IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

3D Reconstruction of Common Objects from a Single RGB Image (May 2023)

Mark Crowley[1], Soumyabrata Dev[2]

[1]School of Electronic and Electrical Engineering, College of Engineering and Architecture, University College Dublin
[2]School of Computer Science, University College Dublin

Corresponding author: Soumyabrata Dev (e-mail: soumyabrata.dev@adaptcentre.ie).

Software developed during this project can be found at: https://github.com/markcrowley1/3D-Reconstruction

ABSTRACT This paper aims to explore existing methods in the field of Single View 3D Reconstruction and present an end-to-end software framework that enables 3D Reconstruction of common objects given only a single RGB image as an input. Single View 3D reconstruction refers to the recovery of 3D shapes from a single viewpoint. This is an ill-posed problem that has been subject to extensive research within the past decade. State of the art techniques implement deep neural networks which can attribute learned 3D shapes to objects observed from only a single viewpoint with a high degree of accuracy. A number of these methods are explored in this paper. The software framework that is presented implements a neural network inspired by PSG architecture to generate a set of points used to represent the shape of an object in 3D space. Postprocessing is carried out on this point set to obtain a 3D mesh. This 3D mesh supports texture mapping, which allows for the output of a textured 3D model given only a single RGB image as an input. Additionally, a generative deep learning network was developed that outputs a synthesized RGB image of an object from a novel viewpoint given an input image and a vector describing the desired viewpoint of the output image.

INDEX TERMS 3D Reconstruction, Computer Vision, Convolutional Neural Networks, Mesh, Point Cloud

I. INTRODUCTION

FIGURE 1. 3D Model Airplane

Give a person a single image like the one shown in figure 1 and they will likely be able to infer the shape of the object shown, and even guess what the object would look like from another angle. This is the essence of what Single View 3D Reconstruction hopes to achieve. This paper explores existing approaches in this field of research and presents an end-to-end software framework capable of outputting a textured 3D model based on a single RGB image, with a focus on the reconstruction of an object within a scene rather than a scene in its entirety.

In recent years, computer vision researchers have experimented with deep learning techniques in their solutions to this problem. These deep learning based methods can be classified into voxel based [1, 2], point cloud based [3, 4, 5], mesh based [6, 7, 8, 9] and implicit function based methods [10]. This paper will focus on a combination of point cloud based and mesh based approaches for 3D reconstruction, applying ideas discussed in papers by Fan et Al.[3] in particular. The decision to follow this type of approach was made based on its suitability for the applications of the proposed framework, as well as its relative simplicity to implement when compared to other methods.

One main use case of the proposed framework is a productivity tool for 3D art. This would have applications in numerous industries, including retail, game development and architecture. Two important features are identified for the proposed framework with these use cases in mind. The first is to support an industry-standard 3D representation for the reconstructed models. The second is to support aesthetically pleasing textured 3D models. Textures are 2D images that are mapped to the surface of a 3D model. This is an important feature as any model used in our target applications will likely be textured. Therefore, it was decided that the output of the software framework should be a 3D mesh. Meshes are the standard format in 3D modelling industries and support texture mapping with relative ease.

The framework was also designed with flexibility and good general performance in mind. A major drawback to many deep learning solutions is that they perform better when reconstructing categories of objects that they have trained on than categories they haven't previously seen. However, there has been research that looks into training networks for high performance on unseen object categories. It has been observed that generating a point cloud before converting that to a mesh can perform well on unseen categories [7, 9], and so this approach was used for the proposed software framework.

Deep learning methods that allow for complex texture prediction [11, 12] and mapping [6] are also investigated. A network inspired by Tatarchenko et. Al[11] was developed in addition to the end-to-end software framework which is capable of generating images of objects from novel viewpoints given an input image and a viewpoint transformation vector. This was pursued in the hope of implementing a highly detailed texture generation and mapping module in the end-to-end framework. The complexity of the texture mapping operation prevented this network from being integrated with the greater framework at this time, and instead a simpler texture generation and mapping module was included so that textured 3D meshes are still supported.

The primary goals of this paper are summarized as:

- Perform a literature review on recently developed and commonly used methods for Single View 3D Reconstruction.
- Design and implement an end to end software framework capable of taking a single RGB image as an input and generating a textured 3D mesh that represents the subject of the input image.

The rest of this paper is structured as follows:

- Literature Review – discussion about existing methods that enable Single View 3D reconstruction with a particular focus on research that contributed to this work.
- Methodology – the design of the proposed software framework and an additional view synthesis network is discussed in detail in this section.
- Experiments and Results – the performance of the newly developed software is analysed and discussed.
- Conclusion

II. LITERATURE REVIEW

A. POINT CLOUD GENERATION

PSG (Point Set Gen)[3] is a paper released in 2017 that explored a convolutional neural network architectures capable of outputting a point cloud given a single RGB image as an input. A point cloud is a set of points in 3D space used to represent an object. Notation is shown in equation 1.

$$P = \{(x_i, y_i, z_i)\}_{i=1}^{N}$$

Equation 1: Point Set Notation

The researchers present 2 architectures in the paper, the first "vanilla version" is a convnet with a sequential set of convolutional layers acting as an encoder, followed by a single fully connected layer acting as a decoder. This network was shown to be proficient at reconstructing objects with intricate structures. The second version of the network has 2 prediction branches and shows improved results on large, smooth surfaces that are common in natural objects. The network was trained on a set of images paired with ground truth point clouds. At the time of release, this paper was the first to research neural network architectures that output point cloud representations for 3D objects. They observed results that outperformed other state of the art methods such as 3D-R2N2[1]. In this research, we look to implement our own network capable of generating a 3D point set, with architecture inspired by the PSG network.

B. POINT CLOUD TO MESH CONVERSION

A 3D mesh is a collection of vertices, edges and faces that define the shape of a polyhedral object. There are several computational algorithms that are capable of generating 3D meshes given a point set as an input, including the Ball Pivoting Algorithm[15], Poisson Surface Reconstruction algorithm[16] and the Alpha Shape algorithm[17]. The Alpha Shape algorithm is selected as the method of point set to mesh conversion in the proposed framework (chosen after experimentation). It is a generalization of a convex hull. While this algorithm is not capable of producing as detailed meshes as the Ball Pivoting or Poisson Surface Reconstruction algorithms, it is more suited to the needs of the proposed framework. The set of points returned from the point set generations network will likely be quite small and the points' surface normals will be unknown, which will likely yield poor results in the case of the other 2 algorithms.

C. TEXTURE PREDICTION AND MAPPING

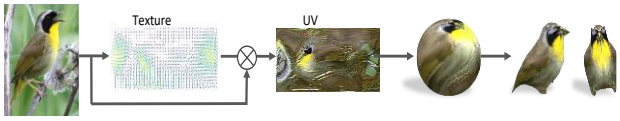

FIGURE 2. CMR Texture Prediction and Mapping

Kanazawa et. Al[6] presented a software framework, named CMR (Category Specific Mesh Reconstruction), that utilised a deep neural network trained on 2D images of specific shape categories capable of producing textured 3D meshes. CMR was shown to produce good quality textured 3D meshes for objects belonging to previously learned categories of objects. The texture prediction and mapping module implemented in this framework was inspired by view synthesis by appearance flow[12].

$$L_{texture} = \sum_{i} dist(S_i \odot I_i,\ S_i \odot R)$$

$$L_{dt} = \sum_{i} \sum_{u,\ v} G\left(D_{S_i}; F_i\right)(u, v)$$

Equation 2 and 3: Texture Loss and Distance Loss

The network proposed texture prediction module was implemented using a neural network that was trained on two loss functions. These were identified as a texture loss function and a distance loss function. The texture loss encourages the texture to match the foreground image and the distance loss specifically encourages pixel selection from the foreground of the image.

To determine the texture loss the distance matrix is computed for the element-wise product of an instance segmentation Si and an input image I_i, and the element-wise product of Si and the predicted textured mesh rendering R. This is done for all training data i and summed. The distance loss function looks to minimise the sum of the bilinear samples G of the distance transform of the foreground mask D_{S_i} according to the flow field F for every image for every pixel defined by (u, v).

The main drawback to the method used by CMR is that it is only effective at predicting textures for categories of shapes it has trained on, and there are multiple reasons for this. The mesh generation and texture prediction networks implemented in CMR are trained on images of specific object categories that have been labelled with semantic key points (e.g. the beak, legs or tail of bird). During this initial training, a template 3D mesh with semantic key points is generated for a particular category. This mesh can later be deformed but keeps a similar basic shape. This is allows for ease of mapping of the generated texture to the 3D mesh.

The framework that this paper has proposed aims to be more flexible than this by generating a set of points for each input image from scratch. Therefore, the 3D mesh will not have any semantic key points and will require some form of extra analysis to enable the mapping process. This could potentially be achieved by using a combination of networks such as PointNet[18, 19], which provides analysis of point clouds, and ResNet[20], which extracts feature maps from 2D images.

D. OTHER APPROACHES

Although beyond the scope of this paper, differential renderers and generative adversarial networks (GANs) are two areas of research that show impressive results in the field of single view 3D reconstruction.

Differentiable renderers [21, 22, 23] allow 2D image pixels to be related back to 3D properties of a scene. 3D scenes are predicted by a neural network before 2D renders of this scene are created. A loss function is then created by comparing the newly created 2D image to the ground truth 2D image. This is then used to train the network. GAN networks [24, 25, 26] have 2 key components: a generator and a discriminator. The generator learns to create plausible instances of data, which become negative training examples for the discriminator. The discriminator is trained to distinguish between the generator's fake data and the real data. The generator attempts to create instances that are indistinguishable from real data. There have been recent publications that utilize this type of network design in single view 3D reconstruction[24].

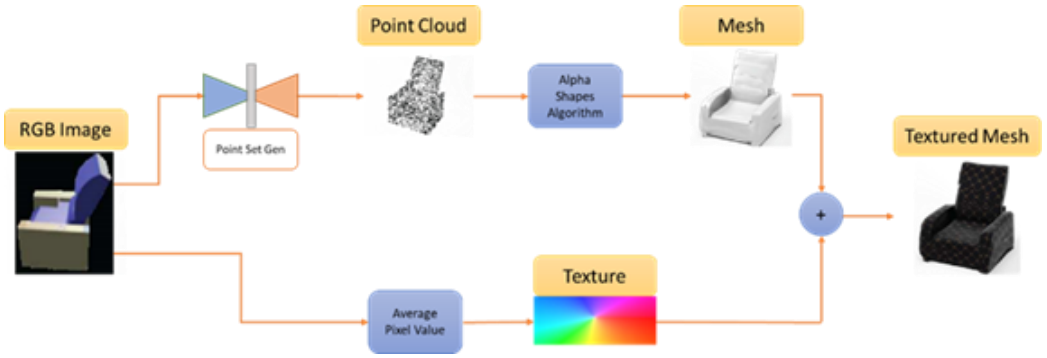

FIGURE 3. Software Framework Architecture

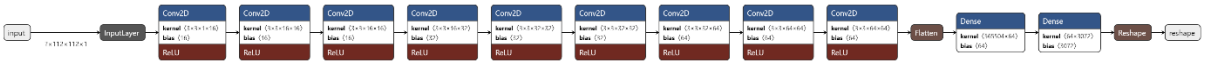

FIGURE 4. Point Set Generation Network Architecture

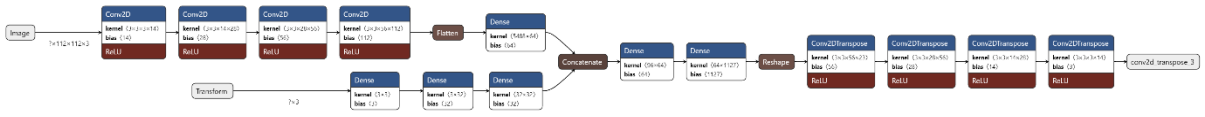

FIGURE 5. View Synthesis Network Architecture

III. METHODOLOGY

An end to end software framework was designed and implemented that takes a single RGB image as an input and generates a textured 3D mesh as an output. This section outlines the overall system design and discusses the details of individual modules within the greater software framework.

A. SYSTEM ARCHITECTURE

The software framework, shown in figure 3 above, is comprised of 3 key modules: a point cloud generation module, a point cloud to mesh conversion module and a texture generation module. A neural network is used to generate a point set from the input image, which is then converted to a mesh using the Alpha Shapes algorithm[12]. A texture of uniform color is generated for the mesh using a basic pixel sampling technique. This texture is then applied to the 3D mesh.

Once the textured 3D model has been generated, it is saved using obj and mtl file formats to capture the shape and texture of the model respectively. The model should then be accessible through standard 3D modelling software such as Blender or Windows 3D Viewer.

B. POINT CLOUD GENERATION

The point cloud generator is convolutional neural network comprising of an encoder and decoder. The encoder is made up of a sequence of convolution layers. Its purpose is to take an image as an input and convert it into a high dimensional feature vector. The decoder is made up of a fully connected layer which interprets the feature vector and converts it to a 1 dimensional list of values. A reshape operation is then used to convert this 1D vector into a 3 dimensional matrix. This way the data can be interpreted as a set of points in 3 dimensional space. The network outputs a point set of size N=1024, which allows for good representation of most common objects according to Fan et al.[3].

This network is trained on data from the ShapeNet[13] and ShapeNetRendering[1] datasets. ShapeNet contains a large amount of textured mesh models and ShapeNetRendering contains renderings of a number of these models from different camera poses. ShapeNet has been used in multiple notable works in the past [3, 7] and is well suited to training this point set generating network. Poisson disk sampling was used to generate ground truth point clouds from the ShapeNet meshes. The ground truth point clouds were then paired with images from ShapeNetRendering and oriented so that they match the image. These point set and image pairings could then be used to train the network, passing the images as inputs and the ground truth point clouds as the expected output. Pixel values are normalized so that they are restricted between 0 and 1 before they are passed as inputs to the network. This is an important step as it ensures that features of the input are on a similar scale, which improves the training stability of the model. The dimensions of the ShapeNet models are already normalized, so no further processing is required for the point sets once they are sampled.

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{x \in S_2} \min_{y \in S_1} \|x - y\|_2^2$$

Equation 4: Chamfer Distance Loss Function

The loss function that is used to train the network attempts to minimize the chamfer distance between the predicted point cloud and the ground truth point cloud. For each point (x) in both the predicted and ground truth point clouds (S¬1 and S2) the chamfer distance algorithm finds the closest point in the opposite point cloud (y) and sums the square of the distance between them. This approach was inspired by point set gen[3].

Note that for this training data, all objects are located in the center of the images. Therefore, we can expect that the model will perform worse if the input image contains an object that is not in the center of the frame. Poor performance is also expected on raw real world images due to additional noise, as the network is trained on images with empty backgrounds. Real world images should be provided with a segmentation mask to indicate the scope of the object and remove unwanted noise from the image.

C. MESH AND TEXTURE GENERATION

The Alpha shapes algorithm [17] is used to construct the surface of the mesh given the predicted point set. Experiments were performed with the ball pivoting algorithm[15] as well as Poisson surface reconstruction[16] but the Alpha shapes algorithm was observed to provide the best results with the information provided by the point set generation network – a relatively small number of points without surface normals. Once the mesh has been created a Taubin filter[27] is used to smooth the mesh for a number of iterations. Python's Open3D module[28] was used to carry out these algorithms on the point set generated from our network.

A basic texture generation module is included in the framework that functions by sampling pixels from the center of the input image and calculating the average pixel value. A uniform texture is then created with this average color. This texture is then applied all across the surface of the mesh, providing a fully textured output whose color is somewhat representative of the object in the input image.

D. VIEW SYNTHESIS NETWORK

The texture generation module implemented in the end-to-end framework is simplistic, and while the texture it provides is somewhat aesthetically pleasing and representative of the original image, it is not very accurate to the ground truth. Further investigation into texture generation methods was carried out and view synthesis network inspired by Tatarchenko et. Al[11] was developed. This was not implemented in the end to end framework due to the complexity of developing a corresponding texture mapping method.

The network uses an autoencoder architecture, and expects 2 inputs: an image and a vector θ that specifies how the image should be transformed. The image is passed through an encoder made up of a sequence of convolution layers and the vector θ is passed to a sequence of fully connected layers. The decoder up-convolves the high dimensional feature vector from the decoder and outputs an RGB image. This synthesised image should show the object in the original from the viewpoint specified by θ. The architecture of this network is directly inspired by Tatarchenko et. Al[11], and is a commonly used structure in many generative deep learning networks.

$$\theta = (azimuth, elevation, distance)$$

Equation 5: Components of vector θ

The vector θ has 3 elements representing the change in azimuth, elevation and distance between the input and target output images. The azimuth and elevation are calculated as degrees, and then normalized so that they will always lie between -1 and 1. The changes in distance between the rendered images is relatively small (between the values of 0 and 1) in the vast majority of cases. This is because the dimensions of the ShapeNet models are already normalized.

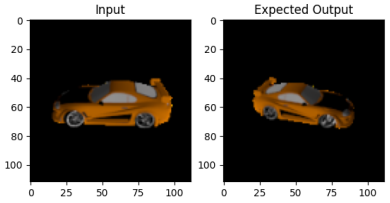

FIGURE 6. Input and Expected Output in Training Tuple

$$L = \sum_i \|y_i - \hat{y}_i\|_2^2$$

Equation 6: Squared Euclidean Distance Loss Function

The training dataset for this network was compiled directly from the ShapeNetRendering dataset. Input and expected output images were paired along with the corresponding θ values. These training tuples were then stored in batches which were loaded into memory during training. As in the case of the PSG network, the input and output images are normalised before they are used in training. The network was trained by minimising the squared Euclidean loss function shown in equation 6. The symbol y_i represents the target output image and $\hat{y}_i$ represents the prediction from the network.

The resulting network can be expected to suffer from some of the same limitations as the PSG network, including the requirement for the subject of the image to be located in the centre, and poor performance on real world images with noisy backgrounds. However, this network does provide a good template for a texture generation module. A drawback of this design is that it does not provide a single texture that can be mapped to the entirety of a 3D mesh, but it is capable of predicting unknown textures which have the potential to be mapped to subsections of a mesh. If done iteratively, this could cover an entire mesh.

IV. EXPERIMENTS AND RESULTS

A set of experiments was carried out to test the performance of the point set generation network, the view synthesis network and the end-to-end software framework. The results of these experiments are analysed both numerically and visually. All networks were trained and tested on models and images from the ShapeNet[13] and ShapeNetRendering[1] datasets.

A. PSG NETWORK PERFORMANCE

After experimenting with a number of different network architectures and methods of training, a prototype of the end to end software framework was completed. This prototype is capable of taking a single RGB image as an input, generating a point cloud and converting this into a textured mesh output. This was implemented with a convolutional neural network (architecture shown in figure 4) that was trained on 50k image/point set pairs belonging to 5 different object categories (Chairs, Couches, Benches, Tables, and Lamps) for 20 epochs.

The performance of the network that generates the point cloud, is investigated in this section. The effects of overfitting in the model are also investigated and possible avenues of improvement to the model as a whole are discussed. The loss function used to train the network seeks to minimise the chamfer distance, and this value is used to measure performance in testing. The lower the chamfer distance (loss) evaluated during testing, the better the model (at least in general). It is widely used as a standard measurement of point cloud accuracy. The chamfer distance has no unit of measurement in this case, but all models are normalised so the distances are relevant to one another.

Category Type	Seen	Unseen (Planes)
Test Loss	0.0070	0.007751

Table 1: 5 Category Network Performance

Figure 7 shows the input images, ground truth point clouds and corresponding predictions made by the point set generation network for categories of objects that it has previously seen. The average chamfer distance (test loss) was calculated on these image/ground truth point cloud pairs, and found to be a value of 0.0070. Figure 8 shows the predictions that the network has made on images of planes, noting that the network has never trained on images of planes before. The average chamfer distance in this case is 0.007751.

An interesting observation here is that while the average chamfer distance in each case is not so different, the visual quality of the point set is far superior when the network is

asked to reconstruct objects that it has already seen. Further experiment shows that the network performs far better when trained examples of planes, providing better visual outputs and better numerical testing results (see figure 11 and table 2).

The general shapes of the 5 object categories that this network was trained are far different to planes, and so it was expected that performance would not be as impressive. Although not recognisable as a plane, the network does succeed in providing rough 3D representations for the images in figure 8, and tends to orient them correctly.

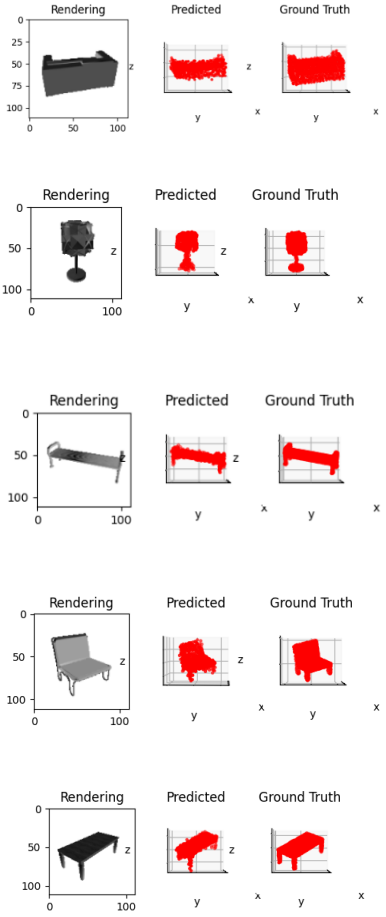

FIGURE 7. Point Cloud Predictions on Objects from Previously Seen Categories

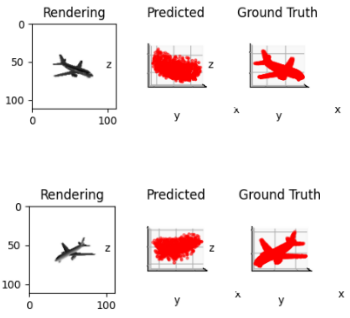

FIGURE 8. Predictions on Objects from Previously Unseen Categories

B. PSG NETWORK OVERFITTING ANALYSIS

Epochs	5	10	20
Test Loss	0.001436	0.001371	0.001756

Table 2: Test Loss across Training Epochs (Plane Only Network)

Two models were trained on 50k image/point cloud pairs including 5 different object categories (Chairs, Couches, Benches, Tables, and Lamps) for 20 epochs. One of these models incorporated a dropout layer and weight decay parameters while the other did not, and the results were compared. Another set of networks were trained only on planes for 5, 10 and 20 epochs. The performance of these models was tested and the inclusion of a dropout layer was also experimented with in certain models.

It was observed that validation and testing loss failed to decrease with training loss when the number of epochs became large, seen in the case of the 5 category and single category networks (figure 9 and table 2). This suggests that the models are overfitting to the training data, which should generally be avoided. Numerically worse testing results were observed on unseen data when the number of epochs was increased above a certain point on both 5 category networks and the plane only network. The inclusion of dropout layers and weight decay was experimented with while training the 5 category network. The intention of adding these to the network architecture is to reduce overfitting[29]. However, as seen in figure 10 this failed to reduce the validation loss further and so it was not included in the final design.

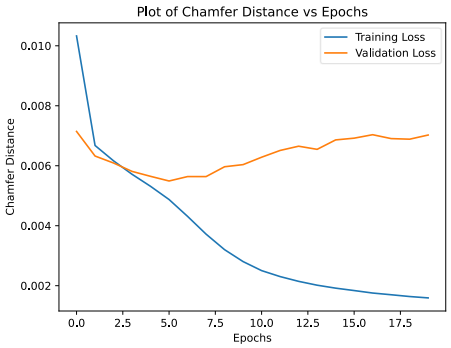

FIGURE 9. 5 Category Network Training and Validation Loss

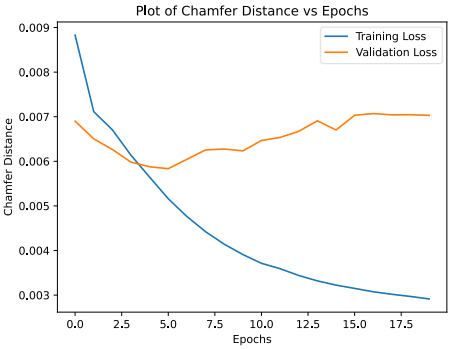

FIGURE 10. 5 Category Network Training and Validation Loss

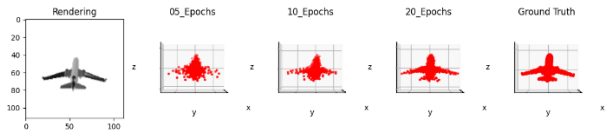

FIGURE 11. Testing Outputs across Training Epochs (Plane Only Network)

Contrary to what the numerical data would suggest, qualitatively better results were observed when the models were trained for a larger number of epochs. Looking at the outputs from the network, recognisable features are more clearly defined in the point cloud and there are fewer points that lie far outside the boundary of the intended shape (see figure 11).

The network also has a tendency to commit more points to the centre of the 3D shape at lower epochs which is effective at minimising the chamfer distance but does not improve the visual quality of the output. For the purposes of this application, it is more desirable that the network is somewhat overfit if the output is a more defined shape, as this is more aesthetically pleasing and provides a better input to the mesh generation module.

The downside of overfitting in this case is that the model will likely perform worse on previously unseen categories of objects, and may include features it has learned from seen categories where they do not exist. To further improve the visual and practical quality of the output, as well as reduce overfitting, the number of training data pairs should be increased and increasing the depth of the network could also be experimented with.

C. VIEW SYNTHESIS NETWORK PERFORMANCE

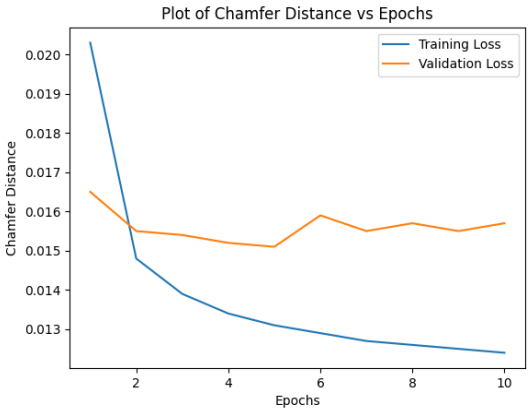

FIGURE 12. Training and Validation Loss for View Synthesis Network

The view synthesis network developed for this project was trained on 386k training tuples. Each tuple contained an input image, input vector θ and a target output image. Image pairs and corresponding θ vectors were obtained from a set of 700 car models from the ShapeNet dataset. 100 car models were used for validation during training and 200 car models were used for testing.

Initially the network was trained for 10 epochs but overfitting was observed in this case. It was decided that the model should be trained for 5 epochs given the improvement in performance on validation data up to that point. Even at this point the model is still overfitting, and training for just 2 epochs could be experimented with, as it is at this point where the training loss becomes noticeably less than the validation loss.

Squared Euclidean Distance	0.016

Table 3: Test Loss for View Synthesis Network (Square Euclidean Distance)

Once the network is trained it is evaluated on the training dataset. The metric used for evaluation in this case is the squared Euclidean distance, which is the also the loss function used during training. The average squared Euclidean distance across all test sets is found to be 0.016. This value agrees with the validation loss that is plotted above. Judging from the visual quality of the results the network does a good job of capturing basic features of the input (for example the colour, wheels, windows and overall shape of the cars) when it is of a similar shape to the typical training example. It is also effective at synthesising the image of the car from the expected viewpoint – we can see that the objects in the target output and network outputs have similar orientations. However, when a car's shape is abnormal (for example a monster truck) the network fails to return an accurate prediction of the target output image.

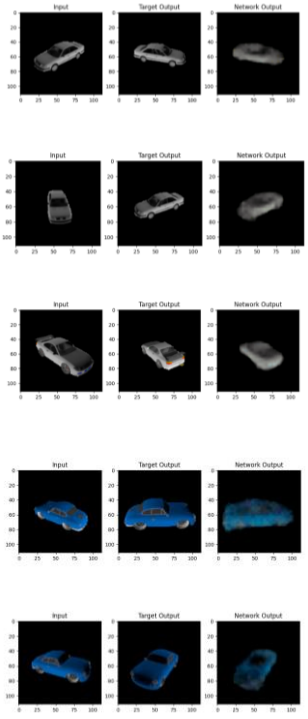

FIGURE 13. View Synthesis Network Good Quality Predictions

D. FRAMEWORK PERFORMANCE

The first 6 images in figure 14 show the textured mesh outputs (right) that the network is capable of generating based on a corresponding input image (left). In general, the conversion from point cloud to mesh tends to lose some of the finer details of the models, for example the arm rests on the bench. This is likely due to only a small number of points being used represent these smaller, intricate structures. The shapes of the meshes tend to be bit distorted, which is an artefact caused during the generation of the point set. That said, the meshes are still quite representative of the objects in the input images. The back of the couch in the first example is not so clearly defined, but a small ridge where it should be is defined.

In the case of the lamp (bottom), the base is unnaturally elongated, but the lamp shade is clearly represented although somewhat misshapen. The textures generated for the models are quite representative in these simple cases, and meshes themselves are observed to be smooth and without too many gaps due to the post processing techniques that were applied (Taubin filter). On a side note, It is also verified that the model can be saved successfully using obj and mtl file formats, and opened for viewing in standard 3D modelling software.

Due to their noisy backgrounds, we can expect that the point set generation network will be unable to generate accurate outputs for raw real world images. This is because the network has been trained on images with no additional noise. However, the network and framework as a whole were tested on a small amount of computer synthesised images that represent a practical use case for the software – a productivity tool for the creation of 3D art. The final image in figure 14 shows the framework's output for an artistic impression of the table generated by the authors. The framework is able to generate a textured mesh that is reasonably representative of the input image, notably including the table top and table legs.

While results are not comparable with state of the art techniques, this output shows that the framework has been developed such that it is capable of functioning as intended on images found outside of the training data source – ShapeNet. It also fair to state that a functional prototype incorporating all elements of the original design has been successfully completed.

FIGURE 14. **View Synthesis Network Good Quality Predictions**

E. DISCUSSION

The point set generation network was tested on a set of unseen data taken from the ShapeNet dataset. Qualitative and quantitative analysis was carried out for the network predictions, and it is fair to say that it performs well when asked to reconstruct an object in a 2D image given that the object belongs to a category that the network has seen during training. When asked to reconstruct an object that belongs to a category it has not trained on, it was not capable of generating a representative 3D shape. The network's capability to construct representative 3D shapes for objects belonging to seen and unseen categories could both be improved by increasing amount of data used in training. The number of convolution layers in the encoder, and the number of nodes in the dense layers of the decoder could also be increased to improve performance. In its current iteration, the network is capable of generating point sets of high enough quality to form reasonably accurate and smooth meshes, which is its intended functionality in the end-to-end framework that this paper presents. It provides this functionality across 5 different categories of common objects – benches, chairs, couches, tables and lamps.

The end-to-end software framework is also tested on a number of models from the ShapeNet dataset and the visual quality of the results was analyzed. Due to the limitations of the point set generation network, these models all belonged to object categories that the network had trained on. In general, the software was shown capable of generating representative, smooth meshes from these ShapeNet examples. It was observed that the output supports a uniform color texture that is automatically generated from the input image. Furthermore, the framework was tested on images of tables synthesized by the authors in commonly used 2D art programs and was observed to produce quite representative 3D reconstructions. It was verified that the predicted model could imported into standard 3D modelling software with ease, which is of high importance if this software is to be used in industry.

The view synthesis network was tested on an unseen set of images of cars from the ShapeNetRendering dataset. The performance was numerically evaluated based on the average squared Euclidean distance between predicted images and target images. The visual quality of the output was also assessed. Although blurry, the predicted outputs were oriented correctly and captured the general outline of the cars in most images. The colors of the predicted output were observed to sometimes be slightly different than the target image, but in general were quite similar. This could potentially be due to the model overfitting to colors in the testing dataset. Certain features such as car windows and wheels could also be seen in many examples.

This experimentation verifies that the overarching goal of the project has been a achieved – a software framework capable of taking a single RGB image and generating a textured 3D model has been successfully developed. Additionally, a view synthesis network has been implemented that lays the foundation for a more complex texture generation module to be implemented into the end-to-end software framework.

V. CONCLUSION

Two generative deep neural networks were successfully developed, one capable of generating a point cloud from a single input image, and the second capable of synthesizing an alternate view of an object given an input image and a vector specifying the change in viewpoint. Each network was shown to produce qualitatively impressive results on unseen testing data. The effects of overfitting in the PSG network were also investigated, where it was counterintuitively observed that the model tends to return more visually accurate results when it has been overfit to the training data.

This paper presents an end-to-end software framework that is capable of generating a textured 3D mesh given a single RGB image as an input. This was achieved by implementing a PSG based network to generate a point cloud, along with computational techniques that convert this point cloud to a 3D mesh and support basic texture creation and mapping. The framework was observed to produce 3D outputs that were visually representative of test images, and it was also shown capable of generating outputs for practical applications.

The software developed throughout this project is open source and may function as a useful resource or template to others who wish to develop software for computer vision and 3D reconstruction. This is available at https://github.com/markcrowley1/3D-Reconstruction.

REFERENCES

[1] Choy, Christopher B., et al. "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction." European conference on computer vision. Springer, Cham, 2016.

[2] Xie, Haozhe, et al. "Pix2vox: Context-aware 3d reconstruction from single and multi-view images." Proceedings of the IEEE/CVF international conference on computer vision. 2019.

[3] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 605–613

[4] Z. Han, C. Chen, Y.-S. Liu, and M. Zwicker, "DRWR: A differentiable renderer without rendering for unsupervised 3D structure learning from silhouette images," in International Conference on Machine Learning, 2020.

[5] C. Chen, Z. Han, Y. shen Liu, and M. Zwicker, "Unsupervised learning of fine structure generation for 3d point clouds by 2d projection matching," in Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2021.

[6] Kanazawa, Angjoo, et al. "Learning category-specific mesh reconstruction from image collections." Proceedings of the European Conference on Computer Vision (ECCV). 2018

[7] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mach ˆ e approach to learning 3d surface generation," in ´ Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 216–224

[8] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2mesh: Generating 3d mesh models from single rgb images," in

Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 52–67.

[9] Yang, Xianghui, Guosheng Lin, and Luping Zhou. "ZeroMesh: Zero-shot Single-view 3D Mesh Reconstruction." arXiv preprint arXiv:2208.02676 (2022).

[10] Mescheder, Lars, et al. "Occupancy networks: Learning 3d reconstruction in function space." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.

[11] Tatarchenko, Maxim, Alexey Dosovitskiy, and Thomas Brox. "Multi-view 3d models from single images with a convolutional network." Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14. Springer International Publishing, 2016.

[12] Zhou, T., Tulsiani, S., Sun, W., Malik, J., Efros, A.A. (2016). View Synthesis by Appearance Flow. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science(), vol 9908. Springer, Cham.

[13] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su et al., "Shapenet: An informationrich 3d model repository," arXiv preprint arXiv:1512.03012, 2015

[14] Mo, Kaichun, et al. "Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.

[15] Bernardini, Fausto, et al. "The ball-pivoting algorithm for surface reconstruction." IEEE transactions on visualization and computer graphics 5.4 (1999): 349-359.

[16] Kazhdan, Michael, Matthew Bolitho, and Hugues Hoppe. "Poisson surface reconstruction." Proceedings of the fourth Eurographics symposium on Geometry processing. Vol. 7. 2006.

[17] Edelsbrunner, Herbert, and Ernst P. Mücke. "Three-dimensional alpha shapes." ACM Transactions On Graphics (TOG) 13.1 (1994): 43-72.

[18] Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[19] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," Advances in neural information processing systems, vol. 30, 2017

[20] Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." Thirty-first AAAI conference on artificial intelligence. 2017.

[21] Loper, Matthew M., and Michael J. Black. "OpenDR: An approximate differentiable renderer." European Conference on Computer Vision. Springer, Cham, 2014.

[22] Kato, Hiroharu, Yoshitaka Ushiku, and Tatsuya Harada. "Neural 3d mesh renderer." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[23] Chen, Wenzheng, et al. "Learning to predict 3d objects with an interpolation-based differentiable renderer." Advances in Neural Information Processing Systems 32 (2019).

[24] Bhattad, Anand, et al. "View generalization for single image textured 3d models." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

[25] Wang, Ting-Chun, et al. "High-resolution image synthesis and semantic manipulation with conditional gans." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[26] Heusel, Martin, et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium." Advances in neural information processing systems 30 (2017).

[27] Taubin, Gabriel, Tong Zhang, and Gene Golub. "Optimal surface smoothing as filter design." Computer Vision—ECCV'96: 4th European Conference on Computer Vision Cambridge, UK, April 15–18, 1996 Proceedings, Volume I 4. Springer Berlin Heidelberg, 1996.

[28] Zhou, Qian-Yi, Jaesik Park, and Vladlen Koltun. "Open3D: A modern library for 3D data processing." arXiv preprint arXiv:1801.09847 (2018).

[29] Ying, Xue. "An overview of overfitting and its solutions." Journal of physics: Conference series. Vol. 1168. IOP Publishing, 2019.

Mark Crowley is pursuing an ME in Electronic and Computer Engineering at UCD. His research project involves the development of generative deep neural networks for 3D reconstruction and is supervised by Dr Soumyabrata Dev. His main areas of interest are in data science, deep learning and computer vision.

Soumyabrata Dev is an Assistant Professor in the School of Computer Science at University College Dublin. He obtained his PhD from Nanyang Technological University (NTU) Singapore in 2017. From August-December 2015, he was a visiting doctoral student at Audiovisual Communication Laboratory (LCAV), École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. He worked at Ericsson as a network engineer from 2010 to 2012. Prior to this, he graduated summa cum laude from National Institute of Technology Silchar, India with a BTech in 2010. His research interests include remote sensing, statistical image processing, machine learning, and deep learning.