Assisted Modeling:

Leveraging Human Analysis of Topology and Symmetry in Convolutional Neural Networks to Generate Improved 3D Mesh Models from Single RGB Images

Author: Colton Bishop
Advisor: Jia Deng, Ph.D.

Junior Independent Work, Spring 2019

Abstract

This paper proposes and evaluates a topology and symmetry-aware deep learning architecture that incorporates user analysis to improve the performance of a state of the art mesh modeling implementation: Pixel2Mesh. In particular, this new system seeks to generate more accurate and appealing mesh models from single RGB images by considering planes of symmetry, complexity demarcations, and domain shape specifications input by the user. Experimentation shows that these additional forms of human analysis can, in some instances, effectively inform the model during and after processing to produce more accurate 3D shapes. Specifically, this paper explores how these user inputs work to help Pixel2Mesh overcome it's object complexity limitations, perspective bias, and difficulties with back estimation.

Table of Contents

1 Introduction

The technique of aiding deep learning approaches with additional human input to tackle difficult computer vision problems has proven itself to be a very effective approach to solving many tasks: consider, for instance, the Microsoft PowerPoint photo-editing feature "Remove Background" that takes as input an image of an object with a noisy background along with a set of user selected points that are to be included in the object. The software then automatically masks the object (or returns an image of the object with the background removed).[1] In this paper, we explore ways in which additional human input might be leveraged to tackle another difficult computer vision problem: the task of generating three-dimensional representations of objects from two-dimensional images.

This modeling task is a problem that has been confronted and studied extensively for several decades.[2] Recently, new developments and advancements in deep learning have allowed us to come closer than ever before to accomplishing this goal. However, despite this progress there are still significant shortcomings present even in the most cutting-edge approaches that seek to generate three-dimensional models from two-dimensional images. This paper examines one such cutting-edge

1 "Remove the Background of a Picture." Office Support, support.office.com/en-us/article/remove-the-background-of-

2 "Multi-Image 3D Reconstruction Data Evaluation." Journal of Cultural Heritage, Elsevier Masson, 17 Jan. 2013, www.sciencedirect.com/science/article/abs/pii/S1296207412001926.

approach (known as Pixel2Mesh[3]), and explores the ways in which additional user input might be used to address some of its most serious weaknesses.

1.1 Motivation

The applications of an effective and efficient system to produce three-dimensional models from two-dimensional images are manifold. Such a system could have positive implications for a wide array of industries and fields such as the visualization of medical imaging[4], architectural modeling[5], graphics and video game design[6], or even the analysis of astronomical data[7].

From an economic vantage point, if the time and resources needed to manually analyze properties of the object before modeling would be less than the time and resources that would need to go into post-processing of poorly modeled objects, this approach could effectively increase the efficiency and reduce the costs for any business that has applications for 3D reconstruction, or constructing 3D models from 2D images. Secondly, from an academic vantage point beyond potential industrial applications, this project seeks to research and analyze how successful different forms of human input are at improving the performance of convolutional

3 Wang, Nanyang, et al. "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images." Computer Vision – ECCV 2018 Lecture Notes in Computer Science, 2018, pp. 55–71., doi:10.1007/978-3-030-01252-6_4.

4 "Three-Dimensional Reconstruction for Medical-CAD Modeling." Taylor & Francis, www.tandfonline.com/doi/abs/10.1080/16864360.2005.10738392.

5"Instant Architecture." ACM Transactions on Graphics (TOG), ACM, dl.acm.org/citation.cfm?id=882324.

6 "US20080246759A1 - Automatic Scene Modeling for the 3D Camera and 3D Video." Google Patents, Google, patents.google.com/patent/US20080246759A1/en.

7 "Visualizing Space Science Data in 3D." Visualizing Space Science Data in 3D - IEEE Journals & Magazine, ieeexplore.ieee.org/abstract/document/544066.

neural networks for 3D modeling. Furthermore, as will be explained further in the subsequent sections, this paper seeks to demonstrate the virtues of a mesh-based approach to 3D modeling (as opposed to more common base representations such as the 3D volume or point-cloud representations) by taking advantage of unique features of the mesh representation to leverage user input and improve reconstruction.

1.2 Context for 3D Reconstruction

As noted previously, there has been much work done toward using deep learning to generate three-dimensional representations of objects from two-dimensional images (henceforth referred to as 3D reconstruction).

1.2.1 A History of the Problem and Related Work

As Dr. Peter Strum, Deputy Scientific Director of the French National Institute for Research in Computer Science and Control (INRIA), explains in his presentation "On the History of Image-Based 3D Modeling," 3D geometric reconstruction emerged as a discipline in the early 1980's.[8] The earliest approaches relied on epipolar geometry (or the geometry of stereo vision) to reconstruct an object from several differently angled images.[9] As the discipline advanced into the 1990's, improved numerical algorithms and more systematically robust statistical approaches

8 Sturm, Peter. "On the History of Image-Based 3D Modeling." INRIA. INRIA. http://gurdjos.perso.enseeiht.fr/vision24janvier2014/Sturm-24jan-TLSE.pdf

9 "Computing Matched-Epipolar Projections." Computing Matched-Epipolar Projections - IEEE Conference Publication, ieeexplore.ieee.org/abstract/document/341076.

emerged until finally, near the turn of the century, we achieved what Dr. Strum refers to as “true” 3D surface modeling.[10]

Though effective, these multi-view reconstruction techniques were limited by their dependence on multiple images from several different vantage points to reconstruct shapes, and they were unable to effectively model objects that were reflective, transparent, or textured.

Since then, as a consequence of the rapid development of deep learning techniques, optimizations to convolutional neural network architectures, and the emergence of extensive datasets like ShapeNet that map 2D images to ground truth 3D shapes, many modern attempts to address this problem have relied heavily on training machine learning models to do the heavy lifting and as a result have achieved unprecedented levels of accuracy in single view reconstruction. Today, some of the most successful state of the art approaches include the 3D Recurrent Reconstruction Neural Network (3D-R2N2)[11] which models shapes using a 3D volume representation, the Point Set Generation Network (PSG)[12] which models shapes using a point cloud representation, the TL-embedding network architecture[13] which

10 Sturm, Peter. “On the History of Image-Based 3D Modeling.” INRIA. INRIA. http://gurdjos.perso.enseeiht.fr/vision24janvier2014/Sturm-24jan-TLSE.pdf

11 Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In: ECCV (2016)

12 Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3d object reconstruction from a single image. In: CVPR (2017)

13 https://arxiv.org/pdf/1603.08637.pdf

models shapes using a 3D voxel representation, and the Pixel2Mesh system[14] which models shapes based on a mesh (or waveform) representation.

1.2.2 The Debate Over Representation

A major point of contention in the discipline of 3D reconstruction is that of representation: what is the most effective way to represent our 3D shape during generation? As Girdhar et. al. suggests, a good vector representation of an object "should be generative in 3D, in the sense that it can produce new 3D objects, as well as be predictable from 2D, in the sense that it can be perceived from 2D images."[15] Over the years, there have been proponents and pilots for several different approaches, including 3D voxel representation, octree representation, 3D volume representation, geometric image representation, and the point cloud representation.

The vast majority of implementations, such as the TL-embedding network architecture or the 3D Recurrent Reconstruction Neural Network, are "restricted by the prevalent grid-based deep learning architectures"[16] and output their models as low resolution 3D voxels or octree representations which, in order to produce accurate models, require substantially more computational power and memory than most modern GPUs can often provide. Furthermore, as Fan et. al. describes, these representations "obscure the natural invariance of 3D shapes under geometric

14 Wang, Nanyang, et al. "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images." Computer Vision – ECCV 2018 Lecture Notes in Computer Science, 2018, pp. 55–71., doi:10.1007/978-3-030-01252-6_4.

15 Girdhar, Rohit. Learning a Predictable and Generative Vector Representation for Objects. Robotics Institute, Carnegie Mellon University, arxiv.org/pdf/1603.08637.pdf.

16 Wang, Nanyang, et al. "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images." Computer Vision – ECCV 2018 Lecture Notes in Computer Science, 2018, pp. 55–71., doi:10.1007/978-3-030-01252-6_4.

transformations," causing structural instability and inconsistencies. Implementations such as the Point Set Generation Network attempt to overcome these weaknesses by utilizing a point cloud representation; however, the absence of local connections between nodes in a point cloud result in an excessive degree of freedom such that the model loses important surface details. Additionally, as Wang. et. al. describes, it is "is non-trivial to reconstruct a surface model" from a point cloud representation.

Most recently, the waveform or *'mesh'* representation has emerged as a promising contender in the representation debate. In addition to overcoming the memory constraints and accuracy limitations of the aforementioned representations, mesh models are also more versatile for many real-world applications due to their ability to capture high-level surface details such as texture or color, their lightweight nature, and the ease with which they are deformed.[17] Currently, there are very few public implementations that attempt 3D reconstruction using mesh models; of these, the most prominent and best performing is Pixel2Mesh.

17 Wang, Nanyang, et al. "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images." Computer Vision – ECCV 2018 Lecture Notes in Computer Science, 2018, pp. 55–71., doi:10.1007/978-3-030-01252-6_4.

1.2.3 Pixel2Mesh

Pixel2Mesh is an end-to-end deep learning architecture proposed by Wang et. al. that represents a 3D mesh model in a graph-based convolutional neural network and "produces correct geometry by progressively deforming an ellipsoid, leveraging perceptual features extracted from the input image"[18] as illustrated in Figure 1 below.

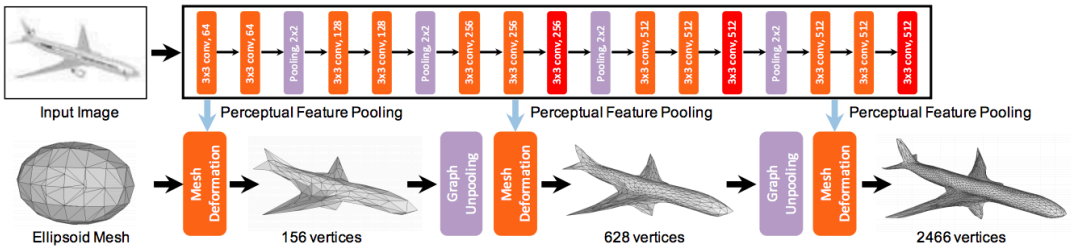

Figure 1: *Illustrating the mesh deformation and pooling layers of the architecture.* [19]

Four mesh-based loss functions are defined with equal weights: a Chamfer Loss to constrain the location of mesh vertices, a normal loss function to enforce the consistency of surface normal, a Laplacan regularization loss function to maintain relative location between neighboring vertices during mesh deformation, and an edge length regularization to prevent outliers.

Extensive experimentation provided by Wang et. al. reveals the Pixel2Mesh implementation to be both quantitatively superior (in terms of shape estimation) and qualitatively superior (in terms of surface details like smoothness and

18 Wang, Nanyang, et al. "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images." Computer Vision – ECCV 2018 Lecture Notes in Computer Science, 2018, pp. 55–71., doi:10.1007/978-3-030-01252-6_4.

19 Ibid.

continuity) when compared to other previously mentioned state of the art methods such as the 3D Recurrent Reconstruction Neural Network and the Point Set Generation Network. It also performs better than Neural 3D Mesh Renderer (N3MR)[20] which at the moment is the only other public, mesh-based 3D reconstruction model.

2 Approach

How can additional user input be utilized to improve the Pixel2Mesh system? We observe that despite the qualitative and quantitative superiority of Pixel2Mesh and the many limitations it overcomes by utilizing a mesh-based representation, the system still has several points of weakness. As previously described, the system works by progressively deforming a base ellipsoid. As a result, the generated models are bound by certain topological constraints and features of this ellipsoid. For instance, the system cannot generate complex objects (or objects of genus greater than zero, with one or more holes). Additionally, the use of a generalized ellipsoid shape for the base mesh wastes the potential for initial specification of form and topology and leaves no space to encode prior knowledge of the shape . Finally, experimentation and analysis reveal that Pixel2Mesh often suffers from a significant perspective bias, resulting in generated models that are proportionally skewed depending on the perspective angle, as shown in Figure 2 on the following page.

20 Kato, H., Ushiku, Y., Harada, T.: Neural 3d mesh renderer. In: CVPR (2018)

Figure 2: *Illustrating an example output model of a chair heavily skewed by perspective bias.* [21]

As is evident from the above example (Figure 2), this problem can result in models that are incorrectly asymmetric and lopsided. Furthermore, Pixel2Mesh suffers from the unavoidable weakness of any single-view 3D reconstruction attempt: back estimation.

Thus, we have identified three potential areas for improvement in the Pixel2Mesh system: a) perspective bias and back estimation difficulties, b) object complexity limitations, and c) lost potential for encoding prior knowledge of shape and form due to topological assumptions in the base ellipsoid.

This paper proposes a user input-centered approach to resolve these issues, in which a human user provides the system with additional information to help it generate an accurate model of a given object. Concretely, it explores how users can specify a) a plane of symmetry to help the system overcome its perspective bias and back estimation difficulties, b) a line of complexity to aid the system in modeling

21 Chang, et al. "ShapeNet: An Information-Rich 3D Model Repository." ArXiv.org, 9 Dec. 2015, arxiv.org/abs/1512.03012.

complex objects, and c) analysis of topology and shape to adjust the topological assumptions made by the system.

2.1 Leveraging the Plane of Symmetry

The intuition behind leveraging a plane of symmetry to help improve the system comes first from the observation that many of the models generated by Pixel2Mesh are significantly asymmetrical despite the obvious (to a human observer) symmetry of the object. While it is difficult for a machine to intuit three dimensional symmetry from a two dimensional image, this task is trivial for a human user with basic planar visualization software, as shown in Figure 3.[22] Thus, this project proposes an optional plane of symmetry argument that will a) inform the system that symmetry exists in the object and b) aid the system in realizing this symmetry during generation.

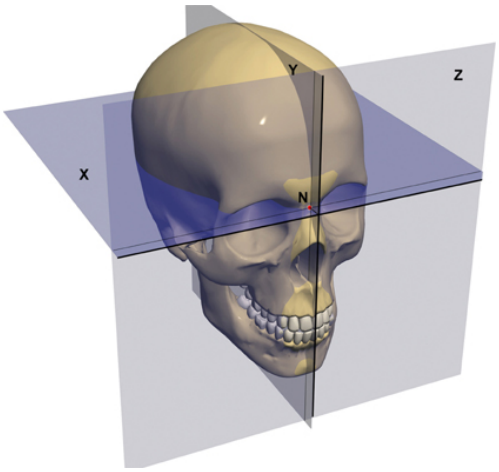

Figure 3: *Illustrating the y-plane of symmetry (x =0) in the skull image above.*

The input of this planar argument will be of the form Ax + By + Cz + D = 0, where the user inputs the numeric parameters A, B, C, and D to define the plane of symmetry. This paper will explore two methodologies for leveraging this additional information: one that is applied post-generation and one that informs the system during generation.

22 Image obtained from https://e-kjo.org/ArticleImage/1123KJOD/kjod-43-62-g001-l.jpg

The first approach, automated geometric post-processing, will seek to deform an output mesh such that it becomes more symmetrical about its plane of symmetry. The second approach seeks to design a deep learning architecture that is symmetry-aware. To accomplish this, we will integrate the plane of symmetry as an additional input into the system and retrain the model with a ground truth plane of symmetry for each instance in the training dataset. We will add an additional loss function that punishes deformations during training that are less symmetric about the specified plane of symmetry. The goal of both these approaches is to alleviate the error caused by perspective bias and back estimation difficulties to produce a mesh model that is both more visually appealing and more accurate according to the ground truth mesh model.

2.2 Leveraging Complexity Analysis

The intuition behind leveraging complexity analysis stems from the system's current inability to model complex objects, or any objects with one or more holes. If a user provides insight upfront relating to the complexity of the object, could this be used to aid the process of modeling complex objects?

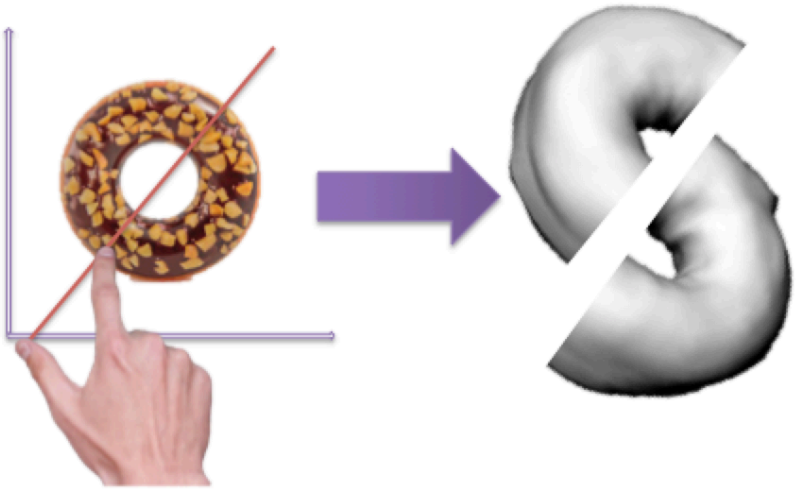

Figure 4: *Illustrating how user might input the line of complexity.*

The approach proposed by this paper will allow the user to input an optional complexity demarcation argument in the form of a line of complexity. This line

segments a complex object of genus greater than 0 into two simple objects of genus equal to zero. This line can be input either via a graphical user interface (as shown in Figure 4[23] to the right) or in the parameterized form y = Mx + B, where the user inputs the numeric parameters M and B to define the line of complexity. With the line of complexity, the system could then feed each of the simple object segments into the system to be modeled individually, as illustrated in Figure 4 above. These pieces could then be rejoined to form a complete model of a complex object.

2.3 Leveraging Topological Analysis

The intuition behind leveraging topological analysis of form and shape comes from the observation that the current system leaves no room for encoding of prior knowledge into the base mesh and instead uses a general base ellipsoid shape for modeling all objects. If the user were to specify information beforehand, the base mesh could be adjusted accordingly and the difficult and costly initial mesh deformations could be avoided.

The approach we will investigate involves the user providing a topological analysis in one of two ways. Firstly, they could specify the object domain or class (such as hand, body, plane, or dog, for instance) so that known key universal features of this domain and generic form can be encoded in the base ellipsoid before modeling. Secondly, the user could input a mesh model that represents form or shape

23 Image obtained from https://pngtree.com/freepng/donut-png-image-collection_510856.html

information about the target object, and the base ellipsoid could be encoded with this information.

3 Implementation

The first implementation task was to design a simple application for easier testing, usage, and demonstration of the new system. I chose to deploy a Djangio-based[24] web application to Heroku[25] that uses Princeton University's computing clusters[26] and graphical processing units as a backend to perform the mesh modeling. The generated model can then be downloaded from the browser. This application can be found at online[27] and currently supports the following inputs: an input image, a parameterized plane of symmetry, a parameterized line of complexity, and a single example domain specification ('hand').

3.1 Leveraging the Plane of Symmetry

With the input infrastructure built, I next moved to incorporate knowledge of symmetry into the system, beginning with automated post-processing.

3.1.1 Automated Symmetric Post-Processing

My first task was to design an algorithm to implement geometric post-processing. Despite the versatility and popularity of the mesh or waveform model format in

24 Django (Version 1.5) [Computer Software]. (2013). Retrieved from https://djangoproject.com.

25 "Heroku." Heroku Architecture | Heroku Dev Center, devcenter.heroku.com/categories/heroku-architecture.

26 "Princeton Computing Clusters." Princeton University, The Trustees of Princeton University, csguide.cs.princeton.edu/resources/clusters.

27 Web application can be accessed at https://deepmesh.herokuapp.com

many fields such as graphic design or the gaming industry, the vast majority of mesh editing software is built for manual editing rather than programmatic manipulation. Due to the limited public software available for complex and programmatic waveform manipulation, my first task was to write a program that takes as input a plane of symmetry and a mesh model and programmatically deforms the model to make it more symmetric about the input plane of symmetry, as can be seen in Figure 5[28] below. To do this, I built software (building upon a simple open source mesh manipulation software, OBJ-Magic[29]) that implements the following algorithm:

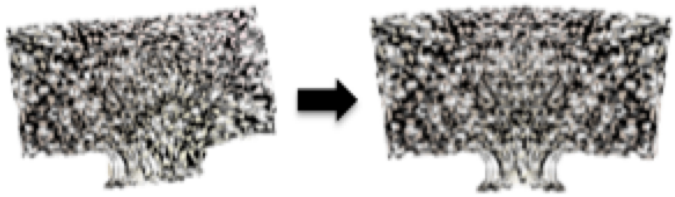

Figure 5: *Illustrating the goal of symmetric post-processing.*

The program must first understand how the plane of symmetry bisects the model into two halves and then designate which side will *dominated*, or overridden to become symmetrical with the other *dominant* side. To do this, it first determines three non-collinear points on the input plane of symmetry (points a, b, and c). Then, for each point x in the mesh, it constructs the following 3x3 matrix M_x = $[a-c \quad b-c \quad x-c\,]$, where the first, second, and third columns are the 3x1 vector representing the difference between the coordinates of point a and c, b and c,

28 Chang, et al. "ShapeNet: An Information-Rich 3D Model Repository." ArXiv.org, 9 Dec. 2015, arxiv.org/abs/1512.03012.

29 Tapio. "OBJ-Magic." GitHub, 16 Feb. 2016, github.com/tapio/obj-magic.

and x and c respectively. This matrix is useful because the determinant of matrix M_x informs us of which side of the input plane point x is on: the matrices of all points on one side of the plane will have a positive determinant and the matrices of all points on the other side will have a negative determinant.[30] An arbitrary side of the plane is then selected and designated as the dominant side.

Our next task is to override all dominated points (on the non-dominant side of the object) with the reflection of all points on the dominant side over the plane of symmetry. This task is more difficult than it initially seems due to the encoding format of waveform objects. Each vertex of the mesh is encoded by a line in the format 'v x y z', where x, y, and z represent the coordinate of the vertex. Each triangular face is then represented in the format 'f a b c', where a, b, and c each refer to a vertex by the line number on which it appears in the file. Thus, one cannot simply begin adding or removing vertices from the mesh without inadvertently changing the ordered relationship between the vertices and faces.

The approach we take is as follows: we iterate again through all points X in the original mesh. For each point, if the determinant of its corresponding matrix M_x is positive, it is on the dominant side of the object. The vertex is printed to the post-processed mesh file. Additionally, the reflection of the point over the plane of symmetry is inserted into a queue data structure. If the point is on the dominated side, the last dominant vertex is reprinted to preserve ordering. Next, we begin popping reflected dominant vertices from the queue to reconstruct the reflected

30 Smith, John. "Point on the Left or Right Side of a Plane in 3D Space." Mathematics Stack Exchange, math.stackexchange.com/questions/214187/point-on-the-left-or-right-side-of-a-plane-in-3d-space.

dominant side, repeating when necessary to preserve order. For each face, we print the line first unaltered and again with each of its values incremented by the number of dominant vertices, to add the faces to the reflected side. We then run this generated mesh through an algorithm (provided by the PyMesh library) that removes duplicate faces and vertices from a waveform object to output our final post-processed mesh.

3.1.1 Symmetry-Aware Architecture

I next moved to design and construct a symmetry-aware deep learning architecture that considers the input plane of symmetry during generation of the mesh model. To do this, I adapted the original Pixel2Mesh architecture to accept and understand input planes of symmetry.

As described earlier, the system already has several mesh-based loss functions defined that seek to encourage things like smoothness, continuity and uniform distribution of vertices during the layers of mesh deformation. I added an additional symmetry loss function that punishes deformations that are less symmetrical about the input plane of symmetry. This loss function adds a loss proportional to the number of points of all points whose reflection over the specified plane is not present in the mesh.

My next task was to train the model with symmetry. I chose to use the same dataset that the original Pixel2Mesh model was trained with: ShapeNet, one of the largest 3D and most frequently used training datasets when it comes to deep learning and computer vision. However, to train the model with symmetry I would need to input

accurate plane of symmetry parameters for each model in the training dataset. With no publically available software for determining the equation of the plane of symmetry given a waveform model, my next step was to create a program that took as input a 3D mesh model and output the parameters of a plane of symmetry.

The algorithm I implemented receives as input a mesh model and determines the parameters of the plane of symmetry as follows. First, a pivot point is set at the center of the oriented bounding box (oriented around the height, width, and length of the object as opposed to the x, y, and z axes) surrounding the model. The algorithm then begins iteratively testing planes of symmetry by rotating the trial planes 360 degrees around the height, width, and length axes. Checking for symmetry over 1080 planes can potentially be time consuming work, and there were many models in the ShapeNet dataset that had to be analyzed. Thus, I needed to design an algorithm that discarded poor planes of symmetry quickly.

To accomplish this, I checked the symmetry of each plane in the following way. I first reflected the model over the plane in question, which is a relatively quick operation. I next checked the new coordinates of the bounding box corners and the new mesh center of mass. If either of these metrics were different past a certain threshold, the plane is discarded and the next one is examined. Those planes which pass this initial filter are then inspected more rigorously (by iterating over all points and checking if their reflection is close to the surface of the mesh) to verify if they are acceptable planes of symmetry. The program returns the first acceptable plane of symmetry (within a certain threshold of similarity) that it finds.

With the loss function added, and acceptable planes of symmetry found for each ground truth mesh, the system was ready to be trained with symmetry. Furthermore, repeated patterns in the ShapeNet data allowed planes of symmetry to be determined more quickly, as they repeated in a predictable order, which significantly sped up the pre-processing of the data and training of the model.

3.2 Leveraging Complexity Analysis

My goal in this area was to open up the space for generation of more complex objects. To do this, I wrote a program that provides a simple graphical user interface that allows the user to demarcate the line of complexity (as described above) to segment the complex object into simple components. This line of complexity can also be input as a parameterized plane if we let the left and bottom borders of the image be the y and x axes respectively. This program then automatically models each simple component separately (by feeding them as input into system) and returns the amalgamation of the two as a single mesh model that the user must then refine and edit to produce the final mesh. The process of remerging the two halves into a whole complex object is a complicated task that I chose to reserve for potential future work. As it stands, the system requires the user to manually align the generated mesh halves to create the final complex model.

3.3 Leveraging Topological Analysis

As described, the architecture generates the final mesh model by a series of progressive deformations of a base ellipsoid. As Dr. Yinda Zhang describes, the

earliest deformations are the most costly and difficult for the network to process.[31] However, if the user provided insight about the shape and topology of the target object beforehand, this knowledge could be leveraged and encoded into the base structure to improve the accuracy and details of the generated model. Thus, my next objective was to design a way to encode this information into the base mesh model. This tasks turns out to be more difficult then simply substituting a different base mesh into the structure. The system represents the base ellipsoid and progressive deformations in both mesh and graph representations, and the relationship between each of the vertices and faces is critical to the functionality of the architecture.

To overcome this obstacle, I wrote a program that takes as input the original base ellipsoid and a mesh model (as graphically visualized in Figure 6 below) with the target topology, converts them to a graph representation, and then performs the non-rigid iterative closest point algorithm[32] to progressively deform the initial base ellipsoid to conform to the desired target topology.

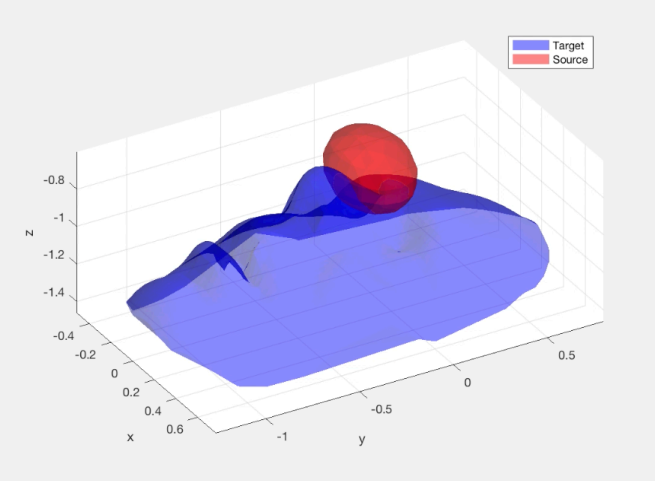

Figure 6: *Illustrating the original base ellipsoid (in red) and the mesh model with the target topology (in blue).*

31 Personal communication with Dr. Yinda Zhang, one of the developers of Pixel2Mesh.

32 Nash, Charlie. "NRICP." GitHub, 3 Apr. 2018, github.com/charlienash/nricp.

This process allows for the additional topological information to be encoded without altering the relationship between or ordering of the vertices and faces of the base model. The new restructured base mesh model is then integrated into the system. As an alternative to inputting a mesh object with the desired target topology, the user can also specify a domain area (such as hand, face, body, etc.) and the appropriate generic base mesh of this object type will be integrated into the system.

4 Evaluation

The quantitative evaluation framework I chose to employ is an adapted version of the standard 3D reconstruction metric as defined by Wang et. al. which centers on the uniform sampling and comparison of points in the result and ground truth and the subsequent calculation of precision and recall by analyzing the distance of the points from a nearest neighbor.

Concretely, my quantitative mesh comparison algorithm works as follows. First, the result mesh and ground truth mesh are normalized through appropriate centering and scaling. Next, for each point s in the set of all ground truth points (S) I check whether that point is contained in the generated mesh object. If it is not, I determine the distance of the point

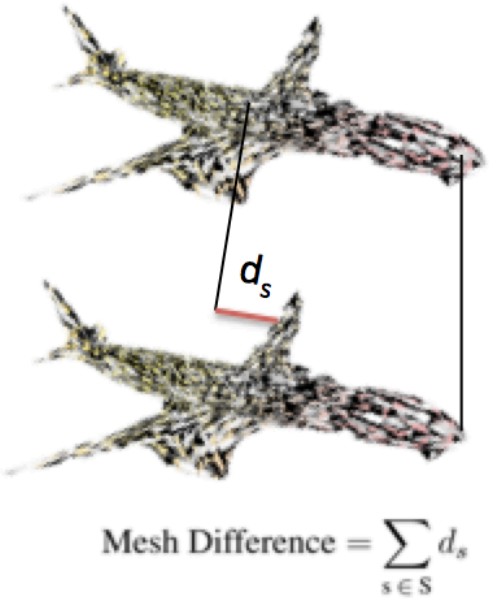

Figure 7: *Illustrating the quantitative mesh evaluation algorithm.*

to the generated mesh surface (d_s) and add this to the running total to determine how far off the generated mesh, as illustrated in Figure 7 on the previous page.

I then perform the reverse process and sum the distances d_g between each point g in the set of all points in the generated mesh (G) from the ground truth model. The total combined value of these summed mesh differences represents how far off the generated mesh is from the ground truth, with a lower value indicating a more quantitatively accurate generated model. Henceforth, we will refer to this total mesh difference as the τ value, where:

$$\tau = \sum_{s \in S} d_s + \sum_{g \in G} d_g$$

However, it becomes clear that additional metrics are needed to fully understand and evaluate the results of this work. We observe that when it comes to evaluating generated models, geometric point-wise accuracy alone is not a sufficient metric. As Wang. et. al. describes, "the commonly used evaluation metrics for shape generation may not thoroughly reflect shape quality. They easily capture occupancy or point-wise distance but neglect other important surface properties such as continuity, smoothness, or high-order details for which a standard evaluation metric is absent in literature."[33] To understand this, we observe that certain loss functions improve the quality of generated meshes but actually decrease the quantitative point-wise accuracy of these meshes. Take, for instance, the edge length regularization loss function which works to "maintain relative location between neighboring vertices

33 Wang, Nanyang, et al. "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images." Computer Vision – ECCV 2018 Lecture Notes in Computer Science, 2018, pp. 55–71., doi:10.1007/978-3-030-01252-6_4.

during deformation."[34] When the system is trained without this loss function, the produced models are quantitatively more accurate, but qualitatively they are far poorer, as can be seen in Figure 8[35] below.

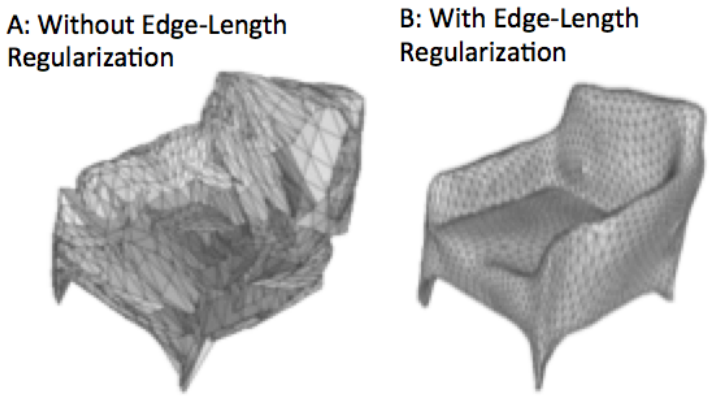

Figure 8: *Illustrating two meshes, one with edge-length regularization (B) and one without (A) generated from the same image. Mesh A performs quantitatively better than mesh B but is qualitatively worse.*

As can be seen, mesh B above is qualitatively much better in terms of symmetry, continuity, and smoothness. Despite this, it is actually quantitatively further from the ground truth model than mesh A when evaluated using the standard mesh evaluation framework. Thus, in addition to the quantitative point-wise metric, this paper proposes a set of qualitative evaluation criteria: namely, we will qualitatively evaluate the generated models based on symmetry, complexity, and the presence of key high-level details.

We begin with the evaluation of each component separately, first turning our attention to the effectiveness of leveraging the plane of symmetry through automated geometric post-processing. For the quantitative evaluation, I selected a

34 Wang, Nanyang, et al. "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images." Computer Vision – ECCV 2018 Lecture Notes in Computer Science, 2018, pp. 55–71., doi:10.1007/978-3-030-01252-6_4.

35 Ibid.

small subset of symmetrical image-model pairs from the ShapeNet database to be used as my testing dataset. As a baseline, I used the models generated by the original Pixel2Mesh system. The average τ value that the original system achieves for this test dataset is 2.65. After post-processing using the newly input knowledge of symmetry, the system achieves a τ value of 2.41. Thus, we observe that on average the system performs quantitatively better after symmetric post-processing. However, despite this we also observe significant variance in the τ value across the models in the testing dataset. While in some cases the post-processing improves accuracy and helps the model alleviate the mistakes made due to perspective bias or back estimation difficulties, in others it can accentuate these errors and result in quantitatively worse mesh models. On the qualitative side of the evaluation spectrum, the produced models are always symmetric about the given plane of symmetry, which improves the appearance of the generated mesh models.

We next examine the effectiveness of leveraging the plane of symmetry by developing and training a symmetry-aware model. For the quantitative evaluation, the testing dataset was comprised of symmetrical image-model pairs selected from the Free3D and ShapeNet databases. As a baseline, I trained the original Pixel2Mesh architecture with a subset of the ShapeNet dataset. I then trained the symmetry-aware architecture with the same dataset and for the same amount of time, and compared the 3D models reconstructed by each system. The average τ value that the original system achieves for this test dataset is 4.22. When the architecture understands and prioritizes symmetry, the system achieves a τ value of 4.76. Thus,

we observe that on average the system performs quantitatively worse when symmetry is introduced into the architecture. Again, however, there is considerable variance across different examples; on some examples, for instance, the symmetry-aware architecture performs better than the original. Qualitatively, the symmetry-aware system produces models that tend more toward symmetry than the original system, which is a positive result.

We next examine the effectiveness of leveraging complexity analysis to generate models of genus greater than zero. For the quantitative evaluation, the testing dataset was comprised of complex image-model pairs selected from the Free3D and ShapeNet databases. As a baseline, I used the models generated by the original Pixel2Mesh system. The average τ value that the original system achieves for this test dataset is 3.51. After the simple components of the object are modeled and manually merged, the system achieves a τ value of 3.36. Thus, we observe that on average the system performs quantitatively better when it comes to generating these complex objects. Furthermore, from a qualitative perspective, the improved system generates complex models that are of the correct genus and topology. However, one significant weakness to this addition is its dependence on manual remerging and refinement of the simple mesh components into a final complex object.

Finally, we examine the effectiveness of encoding prior topological knowledge into the base ellipsoid. For the quantitative evaluation, the testing dataset was comprised of human face models selected from the Turbosquid and Free3D

databases. As a baseline, I used the face models generated by the original Pixel2Mesh system. The average τ value that the original system achieves for this test dataset is 2.40. When the additional knowledge of topology is encoded into the base ellipsoid, the system achieves a τ value of 2.02. Thus, we observe that on average the system performs quantitatively better when prior knowledge is considered and encoded into the mesh before generation. The new system was also able to generate models with slightly more defined facial features (like nose and jawline definition).

As stated in the motivation section of this paper, the ultimate goal of aiding 3D reconstruction with additional user inputs is to increase the effectiveness and efficiency of the process. The quantitative results of this work are mixed. I was excited by the improved performance of the system after incorporation of topological analysis, complexity analyses, and symmetric post-processing. However, while the system on average performs better when these new inputs are leveraged, there is significant variability in the success of these approaches across different models. Furthermore, the decreased quantitative effectiveness of the symmetry-aware deep learning architecture was disappointing and surprising despite its success on certain individual examples. Nevertheless, the potential for additional user input to be leveraged in order to improve the accuracy of reconstruction systems has been effectively demonstrated. These results, though limited, speak to the very real potential of significantly improving the process of 3D reconstruction with human input.

5 Conclusions

The are many routes for further development and exploration of the approaches presented in this paper. One future front-end development I would like to make is incorporating a plane visualizer into the web application to aid the users in inputting the correct plane of symmetry. I would also like to explore the different ways in which the separate components generated after complexity analysis can be programmatically and accurately remerged without so much manual labor from the user. I would also love to further train and develop the symmetry-aware architecture. I believe this approach in particular has much more potential than is evident from the results I achieved, and I would like to demonstrate that.

There are several conclusions to take away from the results of this paper. Concretely, the applicable contributions this project has made to the area of mesh-based 3D reconstruction include waveform manipulation and analysis software that previously did not exist and a public and easily accessible web application that allows users to generated mesh models from 2D images and refine this generation with additional inputs. There are also conclusions we draw that can inform how the task of 3D reconstruction with deep learning is approached in the future.

Firstly, we observe that our results suggest that the task of 3D reconstruction, like many other deep learning and computer vision tasks, may best be accomplished by assisting neural networks with additional human analysis. Secondly, we derive from the results of this paper a conclusion that has implications in the debate over model representation. As has been established, mesh-based 3D reconstruction is

one of the most promising new contenders in the field of 3D modeling. The lightweight nature of mesh objects, along with their easy deformability and their capacity to be encoded with higher-level information, provide many opportunities that other 3D model representations do not. The approaches explored by this paper take advantage of these unique properties to leverage user input and improve the accuracy of generated models in certain cases. In fact, these techniques are dependent on the positive properties of the mesh representation. Thus, the results of this paper serve not only as a defense of user input-centered 3D reconstruction, but also as a defense of the mesh-based approach to 3D reconstruction.

6 Acknowledgements

I would first like to thank my independent work advisor Dr. Jia Deng for the guidance he has provided throughout this process. Both computer vision and deep learning were very new to me at the start of this semester, and he provided invaluable guidance on how to navigate these fields and design my project. I would also like to give a big thank you to Hei Law, the teaching assistant for my independent work seminar, for sharing his time and energy to help me understand how to work with new systems and technologies. Finally, I would like to thank Dr. Yinda Zhang and Nanyang Wang, two of the contributors to the Pixel2Mesh system, who shared their insight over email and video conference and whose guidance helped me to both envision my project and bring it to fruition. Without the people above, I would not have been able to accomplish the work I did, and I am very grateful!

7 Bibliography

Sources References in the Paper

1) “Computing Matched-Epipolar Projections.” Computing Matched-Epipolar Projections - IEEE Conference Publication, ieeexplore.ieee.org/abstract/document/341076.

2) “Heroku.” Heroku Architecture | Heroku Dev Center, devcenter.heroku.com/categories/heroku-architecture.

3) “Instant Architecture.” ACM Transactions on Graphics (TOG), ACM, dl.acm.org/citation.cfm?id=882324.

4) “Multi-Image 3D Reconstruction Data Evaluation.” Journal of Cultural Heritage, Elsevier Masson, 17 Jan. 2013, www.sciencedirect.com/science/article/abs/pii/S1296207412001926.

5) “Princeton Computing Clusters.” Princeton University, The Trustees of Princeton University, csguide.cs.princeton.edu/resources/clusters.

6) “Remove the Background of a Picture.” Office Support, support.office.com/en-us/article/remove-the-background-of-a-picture-c0819a62-6844-4190-8d67-6fb1713a12bf.

7) “Three-Dimensional Reconstruction for Medical-CAD Modeling.” Taylor & Francis, www.tandfonline.com/doi/abs/10.1080/16864360.2005.10738392.

8) “Automatic Scene Modeling for the 3D Camera and 3D Video.” Google Patents, Google, patents.google.com/patent/US20080246759A1/en.

9) “Visualizing Space Science Data in 3D.” Visualizing Space Science Data in 3D - IEEE Journals & Magazine, ieeexplore.ieee.org/abstract/document/544066.

10) Chang, et al. “ShapeNet: An Information-Rich 3D Model Repository.” ArXiv.org, 9 Dec. 2015, arxiv.org/abs/1512.03012.

11) Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In: ECCV (2016)

12) Django (Version 1.5) [Computer Software]. (2013). Retrieved from https://djangoproject.com.

13) Donut image obtained from https://pngtree.com/freepng/donut-png-image-collection_510856.html

14) Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3d object reconstruction from a single image. In: CVPR (2017)

15) Girdhar, Rohit. Learning a Predictable and Generative Vector Representation for Objects. Robotics Institute, Carnegie Mellon University, arxiv.org/pdf/1603.08637.pdf.

16) Kato, H., Ushiku, Y., Harada, T.: Neural 3d mesh renderer. In: CVPR (2018)

17) Nash, Charlie. "NRICP." GitHub, 3 Apr. 2018, github.com/charlienash/nricp.

18) Skull mage obtained from https://e-kjo.org/ArticleImage/1123KJOD/kjod-43-62-g001-l.jpg

19) Smith, John. "Point on the Left or Right Side of a Plane in 3D Space." Mathematics Stack Exchange, math.stackexchange.com/questions/214187/point-on-the-left-or-right-side-of-a-plane-in-3d-space.

20) Sturm, Peter. "On the History of Image-Based 3D Modeling." INRIA. INRIA. http://gurdjos.perso.enseeiht.fr/vision24janvier2014/Sturm-24jan-TLSE.pdf

21) Tapio. "OBJ-Magic." GitHub, 16 Feb. 2016, github.com/tapio/obj-magic.

22) Wang, Nanyang, et al. "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images." Computer Vision – ECCV 2018 Lecture Notes in Computer Science, 2018, pp. 55–71., doi:10.1007/978-3-030-01252-6_4.

Additional sources utilized in web application (unreferenced in paper):

1) Code for Countdown Timer in Web Application Accessed from: https://codepen.io/anon/pen/oOvEVB

2) Design and Layout Template for Front End Application Accessed from: https://onepagelove.com/holly