Fakultät für Informatik
Lehrstuhl für Robotik, Künstliche Intelligenz und Echtzeitsysteme und Robotik

Mesh Generation from RGB Images:

Improvement and Conversion of Pixel2Mesh in Pytorch, with ResNet and Stereo Input

Alexandre Cafaro

Liam Hulsman-Benson

Sebastian Lohr

Praktikum Vision-Based Navigation and Learning for Autonomous Vehicles

Advisor:	Xiang Gao
Supervisor:	Prof. Alois Knoll
Submission:	10. April 2020

I. Introduction

3D Mesh generation from single RGB images has been a great challenge in computer vision. We propose an improvement of the model presented in [7] by using the more efficient ResNet instead of VGG for convolutional architecture, computing by batch, and extending the generation with stereo images as input, leading to more precise and accurate mesh models. We converted the Tensorflow version to PyTorch and trained the new models after transfer of the previous model.

II. Original Pixel2Mesh paper

[7] introduced a novel approach on how to infer a 3D shape from a single perspective. It generates a 3D triangular mesh for the entire object and achieved good results compared to volume [3] or point cloud [4] approaches.
The original paper uses a VGG neural network to extract features of the object from the input image. An ellipsoid with a fixed sized is deformed according to the extracted features and even reconstructs unseen parts of the object. The mesh of the deformed body will be refined to reconstruct more detailed parts of the object.

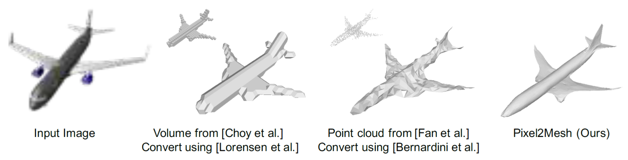

Input Image
Volume from [Choy et al.] Convert using [Lorensen et al.]
Point cloud from [Fan et al.] Convert using [Bernardini et al.]
Pixel2Mesh (Ours)

Fig. 1: Comparison of mesh output between the original paper, point cloud and volume approaches

A. Model and Architecture

1) Initial Ellipsoid: An initial ellipsoid is created containing 156 vertices. The network will aim at modifying the ellipsoid in a way to mould the object. We use this ellipsoid to initialize our network input graph, where the initial feature contains only the 3D coordinate of each vertex. Throughout the deformation process the number of nodes is increased to up to 2466.i

2) Mesh deformation block: In order to generate a 3D mesh model that is consistent with the object shown in the input image, the deformation block needs to pool features from the input image. After extraction by a perceptual pooling layer VGG-16, the feature is concatenated with the 3D shape feature attached on the vertex. The entire model is fed into a graph based convolutional neural network G-ResNet, to predict the new location and 3D shape features for each vertex.

3) Graph unpooling layer: After an unpooling layer is applied, the number of vertices in the G-ResNet is increased. This allows for a more precise and complex mesh with more vertices. More vertices are added in specific areas when required through triangularisation to produce more accurate results.

4) Losses: Four types of losses are applied to constrain the property of the output shape and the deformation procedure to guarantee optimal results. Chamfer loss [1] is used to constrain the location of mesh vertices, a normal loss to enforce the consistency of surface normal, laplacian regularization to maintain relative location between neighboring vertices during deformation, and an edge length regularization to prevent outliers.

B. Data

The dataset used is obtained from [3]. The dataset contains 2D renders of 50,000 models belonging to 13 object categories from ShapeNet [1], a collection of 3D CAD models that are organized according to the WordNet hierarchy. A model is rendered from various camera viewpoints, and camera intrinsic and extrinsic matrices are recorded.

III. Our approach

A. Improvements

- Conversion of the Pixel2Mesh model from Tensorflow to PyTorch
- Replacement of VGG with ResNet : compared to VGG-16, the training progress of ResNet16 is more stable and the final performance is higher.
- Use of Stereo input instead of Mono input : Two images from different camera angles are used as input to a newly designed network to improve the output.
- Increase of Batch size : Forward pass and back-propagation in batches, loss is computed batch by batch.
- Transfer and conversion of checkpoint of VGG of Tensorflow to PyTorch to initialize training for ResNet.
- Transfer of checkpoint of ResNet to initialize training for Stereo-ResNet.
- Computation of F-scores with PyTorch models : the evaluation of the output is done similar to the original paper. Points are sampled uniformly on the surface of the model of the ground truth and the output of the network. The distance for each point to the nearest neighbour in the other model is calculated. The F-score is obtained by calculating a percentile of points that lie within a certain threshold.

B. PyTorch

The original paper was implemented using Google's Tensorflow framework. Since its release, PyTorch has gained a large popularity in the scientific community. It is designed

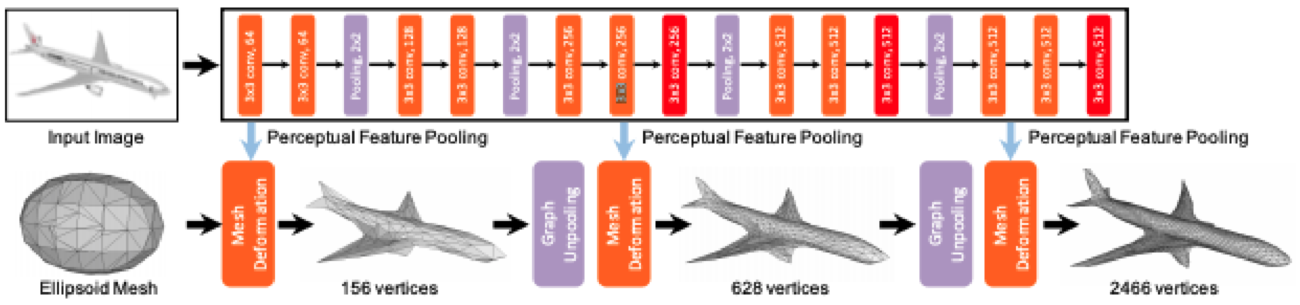


Fig. 2: Architecture of the Pixel2Mesh

for python, allowing for ease of use with a range of other packages. It offers dynamic computation graphs and generally performs better than Tensorflow with low computational power.
For this project, the functions of the Pixel2Mesh network have been rewritten using equivalent functions in PyTorch.
For the chamfer distance, the external library Pytorch Chamfer Distance [5] has been used.

C. ResNet

For the perceptual pooling layer, we implemented a ResNet following the structure and complexity of the VGG-16 from the previous paper. As an improvement, skip connections have been introduced to the network. This allows more transfer of information across the network and provides more robust training. Figure 3 shows the transformation from the VGG model to our ResNet model. The Pooling layers in Fig. 2 have been replaced by convolutional layers with stride 2 as it is done equivalently in Pixel2Mesh.
Similar types of features are also pooled for deformation of the meshes.

D. Stereo-ResNet

The introduction of new information from diverse angles leads to generation of a more accurate and precise meshes. This has been shown with the improved results of Pixel2Mesh++ [8].
After completing the ResNet architecture, the same network is used in parallel to process two images, each producing a feature vector for mesh deformation.
Each of the features are concatenated with the corresponding one from the other image and then processed by additional Conv2D layers, show in Fig. 7c These layers additionally reduce the size of the concatenated feature vector back to the original size of the output of the single ResNet architecture. Further processing can be then done the same way as the ResNet architecture.
The weights are shared and trained together as both networks should be able to extract information from any angle and be invariant, as the angle may vary and has not been assumed to be fixed.
By choosing a high similarity to the single image ResNet architecture, it was expected to require less training time for the Stereo Network as parameters can directly be transferred and used from the previous training of the single image ResNet. This design choice was made due to the limited time with server access as only the additional Conv2D layers need to be trained from scratch.

E. Challenges and Solutions

1) Transfer the VGG checkpoint: In the beginning of the training, the weights from the VGG network in Tensorflow have been transferred to PyTorch. With that far better results have been achieved for the same duration than if initialised by random. The weights have been mapped between the VGG network in Tensorflow to the according ones in PyTorch (check archives in the git repository for more details).

2) Make it run by batches: To achieve faster training results, the forward and backward of the model has been made possible in batches compared to previous paper where it was made one by one. The loss had to stay computed one-by-one because of non consistent sizes between meshes from different objects. The size of the batch was limited by the memory of the GPU, but can be changed to any value and adapted to the hardware. Due to time limitations the influence of this change on the training could not be analysed specifically.

3) Convert losses: The paper used four different losses to constrain the output of the model: Chamfer loss, normal loss, Laplacian regularization and edge length regularization.
For a good performance of the model in terms of training time, those loss calculations have to be performed efficiently. Equivalent functions have been used of PyTorch that parallelize the calculations for the points of the mesh. It pushed us to go deeper in the theory of these losses.
The distance calculation is performed using Chamfer loss as shown in Fig. 5 provides a ready to use library implemented with PyTorch that computes the Chamfer loss that has been used for calculation of the loss during training and evaluation of the final result with F-scores. As mentioned in the original paper, the Chamfer loss doesn't measure surface quality and high order details and results need further visual analysis to evaluate the results.

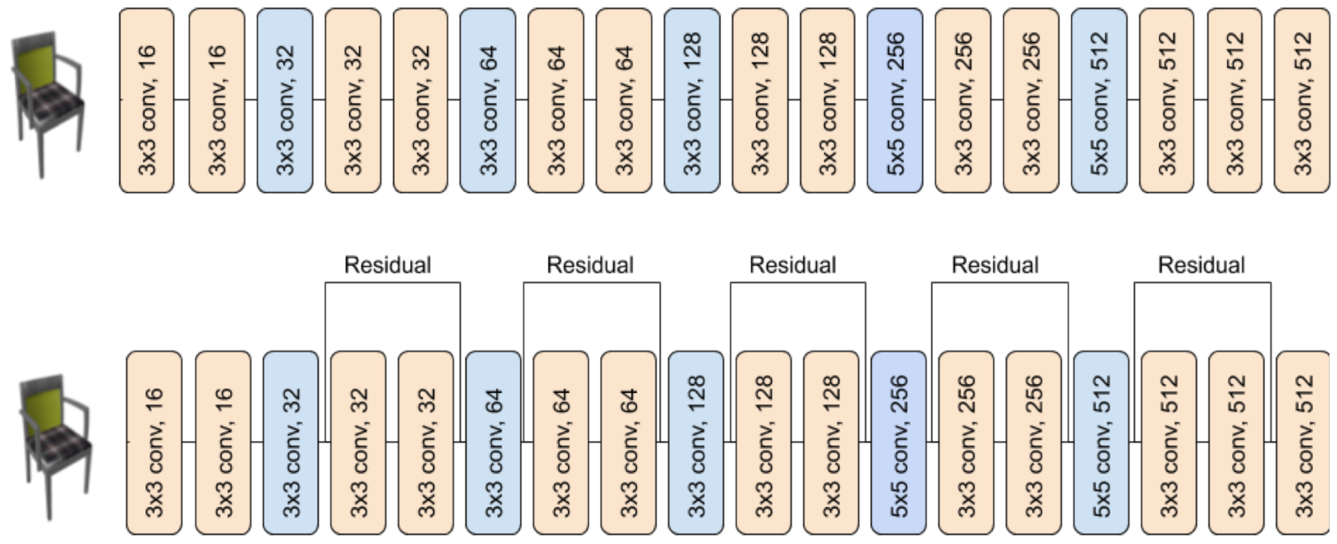


Fig. 3: Comparison of VGG and ResNet

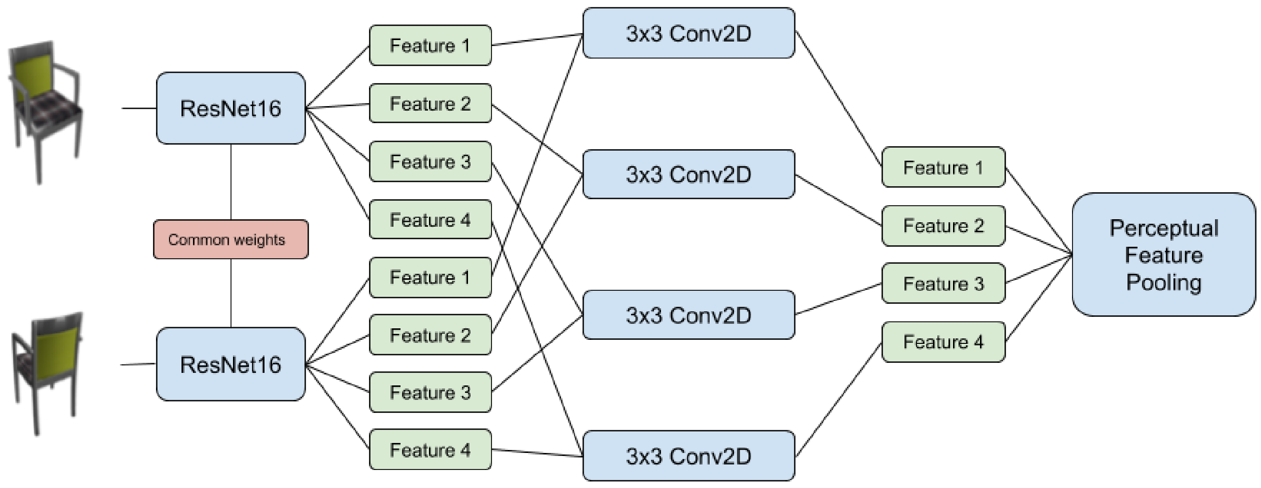


Fig. 4: Stereo-Perceptual Feature Pooling

4) Download the data: The downloading process has also been automated. The file structure of the training set has been adapted and stored on our Google Drive. As we transferred the weights from the Pixel2Mesh network, the size of the dataset was also reduced to achieve faster training iterations and good results.

F. GitHub Repository

Our GitHub Repository can be found at https://github.com/Wapity/Pixel2Mesh-PyTorch-TUM

IV. Experimental Results

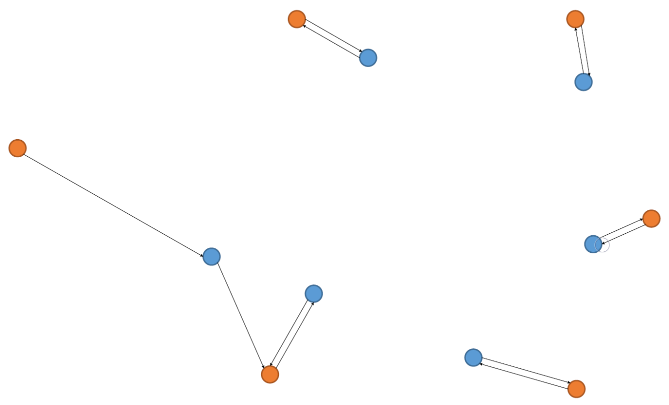
A. Training and Runtime

1) ResNet: The ResNet has been initialised with the training checkpoint from the VGG network.

Hyperparameters:

- batch size: 16/100 due to the limit of the CUDA memory
- learning rate: 5e-5
- learning rate decay: 0.98 every two epochs
- weight decay: 1e-5

.



Chamfer distance (CD)

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

Fig. 5: Chamfer distance between two data sets, [6]

2) Stereo-ResNet: The ResNet used for both images have been initialised with the training checkpoint from ResNet training. The weights are not frozen and the whole network is retrained with the added convolutional layers.

Hyperparameters:

- batch size: 8/50 due to the limit of the CUDA memory, reduced as there are now more parameters
- learning rate: 5e-5
- learning rate decay: 0.98 every two epochs
- weight decay: 1e-5

3) Training: The results were obtained by training with GPU Tesla P4 and P100 on Google Cloud Platform on around 2/3 of ShapeNet dataset [1] (~35k objects - ~100k images for ResNet - ~80k images for Stereo-ResNet) for 20 epochs for ResNet (130h) and for 10 new epochs for Stereo-ResNet (50h).

B. Numerical Comparison

Tab. 1 shows the F-score of different methods with a threshold of 1e-4 and 2e-4. The testing set chosen is based on a sample of the dataset but is not the same as for VGG so comparison with it is not representative. From [7] the F-score are : for 1e-4 of 59.72%, for 2e-4 of 74.19% taken the mean all categories combined. The performace could be slightly improved using our stereo approach.

TABLE I: Comparison of F-Scores of the models

	ResNet	**Stereo-ResNet**
Threshold 1e-4	51.11	53.26
Threshold 2e-4	69.02	70.21

C. Visual Comparison

Figure 7 shows the output of our ResNet and Stereo model compared to the original VGG output for the input shown in figure 6. It can be sef that both our networks produce a chair similar to the input model while not achieving the same details around the legs and smoothness on even surfaces. Also the output of the Stereo and ResNet models are very similar which is probably caused by the weight sharing and weight transfer from the ResNet. But the result should be more accurate as more information is provided to the network, with information taken from another angle of view covering partly hidden spaces with one input. Many views from all possible angles should lead to the best results like in 3D scanning of an object.

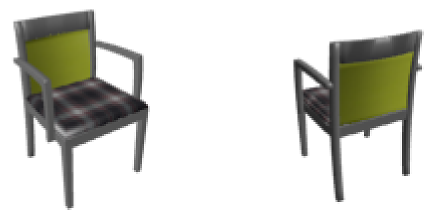


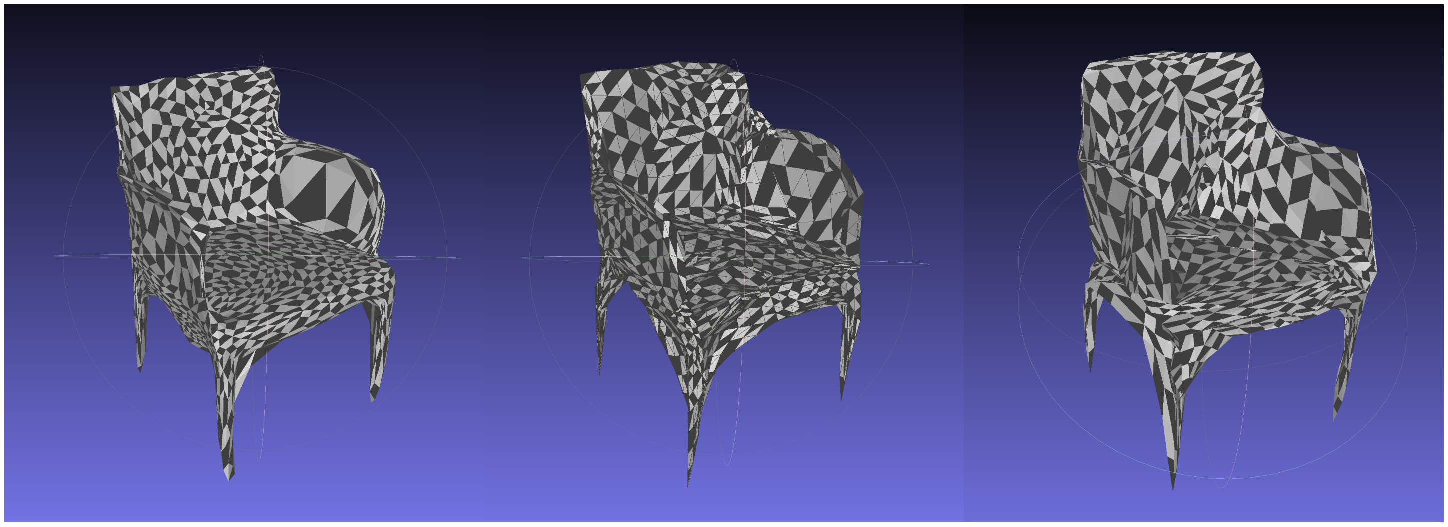
Fig. 6: Input RGB image(s) taken from different camera angles

V. Task Arrangement

The tasks distribution refers to the list in section III-A.
Tasks 2,3,5: Alexandre Cafaro
Tasks 1,4,6,7: Sebastian Lohr & Liam Hulsman-Benson

VI. Conclusion and Future Work

Throughout the course of the project, implementation of the reference paper has been achieved in PyTorch replacing Tensorflow originally used, as well as improving the network by replacing the VGG network by ResNet and proposing a method for stereo imaging.
With the results shown in Fig. 7 it can be seen that both the ResNet and the Stereo approach produce mesh deformations similar to the original paper.
Due to the limited access time to a server both networks had very limited training time, therefore the results of the original paper could not be improved upon. Performance improvements were still visible with each epoch at the end of the training time.
This was surprising as we transferred the weights from the VGG network and had a comparatively high training time to the 72 hours of the original Pixel2Mesh network but with a less powerful GPU.
To perform a final analysis of the chosen approach, additional training time is needed so that its full performance is demonstrated.
It would have been interesting to be able to vary more the hyper parameters (e.g. learning rate, learning rate decay, weight decay) for the training of the networks. The convergence of such networks remains hard.
Concerning Stereo or more images as input, the advances made in Pixel2Mesh++ [8] leading to state-of-the-art results



(a) VGG (b) ResNet (c) Stereo

Fig. 7: Visual comparison

show the way of future improvements, by merging directly output meshes of several mono input images, deforming and combining them in an efficient way. The complexity of the Pixel2Mesh network itself does not need to be increased, as good results have been achieved with one input and VGG. Rather, more efficient and low-computation cost nets can be used and allow further post-processing and fusion of multiple input images.

Adding stereo inputs for better photometric consistency and generalizing to whole scenes could be the next challenges.

References

[1] Xu Cao and Katashi Nagao. Point Cloud Colorization Based on Densely Annotated 3D Shape Dataset. Technical report.

[2] Chris Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. *Arxiv*, 04 2016.

[3] Christopher B. Choy, Danfei Xu, Jun Young Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9912 LNCS, pages 628–644. Springer Verlag, apr 2016.

[4] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image, 2016.

[5] Thibault Groueix. Pytorch chamfer distance, 2020.

[6] Stanford. Pytorch chamfer distance, 2017.

[7] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11215 LNCS:55–71, apr 2018.

[8] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. Pixel2Mesh++: Multi-View 3D Mesh Generation via Deformation. pages 1042–1051, aug 2019.