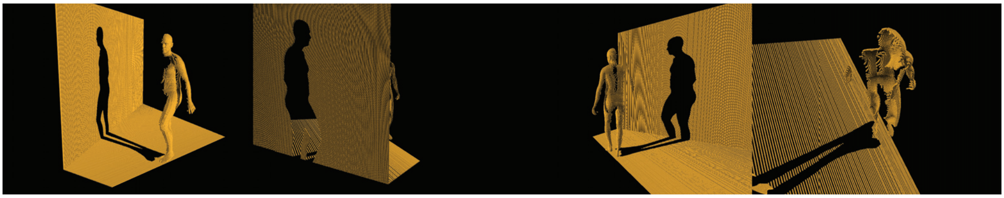


Figure 3. An example of neural network input that is created rendered depth frame converted to pointclouds with the help of camera intrinsic matrix K.

8.3.4. Results

8.3.4.1. Clipping Results

In order to evaluate the accuracy of our clipping node we used Jaccards index [60, 61, 62] to compare the quality of our three-dimentional bounding boxes, which is widely adopted as a metric to compare bounding boxes. Our results (seen in Figure 4) indicate that for most of our anchors but one our $I \cap U \approx 80\%$, with overall accuracy being 79.07%, with some clipping error was able to be improved by slightly expanding the bounding boxes thus potentially improving bounding boxes which were very close to ground truth. The Jaccard index of Anchor 3 being much lower than others may be due to imbalanced number of samples belonging to each dataset.

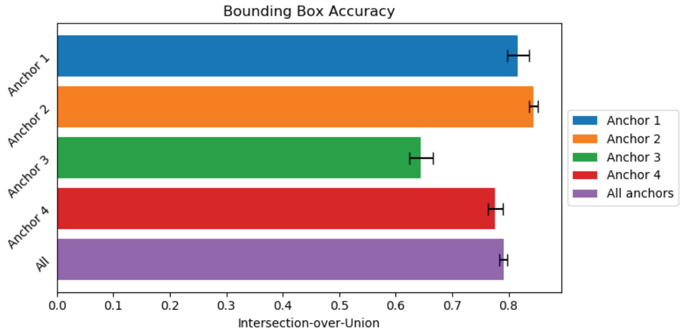


Figure 4. Bounding box accuracy expressed as $I \cap U$ for anchors. Higher is better.

8.3.4.2. Reconstruction Results

The purpose of our network was to reconstruct the human body shapes. To determine the quality of our reconstructions we needed an objective metric to compare results. For this reason we used two main metrics to evaluate model quality Chamfer Distance and Earth Movers Distance (Equation (3)), which is summarized in Figure 11.

$$\epsilon_{cd}(S,\hat{S}) = \frac{1}{2}\left(\frac{1}{|S|}\sum_{x\in S}\min_{y\in\hat{S}}||x-y||_2^2 + \frac{1}{\hat{S}}\sum_{y\in\hat{S}}\min_{x\in S}||x-y||_2^2\right) \tag{10}$$

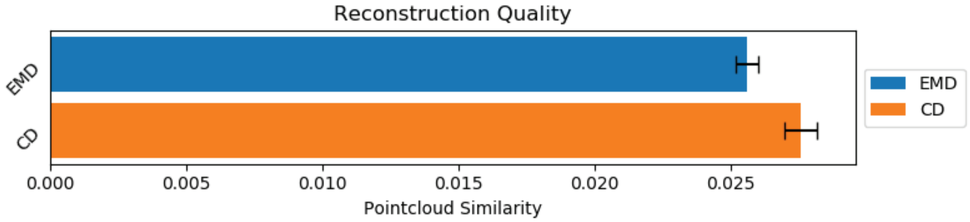


Figure 5. Reconstruction similarity using both Earth Movers Distance and Chamfer Distance. Lower value is better.

We also summarize the distribution of the errors in terms of histogram as Figure 6. 95% of Earth Movers Distance was lower than 0.054, and for Chamfer Distance, lower than 0.078.

We cannot directly compare our results to other researchers' reconstruction results, due to us using a completely different dataset than other state-of-the-art research uses. The approaches we have tested were unable to

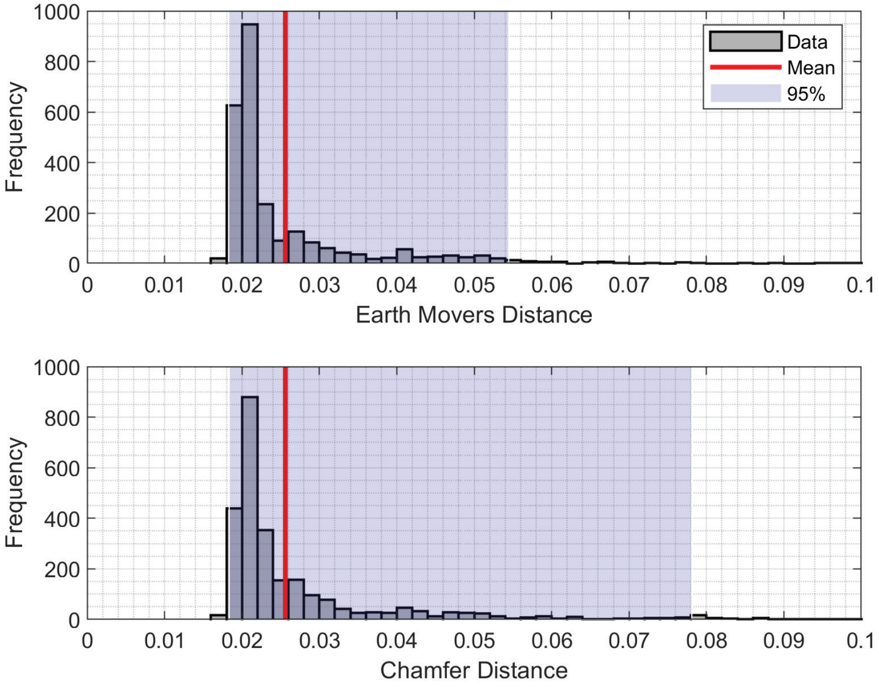


Figure 6. Distribution of Earth Movers Distance and Chamfer Distance values. Box shows 95% of values are between 2.5% and 97.5% percentiles of values.

deal with the additional noise our dataset contains in the form of backgrounds and depth shadows as they lacked a Region of Interest mechanism. However, if we compare the metrics provided with other state-of-the-art methods (see Table 2) we can see that our reconstruction results were similar with the added robustness and flexibility by only reconstructing RoIs.

Table 5. Reconstruction metric comparison between other methods and ours. While direct comparison cannot be drawn due different datasets and techniques being adopted we can see that the reconstruction values are at least very similar to state-of-the-art when compared to *ShapeNet* dataset. As per *Liu et al.* (2020) reference values.

Method	**EMD**	**CD**	**Dataset**
PointNet w/ FCAE	0.0832	0.0182	ShapeNet
PCN	0.0734	0.0121	ShapeNet
AtlasNet	0.0653	0.0182	ShapeNet
MSN	0.0378	0.0114	ShapeNet
Ours	0.0256	0.0276	AMASS

Another way to inspect prediction results that is not objective, nonetheless very important, is visually. Figure 7 displays same pointclouds from four different angles. The first row contains different views of input pointcloud that the first tier network responsible for clipping and resampling was fed. Once the prediction was made, the second row displays the pointcloud after clipping removed points that did not belong to the Region of Interest and downsampled them to 4096 points. The third row is the prediction made by the second tier network, responsible for the human body reconstruction. The final row shows first and second tier network results overlapped. As we can see, the prediction network managed to rebuild entirely missing features based on the most probable guess. Due to depth self-obstruction depth shadows were cast. This caused the input frame to be missing these features: half of the torso, half left hand, half of left hand, almost entire right hand, and half right leg. As we can see the prediction managed to guess very realistic right leg and right arm orientations based on the very few points that were provided by such features as the angle of the right shoulder and elbow. From this we can assert that our network had human-like speculative probabilities on how the obstructed parts of the body may be orientated. Additional validation of this assertion can be seen in Figure 12 comparing ground truth pointcloud and prediction made by the deep neural network. As we can see while there were imperfections in the predicted pointcloud the reconstructed object did in fact reconstruct the entire object shape.

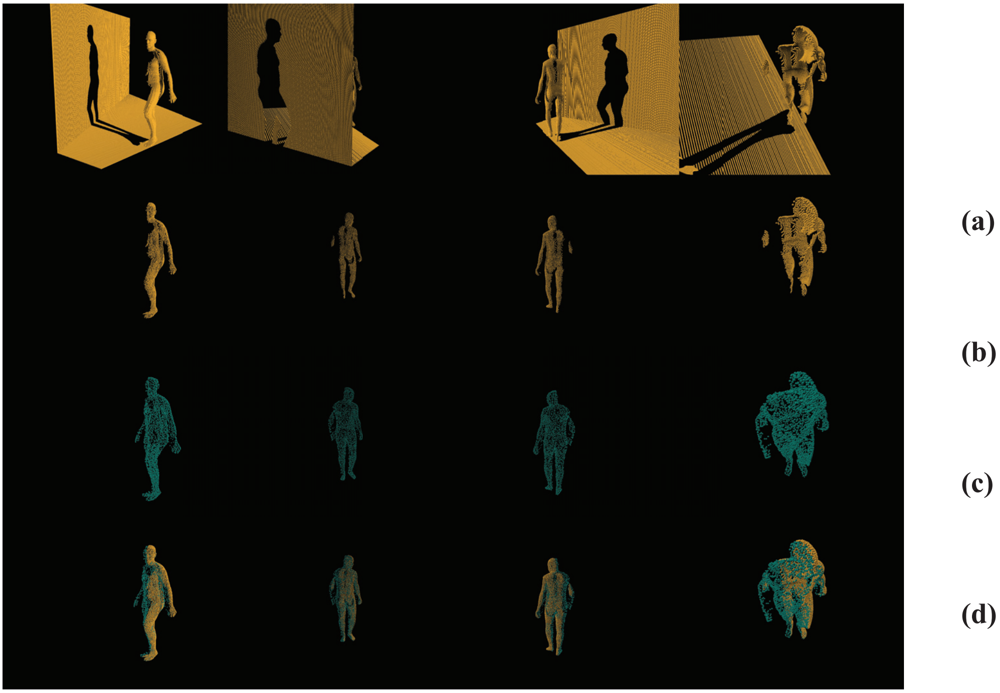


Figure 7. Different viewpoints of same pointclouds. Contains stacked from top to bottom: (a) input pointcloud; (b) clipped and sampled pointcloud; (c) predicted pointcloud; (d) combined (b) and (c).

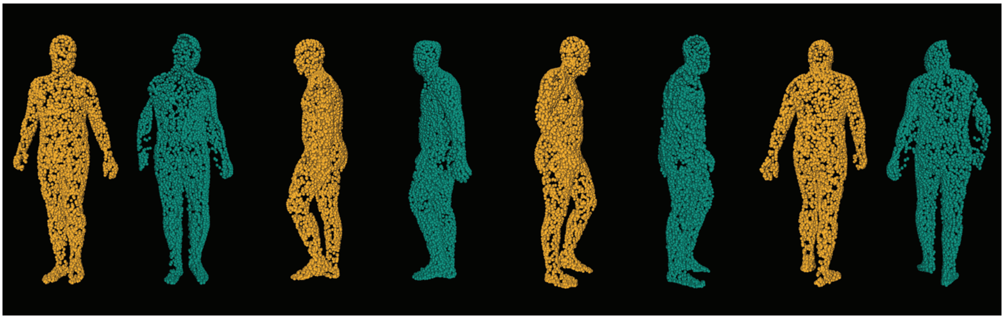


Figure 8. Comparison of ground truth (left/orange) and prediction (right/teal) from different viewpoints.

If we break down the reconstruction results by the pose, which is presented in Figure 10, we can see that the majority of our poses fell below 0.05 value of EMD and CD. Therefore the neural network was in fact able to perform pattern matching to the human pose. In further breakdown of our results (see Figure 9), we can see that there was very little disparity between the gender results, too. This implies that the suggested solution was body shape agnostic, as it was able to reconstruct both male and female human body shapes that were provided by the AMASS dataset with similar results. While a part of this gender reconstruction similarities can be attributed to the general similarities of the human shape, further visual inspection shows that the network was able to restore the distinctly male or female features.

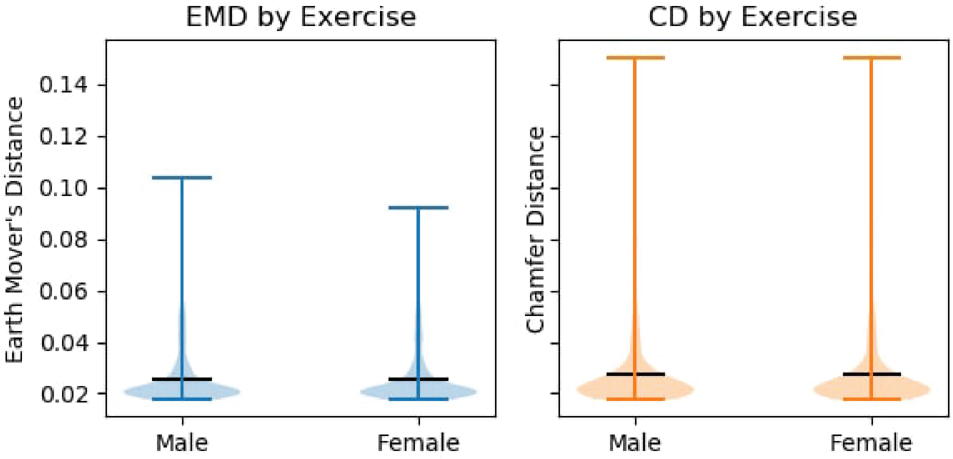


Figure 9. Reconstruction quality breakdown of the subjects' by gender.

8.3.5. Discussion and Concluding Remarks

8.3.5.1. Discussion

The main advantage of the proposed two-tiered neural network architecture as compared to existing reconstruction algorithms is the addition of the first tier Region of Interest (RoI) extraction node. Existing object reconstruction implementations deal with pre-masked user data. Therefore, they are not fit for real-world-like input data, where additional background noise exists along with the object we are attempting to reconstruct. Unfortunately, in addition to background noise, real-world depth sensors also produce a lot of distortions in their depth frames, for which our approach was not able to account for. This requires further research in the field by either creating a real world dataset akin to our synthetic, or an attempt to recreate the distortions for the synthetic dataset which could be used as an augmentation. Additionally, our RoI node is not strongly coupled to the reconstruction branch. This allows us to replace one part of the model completely without retraining the other. For example, our current implementation is unable to extract multiple Regions of Interest from a depth frame. However, if such changes were to be applied, we would be able to keep the existing reconstruction weights. This would allow us to run a separate reconstruction task for each region of interest, without changing the

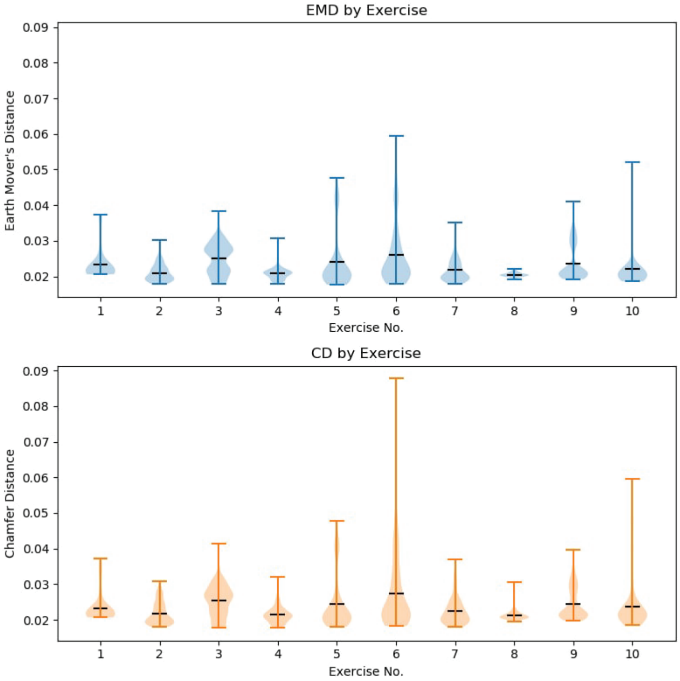


Figure 10. Reconstruction quality breakdown by the recorded motion capture exercise.

entire reconstruction network architecture, thus reducing the amount of Graphical Processing Unit (GPU) time required to train it. Finally, unlike a lot of previous methods, that attempt to rebuild the object shape using voxel grid, non-normalized pointcloud approach inherently does not need to solve for homography, which removes the requirement of extracting the objects world transformation matrix. Instead, the pointcloud based approaches that do not apply normalization to the pointcloud in attempt to improve training process, reconstruct the vertices in their positions in relation to camera space. This removes the need of translating world space coordinates into camera space post-reconstruction and therefore can be easily applied in such applications as Virtual Reality in conjunction with Augmented Reality.

8.3.5.2. Concluding Remarks

We have proposed a two-tiered neural network architecture which has successfully achieved the desired goal of reconstructing human shaped pointcloud.

The proposed network achieved Jaccards' Index of 0.7907 for the first tier which is used to extract Region of Interest from the pointcloud. Second tier of the network has achieved Earth Movers distance of 0.0256 and Chamfer distance of 0.276 indicating good experimental results. Further, subjective reconstruction results inspection shows strong predictive capabilities of the network, with the solution being able to reconstruct limb positions from very few object details.

Finally, unlike previous research, due to the use of anchor boxes our solution does not rely on the user given mask in order to perform reconstruction step giving us a clear advantage over other approaches and theoretical ability to reconstruct multiple objects per scene.

8.3.5.3. Future Work

Our current implementation has been trained and tested using a noiseless synthetic dataset only. Real world depth frames generally contain a lot of imperfections when using consumer grade sensors for that reason future work would have to adapt the proposed solution to be able to reconstruct real world data. Producing such a dataset is a tedious task as it requires labeling 3D data by manually extracting the three-dimensional bounding boxes from a given depth frame in addition to creating an appropriate pointcloud representations to be used as ground truths during the training process. The later can be achieved by creating a dataset containing pointcloud fusion of multiple camera perspectives. Additionally, our dataset only deals with the reconstruction of a single object, where there are no additional objects in the scene, therefore a human body which is occluded by other objects within the scene would not be properly reconstructed.

Author Contributions Conceptualization, R.M.; methodology, R.M.; software, A.K.; validation, A.K., R.M., and R.D.; formal analysis, R.D. and R.S.; investigation, A.K., R.M., and R.D.; resources, R.M.; data curation, A.K.; writing—original draft preparation, A.K., and R.M.; writing—review and editing, R.D. and R.S.; visualization, A.K. and R.M.; supervision, R.M.; funding acquisition, R.S. All authors have read and agreed to the published version of the manuscript.

Funding This research received no external funding.

Institutional Review Board Statement Not applicable.

Informed Consent Statement Not applicable.

Data Availability Statement Data is available from the corresponding author upon reasonable request.

Conflicts of Interest The authors declare no conflict of interest.

References

1. A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis. “Deep Learning for Computer Vision: A Brief Review”. In: *Computational Intelligence and Neuroscience* 2018 (2018).
2. U. Malūkas, R. Maskeliūnas, R. Damaševičius, and M. Woźniak. “Real time path finding for assisted living using deep learning”. In: *Journal of Universal Computer Science* 24.4 (2018), pp. 475–487.
3. K. Ryselis, T. Petkus, T. Blažauskas, R. Maskeliūnas, and R. Damaševičius. “Multiple Kinect based system to monitor and analyze key performance indicators of physical training”. In: *Human-centric Computing and Information Sciences* 10.1 (2020).
4. K. Fu, J. Peng, Q. He, and H. Zhang. “Single image 3D object reconstruction based on deep learning: A review”. In: *Multimedia Tools and Applications* 80.1 (2021), pp. 463–498.
5. Alberto Díaz-Álvarez, Miguel Clavijo, Felipe Jiménez, and Francisco Serradilla. “Inferring the Driver’s Lane Change Intention through LiDAR-Based Environment Analysis Using Convolutional Neural Networks”. In: *Sensors* 21.2 (2021). issn: 1424-8220. doi: `10.3390/s21020475`. online: `https://www.mdpi.com/1424-8220/21/2/475`.

6. Melissa Latella, Fabio Sola, and Carlo Camporeale. "A Density-Based Algorithm for the Detection of Individual Trees from LiDAR Data". In: *Remote Sensing* 13.2 (2021). issn: 2072-4292. doi: `10.3390/rs13020322`. online: `https://www.mdpi.com/2072-4292/13/2/322`.
7. Bruno Fanini, Alfonsina Pagano, and Daniele Ferdani. "A Novel Immersive VR Game Model for Recontextualization in Virtual Environments: The uVRModel". In: *Multimodal Technologies and Interaction* 2.2 (Apr. 2018), p. 20. doi: `10.3390/mti2020020`. online: `https://doi.org/10.3390/mti2020020`.
8. Alex Ibañez-Etxeberria, Cosme J. Gómez-Carrasco, Olaia Fontal, and Silvia García-Ceballos. "Virtual Environments and Augmented Reality Applied to Heritage Education. An Evaluative Study". In: *Applied Sciences* 10.7 (2020). issn: 2076-3417. doi: `10.3390/app10072352`. online: `https://www.mdpi.com/2076-3417/10/7/2352`.
9. Åsa Fast-Berglund, Liang Gong, and Dan Li. "Testing and validating Extended Reality (xR) technologies in manufacturing". In: *Procedia Manufacturing* 25 (2018). Proceedings of the 8th Swedish Production Symposium (SPS 2018), pp. 31–38. issn: 2351-9789. doi: `https://doi.org/10.1016/j.promfg.2018.06.054`. online: `http://www.sciencedirect.com/science/article/pii/S2351978918305730`.
10. G. Plouffe and A. Cretu. "Static and Dynamic Hand Gesture Recognition in Depth Data Using Dynamic Time Warping". In: *IEEE Transactions on Instrumentation and Measurement* 65.2 (2016), pp. 305–316. doi: `10.1109/TIM.2015.2498560`.
11. L. Ma and W. Huang. "A Static Hand Gesture Recognition Method Based on the Depth Information". In: *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*. Vol. 02. 2016, pp. 136–139. doi: `10.1109/IHMSC.2016.159`.
12. W. Ren, O. Ma, H. Ji, and X. Liu. "Human Posture Recognition Using a Hybrid of Fuzzy Logic and Machine Learning Approaches". In: *IEEE Access* 8 (2020), pp. 135628–135639. doi: `10.1109/ACCESS.2020.3011697`.
13. Adnan Ahmed Rafique, Ahmad Jalal, and Kibum Kim. "Automated Sustainable Multi-Object Segmentation and Recognition via Modified Sampling Consensus and Kernel Sliding Perceptron". In: *Symmetry* 12.11 (2020). issn: 2073-8994. doi: `10.3390/sym12111928`. online: `https://www.mdpi.com/2073-8994/12/11/1928`.
14. Maria João Sousa, Alexandra Moutinho, and Miguel Almeida. "Thermal Infrared Sensing for Near Real-Time Data-Driven Fire Detection and Monitoring Systems". In: *Sensors* 20.23 (2020). issn: 1424-8220. doi: `10.3390/s20236803`. online: `https://www.mdpi.com/1424-8220/20/23/6803`.
15. Javier Pérez, Mitch Bryson, Stefan B. Williams, and Pedro J. Sanz. "Recovering Depth from Still Images for Underwater Dehazing Using Deep Learning". In: *Sensors* 20.16 (2020). issn: 1424-8220. doi: `10.3390/s20164580`. online: `https://www.mdpi.com/1424-8220/20/16/4580`.
16. Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016.
17. Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. "Deep Metric Learning via Lifted Structured Feature Embedding". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
18. Angel X. Chang, Thomas A. Funkhouser, Leonidas J Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. "ShapeNet: An Information-Rich 3D Model Repository". In: *CoRR* abs/1512.03012 (2015).

19. Tingsong Ma, Ping Kuang, and Wenhong Tian. “An improved recurrent neural networks for 3d object reconstruction”. In: *Applied Intelligence* (2019). doi: `10.1007/s10489-019-01523-3`.
20. Audrius Kulikajevas, Rytis Maskeliūnas, Robertas Damaševičius, and Sanjay Misra. “Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from ShapeNetCore Dataset”. In: *Sensors* 19.7 (2019). issn: 1424-8220. doi: `10.3390/s19071553`. online: `https://www.mdpi.com/1424-8220/19/7/1553`.
21. Audrius Kulikajevas, Rytis Maskeliūnas, Robertas Damaševičius, and Edmond S. L. Ho. “3D Object Reconstruction from Imperfect Depth Data Using Extended YOLOv3 Network”. In: *Sensors* 20.7 (2020). issn: 1424-8220. doi: `10.3390/s20072025`. online: `https://www.mdpi.com/1424-8220/20/7/2025`.
22. E. Piazza, A. Romanoni, and M. Matteucci. “Real-Time CPU-Based Large-Scale Three-Dimensional Mesh Reconstruction”. In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1584–1591.
23. S. Bounareli, I. Kleitsiotis, L. Leontaris, N. Dimitriou, A. Pilalitou, N. Valmantonis, E. Pachos, K. Votis, and D. Tzovaras. “An integrated system for automated 3D visualization and monitoring of vehicles”. In: *International Journal of Advanced Manufacturing Technology* 111.5-6 (2020), pp. 1797–1809.
24. E. Nocerino, E. K. Stathopoulou, S. Rigon, and F. Remondino. “Surface reconstruction assessment in photogrammetric applications”. In: *Sensors (Switzerland)* 20.20 (2020), pp. 1–25.
25. J. Zhao, C. Zong, L. Cao, S. Chen, G. Liu, J. Xu, and S. Xin. “Automatically modeling piecewise planar furniture shapes from unorganized point cloud”. In: *Computers and Graphics (Pergamon)* 90 (2020), pp. 116–125.
26. M. Kulawiak and Z. Lubniewski. “Improving the accuracy of automatic reconstruction of 3D complex buildings models from airborne lidar point clouds”. In: *Remote Sensing* 12.10 (2020).
27. H. -. Li, M. Zhang, K. Yu, X. Qi, Q. Hua, and Y. Zhu. “R3MR: Region Growing Based 3D Mesh Reconstruction for Big Data Platform”. In: *IEEE Access* 8 (2020), pp. 91740–91750.
28. M. Tatarchenko, A. Dosovitskiy, and T. Brox. “Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2107–2115. doi: `10.1109/ICCV.2017.230`.
29. Z. Mi, Y. Luo, and W. Tao. “SSRNet: Scalable 3D Surface Reconstruction Network”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 967–976. doi: `10.1109/CVPR42600.2020.00105`.
30. Haoqiang Fan, Hao Su, and Leonidas Guibas. “A Point Set Generation Network for 3D Object Reconstruction from a Single Image”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017). doi: `10.1109/cvpr.2017.264`.
31. R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 77–85. doi: `10.1109/CVPR.2017.16`.
32. W. Wu, Z. Qi, and L. Fuxin. “PointConv: Deep Convolutional Networks on 3D Point Clouds”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9613–9622. doi: `10.1109/CVPR.2019.00985`.
33. K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. “LSTM: A Search Space Odyssey”. In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (2017), pp. 2222–2232. doi: `10.1109/TNNLS.2016.2582924`.

34. W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang. "Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network". In: *IEEE Transactions on Smart Grid* 10.1 (2019), pp. 841–851. doi: `10.1109/TSG.2017.2753802`.
35. Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *CoRR* abs/1804.02767 (2018). online: `http://arxiv.org/abs/1804.02767`.
36. Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *CoRR* abs/1612.00593 (2016). arXiv: `1612.00593`. online: `http://arxiv.org/abs/1612.00593`.
37. Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. "PCN: Point Completion Network". In: *CoRR* abs/1808.00671 (2018). arXiv: `1808.00671`. online: `http://arxiv.org/abs/1808.00671`.
38. Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. "AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation". In: *CoRR* abs/1802.05384 (2018). arXiv: `1802.05384`. online: `http://arxiv.org/abs/1802.05384`.
39. Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. "Morphing and Sampling Network for Dense Point Cloud Completion". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.07 (Apr. 2020), pp. 11596–11603. doi: `10.1609/aaai.v34i07.6827`. online: `https://ojs.aaai.org/index.php/AAAI/article/view/6827`.
40. Mohamed Omran, Christoph Lassner, Gerard Pons-Moll, Peter V. Gehler, and Bernt Schiele. "Neural Body Fitting: Unifying Deep Learning and Model-Based Human Pose and Shape Estimation". In: *CoRR* abs/1808.05942 (2018). arXiv: `1808.05942`. online: `http://arxiv.org/abs/1808.05942`.
41. Lama Seoud, Jonathan Boisvert, Marc-Antoine Drouin, Michel Picard, and Guy Godin. "Increasing the robustness of CNN-based human body segmentation in range images by modeling sensor-specific artifacts". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. Sept. 2018.
42. Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. *Frustum PointNets for 3D Object Detection from RGB-D Data*. 2018. arXiv: `1711.08488 [cs.CV]`.
43. Audrius Kulikajevas, Rytis Maskeliunas, and Robertas Damaševičius. "Detection of sitting posture using hierarchical image composition and deep learning". In: *PeerJ Computer Science* 7 (2021). doi: `10.7717/peerj-cs.442`.
44. Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. "Mask R-CNN". In: *CoRR* abs/1703.06870 (2017). arXiv: `1703.06870`. online: `http://arxiv.org/abs/1703.06870`.
45. Linwei Chen, Ying Fu, Shaodi You, and Hongzhe Liu. "Efficient Hybrid Supervision for Instance Segmentation in Aerial Images". In: *Remote Sensing* 13.2 (2021). issn: 2072-4292. doi: `10.3390/rs13020252`. online: `https://www.mdpi.com/2072-4292/13/2/252`.
46. Hao Chen, Kunyang Sun, Zhi Tian, Chunhua Shen, Yongming Huang, and Youliang Yan. "BlendMask: Top-Down Meets Bottom-Up for Instance Segmentation". In: *CoRR* abs/2001.00309 (2020). arXiv: `2001.00309`. online: `http://arxiv.org/abs/2001.00309`.
47. Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 5105–5114. isbn: 9781510860964.
48. Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan,

Quoc V. Le, and Hartwig Adam. “Searching for MobileNetV3”. In: *CoRR* abs/1905.02244 (2019). arXiv: `1905.02244`. online: `http://arxiv.org/abs/1905.02244`.

49. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. “SSD: Single Shot MultiBox Detector”. In: *Lecture Notes in Computer Science* (2016), pp. 21–37. issn: 1611-3349. doi: `10.1007/978-3-319-46448-0_2`. online: `http://dx.doi.org/10.1007/978-3-319-46448-0_2`.
50. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). online: `http://arxiv.org/abs/1512.03385`.
51. Ross B. Girshick. “Fast R-CNN”. In: *CoRR* abs/1504.08083 (2015). arXiv: `1504.08083`. online: `http://arxiv.org/abs/1504.08083`.
52. Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. “Representation Learning and Adversarial Generation of 3D Point Clouds”. In: *CoRR* abs/1707.02392 (2017). arXiv: `1707.02392`. online: `http://arxiv.org/abs/1707.02392`.
53. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. “Microsoft COCO: Common Objects in Context”. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars. Cham: Springer International Publishing, 2014, pp. 740–755. isbn: 978-3-319-10602-1.
54. Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. “The Pascal Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338. issn: 1573-1405. doi: `10.1007/s11263-009-0275-4`. online: `https://doi.org/10.1007/s11263-009-0275-4`.
55. Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. “ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes”. In: *CoRR* abs/1702.04405 (2017). arXiv: `1702.04405`. online: `http://arxiv.org/abs/1702.04405`.
56. Steffen Flaischlen and Gregor D. Wehinger. “Synthetic Packed-Bed Generation for CFD Simulations: Blender vs. STAR-CCM+”. In: *ChemEngineering* 3.2 (2019). issn: 2305-7084. doi: `10.3390/chemengineering3020052`. online: `https://www.mdpi.com/2305-7084/3/2/52`.
57. Saeed Ghorbani, Kimia Mahdaviani, Anne Thaler, Konrad Kording, Douglas James Cook, Gunnar Blohm, and Nikolaus F. Troje. *MoVi: A Large Multipurpose Motion and Video Dataset*. 2020. arXiv: `2003.01888 [cs.CV]`.
58. Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. “AMASS: Archive of Motion Capture as Surface Shapes”. In: *International Conference on Computer Vision*. Oct. 2019, pp. 5442–5451.
59. Franz Kainz, Rebecca R. Bogart, and David K. Hess. “The OpenEXR image file format”. In: 2004.
60. D. Zhou, J. Fang, X. Song, C. Guan, J. Yin, Y. Dai, and R. Yang. “IoU Loss for 2D/3D Object Detection”. In: *2019 International Conference on 3D Vision (3DV)*. 2019, pp. 85–94. doi: `10.1109/3DV.2019.00019`.
61. Waleed Ali, Sherif Abdelkarim, Mohamed Zahran, Mahmoud Zidan, and Ahmad El Sallab. *YOLO3D: End-to-end real-time 3D Oriented Object Bounding Box Detection from LiDAR Point Cloud*. 2018. arXiv: `1808.02350 [cs.CV]`.
62. Abdel Aziz Taha and Allan Hanbury. “Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool”. In: *BMC Medical Imaging* 15.1 (2015). doi: `10.1186/s12880-015-0068-x`.

8.4. Auto-Refining Reconstruction Algorithm for Recreation of Limited Angle Humanoid Depth Data

Authors Audrius Kulikajevas[1], Rytis Maskeliūnas[2], Robertas Damaševičius[3] and and Marta Wlodarczyk-Sielicka[3]

[1] Department of Multimedia Engineering, Kaunas University of Technology, 51368 Kaunas, Lithuania; audrius.kulikajevas@ktu.lt (A.K.); rytis.maskeliunas@ktu.lt (R.M.)
[2] Faculty of Applied Mathematics, Silesian University of Technology, 44-100 Gliwice, Poland; robertas.damasevicius@polsl.pl
[3] Department of Geoinformatics, Maritime University of Szczecin, Waly Chrobrego 1-2, 70-500 Szczecin, Poland

Academic Editors: Roberto Vezzani and Guido Borghi

Sensors 2021, 21(11), 3702; https://doi.org/10.3390/s21113702

Received: 20 April 2021 / Revised: 11 May 2021 / Accepted: 24 May 2021 / Published: 26 May 2021

Abstract With the majority of research, in relation to 3D object reconstruction, focusing on single static synthetic object reconstruction, there is a need for a method capable of reconstructing morphing objects in dynamic scenes without external influence. However, such research requires a time-consuming creation of real world object ground truths. To solve this, we propose a novel three-staged deep adversarial neural network architecture capable of denoising and refining real-world depth sensor input for full human body posture reconstruction. The proposed network has achieved *Earth Mover* and *Chamfer* distances of 0.059 and 0.079 on synthetic datasets, respectively, which indicates on-par experimental results with other approaches, in addition to the ability of reconstructing from maskless real world depth frames. Additional visual inspection to the reconstructed pointclouds has shown that the suggested approach manages to deal with the majority of the real world depth sensor noise, with the exception of large deformities to the depth field.

Keywords pointcloud reconstruction; adversarial auto-refinement; human shape reconstruction

8.4.1. Introduction

One of the most rapidly expanding scientific research fields, thanks in part to the advancements in artificial intelligence and, specifically, Deep Neural Networks (DNNs), is computer vision. Whereas regular cameras have already been widely adopted in various object detection tasks, however, depth sensors still have narrow range of research dedicated to them. This can be attributed to them not being easily available for personal use until relatively recently with the introduction of the original *Kinect* sensor [1]. Unfortunately, while the *Kinect* technology made the depth sensors affordable they have not had wide consumer adoption outside of entertainment [2, 3], although health-related applications were also considered [4, 5, 6]. This is, however, likely a rapidly shifting trend with more consumer grade sensors, such as *Intel Realsense* [7], being released and depth scanning systems being integrated as part of mobile devices. With rapidly evolving field of three-dimensional object reconstruction, such depth sensing systems [8, 9] may be the key to boosting object reconstruction quality.

Quite a few real world applications would benefit greatly from depth sensors and/or real-time object reconstruction, starting with collision avoidance in autonomous vehicles [10, 11], robotics [12, 13], or posture recognition [14, 15]. Other object reconstruction applications may involve interactive medium, like obstacle avoidance in virtual reality [16, 17], augmented reality [18], extended reality [19], and more. Even though 3D object reconstruction opens up a lot of possibilities to various fields, the main issue with object reconstruction is that it generally requires either intricate camera setups or moving the camera around the object in order to scan the entirety of the object and to build its full profile from all sides. This makes the reconstruction have a high accessibility barrier, as daily users cannot be expected to have professional filming setups containing laser sensors arrays capable of single-shot scanning entire object from all perspectives, nor expected to bother carefully scanning an object from all of its sides to reconstruct the object iteratively. This is in conjunction with potentially requiring a lot of computing power to perform high fidelity pointcloud fusion, which greatly impacts end-user experience.

Therefore, there is a strong desire for a solution that is able of performing object reconstruction task by using only a single camera view. There already are existing solutions that attempt to solve the previously mentioned problems with multi-view perspective reconstruction by using a priori knowledge. These methods usually involve using artificial intelligence, specifically deep neural networks which are able to approximate the occluded object information leveraging reinforcement learning. This type of learning is reminiscent of how a person is able to generally infer the objects shape based on the mental object model they had built from their life experience. There already exist methods capable of performing object reconstruction from a single perspective, one of the most popular solutions due to its simple implementation is volumetric pixel (voxel)-based reconstruction. However, high fidelity models using voxel-based models require large amounts of operative memory to represent.

To mitigate this, certain reconstruction solutions instead of attempting to reconstruct voxels try to predict pointclouds. Unlike voxel-based solutions, pointclouds require very little memory overhead for their representation. However, their comparison functions are much more complex, making them a lot harder to train due to vertices being able to occupy any coordinate in three-dimensional space. One of the first of such type of solutions being *PointOutnet* [20], which has shown the capability of predicting and generating plausible 3D shapes of objects. Although the solution has shown good prediction results, it relies on hand drawn object segmentation masks, which makes this solution not really applicable to real world applications. Additionally, this solution worked on flat 2D images, which lose a lot of important depth information. Nevertheless, there are already existing solutions capable of leveraging pointcloud information in order to improve generalization and prediction quality [21]. However, these solutions generally are only applicable to either synthetic or manually pre-processed real world data, which makes them unsuitable for real-time applications.

To solve this, we propose a novel unsupervised adversarial auto-refiner capable of full human body pointcloud reconstruction using only a single self-occluding depth view capable of reconstructing real depth sensor data with no masking nor any other direct interference. Our contribution to the field of reconstruction is the three-stage (cleanup, coarse, and fine) adversarial network capable of cleaning up the noise from real world input, without losing the underlying shape or position of the body posture.

The structure of the remaining parts of the paper is as follows. Section 8.4.2 discusses the related work. Section 8.4.3 describes our synthetic dataset and the proposed methodology. Section 8.4.4 presents the results. Section 8.4.5 discusses the results of this study. Section 8.4.6 presents the conclusions.

8.4.2. Related Work

Thanks to advancements in artificial intelligence and deep neural networks, object reconstruction is a rapidly expanding computer vision field. Currently, there are two main approaches in order to reconstruct a 3D, voxel-based and pointcloud-based. One of the most well known voxel-based solutions is *3D-R2N2* [22] that uses *Sanford Online Products* [23] and *ShapeNet* [24] datasets as a priori knowledge in order to predict objects shape using both either single or multi-perspective reconstruction. It uses deep recurrent neural networks with *Long Short Term Memory* [25, 26] (LSTM) to learn objects features using multiple views as training input, while still being capable of predicting objects shape from a single perspective when performing predictions. Unfortunately, the method requires additional masks that need to be provided separately in order to make a prediction. One of the solutions that attempted to resolve this drawback has extended *YoloV3* [27] network by merging the reconstruction and object prediction tasks (*YoloExt*) [28]. Unlike *3D-R2N2*, it was capable of detecting and performing *Region of Interest* (RoI) segmentation independently before passing the mask to the object reconstruction branches. This solution allowed it to be independent of outside interference and could work with real world input. Some other attempts were made using hybridized neural network models [29] which are separately trained on groups of objects for faster model convergence and ability to reconstruct in real-time due to reduced network complexity. Despite voxel-based approaches being easy to represent and train due to low mathematical complexity of loss function, they suffer from a major flaw: the exponential memory requirements in order to train high granularity model, which would be required in order to reconstruct complex models containing a lot of details. While there have been attempts to solve this issue of ever-increasing memory requirements by using more compact data representation styles, such as octrees [30, 31], thus greatly reducing the amount of required data to represent the same model, these still suffer from overheads.

However, a much better solution of representing 3D volumes than voxels is pointclouds. Unlike voxel-based solution, pointclouds have much lower memory footprint both during training and prediction stages, this allows for much higher fidelity reconstruction to be performed. Unfortunately, due to their very nature training pointclouds is difficult, due to the high complexity of loss function that is required to compare ground truth and prediction. One of the first solutions being *PointOutNet*. Same as with *3D-R2N2*, it requires an external mask to be provided with the input in order for the network to properly reconstruct from an RGB image. However, unlike its voxel-based predecessor, it is able to reconstruct the shape using unstructured pointcloud. The approach suggests both *Chamfer* [32] and *Earth Mover's* [33] distances (CD and EMD, respectively) as loss metrics. Subsequent research in pointcloud reconstruction instead of using RGB frame that loses depth information attempted to use pointclouds as inputs [34, 35]. One of the main drawbacks when using unstructured pointclouds is that it is not possible to use well-known feature extracting convolutional neural networks, as due to unstructured nature of the pointcloud data both 2D and 3D convolutional kernels are not applicable to the input. To resolve this issue, *PointNet* attempts in learning symmetric functions and local features. When *PointNet* is matched with a fully-connected auto-encoder branch, it was able to fill in missing chunks in malformed pointclouds. Other research proposes the addition of a fine-grained pointcloud completion method; this way PCN [36] manages to maintain only a few parameters during training due to its coarse-to-fine approach. However, *AtlasNet* [37] suggests the addition of patch-based reconstruction that is capable of mapping 2D information into parametric 3D object groups, while others generally focus on *Chamfer* distance as a loss metric and only use *Earth Mover's* distance as an accuracy metric. Moreover, EMD is less sensitive to density distribution, and it also has high computational complexity of $O(n^2)$ for its calculation. An EMD approximation [38], in addition to evenly distributed point sub-sampling method, is proposed for application in *MSN*, which has shown state-of-the-art reconstruction performance.

Table 1 compares different reconstruction solution implementations. As we can see, our solution is capable of performing sensor-to-screen prediction; due to the use of EMD as loss function, we are also able to maintain sensitivity to high density distributions.

8.4.3. Materials and Methods

8.4.3.1. Dataset

There are multiple datasets for object detection which contain image data, such as *COCO* [39] and *Pascal VOC* [40], 3D object datasets, such as *ShapeNet*, and even labeled voxel data [41]. Our task requires human meshes that contain ground truth information, in addition to depth camera information which would match positions. Only a few such datasets exist publicly, like *ITOP* [42] and *EVAL* [43]; unfortunately, in both cases, they use *Kinect* sensors, which have been discontinued by *Microsoft*. Thus, any solutions developed for it are obsolete as generally different manufacturer depth sensors have different types depth errors. Therefore, it is up to us to create dataset that matches our specifications. Because creating a dataset that would have real world ground truths is prohibitively time-consuming, as in creating ground truths for each of the frame of a recorded person manually, we have devised two datasets: dataset containing synthetic data and dataset containing real world data. Synthetic dataset contains data frame samples generated using *Blender* [44], while real dataset contains data pre-recorded human poses using two *Intel Realsense* devices.

8.4.3.1.1. Synthetic Dataset

To create synthetic dataset, we use *MoVi* [45] dataset as a base as it contains a large library containing motion capture data from multiple camera perspectives. Unfortunately, dataset contains no depth information; to solve

Table 1. Comparison between different implementations. Networks capable of performing sensor-to-screen prediction where no additional external help is required are marked as Standalone.

Name	**Voxels**	**Pointcloud**	**Input**	**EMD**	**CD**	**Standalone**
3D-R2N2	✓	✗	RGB	—	—	✗
YoloExt	✓	✗	RGB-D	—	—	✓
PointOutNet	✗	✓	RGB	✓	✓	✗
PointNet w/ FCAE	✗	✓	Pointcloud	✗	✓	✗
PCN	✗	✓	Pointcloud	✗	✓	✗
AtlasNet	✗	✓	Pointcloud	✗	✓	✗
MSN	✗	✓	Pointcloud	✓	✗	✗
Ours	✗	✓	Depth	✓	✗	✓

this, we bind the motion capture data provided to the *AMASS* [46] triangle meshes. An example of *AMASS* dataset can be seen in Figure 1.

In order to create the synthetic dataset from the motion captured models, we render the depth maps from various angles by rotating the camera and the human model itself. This is done to create multiple views of the same event from a single motion capture file. The human model is rotated from $[0°, 360°)$ in 45°increments on *Up* (z) axis, whereas the camera itself is rotated in ranges of $[-35°, 35°]$ in the increments of 15°on *Up* (z) axis. The camera is placed 1.8 m away from person and 1.4 m above ground. This done so that the human position relative to the frame is more in alignment with the real world depth sensor data. The positioned model is then captured using raytracing, and the exported depth frame is saved using *OpenEXR* [47]. This file format is chosen as it does not have any type of compression and is linear and lossless; therefore, it does not lose any depth information that a standard 8-bit channel image format would provide. In addition to the rendered depth frame, we uniformly sample 2048 points of the surface mesh to create ground-truth pointclouds. An example of resampled pointcloud can be seen in Figure 2.

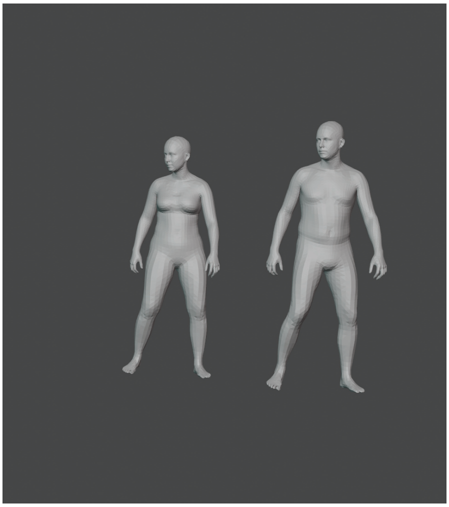


Figure 1. *MoVi* dataset example. Motion capture pose applied to models provided by *AMASS*. Same pose is applied to female and male body type.

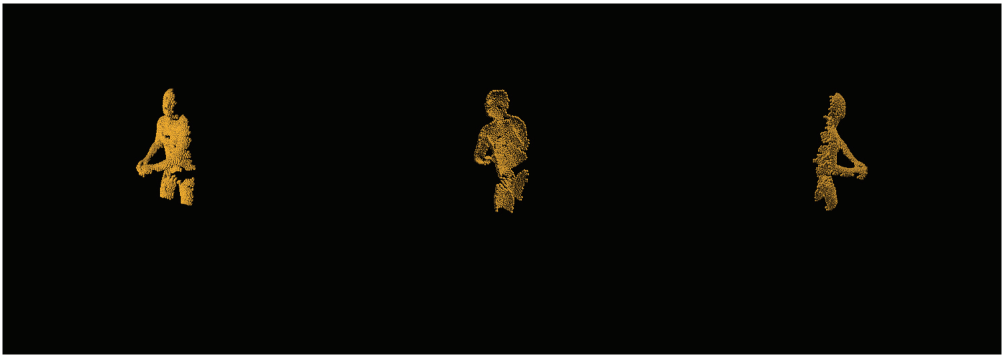


Figure 2. An example of depth frame generated by *Blender* converted into pointcloud using cameras intrinsic parameter matrix *K*.

In order to convert a pinhole typed depth camera frame into pointcloud, we use intrinsic parameter matrix K (see Equation 1), whereby, applying camera intrinsic to each of the pixels in the depth map, we are able to recreate an undistorted pointcloud. The f_x and f_y denotes the image focal points, while c_x and c_y is the sensor center point. The intrinsic parameters are applied to the 640×480 depth frame using Algorithm 4, which maps each of the depth pixels to one point in pointcloud. Points with zero depth can be filtered out, while the rest can be resampled using *Farthest Point Sampling* [48] (FPS) to desired resolution pointcloud.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{1}$$

Algorithm 4 Convert depth image to pointcloud

```
procedure TO_POINTS($w, h, f_x, f_y, c_x, c_y, D$)    ▷ Converts depth D to vertices
    $x \leftarrow 0$
    $y \leftarrow 0$
    $V \leftarrow \{\emptyset\}$    ▷ Create empty output vertex list
    for x < w do
        for y < h do
            $z_i \leftarrow D(x, y)$    ▷ Get depth value from depth frame
            $x_i \leftarrow (x - c_x) \cdot z_i / f_x$
            $y_i \leftarrow (y - c_y) \cdot z_i / f_y$
            insert ($x_i$, $y_i$, $z_i$) into V
        end for
    end for
    return V
end procedure
```

8.4.3.1.2. Real World Dataset

For our real world dataset, we have captured multiple subjects performing various tasks using two *Intel Realsense* devices. The first depth sensor (*Intel Realsense L515*) is positioned in, while the second depth sensor (*Intel Realsense D435i*) is positioned to 90° side of the subject. The subjects are asked to perform these gestures while being filmed from both angles simultaneously: standing in front camera raise the hand forward, place on top of the head, touch the nose, move the hand to the side, raise the hand above the head, or facing camera with the back raise the hand to the back. No additional preprocessing of the camera depth frames is done outside of cutting off anything further than 2.5 m away from the subject. The depth frame is then converted into pointcloud using each of the sensors intrinsic parameters and resampled using *FPS* to 2048 points. An example of resampled depth frames from each of the devices can be seen below in Figure 3. As we can see, the depth tends to be quite noisy when compared to synthetic, and this makes the existing approaches fail completely or have very poor results when trained on synthetic data.

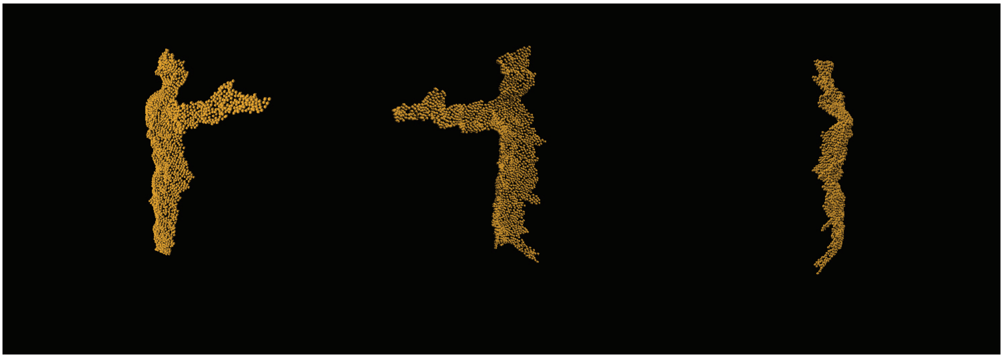


Figure 3. An example of real world depth frame converted into pointcloud using cameras intrinsic parameter matrix *K*.

8.4.3.2. Proposed Adversarial Auto-Refiner Network Architecture

Our proposed adversarial auto-refiner network architecture has three main stages used for object reconstruction. Encoder/Refiner contains the first: cleaning up and refining stage, it is responsible for cleaning out the noise from the original input and capturing the most important features of the pointcloud. The decoder contains two subsequent stages: coarse reconstruction and fine reconstruction. These three stages are our deliverables and responsible for the pointcloud reconstruction. In addition to these three stages, we also have an additional discriminator network attached at the end of reconstruction that acts as a guide to clean up the noise from the real world depth sensor data and make it *synthetic-like* without losing any of the input features. Overview of the entire network architecture can be seen in Figure 4.

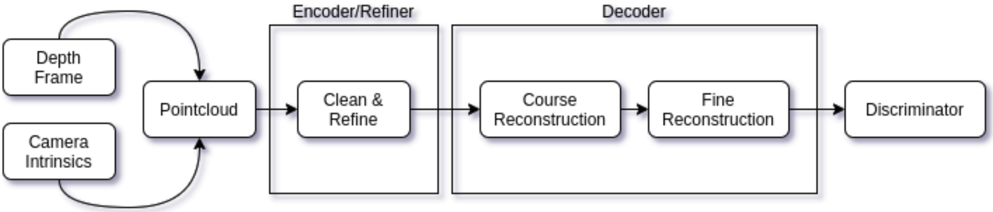


Figure 4. An overview of our adversarial auto-refiner network architecture. It uses captured depth frame and camera intrinsic matrix to convert depth information into a pointcloud. Pointcloud is then fed into encoder stage. Extracted and cleaned up features are sent to decoder where coarse and fine reconstruction stages take place. The resulting fine reconstruction is then evaluated by the discriminator.

Once we have trained our artificial adversarial neural network, we are able to perform sensor-to-screen reconstruction; see Figure 5 for full *UML* activity diagram. To perform a reconstruction, firstly, we initialize the system by loading the trained model weights and initialize the depth sensor; in our case, this is an *Intel Realsense* depth camera sensor. Once the system is ready, we retrieve a single depth frame, as we are not interested in the color information of the captured frame. Afterwards, we filter out invalid pixels by setting all pixels with depth over 2.5 m to zero. This is done to avoid irrelevant background noise. Once we have the filtered depth frame, we convert it to pointcloud using intrinsic camera parameters, and the resulting pointcloud is then filtered again by discarding any of the vertices, in which distance is zero, and flattening the input to a single dimension as the pointcloud itself is not an unstructured data structure. The filtered pointcloud is thereafter downsampled to 2048 points using *FPS* and used as an input in the refiner stage. Refiner outputs input pointcloud features along with cleaned up pointcloud that is then passed through the decoder network for object reconstruction. This gives the final output of reconstructed human mesh with the density of 2048 points.

8.4.3.2.1. Refiner

The refiner architecture is our main contribution to the field of object reconstruction. While the majority of the applications of adversarial neural networks involve generation of new samples [49, 50, 51, 52], this can be done either from random noise, hand drawn-input, etc. Very little research has focused on refining the initial input without distorting the input. While there

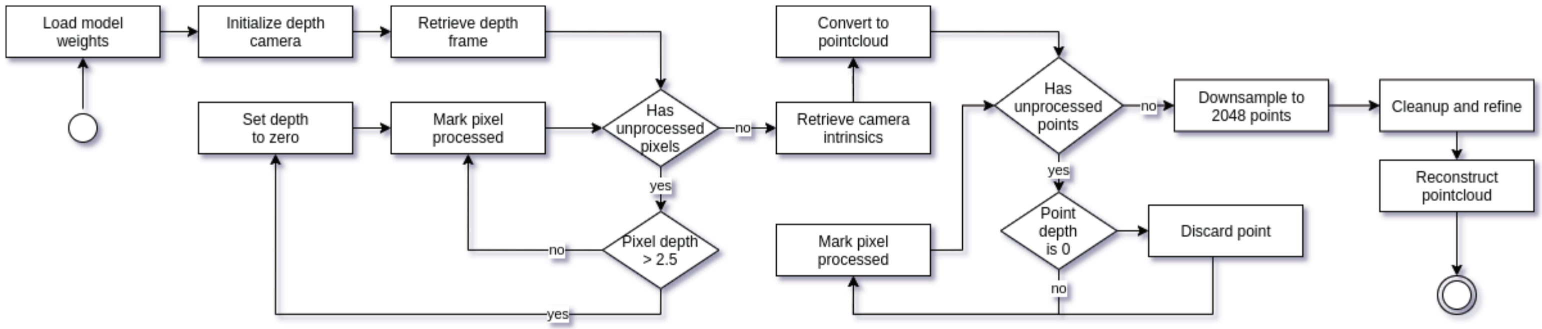


Figure 5. Activity diagram of the proposed method for human posture reconstruction.

have been attempts at refining the synthetic data in order to make it similar to synthetic [53], they still require some sort of knowledge of the given input, as in the case of Reference [53], and it is the pupil direction. Our approach, however, involves of refining real world data to make it more akin to synthetic without knowing any correlation between synthetic and real world data. The refiner network architecture can be seen in Figure 6. Our refiner architecture uses the suggested *PointNetFeat* [21] pointcloud feature extraction architecture directly connected to a fully-connected bottleneck layer of size 256, followed by batch normalization in order to improve generalization and reduce training times [54, 55], connected to non-linearity function. For our non-linearity, we use *Rectified Linear units* [56] (ReLU) for they have fast computation times and have been shown to achieve better results when compared to other non-linearity functions, such as *sigmoid*. Output feature vector is then connected a modified random grid (*RandGrid*) decoder architecture, as suggested by Liu et al. [38], using 8 primitives for reconstruction. The output pointcloud is them resampled using *Minimum Density Sampling* (MDS) in order to more evenly distribute the subset pointclouds. Our main modification to the random grid decoder consists of having the uniform distribution for initial random positions be in range of $[-0.5, 0.5]$ with the offset of input pointcloud center of mass, in addition to the points being on all three axis instead of two. This has improved the convergence by having the initial points more likely to be distributed around the reconstructed object. Resampled pointcloud is part of the two outputs provided by the refiner/encoder network. It acts as the comparison output for our discriminator network and as part of feature vectors that are used for the decoder. To obtain feature vectors from the refined pointcloud, we apply additional feature extraction block as we did with original input. This gives us two feature vectors of shape 256 that we combine into final feature vector of shape 512 that is then used in the decoders' reconstruction phases. For full refiner architecture, see Figure 7.

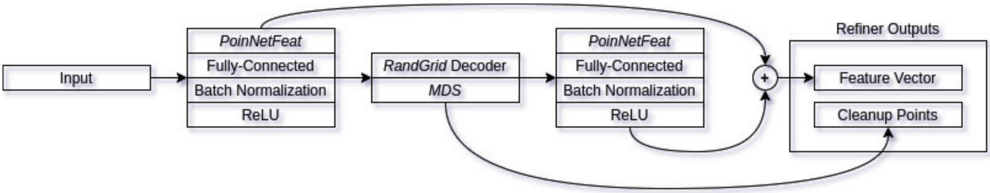


Figure 6. Refiner network architecture. For a given pointcloud, it outputs a cleaned up pointcloud, in addition to a feature vector containing a combination of both cleaned up and original feature vectors.

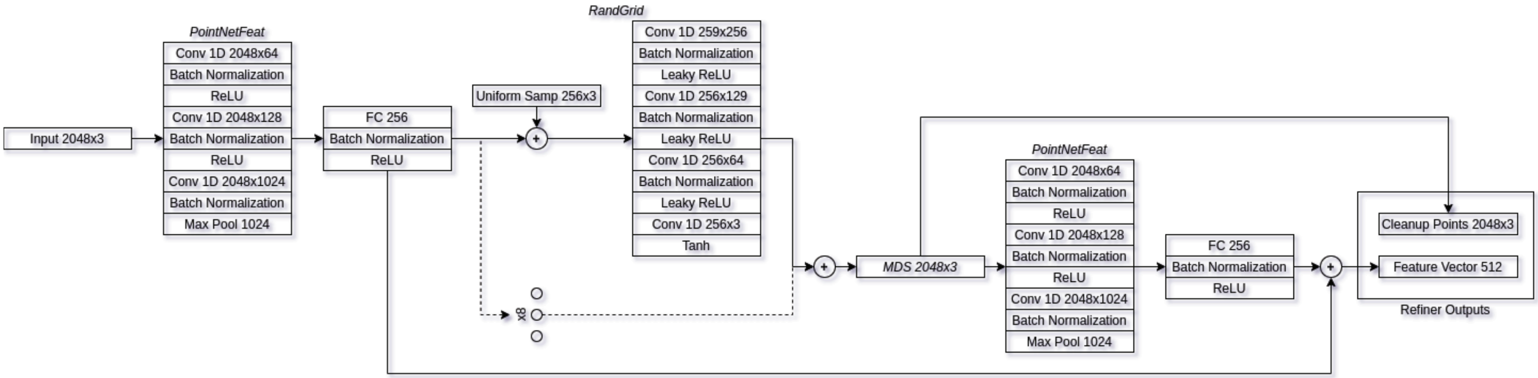


Figure 7. Full refiner network architecture.

8.4.3.2.2. Decoder

Our decoder network resembles Liu et al. [38] decoder architecture, with the main modifications being in the random grid (*RandGrid*) decoder. Unlike the paper suggested architecture, we have modified the random pointcloud to be generated on all three axes, in the range of $[-0.5, 0.5]$ with the offset of refined pointcloud center of mass. The overview of the architecture can be seen in Figure 8. The refiner output feature vector is passed to *RandGrid* decoder using 8 primitives for reconstruction, giving the reconstruction of coarse points. The output coarse pointcloud is then merged with refined pointcloud instead of the input pointcloud and resampled using minimum density sampling. Resampled pointcloud is then passed through residual decoder (*PointNetRes*) giving the final output of fine pointcloud reconstruction. For full decoder architecture, see Figure 9.

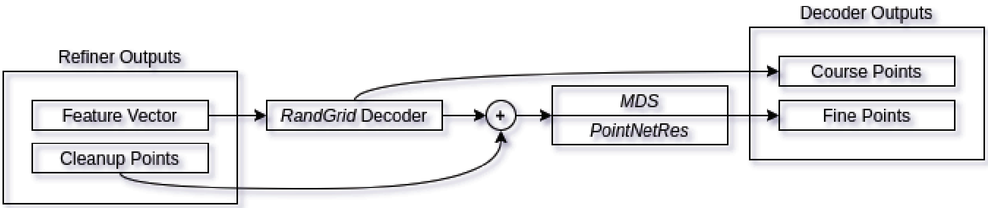


Figure 8. Decoder architecture. Refiner feature vectors are used in order to reconstruct the coarse human pointcloud. coarse features along with cleaned up features are then resampled and passed through residual block, of which output is fine-grained point reconstruction.

8.4.3.2.3. Discriminator

The main job of discriminator is to evaluate whether given input is either synthetic or real input. Our discriminator is shown in Figure 10 below. We take the output of the decoders fine pointcloud reconstruction and use that as an input in the discriminator. The inputs are passed through feature extraction block (*PointNetFeat*), which is subsequently connected to fully-connected layer. Our fully connected layer only has an output of a single neuron that is then passed through sigmoid function. The output of the sigmoid function indicates if the generated pointcloud is synthetic (1) or if it is real (0). This is done in order to make the input pointcloud as close to synthetic samples as possible as we only have ground-truths for synthetic pointclouds, thus our only being able to train the decoder on synthetic examples.

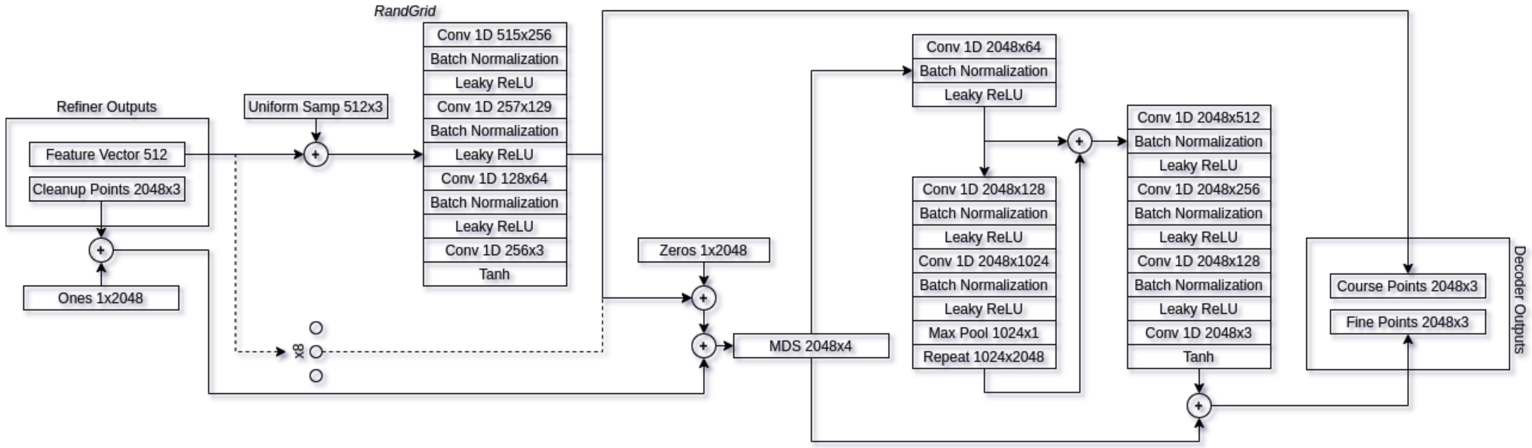


Figure 9. Full decoder network architecture.

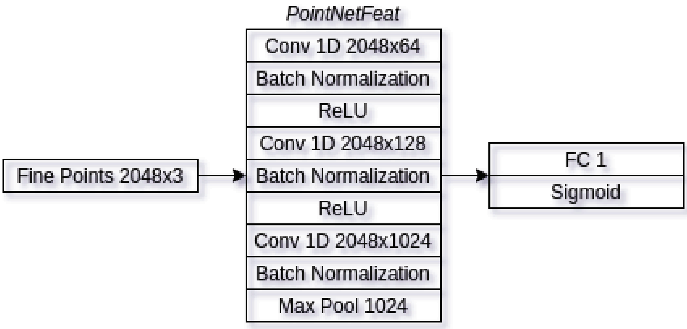


Figure 10. Discriminator architecture. Decoder fine pointcloud reconstruction is used as discriminator input. It is passed through feature extraction block. Extracted features are then connected to fully-connected layer, followed by sigmoidal function.

8.4.3.3. Training Procedure

To train our neural network, we have chosen a four phase approach: encoder training, decoder training, discriminator training, and refiner training. We have chosen four phased training approach as we have found the network to be much easier to train this way, in addition to having much better prediction results. Additionally, while training, we introduce augmentations to the synthetic input in order to try and produce *Realsense*-like depth deformities. This is done in two ways: either removing random patches from the pointcloud or by adding wavelet disturbances to the pointcloud (see Equation (2)). Wavelets have the period of $\omega = [2\pi, 32\pi]$ and the amplitude of $A = (0, 0.03]$. There is a 75% chance of pointcloud having small patches removed and 50% chance of it having wavelet deformities. These two types of augmentations are expected to be cleaned up during the cleanup stage in the refiner/encoder branch. In addition to these augmentations, there is a third type of augmentation which involves adding random scale factor to the model. This is done because real world people are different heights, while the synthetic models only have single height subjects. Height augmentation also has a 50% chance of applying height scaling in the range of $[0.8, 1.8]$.

$$p(x, y, z) = (x + A \cdot cos(\omega \cdot x), y + A \cdot sin(\omega \cdot y), z). \qquad (2)$$

Phase I During first phase, we focus on passing the best weights to the encoder to do this, in this phase we completely ignore the discriminator and ignore its outputs. Instead, we train all three of the reconstruction stages (cleanup, coarse and fine) to try and act as a regular auto-encoder by passing through input values to output with. To compare the auto-encoder result, we need a loss function capable of comparing unstructured pointcloud data. One of the most popular ones for this task is *Chamfer* distance due to its low memory impact and fast computation. However, what we found is that the use of *Chamfer* produces improper features by causing vertices to congregate close to each other instead of spreading around the desired mesh properly. Therefore, we have instead opted to use *Earth Mover's* distance (see Equation (3)) along with suggested penalization criteria (see Equation (4)) in Liu et al. [38], where $d(u, v)$ is Euclidean distance between two vertices in three-dimensional space; $\mathbb{1}$ is the indicator function used to filter which shorter than λl_i with $\lambda = 1.5$ as per suggested value.

$$EMD(S, \hat{S}) = \underset{\phi:S\rightarrow\hat{S}}{min} \frac{1}{|S|} \sum_{x\in S} ||x - \phi(x)||_2, \tag{3}$$

$$EXP(S, \hat{S}) = \frac{1}{KN} \sum_{1\leq i\leq K} \sum_{(u,v\in\tau_i)} \mathbb{1}\{d(u, v) \geq \lambda l_i\} d(u, v). \tag{4}$$

During the Phase I training, our final loss function looks like Equation (5), where $\hat{S}_{clean}$ is the result of the cleaning stage, $\hat{S}_{coarse}$ is the result of coarse stage, $\hat{S}_{fine}$ is the result of fine point reconstruction, and S_{clean} is ground truth for cleaned pointcloud, as the input pointcloud can have additional noise added to it during augmentation. As per previous research, we kept $\gamma = 0.1$ for the expansion penalty factor. The network stays in Phase I training until $\epsilon_{\Phi 1} > 0.13$; this training value was chosen during experiments as a good value to start training next phase. Once this condition is triggered, Phase II of training starts.

$$\begin{aligned} \epsilon_{\Phi 1} = EMD(S_{clean}, \hat{S}_{clean}) + EMD(S_{clean}, \hat{S}_{coarse}) + \\ EMD(S_{clean}, \hat{S}_{fine}) + \cdot \\ \gamma \cdot (EXP(S_{clean}, \hat{S}_{clean}) + EXP(S_{clean}, \hat{S}_{fine})) \end{aligned} \tag{5}$$

Phase II During this training phase, we focus on training the decoder itself; therefore, during this stage, no modifications to the network weights are applied to the refiner or discriminator branches. In addition, unlike in the previous phase, we only train on synthetic dataset. When Phase II condition is triggered for the first time, we need to apply some weight re-initialization to the network in order to clean up previous training phases potential falls into local minimums. This is done to clear all the decoder weights acquired during Phase I training. To drop the weights, we re-initialize all the decoder weights using Xavier initialization [57] and biases with uniform distribution. In addition, we drop any optimizer state that has been built up to this point for both encoder and decoder optimizers. During Phase II, only decoder weights are being modified in order to build a viable profile for both real and synthetic pointcloud reconstructions during Phase III training. Because only decoder weights are being trained during this phase, our loss equation can be simplified to Equation (6), where S_{gt} is the synthetic ground truth for the synthetic input. Phase II is trained while $\epsilon_{\Phi 2} > 0.08$; once loss drops below the threshold, Phase III training starts. Like with Phase I, the threshold value has been chosen experimentally.

$$\begin{aligned} \epsilon_{\Phi 2} = EMD(S_{gt}, \hat{S}_{coarse}) + EMD(S_{gt}, \hat{S}_{fine}) + \\ \gamma \cdot EXP(S_{clean}, \hat{S}_{fine}). \end{aligned} \tag{6}$$

Phase III During this phase, we are concerned with training the discriminator to differentiate between real and synthetic input predicted pointclouds. Training discriminator up until this point makes the weights very unstable; for this reason, training it was relegated to its own phase. The discriminator is fed output of the decoder branch fine pointcloud and the output is either 1 for synthetic dataset element or 0 for real dataset element. Therefore, as loss function, we are able to use binary cross entropy; see Equation (7) below. Discriminator is trained until $\epsilon_{\Phi 3} < 0.05$, then the final training phase can begin.

$$\epsilon_{\Phi 3} = BCE(y, \hat{y}) = \sum_{i=1}^{N} \hat{y_i} \cdot log(y_i) + (1 - \hat{y_i}) \cdot log(1 - y_i). \tag{7}$$

Phase IV During the fourth and final phase, the actual adversarial training is performed for pointcloud refinement. Because we want our training to start afresh, we drop all previous optimizer states. This helps to kick-start the training in case optimizers have built up local minima states. During the adversarial training phase, we train on both synthetic and real datasets. Synthetic dataset is used to further enforce the decoder state and to reinforce discriminator, while real data is used to update the refiner/encoder weights and to reinforce discriminator. Phase IV consists of three different steps for each batch that is being trained with the weights being updated separately. The first step of Phase IV reinforces the entire network using the synthetic dataset, for loss function for step one see Equation (8). The second step of the phase involves training refiner using adversarial loss, where the network attempts to predict such a pointcloud for real world data that the discriminator would be fooled into thinking that this is a synthetic data sample. During this step, only the refiner weights are being updated; the discriminator is left untouched, and only its loss function is used (see Equation (9)). In addition, we do not want the refiner to lose the shape of the point cloud, and, for this reason, we constrain it with pointcloud loss in relation to the input frame with a factor of $\alpha = 0.4$, the value of which was chosen experimentally. This allows the pointcloud to gain adversarial properties and actually refine the model without losing the underlying shape. And, finally, we reinforce the discriminator to recognize the real dataset pointclouds from synthetic (see Equation (10)).

$$\epsilon_{\Phi 4a} = EMD(S_{clean}, \hat{S}_{clean}) + \epsilon_{\Phi 2} + \epsilon_{\Phi 3}, \tag{8}$$

$$\begin{gathered} \epsilon_{\Phi 4b} = \alpha \cdot EMD(S_{clean}, \hat{S}_{clean}) + \\ \gamma \cdot EXP(S_{clean}, \hat{S}_{fine}) + BCE(1 - y, \hat{y}), \end{gathered} \tag{9}$$

$$\epsilon_{\Phi 4c} = \epsilon_{\Phi 3}. \tag{10}$$

8.4.4. Results

The main purpose of our approach is to reconstruct self-occluded human body shapes using real world depth sensor data. However, it is not possible to objectively measure the achieved results for real world data as we have no way of comparing ground truth with prediction, no such ground truth pointclouds exist. For this reason, we will only objectively measure the reconstruction quality using synthetic dataset, while the evaluation of the real world data will be evaluated using expert knowledge. For evaluating synthetic dataset, we will use two quality metrics *Chamfer* (see Equation (11)) and *Earth Mover's* distance. Results for synthetic dataset reconstruction are shown in Figure 11.

$$\begin{gathered} \epsilon_{cd}(S, \hat{S}) = \\ \frac{1}{2}\left(\frac{1}{|S|}\sum_{x\in S}\underset{y\in\hat{S}}{min}||x-y||_2^2 + \frac{1}{\hat{S}}\sum_{y\in\hat{S}}\underset{x\in S}{min}||x-y||_2^2\right). \end{gathered} \tag{11}$$

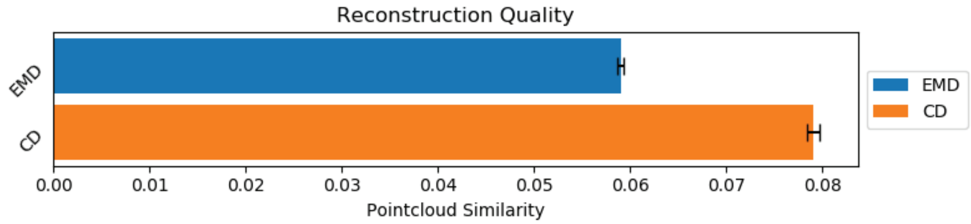


Figure 11. Reconstruction similarity using both *Earth Movers Distance* and *Chamfer Distance*. Lower values are better.

To compare our approach to other state-of-the-art research using both *Earth Mover's* and *Chamfer* distance metrics, from Table 2, we can see that the other approaches, without modifications, completely fail when attempting to deal with our *AMASS* dataset, despite having comparable results (Liu et al. [38]) when applied *ShapeNet* dataset to our results with *AMASS* dataset. Unfortunately, we cannot compare our approach against *ShapeNet* dataset as our suggested approach is meant for the synthesis of real-world object data for reconstruction. Comparing against it would require collection of a real world *ShapeNet*-like dataset using depth sensors, in addition to generation of synthetic frames.

Table 2. Comparison of different reconstruction method metrics for different reconstruction methods and ours. Our approach when applied to *AMASS* dataset has very similar metrics to state-of-the-art approaches on *ShapeNet* datasets, whereas other methods completely fail when reconstructing *AMASS* dataset. Our method is not applicable to *ShapeNet* as our data collection and trainign process is more complicated.

Method	**ShapeNet**		**AMASS**	
	EMD	**CD**	**EMD**	**CD**
PointNet w/ FCAE [21]	0.0832	0.0182	3.3806	4.9042
PCN [36]	0.0734	0.0121	3.0456	4.0955
AtlasNet [37]	0.0653	0.0182	2.0875	6.4343
MSN [38]	0.0378	0.0114	1.1525	0.8016
Our method (Cleanup)	*N/A*	*N/A*	0.0603	0.0292
Our method (Reconstruction)	*N/A*	*N/A*	0.0590	0.0790

Additionally, we inspect synthetic and real world data results visually using expert knowledge. Figure 12 depicts synthetic models input (orange) being compared side by side. As we can see, the majority of the reconstruction flaws occur at the ends of the limbs (both hands and feet) due to those features requiring much finer granularity. Alongside ground-truth to prediction side by side comparisons, we perform input-to-prediction overlap visual inspection as it helps us see to better compartmentalize what features were given as the input to neural network and what it had to make a guess.

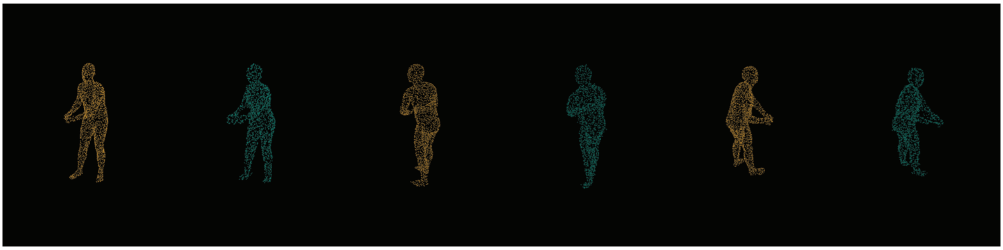


Figure 12. Comparison of ground truth (left/orange) and prediction (right/teal) from different viewpoints.

From Figure 13, we can see that, despite being given very little input (orange) about legs, our network has managed to predict (teal) the entirety of its orientation. As we can see, the network has managed to predict and reconstruct the entire human posture, while being given less than half of the body features. Finally, we compare real world data reconstruct in comparison to the input depth (see Figure 14).

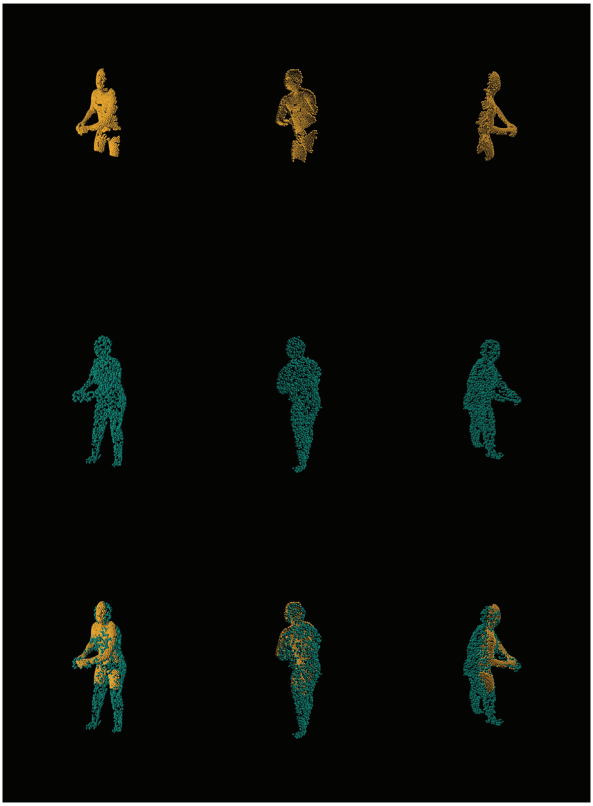


Figure 13. Stacked views for synthetic input (orange) and its prediction (teal).

As we can see, the network has had no issue in reconstructing the human posture and cleaning out the majority of deformations. The biggest defects for real world data reconstruction are where the depth has large deformities; for example, in the top row, we can see that, at the end of the hand, there is an extra lump that was captured by the depth sensor. This has caused the reconstruction prediction to fail cut off part of the hand. Additionally, there visually seem to be small scaling discrepancies between input and reconstruction, although these discrepancies are somewhat hard to evaluate, even with expert knowledge; regardless of that, we can safely assert that the addition of adversarial refinement to the network has allowed the network to understand and reconstruct real world depth sensor data, whereas other approaches have failed without. Finally, we have tried applying our approach to *ITOP* dataset. However, due to very different noise model and point distribution of *Microsoft Kinect* sensor when compared to *Intel Realsense*, the reconstruction results proved to be inconsistent. Having the model trained with *ITOP* dataset would greatly improve the results. Unfortunately, the dataset has additional background noise that we were unable to isolate to be used during the training process. Further research could improve our model by making it compatible with *Kinect*-like device noise; however, due to the *Microsoft Kinect* being a discontinued product, we feel like pursuing this is not as useful.

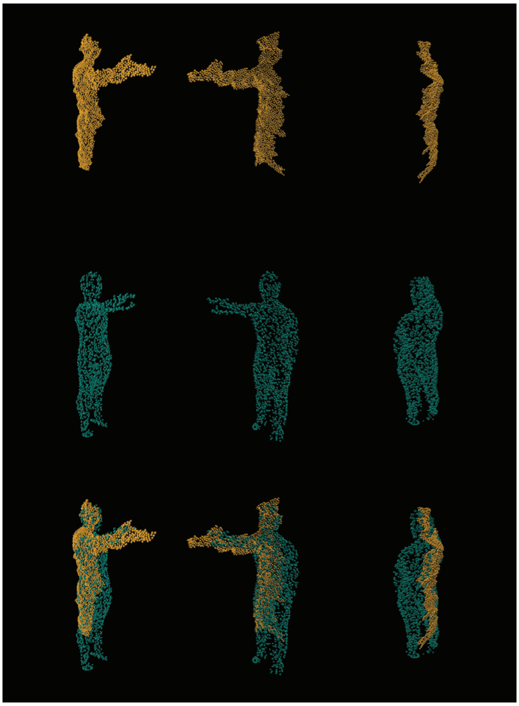


Figure 14. Stacked views for real depth sensor input (orange) and its prediction (teal). Real input is quite distorted due to depth sensor inaccuracies, and the cleanup stage manages to clean up the majority of the noise.

8.4.5. Discussion

The main of advantage over other state-of-the-art approaches involving object reconstruction, is that our three-staged neural network architecture is capable of reconstructing full human body postures with no external interference using real world depth sensor frames. The addition of the cleanup stage has allowed our approach to not only denoise the input data, but it also acted as real world input refinement, which has allowed us to train the network without actually having ground truths for it. Furthermore, our solution, unlike voxel grid-based approaches, does not require us finding the objects' transformation matrix in order to scale and translate it to place it in 3D space. Instead, it is easily adaptable to existing 3D applications, such as AR or VR. Finally, while our solution had no issue in reconstructing the general objects' shape with less than half of the object visible, finer details, like palms and fingers, tend to be too small features for reconstruction, causing ambiguities and bad reconstruction results.

8.4.6. Conclusions

We proposed three-staged adversarial auto-refining reconstruction network, which is capable of reconstructing human body using both: synthetic depth inputs and real world depth sensor data. The network has achieved *Earth Mover's* and *Chamfer* distances of 0.059 and 0.079, respectively, indicating good reconstruction quality, when compared to other state-of-the-art methods. Additionally, when comparing the reconstructed pointclouds visually, it is clear that the network manages to meet the expectations of reconstructing both synthetic and real world samples with most of the defects being concentrated at the ends of the limbs, or in the case of real world data, large defects in the depth map can cause defects in the reconstruction even when cleaning. This is likely due to constraints of refiner attempting to retain the original shape of the object.

Finally, we have proposed a four-phased training approach for training the adversarial auto-refiner. The addition of adversarial refinement to the network has allowed our approach to work with real-world depth sensor data, which other approaches are unable to do.

Author Contributions Conceptualization, R.M.; methodology, R.M.; software, A.K.; validation, A.K., R.M., R.D., and M.W.-S.; formal analysis, R.M. and R.D.; investigation, A.K. and R.M.; resources, R.M.; data curation, A.K.; writing—original draft preparation, A.K. and R.M.; writing—review and editing, R.D. and M.W.-S.; visualization, A.K. and R.M.; supervision, R.M.; funding acquisition, M.W.-S. All authors have read and agreed to the published version of the manuscript.

Funding This research received no external funding.

Institutional Review Board Statement Not applicable.

Informed Consent Statement Not applicable.

Data Availability Statement The data is available from the corresponding author upon reasonable request.

Conflicts of Interest The authors declare no conflict of interest.

References

1. J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. “Real-time human pose recognition in parts from single depth images”. In: *CVPR 2011*. 2011, pp. 1297–1304. doi: `10.1109/CVPR.2011.5995316`.
2. G. Bozgeyikli, E. Bozgeyikli, and V. Işler. “Introducing tangible objects into motion controlled gameplay using Microsoft® Kinect™”. In: *Computer Animation and Virtual Worlds* 24.3-4 (2013), pp. 429–441.
3. R. M. Lozada, L. R. Escriba, and F. T. Molina Granja. “MS-Kinect in the development of educational games for preschoolers”. In: *International Journal of Learning Technology* 13.4 (2018), pp. 277–305.
4. F. Cary, O. Postolache, and P. S. Girao. “Kinect based system and serious game motivating approach for physiotherapy assessment and remote session monitoring”. In: *International Journal on Smart Sensing and Intelligent Systems* 7.5 (2014).
5. K. Ryselis, T. Petkus, T. Blažauskas, R. Maskeliūnas, and R. Damaševičius. “Multiple Kinect based system to monitor and analyze key performance indicators of physical training”. In: *Human-centric Computing and Information Sciences* 10.1 (2020).
6. S. Camalan, G. Sengul, S. Misra, R. Maskeliūnas, and R. Damaševičius. “Gender detection using 3d anthropometric measurements by kinect”. In: *Metrology and Measurement Systems* 25.2 (2018), pp. 253–267.
7. F. Lourenco and H. Araujo. “Intel realsense SR305, D415 and L515: Experimental evaluation and comparison of depth estimation”. In: *VISIGRAPP 2021 - Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. Vol. 4. 2021, pp. 362–369.
8. Jingtian Zhang, Hubert P. H. Shum, Kevin McCay, and Edmond S. L. Ho. “Prior-less 3D Human Shape Reconstruction with an Earth Mover’s Distance Informed CNN”. In: *Motion, Interaction and Games on - MIG19*. ACM Press, 2019. doi: `10.1145/3359566.3364694`.
9. S. Jacob, V. G. Menon, and S. Joseph. “Depth Information Enhancement Using Block Matching and Image Pyramiding Stereo Vision Enabled RGB-D Sensor”. In: *IEEE Sensors Journal* 20.10 (2020), pp. 5406–5414.
10. Alberto Díaz-Álvarez, Miguel Clavijo, Felipe Jiménez, and Francisco Serradilla. “Inferring the Driver’s Lane Change Intention through LiDAR-Based Environment Analysis Using Convolutional Neural Networks”. In: *Sensors* 21.2 (2021). issn: 1424-8220. doi: `10.3390/s21020475`. online: `https://www.mdpi.com/1424-8220/21/2/475`.
11. Melissa Latella, Fabio Sola, and Carlo Camporeale. “A Density-Based Algorithm for the Detection of Individual Trees from LiDAR Data”. In: *Remote Sensing* 13.2 (2021). issn: 2072-4292. doi: `10.3390/rs13020322`. online: `https://www.mdpi.com/2072-4292/13/2/322`.
12. Maria João Sousa, Alexandra Moutinho, and Miguel Almeida. “Thermal Infrared Sensing for Near Real-Time Data-Driven Fire Detection and Monitoring Systems”. In: *Sensors* 20.23 (2020). issn: 1424-8220. doi: `10.3390/s20236803`. online: `https://www.mdpi.com/1424-8220/20/23/6803`.
13. Javier Pérez, Mitch Bryson, Stefan B. Williams, and Pedro J. Sanz. “Recovering Depth from Still Images for Underwater Dehazing Using Deep Learning”. In: *Sensors* 20.16 (2020). issn: 1424-8220. doi: `10.3390/s20164580`. online: `https://www.mdpi.com/1424-8220/20/16/4580`.
14. W. Ren, O. Ma, H. Ji, and X. Liu. “Human Posture Recognition Using a Hybrid of Fuzzy Logic and Machine Learning Approaches”. In: *IEEE Access* 8 (2020), pp. 135628–135639. doi: `10.1109/ACCESS.2020.3011697`.
15. Audrius Kulikajevas, Rytis Maskeliunas, and Robertas Damaševičius. “Detection of sitting posture using hierarchical image composition and deep learning”. In: *PeerJ Computer Science* 7 (2021). doi: `10.7717/peerj-cs.442`.

16. Bert Coolen, Peter J. Beek, Daphne J. Geerse, and Melvyn Roerdink. “Avoiding 3D Obstacles in Mixed Reality: Does It Differ from Negotiating Real Obstacles?” In: *Sensors* 20.4 (2020), p. 1095. doi: `10.3390/s20041095`.

17. Bruno Fanini, Alfonsina Pagano, and Daniele Ferdani. “A Novel Immersive VR Game Model for Recontextualization in Virtual Environments: The uVRModel”. In: *Multimodal Technologies and Interaction* 2.2 (Apr. 2018), p. 20. doi: `10.3390/mti2020020`. online: `https://doi.org/10.3390/mti2020020`.

18. Alex Ibañez-Etxeberria, Cosme J. Gómez-Carrasco, Olaia Fontal, and Silvia García-Ceballos. “Virtual Environments and Augmented Reality Applied to Heritage Education. An Evaluative Study”. In: *Applied Sciences* 10.7 (2020). issn: 2076-3417. doi: `10.3390/app10072352`. online: `https://www.mdpi.com/2076-3417/10/7/2352`.

19. Åsa Fast-Berglund, Liang Gong, and Dan Li. “Testing and validating Extended Reality (xR) technologies in manufacturing”. In: *Procedia Manufacturing* 25 (2018). Proceedings of the 8th Swedish Production Symposium (SPS 2018), pp. 31–38. issn: 2351-9789. doi: `https://doi.org/10.1016/j.promfg.2018.06.054`. online: `http://www.sciencedirect.com/science/article/pii/S2351978918305730`.

20. Haoqiang Fan, Hao Su, and Leonidas Guibas. “A Point Set Generation Network for 3D Object Reconstruction from a Single Image”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017). doi: `10.1109/cvpr.2017.264`.

21. R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 77–85. doi: `10.1109/CVPR.2017.16`.

22. Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. “3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016.

23. Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. “Deep Metric Learning via Lifted Structured Feature Embedding”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

24. Angel X. Chang, Thomas A. Funkhouser, Leonidas J Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. “ShapeNet: An Information-Rich 3D Model Repository”. In: *CoRR* abs/1512.03012 (2015).

25. K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. “LSTM: A Search Space Odyssey”. In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (2017), pp. 2222–2232. doi: `10.1109/TNNLS.2016.2582924`.

26. W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang. “Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network”. In: *IEEE Transactions on Smart Grid* 10.1 (2019), pp. 841–851. doi: `10.1109/TSG.2017.2753802`.

27. Audrius Kulikajevas, Rytis Maskeliūnas, Robertas Damaševičius, and Edmond S. L. Ho. “3D Object Reconstruction from Imperfect Depth Data Using Extended YOLOv3 Network”. In: *Sensors* 20.7 (2020). issn: 1424-8220. doi: `10.3390/s20072025`. online: `https://www.mdpi.com/1424-8220/20/7/2025`.

28. Joseph Redmon and Ali Farhadi. “YOLOv3: An Incremental Improvement”. In: *CoRR* abs/1804.02767 (2018). online: `http://arxiv.org/abs/1804.02767`.

29. Audrius Kulikajevas, Rytis Maskeliūnas, Robertas Damaševičius, and Sanjay Misra. “Reconstruction of 3D Object Shape Using Hybrid Modular Neural Network Architecture Trained on 3D Models from ShapeNetCore Dataset”. In: *Sensors* 19.7 (2019). issn: 1424-8220. doi: `10.3390/s19071553`. online: `https://www.mdpi.com/1424-8220/19/7/1553`.

30. M. Tatarchenko, A. Dosovitskiy, and T. Brox. “Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2107–2115. doi: `10.1109/ICCV.2017.230`.
31. Z. Mi, Y. Luo, and W. Tao. “SSRNet: Scalable 3D Surface Reconstruction Network”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 967–976. doi: `10.1109/CVPR42600.2020.00105`.
32. Tingsong Ma, Ping Kuang, and Wenhong Tian. “An improved recurrent neural networks for 3d object reconstruction”. In: *Applied Intelligence* (2019). doi: `10.1007/s10489-019-01523-3`.
33. X. Xu, F. Lin, A. Wang, Y. Hu, M. Huang, and W. Xu. “Body-Earth Mover’s Distance: A Matching-Based Approach for Sleep Posture Recognition”. In: *IEEE Transactions on Biomedical Circuits and Systems* 10.5 (2016), pp. 1023–1035. doi: `10.1109/TBCAS.2016.2543686`.
34. W. Wu, Z. Qi, and L. Fuxin. “PointConv: Deep Convolutional Networks on 3D Point Clouds”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9613–9622. doi: `10.1109/CVPR.2019.00985`.
35. Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *CoRR* abs/1612.00593 (2016). arXiv: `1612.00593`. online: `http://arxiv.org/abs/1612.00593`.
36. Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. “PCN: Point Completion Network”. In: *CoRR* abs/1808.00671 (2018). arXiv: `1808.00671`. online: `http://arxiv.org/abs/1808.00671`.
37. Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. “AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation”. In: *CoRR* abs/1802.05384 (2018). arXiv: `1802.05384`. online: `http://arxiv.org/abs/1802.05384`.
38. Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. “Morphing and Sampling Network for Dense Point Cloud Completion”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.07 (Apr. 2020), pp. 11596–11603. doi: `10.1609/aaai.v34i07.6827`. online: `https://ojs.aaai.org/index.php/AAAI/article/view/6827`.
39. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. “Microsoft COCO: Common Objects in Context”. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars. Cham: Springer International Publishing, 2014, pp. 740–755. isbn: 978-3-319-10602-1.
40. Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. “The Pascal Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338. issn: 1573-1405. doi: `10.1007/s11263-009-0275-4`. online: `https://doi.org/10.1007/s11263-009-0275-4`.
41. Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. “ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes”. In: *CoRR* abs/1702.04405 (2017). arXiv: `1702.04405`. online: `http://arxiv.org/abs/1702.04405`.
42. Albert Haque, Boya Peng, Zelun Luo, Alexandre Alahi, Serena Yeung, and Li Fei-Fei. “ITOP Dataset”. Version 1.0. In: (Oct. 2016). doi: `10.5281/zenodo.3932973`. online: `https://doi.org/10.5281/zenodo.3932973`.
43. Varun Ganapathi, Christian Plagemann, Daphne Koller, and Sebastian Thrun. “Real-Time Human Pose Tracking from Range Data”. In: *Computer Vision – ECCV 2012*. Ed. by Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia

Schmid. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 738–751. isbn: 978-3-642-33783-3.

44. Steffen Flaischlen and Gregor D. Wehinger. “Synthetic Packed-Bed Generation for CFD Simulations: Blender vs. STAR-CCM+”. In: *ChemEngineering* 3.2 (2019). issn: 2305-7084. doi: `10.3390/chemengineering3020052`. online: `https://www.mdpi.com/2305-7084/3/2/52`.
45. Saeed Ghorbani, Kimia Mahdaviani, Anne Thaler, Konrad Kording, Douglas James Cook, Gunnar Blohm, and Nikolaus F. Troje. *MoVi: A Large Multipurpose Motion and Video Dataset*. 2020. arXiv: `2003.01888 [cs.CV]`.
46. Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. “AMASS: Archive of Motion Capture as Surface Shapes”. In: *International Conference on Computer Vision*. Oct. 2019, pp. 5442–5451.
47. Franz Kainz, Rebecca R. Bogart, and David K. Hess. “The OpenEXR image file format”. In: 2004.
48. Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 5105–5114. isbn: 9781510860964.
49. Yanghua Jin, Jiakai Zhang, Minjun Li, Yingtao Tian, Huachun Zhu, and Zhihao Fang. “Towards the Automatic Anime Characters Creation with Generative Adversarial Networks”. In: *ArXiv* abs/1708.05509 (2017).
50. Tero Karras, Samuli Laine, and Timo Aila. “A Style-Based Generator Architecture for Generative Adversarial Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.
51. Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. “Image-to-Image Translation with Conditional Adversarial Networks”. In: *CVPR* (2017).
52. Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. 2020. arXiv: `1703.10593 [cs.CV]`.
53. C. Atapattu and B. Rekabdar. “Improving the realism of synthetic images through a combination of adversarial and perceptual losses”. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. 2019, pp. 1–7. doi: `10.1109/IJCNN.2019.8852449`.
54. K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. “Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising”. In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3142–3155. doi: `10.1109/TIP.2017.2662206`.
55. S. Wu, G. Li, L. Deng, L. Liu, D. Wu, Y. Xie, and L. Shi. “$L1$ -Norm Batch Normalization for Efficient Training of Deep Neural Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* 30.7 (2019), pp. 2043–2051. doi: `10.1109/TNNLS.2018.2876179`.
56. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034. doi: `10.1109/ICCV.2015.123`.
57. T. Gao, Y. Chai, and Y. Liu. “Applying long short term momory neural networks for predicting stock closing price”. In: *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. 2017, pp. 575–578. doi: `10.1109/ICSESS.2017.8342981`.

9. CURRICULUM VITAE

Personal information

Audrius Kulikajevas
Date of birth: 24 March 1993
Email address: audrius.kulikajevas@ktu.edu

Institution

Kaunas University of Technology
Faculty of Informatics
Department of Multimedia Engineering
K. Baršausko g. 59, A336, LT-51423, Kaunas, Lithuania

Education

2017-2021	Studied in Informatics doctoral programme at Kaunas University of Technology, Faculty of Informatics, Department of Multimedia Engineering, Kaunas, Lithuania.
2015-2017	Was awarded Software Systems Engineering Master's degree at Kaunas University of Technology, Faculty of Informatics, Department of Software Engineering, Kaunas, Lithuania.
2011-2015	Was awarded Software Systems Bachelor's degree at Kaunas University of Technology, Faculty of Informatics, Department of Software Engineering, Kaunas, Lithuania.

Research interests

Machine learning
Computer graphics
Software engineering

Work experience

2015-present — senior software developer at Indeform, Ltd.

10. LIST OF SCIENTIFIC PAPERS AND CONFERENCES

Web of Science and Scopus database journals with the impact factor:

1. **Kulikajevas, Audrius**; Maskeliunas, Rytis; Damasevicius, Robertas. Adversarial 3D human pointcloud completion from limited angle depth data // IEEE sensors journal. Piscataway, NJ : IEEE. ISSN 1530-437X. eISSN 1558-1748. 2021, vol. 00, iss. 00, p. 1-8. DOI: 10.1109/JSEN.2021.3124451. Author's input: 0.334

2. **Kulikajevas, Audrius**; Maskeliūnas, Rytis; Damaševičius, Robertas. Detection of sitting posture using hierarchical image composition and deep learning // PeerJ computer science. London : PeerJ. ISSN 2376-5992. 2021, vol. 7, art. No. e442, p. 1-20. DOI: 10.7717/peerj-cs.442. Author's input: 0.334.

3. **Kulikajevas, Audrius**; Maskeliūnas, Rytis; Damaševičius, Robertas; Scherer, Rafal. HUMANNET—a two-tiered deep neural network architecture for self-occluding humanoid pose reconstruction // Sensors. Basel : MDPI. ISSN 1424-8220. 2021, vol. 21, iss. 12, art. No. 3945, p. 1-16. DOI: 10.3390/s21123945. Author's input: 0.250.

4. **Kulikajevas, Audrius**; Maskeliūnas, Rytis; Damaševičius, Robertas; Wlodarczyk-Sielicka, Marta. Auto-refining reconstruction algorithm for recreation of limited angle humanoid depth data // Sensors. Basel : MDPI. ISSN 1424-8220. 2021, vol. 21, iss. 11, art. No. 3702, p. 1-17. DOI: 10.3390/s21113702. Author's input: 0.250.

5. **Kulikajevas, Audrius**; Maskeliūnas, Rytis; Damaševičius, Robertas; S.L. Ho, Edmond. 3D object reconstruction from imperfect depth data using extended YOLOv3 network // Sensors. Basel : MDPI. ISSN 1424-8220. 2020, vol. 20, iss. 7, art. No. 2025, p. 1-28. DOI: 10.3390/s20072025. Author's input: 0.500.

6. **Kulikajevas, Audrius**; Maskeliūnas, Rytis; Damaševičius, Robertas; Misra, Sanjay. Reconstruction of 3D object shape using hybrid modular neural network architecture trained on 3D models from ShapeNetCore dataset // Sensors. Basel : MDPI. ISSN 1424-8220. 2019, vol. 19, iss. 7, art. No. 1553, p. 1-21. DOI: 10.3390/s19071553. Author's input: 0.400.

National and international conferences:

1. **Kulikajevas, Audrius**; Maskeliūnas, Rytis; Damaševičius, Robertas; Griškevičius, Julius; Daunoravičienė, Kristina; Žižienė, Jurgita; Lukšys, Donatas; Adomavičienė, Aušra; Exercise Abnormality Detection Using BlazePose Skeleton Reconstruction // ICCSA 2021: Computational Science and Its Applications, Cagliari, Italy, 13-16 September 2021. DOI: 10.1007/978-3-030-86976-2_7. p. 90-104. Author's input: 0.125.
2. **Kulikajevas, Audrius**; Maskeliūnas, Rytis; Damaševičius, Robertas; Woźniak, Marcin. Reconstruction algorithm of invisible sides of a 3D object for depth scanning systems of a 3D object for cost effective truncation of point cloud data // ECOS 2019: Proceedings of the 32nd international conference on efficiency, cost, optimization, simulation and environmental impact of energy systems, Wrocław, Poland, 23-28 June 2019 / edited by: Wojciech Stanek, Paweł Gładysz, Sebastian Werle, Wojciech Adamczyk. Gliwice : Institute of Thermal Technology Silesian University of Technology, 2019. ISBN 9788361506515. p. 4505-4507. Author's input: 0.250.
3. Bhandari, Sandeepak; **Kulikajevas, Audrius**. Ontology based image recognition: a review // CEUR workshop proceedings : IVUS 2018: proceedings of the international conference on information technologies, Kaunas, Lithuania, April 27, 2018 / edited by G. Capizzi, R. Damaševičius, A. Lopata, T. Krilavičius, Ch. Napoli, M. Woźniak. Aachen : CEUR-WS. eISSN 1613-0073. 2018, vol. 2145, p. 13-18. Author's input: 0.500.

2018 June 28th – July 6th participated in international doctoral summer school in Dortmund university, Germany. The course was rated 2 ECTS credits.

Appendix A. ABBREVIATIONS AND TERMS

Object reconstruction/completion — a type of method which performs attempts to fill in the missing object information that was occluded either by another object or by itself, e.g., self-occluded back side of the object;

SLAM — simultaneous localization and mapping;

Point cloud — an unstructured data type used for three-dimensional object representation using points in space;

Point cloud fusion — method based on the process in which multiple point clouds from several perspectives are fused into one final point cloud result. It is generally used in combination with SLAM type algorithms;

Voxel — volumetric pixel. A type of a data point that represents a point in three-dimensional space that has a defined volume;

Voxel grid — a structured data type used for three-dimensional object representation using volumetric points in space;

EMD — Earth Mover's distance;

CD — Chamfer distance;

Unsupervised learning — a type of machine learning algorithms which attempt to analyze an unlabeled dataset in order to find the hidden patterns without additional human intervention;

Adversarial learning — a type of machine learning algorithms where two separate machine learning algorithms, generally neural networks, attempt to deceive each other;

Convex hull — represents the smallest convex shape that can encompass a given three-dimensional object;

Temporally shifting/morphing — objects that can change in the dimension of time;

LSTM — long short-term memory;

Generative Neural Networks — a type of neural network which is capable of generating new data samples from a given input, generally a random noise;

GANN — generative adversarial neural networks;

Object mask — an overlay that when combined with the original image would only segment/extract only the object contained within the masked region;

Intrinsic matrix — a matrix which describes a pinhole type camera or sensors, e.g., focal point, skewing, center offset;

Latent feature vector — a vector describing the result of a neural network's hidden layer;

ReLU — rectified linear unit;

Dead neurons — neurons which for any given input always output a zero, generally a downside found in the rectified linear unit type activation function using neurons;

FC — fully-connected;

Auto-encoder — a type of a machine learning approach where an input is passed through a bottleneck while retaining the output value very similar to that of the input;

GPU — graphics processing unit;

TPU — tensor processing unit;

RoI — region of interest;

Bounding box — a region in two-dimensional or three-dimensional space that describes the minimum object bounds;

FPS — farthest point sampling;

Residual connection — a type of neural network connection that allows the gradient flow to skip connections without having to flow through entire network during back-propagation;

EXP — expansion penalty;

FPS — frames per second.

Appendix B. SYMBOLS AND NOTATION

$\triangleright$ — comment;
$\varnothing$ — empty set;
$\leftarrow$ — value assignment;
$\in$ — in set;
$\textcircled{?}$ — branch selection;
$\otimes$ — binary mask multiplication;
$\oplus$ — residual connection;
$\neg$ — negate;
$\wedge$ — logical and.

UDK 004.932+004.85](043.3)

SL 344. 2022-02-25, 29,25 leidyb. apsk. l. Tiražas 14 egz. Užsakymas 48.
Išleido Kauno technologijos universitetas, K. Donelaičio g. 73, 44249 Kaunas
Spausdino leidyklos „Technologija“ spaustuvė, Studentų g. 54, 51424 Kaunas