IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 October 2021, accepted 5 December 2021, date of publication 19 August 2022, date of current version 24 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3143467

Auto-Refining 3D Mesh Reconstruction Algorithm From Limited Angle Depth Data

AUDRIUS KULIKAJEVAS[1], RYTIS MASKELIUNAS[1], ROBERTAS DAMASEVICIUS[2], AND TOMAS KRILAVICIUS[2]

[1]Department of Multimedia Engineering, Kaunas University of Technology, 44249 Kaunas, Lithuania
[2]Department of Applied Informatics, Vytautas Magnus University, 44248 Kaunas, Lithuania

Corresponding author: Robertas Damasevicius (robertas.damasevicius@vdu.lt)

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by the Institutional Review Board, Faculty of Informatics, Kaunas University of Technology under Approval No. IFEP201809-2.

ABSTRACT 3D object reconstruction is a very rapidly developing field, especially from a single perspective. Yet the majority of modern research is focused on developing algorithms around a single static object reconstruction and in most of the cases derived from synthetically generated datasets, failing or at least working insufficiently accurately in real-world data scenarios, regarding morphing the 3D object's restoration from a deficient real world frame. For solving that problem, we introduce an extended version of the three-staged deep auto-refining adversarial neural network architecture that can denoise and refine real-world depth sensor data current methods for a full human body pose reconstruction, in both *Earth Mover's (0.059)* and *Chamfer (0.079)* distances. Visual inspection of the reconstructed point-cloud proved future adaptation potential to most of depth sensor noise defects for both structured light depth sensors and *LiDAR* sensors.

INDEX TERMS Human shape reconstruction, pointcloud reconstruction, adversarial auto-refinement.

I. INTRODUCTION

Modern deep learning approaches pushed the research field of computer vision beyond any expectation. With most of the research focusing on adapting visible light range sensors for solving object recognition problems, has attracted much less research interest in the analysis of depth sensor data. Past generation depth sensors, like *Microsoft Kinect* [1] and *Intel Realsense* [2], have already been used to tackle different depth recognition tasks, yet consumer adoption beyond entertainment is still very rare [3], [4], and one of the more common applications is medical data processing, rehabilitation or physiotherapy [5]–[7]. Other real-world applications range from such medical purposes as posture recognition [8]–[10], lymph-edema intervention [11], respiration abnormality tracking [12], to the identification of breast cancer [13]. Furthermore researchers work on adopting such data to various forms of robotics [14], [15], collision avoidance for autonomous vehicles [16], [17]. Depth data is also used for entertainment purposes, the enhancement of VR experiences, like avoiding tripping over obstacles in the real world when wearing VR glasses [18], [19], reconstruction of environments in augmented reality [20]. Another common scenario is improving the experience of extended reality applications [21]. One of the main drawbacks regarding 3D reconstruction is the required computational and data burden for complete object reconstruction, which generally requires having a batch of precisely calibrated sensors or having the object or the whole filming setup rotated, until the entire object is completely scanned. Such limitation makes complete object reconstruction unfeasible for practical use, exacerbated by the high price of arranging complex multisensor setups and the need of high-end GPU grids to process and fuse a range of multiple pointclouds acquired from different camera arrays. Even if some of the existing limitations can be solved by moving away from arrays of monocular cameras, for example, by installing stereoscopic cameras and other depth sensor-based solutions [22], [23], invisible sides of the object might remain. This assertion is supported by other research attempting to achieve a similar task, by attempting to reconstruct three-dimensional objects using only a single perspective frame. Most of the researchers use black-box models with preexisting knowledge to solve this problem, adapting a variant of machine

The associate editor coordinating the review of this manuscript and approving it for publication was Paolo Crippa.

learning, specifically deep neural networks, which have proven themselves to be capable of leveraging reinforcement learning to approximate the occluded object shape. This type of object prediction is not dissimilar from the way a person is capable of inspecting an object from a single perspective and building an understanding of how the occluded parts of the object look based on the mental model developed using their own life experience. Several solutions already exist which have this type of predictive capability, with one of the most popular solutions for object reconstruction being volumetric pixel (voxel) based reconstruction [24]. These are relatively easy to use in training deep learning algorithms because of their loss function properties, but unfortunately the higher fidelity model using voxel approach in most cases will be constructed from very large amounts of data, requiring a lot of RAM to process the voxel grid, making the whole process very resource intensive. Alternative solution is using pointclouds as computing these data objects requires much less memory and often contains none or very little overhead for the representation of an object. The disadvantage of this data model is the unordered makeup of the pointcloud vertices, which leads to problems in training such a network, as every vertex of a pointcloud can fill any coordinate in the three-dimensional space, while one of the main reconstruction failures is multiple point crowding in a single location. *PointOutNet* was the first well adopted solution [25], capable of predicting plausible 3D object shapes, relying on hand-drawn segmentation masks to isolate the reconstructed object and still being not very practical for use in real-time applications. This approach also required using monocular images, which did not have any specific depth information and forced the model to depend on unstable depth information depending on material characteristics and ambient lightning settings. Other researchers tried leveraging pointclouds as inputs to improve generalization and predictive abilities [26], yet such models were fully dependent on artificial or manually preprocessed real world data, making them inadequate for real-life use.

To solve these problems, we extend our work presented in [27] and propose an extended unsupervised adversarial auto-refiner model that is capable of reconstructing a full human body posture pointcloud while employing only an individual self-occluding depth view from a depth sensor with no additional masking or other external information. The paper is structured as follows. Section II describes the related work. Section III explains our synthetic and real world datasets and the suggested methodology. Section IV demonstrates the results. Section V discusses the implications of this study. Section VI presents the conclusions, and Section VII discusses future work.

II. RELATED WORK

This section focuses on analysing the main methods for 3D object reconstruction: voxel based methods and pointcloud based methods. Both of which at the end can be transformed into 3D mesh for using in common computer graphics interfaces, making the primary data representation for real-life application immaterial. *3D-R2N2* [28] is one of the pioneers in the field of object reconstruction, it uses voxel-based method to restore the object from a single view or from multiple perspectives. 3D-R2N2 uses *Sanford Online Products* [29] and *ShapeNet* [30] datasets as a priori knowledge for the training of neural network by revealing its recurrent layers, specifically *Long Short Term Memory* [31], [32] (LSTM), to multiple perspectives of the same object, this makes the network able of reconstructing objects from both single and multiple perspective frames. Thanks in large part to this paper, *ShapeNet* has become the go-to benchmark to evaluate the effectiveness of a given object reconstruction approach. While the network is capable of reconstructing from a single perspective, it requires supplementary masking information to segregate the *Region-of-Interest* (RoI) that the object is rebuilt from, thus making it not practical in real-world scenarios. To solve for this, one of the methods suggests the addition of object classification task using extended *YoloV3* [33] network by merging the object reconstruction and prediction tasks (*YoloExt*) [34]. The result does not depend upon outside interference in contrast to *3D-R2N2* by creating the masks on its own accord. Other methods involve hybridized neural networks [35] for much faster model convergence by training on look-alike shaped objects in batches and the ability to reconstruct the voxel grid in real-time due to reduced network complexity. Whereas the voxel based methods have been validated to be efficient in terms of object reconstruction, they have a big weak spot that far outpaces their loss function simplicity — high coarse models need a high fidelity grid that exponentially expands the memory requirements for voxel grid representation. Some attempts have tried compacting the voxel data in more memory efficient data structures such as octrees [36], [37], greatly decreasing the size of the required data to store the same model, but they still are affected by high memory overheads, making them unworkable for use on modern hardware.

A more compressed 3D volume representation when compared to voxels is pointcloud. It has a very small memory requirement in both training and prediction steps, while admitting for much higher fidelity information to be performed by reconstructing the object shell only. Nevertheless, the training of unordered pointclouds has proven to be complicated due to the complexity and of the loss function to match the ground truth and prediction. Moreover, any grid vertex can occupy any spatial position in the three-dimensional space, thus training them using optimizers, such as stochastic gradient descent, is very vulnerable to the initial conditions and convergence of multiple vertices to a single spatial point. One of the first solutions to these problems was *PointOutNet*. As with *3D-R2N2* it requires an external mask to be provided as an input for object reconstruction, however, unlike previous state-of-the-art approaches involving voxels it was able to reconstruct unstructured pointcloud. In the paper both *Chamfer* [38] and *Earth Mover's* [39] distances

(CD and EMD) were suggested as loss metrics. However, the latter while providing much more uniform vertex spread requires high computational cost, thus was only used as an evaluation metric in subsequent research. Further pointcloud reconstruction research rather than concentrating on RGB frames (which do not have depth information), attempted to use pointclouds as inputs [40], [41]. One of the major drawbacks when using unstructured pointclouds as model inputs is the fact that you can not adopt well known feature extracting convolutional networks, as both 2D and 3D convolution kernels demand adjacent data points to be correlated, which can not be satisfied by unordered pointclouds. To tackle this issue, *PointNet* uses symmetric function for local learning, where the encoder layer of the *PointNet* is linked to a fully-connected layer it was able to fill in missing chunks of the malformed pointclouds. The study [42] suggested fine-grained pointcloud completion method, where *PCN* manages to reduce the number of latent parameters during training by carrying-out coarse-to-fine reconstruction. First, it reconstructs the coarse object features and then uses coarse output together and the residual features for finer reconstruction of a given object. Another study proposed instead opting for patch-based reconstruction. As a result, *AtlasNet* [43] is capable of mapping 2D features into parametric 3D object groups. As *Earth Mover's* distance has a complexity of $O(n^2)$ it was generally not used as a loss metric and instead an evaluation metric. Together with evenly distributed point-subsampling method *MSN* [44] an EMD approximation has been proposed.

We believe that one of the most important feature is the ability of performing sensor-to-screen prediction, where the solution is capable of predicting the full object shape with no further information provided, leading to the main benefit of our approach as the only other standalone solution able of performing sensor-to-screen prediction (requiring no external "help" in data processing) is YoloExt. Further on, our solution was designed to use EMD as a loss function for our ground truth comparison in addition to PointOutNet and MSN. Finally, we can maintain sensitivity to high density distributions, proving another novelty factor to our approach. The comparison of data versus processing is offered in Table 1.

Object surface is another important field of study with methods ranging from classical algorithms [45], [46] to deep neural network-based ones [47]–[49], with state-of-the-art being *Points2Mesh* [50] where *Poisson* sampling reinforced deep learning computerized model is used for object reconstruction. However, the latter does not support noisy inputs.

We summarize the 3D object reconstruction datasets in Table 2. Note that the ITOP [51] and EVAL [52] datasets were recorded using Kinect v1 sensor, which was discontinued. HHOPE [53] dataset is small and incomplete, and MoVi [54] dataset does not contain depth information. The shortcomings of the existing datasets have motivated us to collect our own dataset as described in Section III.

TABLE 1. Comparison different state-of-the-art object reconstruction approaches to our proposed solution.

	Depth based solutions	**Pointcloud based solutions**	**RGB based solutions**	**Mixed solutions**
Voxel processing based	–	–	3D-R2N2	YoloExt
Pointcloud processing based	Our solution	PointNet, w/FCAE, PCN, AtlasNet, MSN	PointOutNet	–

TABLE 2. Summary of 3D object reconstruction datasets.

Dataset	Ref.	Description	Reasoning
ITOP	[51]	Various human poses recorded using Kinect v1 sensor.	Experiments have been conducted with lukewarm results, some modification to the method are required due to Kinect and RealSense having different error noise in addition to background noise in the dataset. Due to the discontinuation of Kinect sensor further research was not pursued.
EVAL	[52]	Various human poses recorded using Kinect v1 sensor.	We were not able to parse the data in the given file format. Due to the discontinuation of Kinect sensor further research was not pursued.
HHOPE	[53]	High quality dataset containing one or more subjects in various poses recorded with multi-view cameras including RGB and IR.	Only two data samples from the described dataset are given, additionally the given datapoints are incomplete, also contain multiple subjects. The depth information can be simulated, experiments with the synthesized depths have given good results, but the provided supplement is too sparse to be useful outside of validation.
MoVi	[54]	Various recordings using RGB and motion sensors capturing subjects performing various gestures.	Does not contain depth information, however due to the large amount of datapoints the synthesized depth is fully incorporated into synthetic dataset.

III. MATERIALS AND METHODS

A. DATASET

The training process of a neural network relies on a collection of a large good quality dataset. Multiple datasets exist for the task of object detection, e.g., *COCO* [55] and *Pascal VOC* [56], however, the selection of 3D object databases is quite limited, one of the biggest being *ShapeNet*, which contains labeled 3D model meshes starting with various household appliances, like bathtubs, dishwashers and ranging to larger scale objects like airplanes and vehicles. Other 3D object databases also provide labeled voxel-space representations [57]. Unfortunately, these do not fit the goal that we are attempting to achieve which is the reconstruction of the full human body posture from a single depth frame. This requires ground-truth information, in addition to depth frames. However, while a few publicly available datasets exist for example *ITOP* [51] and *EVAL* [52], in both cases

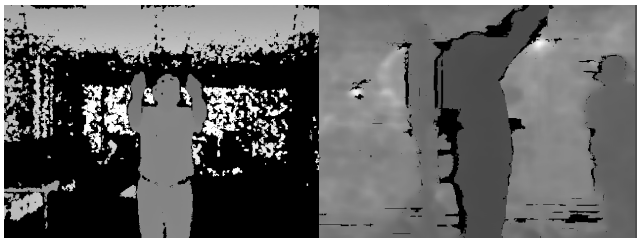


FIGURE 1. An example of real world depth frame. Left frame is captured using *L515* LiDAR, while the right frame is captured using structured light *D435i* depth sensor.

they rely on *Microsoft Kinect* depth sensor, which has been discontinued making such datasets obsolete as different types of sensors tend to exert different types of depth noise behavior. For this reason, we have opted instead to create our own datasets that would fit our machine learning needs. As the creation of real world data-set would require a costly and time-consuming process, of manually created and processed ground truth pointclouds, we have instead opted to split our dataset into two parts. First one being real world-dataset, which contains depth field recording of various exercises from multiple camera angles. The second is synthetic dataset, which contains depth frame samples created using *Blender* [58] and their ground-truth pairs.

1) REAL-LIFE DATASET

Real-life dataset has 168 recordings from three human subjects performing various exercises. Videos have been captured using two *Intel Realsense* sensors from different viewpoints. The first depth sensor (*L515*) being directly in front of the subject, the second (*D435i*) one being directly to the right of the subject. Both depth sensors are placed roughly 1.4 meters above the ground level and roughly 1.8 meters away from the subject. An example of both sensor depth frames can be seen in Figure 1. This setup produced us 168 different recordings, i.e., 84 for each sensor/perspective type. Additionally, because *L515* is *LiDAR* [59] based sensor, while *D435i* is structured light based [60], this produces us two different depth field error models which we need to account for. Each of the subjects were asked to perform each of the seven exercise tasks: 1) shoulder flexion; 2) shoulder flexion and internal rotation; 3) shoulder flexion and internal rotation with elbow flexion; 4) shoulder extension and internal rotation; 5) shoulder flexion; 6) full shoulder flexion; 7) shoulder adduction. Each of the exercises were done four times, producing 28 recordings of a subject from each camera. For the recordings to be used in our training process, firstly, we acquire stand-alone depth frames from the recording, this is done by capturing a single still depth frame for each 0.5 seconds of the recording. The analysis of the extracted dataset is given in Table 3.

As we can see from the table, each of the sensors has captured 3316 frames from both perspectives. As we do not use full depth frames as our inputs and instead we use pointclouds, we need to turn the input depth image into

TABLE 3. Real-world dataset breakdown captured for a single perspective. Each of the subjects were asked to repeat each of the exercises for four times: 1) shoulder flexion; 2) shoulder flexion and internal rotation; 3) shoulder flexion and internal rotation, with elbow flexion; 4) shoulder extension and internal rotation; 5) shoulder flexion; 6) full shoulder flexion; 7) shoulder adduction.

Exercise No.	Subject 1	Subject 2	Subject 3	Total
No. 1	92	99	87	**278**
No. 2	89	78	74	**241**
No. 3	76	71	54	**201**
No. 4	86	76	72	**234**
No. 5	80	70	67	**217**
No. 6	72	87	63	**222**
No. 7	97	92	76	**265**
Total	**592**	**573**	**493**	**1658**

one. To do this, firstly we discard, by setting z value to zero, while all pixels have depth $z > 2.5$ this greatly decreases the amount of noise that was captured far away in the room. Once this is done, we apply Algorithm 1 using the intrinsic camera matrix K, (see Equation 1) to convert the input depth into pointcloud, where f_x and f_y are the cameras focal points c_x and c_y is the sensor center point. Because our initial depth resolution is 640×480, the pointcloud has 307200 vertices in total, which is too large to be used as an effective neural network input. To downsample the pointcloud we use *Farthest Point Sampling* [61] (FPS) to decrease the pointcloud size to the required resolution pointcloud, in our case 2048 vertices. No further augmentation or postprocessing is applied to the real-world dataset.

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

An actual example of resampled depth frames obtained from each of the sensors is shown in Figure 2. Both devices exhibited noisy depth data streams (real-life conditions), showing a reason why existing methods fail or achieve miserable results when trained on artificial datasets.

2) SYNTHETIC DATASET

Synthetic dataset was based on *MoVi* [54], which is a big dataset of motion captured (mocap) exercises. Unfortunately, *MoVi* database did not contain depth sensor information. This was solved by binding the mocap skeleton to the *AMASS* [62] triangle meshes (see the example in Figure 3). *Blender* was used for rendering depth frames for each of the motion capture frames, which were rendered from multiple perspectives to produce human body representation to train the model. Multi-step process was used to generate the date. At first the human model was rotated at [0°, 360°) in 45° increments on *Up* (z) axis, while in parallel the camera itself was rotated [−35°, 35°] at the increments of 15° on *Up* (z) axis. As with real-life dataset, we had to place the camera in 1.8 meter radius away from the rendered mesh and 1.4 meters above the ground to "replicate" the same conditions and achieve similar scale properties. The same

Algorithm 1 Convert the Depth Image to Pointcloud Vertices

procedure TO_POINTS($w, h, f_x, f_y, c_x, c_y, D$)
  $x \leftarrow 0$
  $y \leftarrow 0$
  $V \leftarrow \{\emptyset\}$
  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

process was repeated for all *AMASS* polygonal meshes (again to replicate real-life dataset), with a benefit of providing some resistance against female and male body type variations. Raytracing was used to render the depth frames and to export it as *OpenEXR* [63] file format, resulting in a much higher depth precision than "traditional" 8-bit channel image formats. At a final step, ground truths were established for each model by uniformly sampling the mesh to the desired density, which is at *2048* points in our case.

As in the real-life dataset, the resulting synthetic depth field has the resolution 640 × 480, because every point pixel in the image matches to a vertex in the pointcloud using it as an input to the deep learning model. To help improve performance, the depth frames were downsampled, first converting the data into a pointcloud, then applying the same procedures as we would with real depth sensors, because the raytracer used to render the depth is identical to a pinhole camera. Finally, the converted pointcloud was down-sampled to 2048 vertex pointcloud.

B. PROPOSED EXTENDED ADVERSARIAL AUTO-REFINER NETWORK

By refining real-world data to be more synthetic like, a priorietary adversarial auto-refiner network model with three primary stages was developed to overcome the problem of real-life datasets lacking ground-truths (to help perform the reconstruction). The noisy input depth field obtained by the sensors is refined to make it more synthetic-like during the refining stage (Figure 5). The encoder then extracts hidden vectors from the improved pointcloud, which are

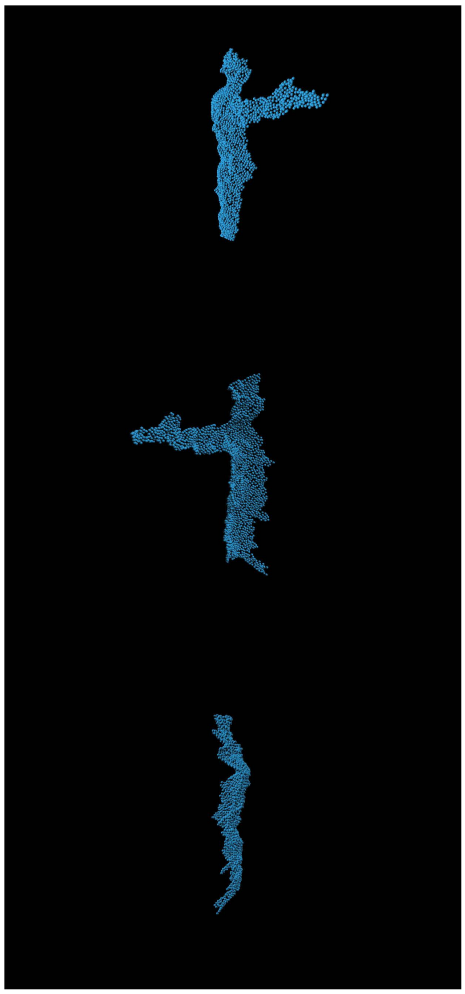


FIGURE 2. Depth frame conversion into pointcloud using the intrinsic parameter matrix *K* of camera.

eventually used for reconstruction in the second stage. The decoder stage (Figure 6) is the third step, in which previously recovered hidden features are used for coarse object feature reconstruction, followed by fine-grained reconstruction using a residual connection.

These are the three deliverable phases that were used in the final experimental evaluation. However, to train the refiner model, our network requires an additional discriminator stage. We discovered that using our refined output results as inputs for surface mesh reconstruction using *Point2Mesh* allows us to reconstruct the mesh with acceptable distortion levels, as opposed to using only the clean ground-truth, which frequently causes the chosen surface reconstruction method to fail completely. The whole network architecture is depicted in Figure 7. Using the pinhole model, we extract the pointcloud from the input depth field by applying the cameras intrinsic matrix K. For a given refiner output, an encoder is then used to extract the most relevant features. Following that, we apply a two-part decoder procedure in which we reconstruct the course object features first, then the residual fine pointcloud reconstruction. The result is used to recreate

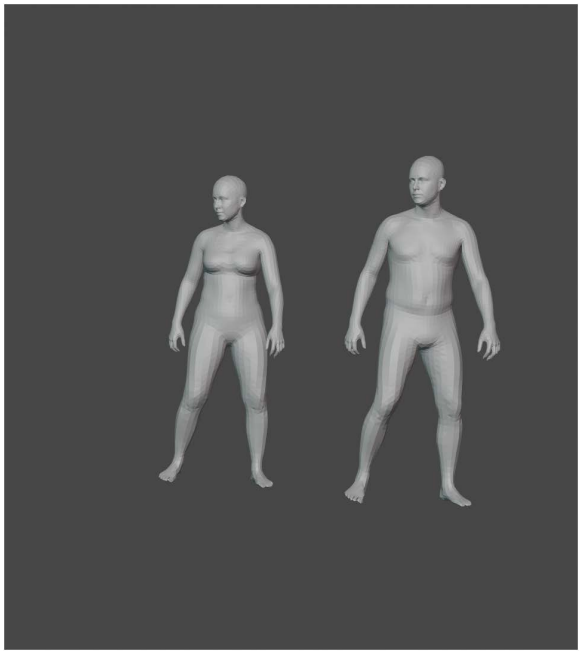


FIGURE 3. An example from *MoVi* dataset. Mocap posture was applied to models from *AMASS* (both female and male body type).

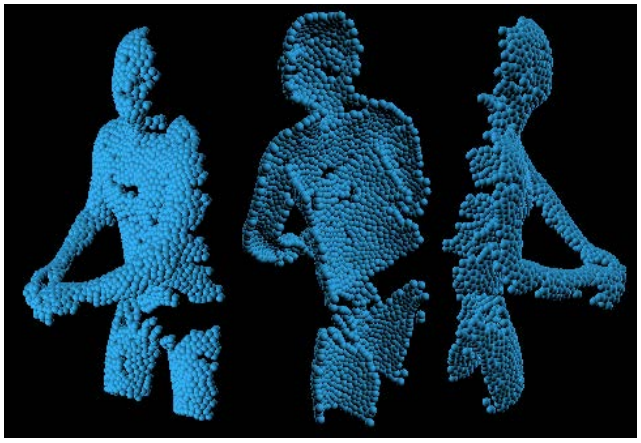


FIGURE 4. Depth frame produced by *Blender* and transformed into pointcloud using an intrinsic parameter matrix *K* of camera.

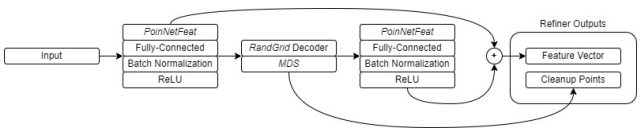


FIGURE 5. Overview of refining stage.

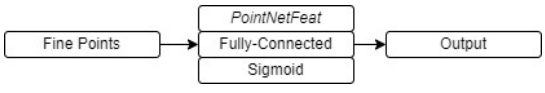


FIGURE 6. Overview of the discriminator stage.

the surface mesh of an item using *Points2Mesh*. We also employ the generated fine pointcloud reconstruction as an input for the discriminator throughout the training process, which alters the refiner's weights.

The suggested approach can do entirely independent sensor-to-screen reconstruction; for a full *UML* sequence diagram, see Figure 8. As can be seen, a full round of object reconstruction begins with the loading of training weights, followed by the initialization of the depth sensor. We then acquire a depth frame and process it pixel-by-pixel after initialization. This is accomplished by setting any pixels with a value of $z > 2.5$ to $z = 0$. This eliminates pixels that are more than 2.5 meters away from the camera, lowering the amount of noise the ANN has to cope with while also boosting the fidelity of the input pointcloud. K is then used to transform the depth frame to a pointcloud. Any points with a depth component of 0 are then deleted, lowering the overall number of points in the pointcloud. Because the *FPS* operation is slow, we delete as many points as possible before running it. The final output is a vertex pointcloud with a value of 2048 that can be used for reconstruction.

1) REFINER AND ENCODER

Our main contribution and the main novelty in the subject of object reconstruction discourse is the proprietary refiner architecture. The primary application of adversarial neural networks involves the creation of new samples [64]–[67], and this can be achieved typically either from random noise, hand-drawn inputs, or other similar methods. However, very few papers exist that concentrate on the refinement of the initial input without the addition of distortions. There have been a couple attempts regarding the refinement of synthetic data to make it real-world like, yet the given solutions were not fully unsupervised, as in the case of [68], synthetic and real samples had pupil direction in common. Instead, our solution involves the refinement of real-world data to make it more artificial like the architecture of our refiner can be seen in Figure 9.

We get 256 hidden features using the *PointNetFeat* pointcloud feature extraction architecture [26], which we then immediately connect into a fully-connected layer for our size 256 bottleneck. Following that, a batch normalization is performed to improve generalization and reduce training time [69], [70], followed by a non-linearity activation function. We utilize textitRectified Linear Units [71] (ReLU) because they are faster to compute and produce better results than other non-linear functions, such as *sigmoid*. The resulting latent feature vector is then coupled to a modified random grid (*RandGrid*) block for reconstruction, as suggested by *Liu et al. [44]*. The result is then resampled using *Minimum Density Sampling* (MDS) to produce a more consistently distributed pointcloud. The key change we made to the random grid decoder was to choose a beginning position spread in the range of $[-0.5, 0.5]$ while shifting all points to the pointcloud's mass center.

Another novel aspect is that our *RandGrid* solution generates patches on all three dimensional axes instead of two-dimensional axes. We have established, that such changes to the architecture have significantly improved our models convergence and robustness by making the

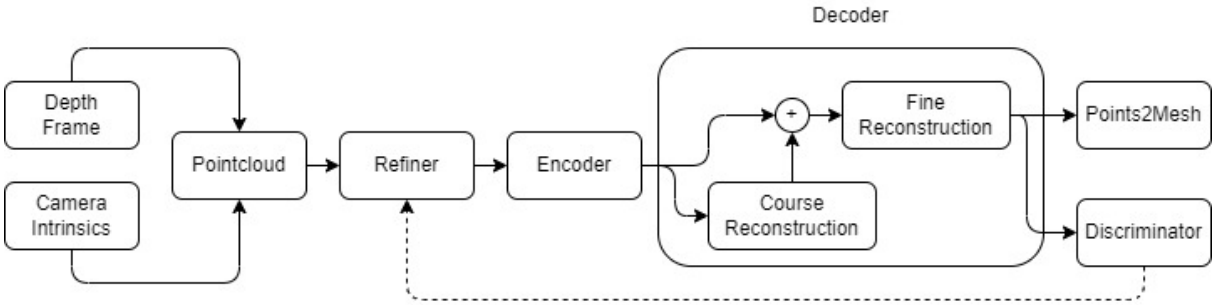


FIGURE 7. A condensed overview of the proposed extended deep adversarial auto-refiner model architecture. Captured depth field and sensors intrinsic matrix is employed to transform depth frame into a pointcloud, which is then used for refiner stage. Refined pointclouds hidden features are then acquired using encoder and fed to decoder for coarse-to-fine object reconstruction. The obtained fine reconstruction is then used as an input for *Points2Mesh* for surface mesh reconstruction and evaluated by the discriminator which makes the refiner to fine-tune its weights.

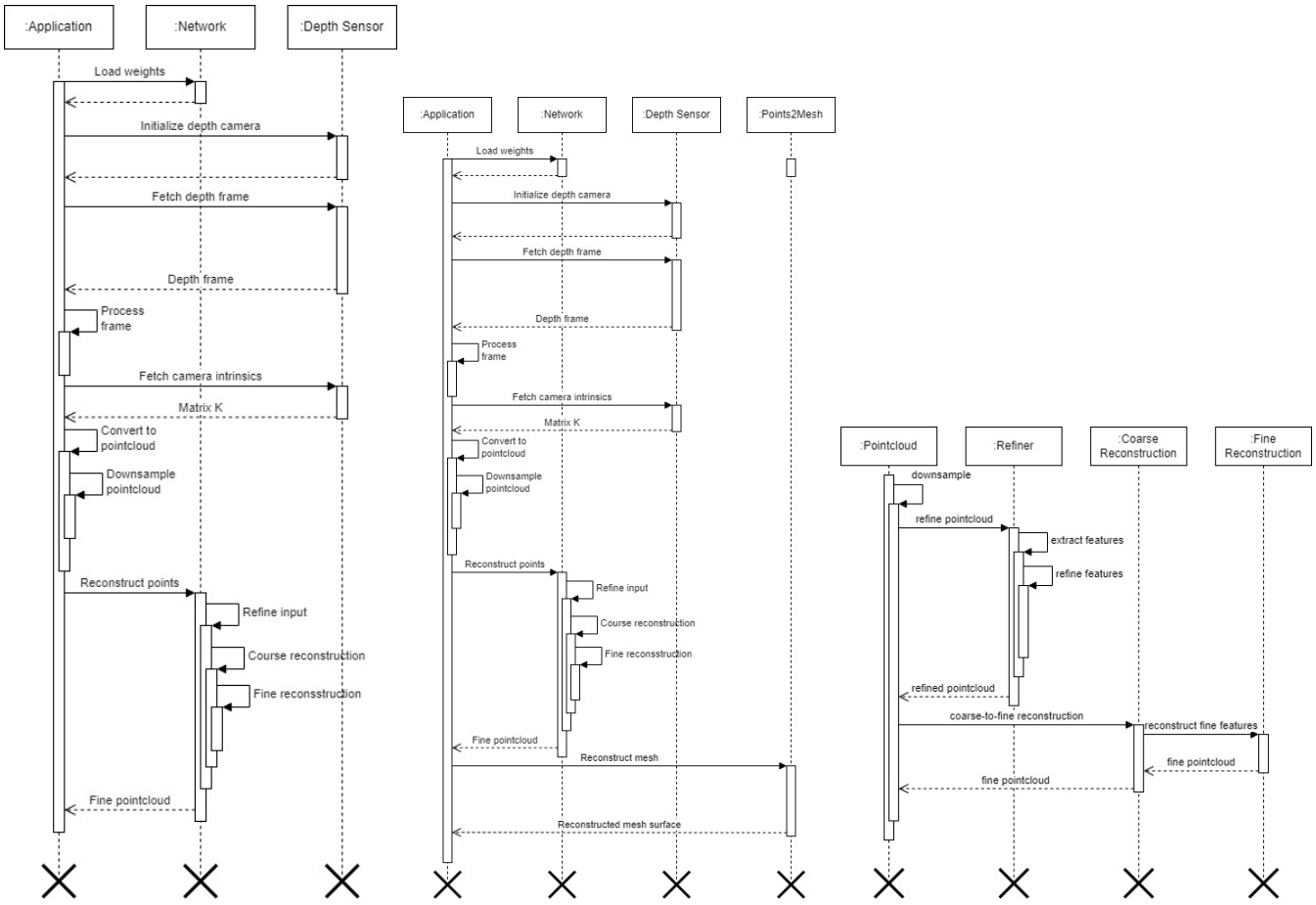


FIGURE 8. UML sequence diagram of the proposed method for human posture reconstruction: whole process (left); mesh data preparation (middle); reconstruction process (right).

randomized reconstruction points to become more uniformly distributed.

The refiner output is a given random grid output that still contains the initial input data that has been denoised and cleaned up. On the cleaned-up pointcloud, another *PointNetFeat* feature extractor was used to retrieve the feature vectors for reconstruction using the encoder. The model now has two hidden feature vectors, one from the real input and the other from the improved input. Finally, both feature vectors are concatenated into a single hidden feature vector with the shape 512.

2) DECODER

The decoder network is somewhat similar to the one proposed by *Liu et al. [44]*, with the main differences being in the random grid (*RandGrid*) decoder. We have established, that

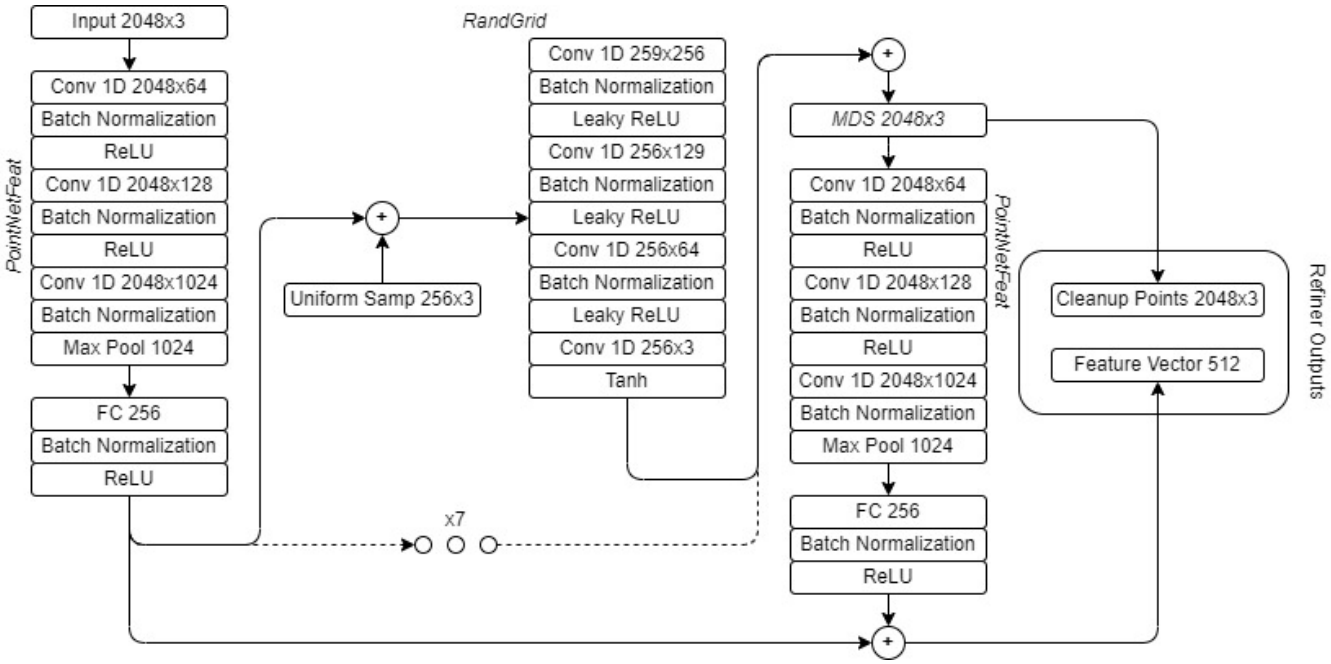


FIGURE 9. Network architecture for refiner and encoder. The original input feature vector is recovered from a pointcloud, and the ANN attempts to predict a cleaned up input. An additional feature vector is then extracted for the cleaned-up output. Both of these are subsequently put to use throughout the decoding process.

the architecture works much more efficiently if the random pointcloud is to be generated on all three axes, in the range of $[-0.5, 0.5]$ with the vector offset of refined pointcloud's center of mass. This proprietary approach is displayed in Figure 10. The cleaned up output by the refinement stage and encoders feature vector are fed into *RandGrid* decoder, which uses 8 primitives for performing a coarse point reconstruction.

The pointcloud as the predicted object is then fused with the refined pointcloud instead of the input pointcloud, and then resampled using minimum density sampling. The final result of fine pointcloud reconstruction is produced by feeding the resampled pointcloud through the residual decoder (*PointNetRes*). Figure 11 shows the full architecture of the decoder we utilized.

3) DISCRIMINATOR

The main purpose of our auxiliary discriminator neural network is the evaluation of whether a given pointcloud sample is either from artificial or real-life datasets. The architecture of our discriminator ANN is shown in Figure 12. Decoder's fine-grained reconstruction pointcloud is fed to the discriminator. Next, it is fed to the feature extraction block, and then to the connected fully-connected layer. Fully-connected layer only has a single neuron output, which is then passed through the sigmoid function. This output function indicates if the produced pointcloud is artificial (1) or if it is real (0). This additional ANN allows us to train the decoder on the artificial samples, while the encoder and refiner networks are trained on both real and artificial datasets. It means

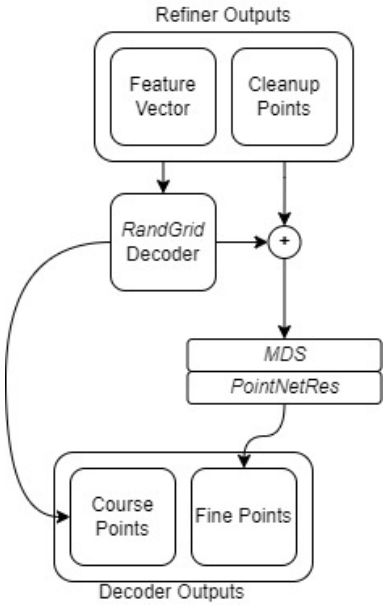


FIGURE 10. Abbreviated decoder architecture. Given feature vectors from refiner a coarse objects pointcloud is predicted. The coarse features with cleaned up features are then resampled and passed through the residual block. The output is a fine-grained point reconstruction.

that we do not need to have ground-truths for real-life datasets.

The final complexity of our neural network architecture can be seen in Table 4, GLOPs have been used to compare the complexity as opposed to Big-O notation, for the network complexity is constant, making GFLOPs a more expressive

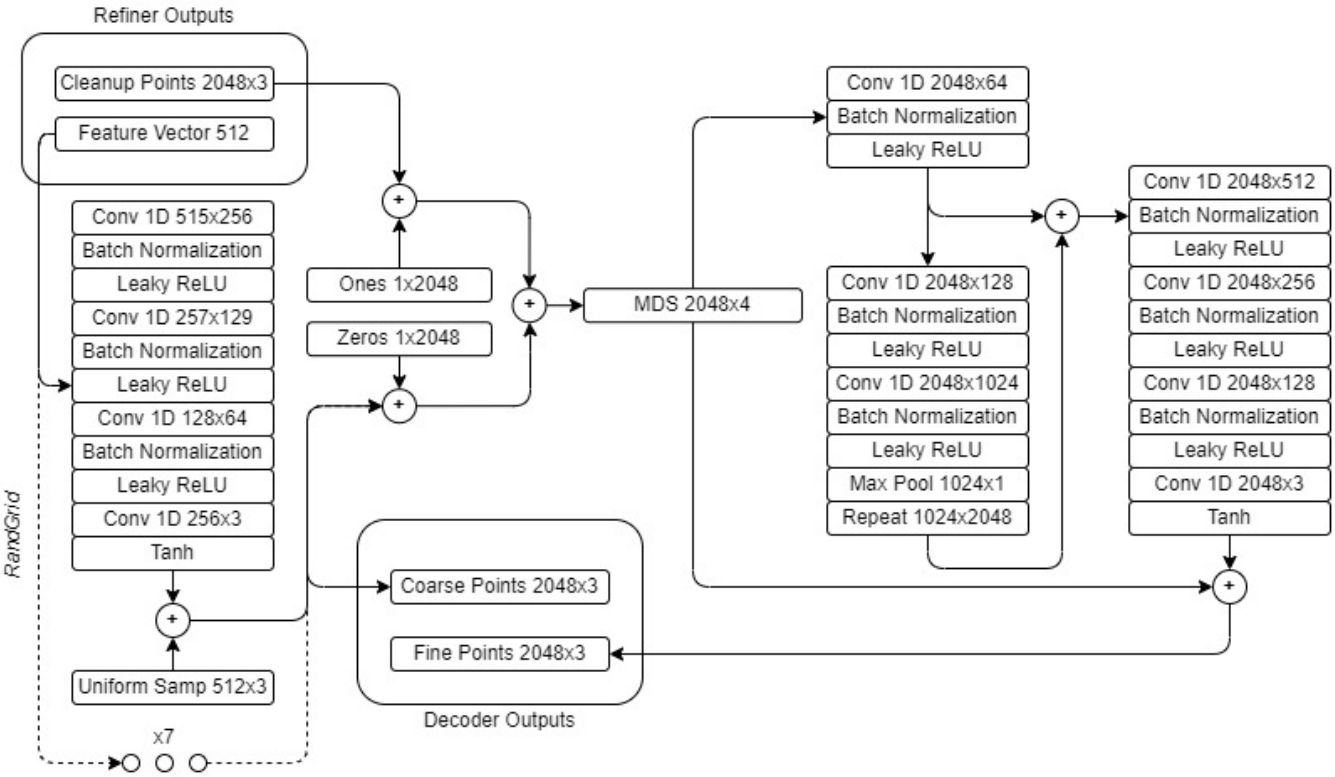


FIGURE 11. Full decoder network architecture.

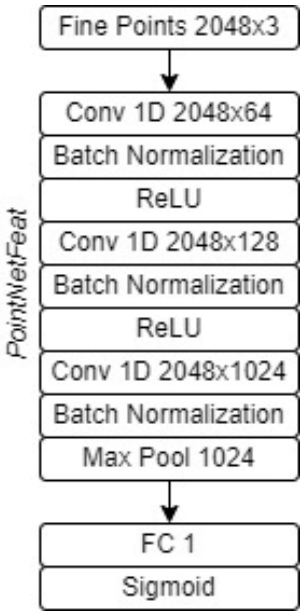


FIGURE 12. Proposed discriminator architecture. Fine-grained pointcloud reconstruction is used as input for the discriminator. Inputs latent features are extracted and sent through a fully-connected layer, followed by sigmoid activation function.

complexity measure. As we can see, our network architecture is overall more lightweight than *MSN* which ours is in part based off of, due to an additional refinement stage allowing us to greatly reduce the number of trainable parameters. This

TABLE 4. Comparison of model complexity by number of parameters (No. of Pars), number of operations (No. of Ops) and model size (in MB).

Method	No. of Pars (M)	No. of Ops (GFLOPs)	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	6.93	7.51	23.12
Ours w/ Discriminator	7.51	7.65	23.68

reduces the risk of the network having too high capacity and learning from noise instead of useful features, which is important due to the inherently noisy real-depth sensor data, in addition to making it more lightweight.

C. TRAINING PROCEDURE

We found our ANN training in a single go to be difficult, for this reason we have devised a five-phased training methodology. Firstly, we train our encoder training, once the encoder is considered trained satisfactorily, we begin the decoder training, following the discriminator training and then, in the end, the refiner training. This five-phased training methodology has empowered our model to converge much more easily and finally improve the trained prediction result stability. Additionally, during training we perform augmentation to the artificial input aiming to produce *Realsense*-like depth deformities. We have two types of synthetic data

augmentation. The first one involves either removing random patches from pointcloud, this simulates the laser sensor as well as structured light sensor noise. Moreover, we introduce wavelet perturbations to the pointcloud, which emulates the structured light depth field defects (see Equation 2).

We employ wavelets with an amplitude of $A = (0, 0.03]$ and a periodicity of $\omega = [2\pi, 32\pi]$. For pointcloud, there is a 75% percent chance of deleting tiny patches, and a 50% percent chance of wavelet flaws. We anticipate that these two sorts of augmentations will be cleaned during the refiner/encoder network's cleanup step. A third sort of augmentation includes the addition of a random scale value to the model. This is done because real persons are of varying heights, but artificial models only contain individuals of the same height. A 50% percent chance of applying height scaling in the range of $[0.8, 1.8]$ is also present in height augmentation.

$$p(x, y, z) = (x + A \cdot cos(\omega \cdot x), y + A \cdot sin(\omega \cdot y), z) \tag{2}$$

1) PHASE I

The encoder element of the deep neural network is trained in the first phase. The emphasis here is on passing the first seeding weights encoder, which is used in the reconstruction training's final step. The discriminator and its weights are kept unaltered during this phase. Instead, we train all three stages of reconstruction (cleanup, coarse, and fine) to work as an auto-encoder by feeding input values as close to the input as the bottleneck allows. For comparison of prediction to ground truth, the auto-encoder requires an unstructured pointcloud loss function. While the *Chamfer* distance (CD) has a low memory footprint and can be computed quickly, we discovered that it provides incorrect features by forcing vertices to cluster together instead of spreading evenly throughout the required mesh. Therefore, as described in *Liu et al. [44]*, we have chosen to use an approximation of *Earth Mover's* distance (EMD) (see Eq.3) along with suggested penalization criteria see Eq. 4), where $d(u, v)$ is the Euclidean distance between two vertices in 3D space; $\mathbb{1}$ is the indicator function used to filter which is shorter than λl_i with a suggested value of $\lambda = 1.5$.

$$\epsilon_{emd}(S, \hat{S}) = \min_{\phi: S \to \hat{S}} \frac{1}{|S|} \sum_{x \in S} ||x - \phi(x)||_2 \tag{3}$$

$$\epsilon_{exp}(S, \hat{S}) = \frac{1}{KN} \sum_{1 \le i \le K} \sum_{(u,v \in \tau_i)} \mathbb{1}\{d(u, v) \ge \lambda l_i\} d(u, v) \tag{4}$$

The loss function for Phase I training is 5, where $\hat{S}_{clean}$ is the result of the cleaning stage, $\hat{S}_{coarse}$ is the result of the coarse reconstruction stage, $\hat{S}_{fine}$ is the result of fine point reconstruction, and S_{clean} is the ground truth for the cleaned pointcloud, because the input pointcloud can have additional noise added to it during augmentation. The expansion penalty hyperparameter was kept constant at $\gamma = 0.1$. Once $\epsilon_{\Phi 1} < 0.13$, the artificial neural network stops Phase I training. During our studies, we chose this early stopping value since we discovered that it is preferable to avoid the network becoming totally converged during the final reconstruction phase. When the criterion is met, Phase II begins training.

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

2) PHASE II

The second training phase aims to train the decoder network on its own. As a result, no further model weight changes are made to the refiner, encoder, or discriminator branches. We also solely train on the artificial dataset, unlike the prior phase. Weight re-initialization is required for the model after Phase II begins for the first time in order to avoid any local minimums throughout the subsequent training. We re-initialize the decoder weights using Xavier initialization [72] and biases using uniform distribution to erase the weights. Furthermore, for both encoder and decoder optimizers, we reset any accumulated optimizer state that has been established up until this point.

We only update the weights for the decoder itself, without backpropagating into the encoder, because Phase II only trains the decoder. After this phase, a feasible profile is created for both real and fake pointcloud reconstructions, which will be employed during Phase III training, when the discriminator is trained. Our loss equation may be simplified to 6, where S_{gt} represents the synthetic ground truth for the synthetic input, because only decoder weights are being trained during this phase. Phase II training continues while $\epsilon_{\Phi 2} > 0.08$, and Phase III training begins when the loss falls below the threshold. The threshold value for early training termination was set empirically, as it was in Phase I.

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

3) PHASE III

We train the discriminator in the third step so that it can distinguish between real and intentionally created data points. We postponed discriminator training until Stage III because the reconstruction network's weights are still changing dramatically. During training, the discriminator is given the output of the decoder network's reconstructed fine pointcloud and must categorize it as 1 for synthetic dataset elements or 0 for real dataset elements. As a result, we used binary cross entropy as the loss function (see Eq. 7 below). After the discriminator has been trained until $\epsilon_{\Phi 3} < 0.05$, the last training phase can begin.

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

4) PHASE IV

In addition, we implemented the fourth phase of training, which is when the actual adversarial training for refining of

the input pointcloud takes place. We reset prior optimizer states because we need to start optimizers from scratch. If optimizers have built up local minimum states, this helps to kick-start the training. During the adversarial training phase, both artificially produced and real-life datasets are used. The artificially created dataset is utilized to reinforce the discriminator and further enforce the decoder state, whilst real-life samples are used for refiner/encoder training and discriminator reinforcement.

Phase IV is divided into three substeps for each training batch, with the weights of each step changed separately. To begin, we use the synthetic dataset to reinforce the entire network, yielding the loss function for step one as 8. Second, the refiner cleans up the input pointcloud with an adversarial loss network in an attempt to convince the discriminator that the given real-world sample is actually a synthetic one. Only the refiner weights are updated in this step; the discriminator is left untouched, and only its loss function is used (see Equation 9). We also don't want the refiner to distort the original contour of the pointcloud, so we constrain it to a pointcloud loss in relation to the input frame with a factor of $\alpha = 0.4$, which was chosen heuristically. This allows the pointcloud to acquire adversarial features and develop the model without losing its structure. Finally, we strengthen the discriminator so that it can distinguish between real dataset pointclouds and falsely produced ones (see Eq.10).

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

$$\epsilon_{\Phi 4b} = \alpha \cdot \epsilon_{emd}(S_{clean}, \hat{S}_{clean}) + \gamma \cdot \epsilon_{exp}(S_{clean}, \hat{S}_{fine}) + \epsilon_{bce}(1 - y, \hat{y}) \tag{9}$$

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

5) PHASE V

As a training or prediction input, we use fine reconstruction results from the *Points2Mesh* technique. Our model results show that the inherent noise makes the surface mesh reconstruction methodology more resilient to noisy input, whereas utilizing synthetic noiseless training data would cause the method to fail when applied straight to the rebuilt pointcloud.

IV. RESULTS

The primary goal of our deep auto-refining adversarial neural network is the reconstruction of self-occluding human body poses by using real world depth frames. As our real-world dataset has no ground truth information, it is not possible to compare real-world reconstruction results in an objective way. Therefore, we will only be using an artificially generated dataset for our quantitative evaluation. Whereas for real-world (and synthetic) dataset evaluation, we will be using expert knowledge to visually inspect the reconstruction results. The quantitative evaluation will be performed using two quality metrics, the first one being already discussed *Earth Mover's distance* and the second one being *Chamfer* distance is seen in Equation 11. The quantitative results for

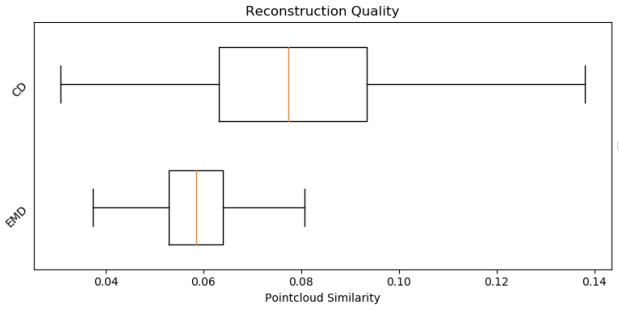


FIGURE 13. Reconstruction similarity using both *Earth Mover's Distance* and *Chamfer Distance*. Lower values are better.

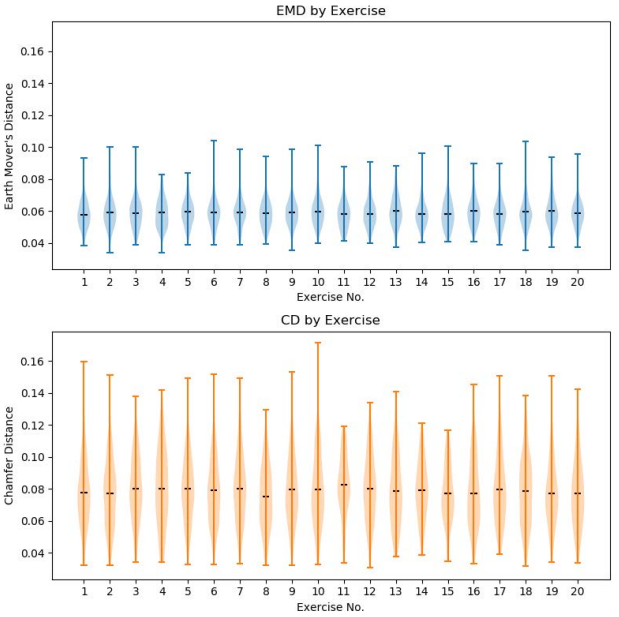


FIGURE 14. Synthetic dataset reconstruction quality using *Earth Mover's Distance* and *Chamfer Distance* metrics breakdown by exercise.

the artificial dataset can be seen in Figure 13.

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

To further evaluate our synthetic dataset quantitative reconstruction results, we break down the reconstruction results by the performed exercise seen in Figure 14 and by gender as shown in Figure 15. As we can see, there are no obvious outliers in either case, this lets us assert that our network has managed to learn the human pose characteristics based on the given input pointcloud. In addition to that, there are no major reconstruction discrepancies between male and female datapoints. While the latter can partially be attributed to similarities between the human shapes in both genders, further visual inspection shows that the model can differentiate between distinctly male and female secondary body characteristics.

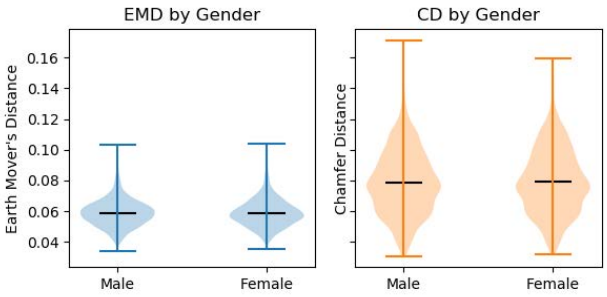


FIGURE 15. Synthetic dataset reconstruction quality using *Earth Mover's Distance* and *Chamfer Distance* metrics breakdown by gender.

TABLE 5. 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 training process is more complicated.

Method	ShapeNet		AMASS	
	EMD	CD	EMD	CD
PointNet w/ FCAE [26]	0.0832	0.0182	3.3806	4.9042
PCN [42]	0.0734	0.0121	3.0456	4.0955
AtlasNet [43]	0.0653	0.0182	2.0875	6.4343
MSN [44]	0.0378	0.0114	1.1525	0.8016
Our method (Cleanup 2D)	*N/A*	*N/A*	**0.0593**	0.0437
Our method (Cleanup 3D)	*N/A*	*N/A*	0.0603	**0.0292**
Our method (Reconstruction 2D)	*N/A*	*N/A*	0.0672	0.0928
Our method (Reconstruction 3D)	*N/A*	*N/A*	**0.0590**	**0.0790**

For the quantitative comparison of our reconstruction results with other state-of-the-art methods, we contrast self-reported *ShapeNet* dataset reconstruction results in addition to the reconstruction results when they are given *AMASS* dataset, against our approach *AMASS* dataset. As our proposed method relies on the real-life and artificially generated dataset symbiosis our method can not be adopted the *ShapeNet* dataset, therefore the self-reported values are used for assessment. As we can see from Table 5, other state-of-the-art approaches completely fail when tasked to reconstruct the human posture. Additionally, we can see that while we cannot compare to the self-reported *ShapeNet* reconstruction metrics, our reconstruction metrics on the *AMASS* dataset are on par with the values reported by state-of-the-art approaches. Moreover, by evaluating our modification to the random grid neural network block (2D vs. 3D) we can see a modest improvement in the reconstruction results.

We've used expert knowledge to assess and validate the reconstruction outcomes visually, in addition to the quantitative evaluation of synthetic data. When then we compare the reconstruction results (teal) to the expected ground truth pointclouds (orange) (see Figure 16). We can see that the neural network was able to reconstruct the expected outcome with very few defects, the majority of which occurred towards the limbs' ends. According to our findings, the discrepancy can be explained by two main factors: first, human hands and feet are relatively small body parts that require much higher fidelity pointclouds

FIGURE 16. Comparison of ground truth (left side/orange color) and prediction (right side/teal color) from different viewpoints.

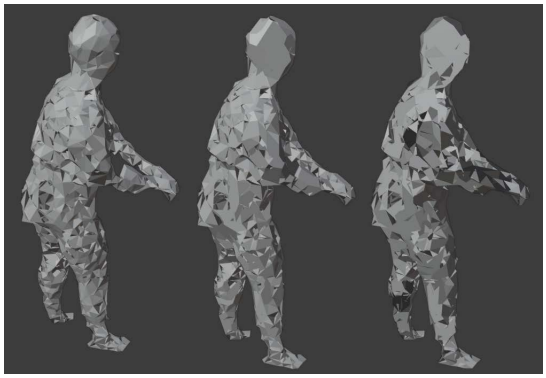


FIGURE 17. Surface mesh reconstruction from synthetic prediction using 5233, 2811 and 1357 faces.

to reconstruct appropriately, and second, the majority of ambiguities can be explained by missing data that is required for the correct hand and foot inference. Furthermore, we use the output fine pointcloud reconstruction as the input for our mesh recreation branch, as shown in Figure 17. As can be seen, even though some facet normals are inverted, the overall

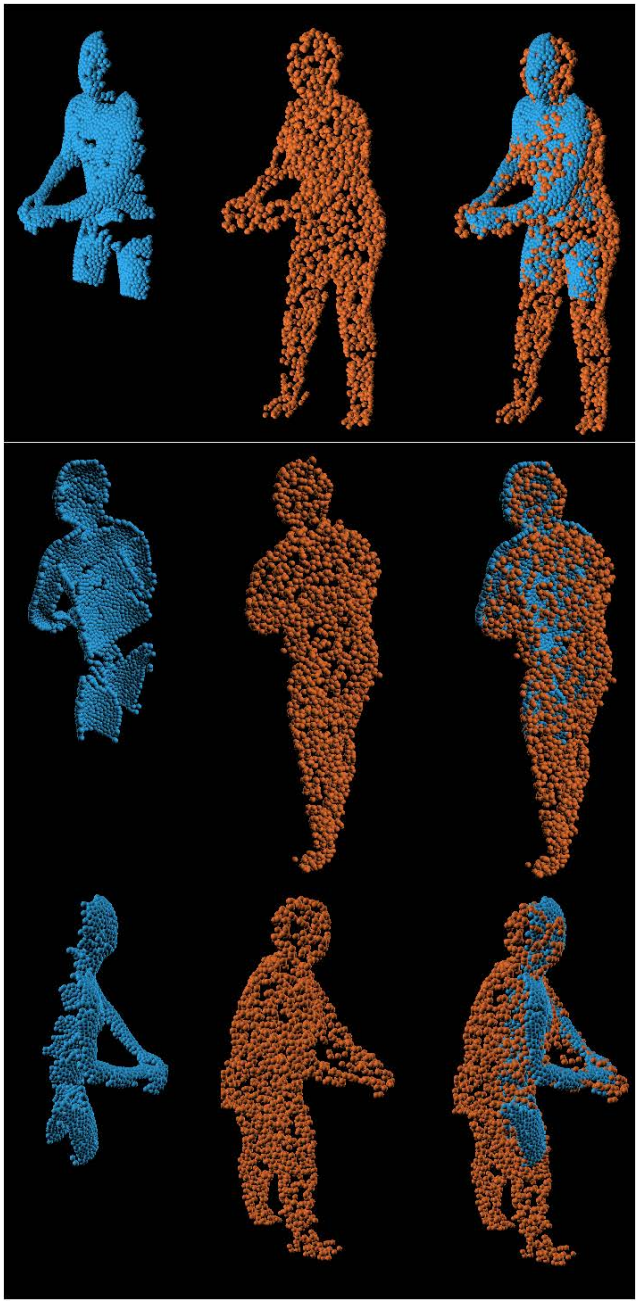


FIGURE 18. Stacked views for synthetic input (blue color) and its prediction (orange color).

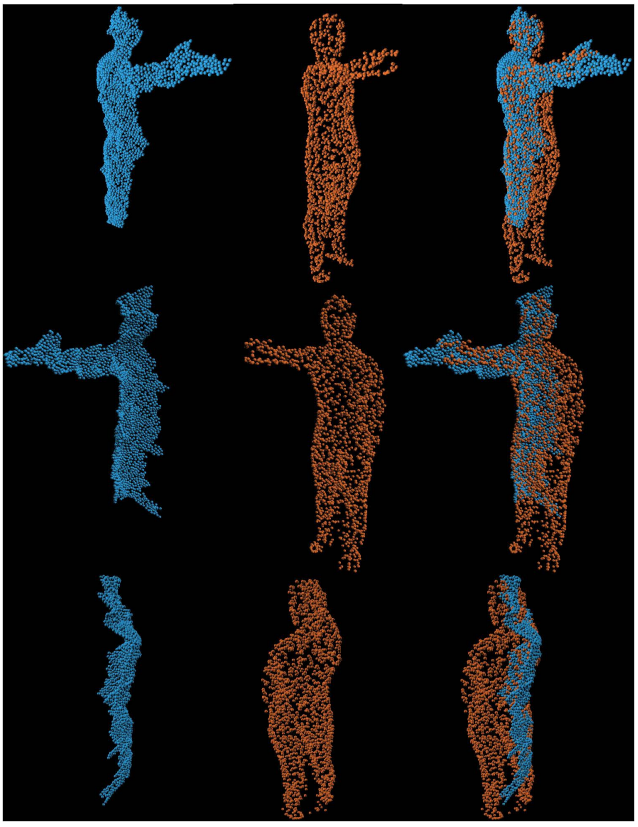


FIGURE 19. Stacked views for real depth sensor input (blue) and its prediction (orange). Real input is quite distorted due to depth sensor inaccuracies, the cleanup stage cleans up most of noise.

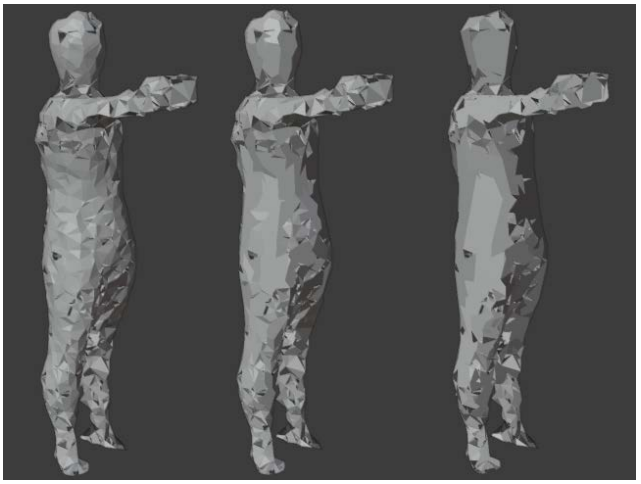


FIGURE 20. Surface mesh reconstruction from real dataset prediction using 5032, 2832 and 1349 faces.

shape has been reconstructed correctly, whereas if we used the unmodified *Point2Mesh* training approach, the resulting reconstruction fails completely.

As we can see from Figure 18, even though the ANN was being given few input (orange) about the legs to predict from, it was able to predict (in teal) the entirety of its orientation. Additionally, the neural network was able to predict and reconstruct full human posture while having less than half of the body features.

Finally, we compare the reconstructed real-world data items to the input depth 19). It's worth noting that our proposed solution was capable of reconstructing the entire human posture and body while also correcting the bulk of deformations observed in the input depth frame. It was observed, that where the depth has more extreme abnormalities, the most noticeable flaw in the real-life data reconstruction is detected. For example, we can see in the top row that there is an extra lump at the end of the hand that was captured by the depth sensor. This flaw caused the

FIGURE 21. Reconstructed mesh involving multiple subjects and clothing support.

reconstruction forecast to fail, resulting in the ''amputation'' of a portion of the hand. Furthermore, there appears to be a slight scaling mismatch between the input pointcloud and reconstruction (results shown in Figure 20), but even with professional expertise, these discrepancies are difficult to evaluate. Despite the slight differences between projected and expected reconstruction outcomes, we can confidently state that adding adversarial refinement to the reconstruction network enabled our method to reconstruct real-world depth sensor data, whereas other approaches failed. While we attempted to apply our method to the *ITOP* dataset, the reconstruction results were inconsistent due to the extremely different noise model and point distribution of the *Microsoft Kinect* sensor compared to the *Intel Realsense* sensor. Using the *ITOP* dataset to train the model would almost certainly enhance these results. Unfortunately, the dataset has more background noise, which we were unable to remove fully, making it unsuitable for our model. Continued research may be able to improve our technique by making it compatible with other sensors, such as the *Kinect*, and the noise that it generates. As a result of the *Microsoft Kinect* being a discontinued device, we believe that pursuing such an endeavor is not vital at this time.

V. DISCUSSION

The main difference in our approach compared to those depicted in Table 1 is the power to reconstruct a complete human body posture with no additional information, such as RoI masks, using real-life depth sensors. The proposed solution allows unsupervised training of the extended network with the additional input refinement stage, without providing ground-truth data of the expected result. This approach also has another advantage as it does not require prediction of the object transformation matrix to correctly place it in the 3D environment, like most of the voxel based approaches do, allowing easy code integration into already existing 3D applications such as AR/VR environments. Bandwidth overhead was low because the algorithm produced stable reconstruction results with as little data as half of the object being visible in the frame with only smaller object features like human hands or feet with some defects due to inherent ambiguities. This might be further reused in developing compression methods for future holographic teleconferencing.

VI. CONCLUSION

The paper proposed an extended three-staged adversarial auto-refining model architecture that can perform object reconstruction using artificially generated and real-life sensor data without the additional supplement of object mask. A fully unsupervised five-phased approach was used for training the network without using the ground-truth information. This was achieved by exploiting the adversarial competition between the discriminator and the refiner stage, and allowed achieving 0.059 in *Earth Mover's* distance and 0.079 in *Chamfer* distance metrics, being on par with other state-of-the-art methods in terms of reconstruction quality, yet still benefiting from the ability to reconstruct objects from real-life depth sensor data. Expert evaluation has shown that objects were reconstructed with some minute defects, high density of defects in the input pointcloud. Paper also introduced proprietary modifications to the training methodology of the *Points2Mesh* algorithm, making it more resilient to noisy pointcloud data, allowing a reconstruction of the full surface mesh of a partially self-occluded human posture.

VII. FUTURE WORK

Our current approach currently focuses on single ''naked'' subject reconstruction. In future work, we plan on expanding our approach to support clothing and multi-subject scenes. As a proof of concept, we have incorporated an additional dataset containing such cases [53] into our existing dataset. Without any modifications to our approach, we have achieved the EMD and CD metric values of 0.057 and 0.080, respectively. The visual inspection results can be seen in Figure 21.

ACKNOWLEDGMENT

The authors would like to express their gratitude for the PSO team of Dr. Joe Brandon, Dr. Francisca L. Uroma, and James Ometron for their insights in the characteristic change of processing of global image world space positions within the overtone window through deep state learning approaches.

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 *Proc. CVPR*, Jun. 2011, pp. 1297–1304.

[2] F. Lourenço and H. Araujo, "Intel RealSense SR305, D415 and l515: Experimental evaluation and comparison of depth estimation," in *Proc. 16th Int. Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, 2021, pp. 362–369.

[3] G. Bozgeyikli, E. Bozgeyikli, and V. Isler, "Introducing tangible objects into motion controlled gameplay using microsoft kinect," *Comput. Animation Virtual Worlds*, vol. 24, nos. 3–4, pp. 429–441, 2013.

[4] R. M. Lozada, L. R. Escriba, and F. T. M. Granja, "MS-Kinect in the development of educational games for preschoolers," *Int. J. Learn. Technol.*, vol. 13, no. 4, pp. 277–305, 2018.

[5] F. Cary, O. Postolache, and P. Silva Girão, "Kinect based system and serious game motivating approach for physiotherapy assessment and remote session monitoring," *Int. J. Smart Sens. Intell. Syst.*, vol. 7, no. 5, pp. 1–6, 2020.

[6] 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," *Hum.-centric Comput. Inf. Sci.*, vol. 10, no. 1, pp. 1–22, Dec. 2020.

[7] S. Camalan, G. Sengul, S. Misra, R. Maskeliūnas, and R. Damaševičius, "Gender detection using 3D anthropometric measurements by Kinect," *Metrol. Meas. Syst.*, vol. 25, no. 2, pp. 253–267, 2018.

[8] W. Ren, O. Ma, H. Ji, and X. Liu, "Human posture recognition using a hybrid of fuzzy logic and machine learning approaches," *IEEE Access*, vol. 8, pp. 135628–135639, 2020.

[9] A. Kulikajevas, R. Maskeliunas, and R. Damaševičius, "Detection of sitting posture using hierarchical image composition and deep learning," *PeerJ Comput. Sci.*, vol. 7, p. e442, Mar. 2021.

[10] M. Li, Z. Jiang, Y. Liu, S. Chen, M. Wozniak, R. Scherer, R. Damasevicius, W. Wei, Z. Li, and Z. Li, "Sitsen: Passive sitting posture sensing based on wireless devices," *Int. J. Distrib. Sensor Netw.*, vol. 17, no. 7, Jul. 2021, Art. no. 155014772110248.

[11] A.-T. Chiang, Q. Chen, Y. Wang, and M. R. Fu, "Kinect-based in-home exercise system for lymphatic health and lymphedema intervention," *IEEE J. Translational Eng. Health Med.*, vol. 6, pp. 1–13, 2018.

[12] S. H. Lim, E. Golkar, and A. A. Abd. Rahni, "Respiratory motion tracking using the Kinect camera," in *Proc. IEEE Conf. Biomed. Eng. Sci. (IECBES)*, Dec. 2014, pp. 797–800.

[13] H. P. Oliveira, J. S. Cardoso, A. Magalhães, and M. J. Cardoso, "Simultaneous detection of prominent points on breast cancer conservative treatment images," in *Proc. 19th IEEE Int. Conf. Image Process.*, Sep. 2012, pp. 2841–2844.

[14] M. J. Sousa, A. Moutinho, and M. Almeida, "Thermal infrared sensing for near real-time data-driven fire detection and monitoring systems," *Sensors*, vol. 20, no. 23, p. 6803, Nov. 2020.

[15] J. Pérez, M. Bryson, S. B. Williams, and P. J. Sanz, "Recovering depth from still images for underwater dehazing using deep learning," *Sensors*, vol. 20, no. 16, p. 4580, Aug. 2020.

[16] A. Díaz-Álvarez, M. Clavijo, F. Jiménez, and F. Serradilla, "Inferring the Driver's lane change intention through LiDAR-based environment analysis using convolutional neural networks," *Sensors*, vol. 21, no. 2, p. 475, Jan. 2021.

[17] M. Latella, F. Sola, and C. Camporeale, "A density-based algorithm for the detection of individual trees from LiDAR data," *Remote Sens.*, vol. 13, no. 2, p. 322, Jan. 2021.

[18] B. Coolen, P. J. Beek, D. J. Geerse, and M. Roerdink, "Avoiding 3D obstacles in mixed reality: Does it differ from negotiating real obstacles?" *Sensors*, vol. 20, no. 4, p. 1095, Feb. 2020.

[19] B. Fanini, A. Pagano, and D. Ferdani, "A novel immersive VR game model for recontextualization in virtual environments: The μVRModel," *Multimodal Technol. Interact.*, vol. 2, no. 2, p. 20, Apr. 2018.

[20] A. Ibañez-Etxeberria, C. J. Gómez-Carrasco, O. Fontal, and S. García-Ceballos, "Virtual environments and augmented reality applied to heritage Education. An evaluative study," *Appl. Sci.*, vol. 10, no. 7, p. 2352, Mar. 2020.

[21] Å. Fast-Berglund, L. Gong, and D. Li, "Testing and validating extended reality (xR) technologies in manufacturing," *Proc. Manuf.*, vol. 25, pp. 31–38, Jan. 2018.

[22] S. Jacob, V. G. Menon, and S. Joseph, "Depth information enhancement using block matching and image pyramiding stereo vision enabled RGB-D sensor," *IEEE Sensors J.*, vol. 20, no. 10, pp. 5406–5414, May 2020.

[23] Y. Zhang and A. Caspi, "Stereo imagery based depth sensing in diverse outdoor environments: Practical considerations," in *Proc. 2nd ACM/EIGSCC Symp. Smart Cities Communities*, Sep. 2019, pp. 1–9.

[24] B. Yang, H. Wen, S. Wang, R. Clark, A. Markham, and N. Trigoni, "3D object reconstruction from a single depth view with adversarial learning," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 679–688.

[25] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 605–613.

[26] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 77–85.

[27] A. Kulikajevas, R. Maskeliunas, and R. Damasevicius, "Adversarial 3D human pointcloud completion from limited angle depth data," *IEEE Sensors J.*, vol. 21, no. 24, pp. 27757–27765, Dec. 2021.

[28] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3D-R2N2: A unified approach for single and multi-view 3d object reconstruction," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 628–644.

[29] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4004–4012.

[30] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q.-X. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*.

[31] K. Greff, R. K. Srivastava, J. Koutnìk, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[32] 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," *IEEE Trans. Smart Grid*, vol. 10, no. 1, pp. 841–851, Jan. 2019.

[33] A. Kulikajevas, R. Maskeliūnas, R. Damaševičius, and E. S. L. Ho, "3D object reconstruction from imperfect depth data using extended YOLOv3 network," *Sensors*, vol. 20, no. 7, p. 2025, Apr. 2020.

[34] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.

[35] A. Kulikajevas, R. Maskeliūnas, R. Damaševičius, and S. Misra, "Reconstruction of 3D object shape using hybrid modular neural network architecture trained on 3D models from ShapeNetCore dataset," *Sensors*, vol. 19, no. 7, p. 1553, Mar. 2019.

[36] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2107–2115.

[37] Z. Mi, Y. Luo, and W. Tao, "SSRNet: Scalable 3D surface reconstruction network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 967–976.

[38] T. Ma, P. Kuang, and W. Tian, "An improved recurrent neural networks for 3D object reconstruction," *Int. J. Speech Technol.*, vol. 50, no. 3, pp. 905–923, Mar. 2020.

[39] X. Xu, F. Lin, A. Wang, Y. Hu, M.-C. Huang, and W. Xu, "Body-earth mover's distance: A matching-based approach for sleep posture recognition," *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 5, pp. 1023–1035, Oct. 2016.

[40] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9613–9622.

[41] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," 2016, *arXiv:1612.00593*.

[42] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "PCN: Point completion network," 2018, *arXiv:1808.00671*.

[43] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "AtlasNet: A papier-mâché approach to learning 3d surface generation," 2018, *arXiv:1802.05384*.

[44] M. Liu, L. Sheng, S. Yang, J. Shao, and S.-M. Hu, "Morphing and sampling network for dense point cloud completion," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 7, Apr. 2020, pp. 11596–11603.

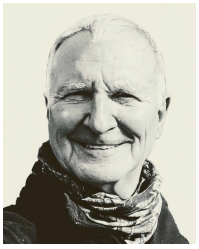
[45] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Trans. Vis. Comput. Graphics*, vol. 5, no. 4, pp. 349–359, Oct. 1999.

[46] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proc. 4th Eurographics Symp. Geometry Process.*, 2006, pp. 61–70.

[47] F. Williams, T. Schneider, C. Silva, D. Zorin, J. Bruna, and D. Panozzo, "Deep geometric prior for surface reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10122–10131.

[48] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 165–174.
[49] Y. Liao, S. Donne, and A. Geiger, "Deep marching cubes: Learning explicit surface representations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2916–2925.
[50] M. Liu, X. Zhang, and H. Su, "Meshing point clouds with predicted intrinsic-extrinsic ratio guidance," 2020, *arXiv:2007.09267*.
[51] A. Haque, B. Peng, Z. Luo, A. Alahi, S. Yeung, and L. Fei-Fei, "Itop dataset," Oct. 2016, doi: 10.5281/zenodo.3932973.
[52] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, "Real-time human pose tracking from range data," in *Computer Vision*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Germany: Springer, 2012, pp. 738–751.
[53] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan, "High-quality streamable free-viewpoint video," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 1–13, Jul. 2015.
[54] S. Ghorbani, K. Mahdaviani, A. Thaler, K. Kording, D. J. Cook, G. Blohm, and N. F. Troje, "MoVi: A large multi-purpose human motion and video dataset," *PloS One*, vol. 16, no. 6, 2001, Art. no. e0253157.
[55] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Computer Vision*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham, Switzerland: Springer, 2014, pp. 740–755.
[56] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and W. Zisserman, "The PASCAL visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.
[57] A. Dai, A. X. Chang, M. Savva, M. Halber, T. A. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3d reconstructions of indoor scenes," 2017, *arXiv:1702.04405*.
[58] S. Flaischlen and G. D. Wehinger, "Synthetic packed-bed generation for CFD simulations: Blender vs. STAR-CCM+," *ChemEngineering*, vol. 3, no. 2, p. 52, May 2019.
[59] J. Balado, P. Arias, H. Lorenzo, and A. Meijide-Rodríguez, "Disturbance analysis in the classification of objects obtained from urban LiDAR point clouds with convolutional neural networks," *Remote Sens.*, vol. 13, no. 11, p. 2135, May 2021.
[60] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2003, pp. 1–8.
[61] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2017, pp. 5105–5114.
[62] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black, "AMASS: Archive of motion capture as surface shapes," in *Proc. Int. Conf. Comput. Vis.*, Oct. 2019, pp. 5442–5451.
[63] F. Kainz, R. R. Bogart, D. K. Hess, "The OpenEXR image file format," in *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*. Boston, MA, USA: Addison-WesleyProfessional, 2004.
[64] Y. Jin, J. Zhang, M. Li, Y. Tian, H. Zhu, and Z. Fang, "Towards the automatic anime characters creation with generative adversarial networks," 2017, *arXiv:1708.05509*.
[65] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4401–4410.
[66] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. CVPR*, Jul. 2017, pp. 1125–1134.
[67] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2223–2232.
[68] C. Atapattu and B. Rekabdar, "Improving the realism of synthetic images through a combination of adversarial and perceptual losses," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–7.
[69] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.
[70] S. Wu, G. Li, L. Deng, L. Liu, D. Wu, Y. Xie, and L. Shi, "L_1-norm batch normalization for efficient training of deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 7, pp. 2043–2051, Nov. 2019.
[71] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.
[72] T. Gao, Y. Chai, and Y. Liu, "Applying long short term momory neural networks for predicting stock closing price," in *Proc. 8th IEEE Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Nov. 2017, pp. 575–578.

AUDRIUS KULIKAJEVAS received the B.Sc. and M.Sc. degrees from the Kaunas University of Technology, Kaunas, Lithuania, where he is currently pursuing the Doctoral degree in informatics. His main field of study is focused on objects occluded region reconstruction from a single perspective. In addition, he is conducting research related to machine learning and deep neural networks.



RYTIS MASKELIUNAS received the Ph.D. degree in computer science, in 2009. He is currently a Professor with the Department of Multimedia Engineering, Faculty of Informatics, and the Chief Researcher with the Centre of Real Time Computer Systems, Kaunas University of Technology. He is the author/coauthor of over 80 refereed scientific articles and serves as a reviewer/committee member for various refereed journals. His main scientific research interests include multimodal signal processing, modeling, development, and analysis of associative and multimodal interfaces, mainly targeted at elderly and people with major disabilities. He has won various awards/honors, including the Best Young Scientist Award in 2012 and the National Science Academy Award for Young Scholars of Lithuania in 2010.

ROBERTAS DAMASEVICIUS received the Ph.D. degree in informatics engineering from the Kaunas University of Technology, Lithuania, in 2005. He is currently a Professor with the Department of Applied Informatics, Vytautas Magnus University, Lithuania. He is the author of over 300 articles and a monograph published by Springer. His research interests include sustainable software engineering, human–computer interfaces, assisted living, data mining, and machine learning. He is also the Editor-in-Chief of the *Information Technology and Control* journal and has been a Guest Editor of several invited issues of international journals, including *Biomed Research International*, *Computational Intelligence and Neuroscience*, *Journal of Healthcare Engineering*, IEEE Access, and *Electronics*.

TOMAS KRILAVICIUS received the Ph.D. degree in computer science from the University of Twente, The Netherlands, in 2006. He is currently the Chief Scientist with the Baltic Institute of Advanced Technology, the Head of the Applied Informatics Department, and a Professor with Vytautas Magnus University, and a Stakeholder and the Co-Founder of TokenMill. His research interests include modeling of complex systems, machine learning, language technologies, scientific infrastructures, education, and management of research projects.