An Effective Loss Function for Generating 3D Models from Single 2D Image Without Rendering

Nikola Zubić, Pietro Liò

An Effective Loss Function for Generating 3D Models from Single 2D Image without Rendering

Nikola Zubić*[1][0000−0001−9816−2718] and Pietro Liò[2][0000−0002−0540−5053]

[1] Faculty of Technical Sciences, Novi Sad 21125, Serbia
nikola.zubic@uns.ac.rs
[2] University of Cambridge, Cambridge CB3 0FD, United Kingdom
pietro.lio@cst.cam.ac.uk

Abstract. Differentiable rendering is a very successful technique that applies to a Single-View 3D Reconstruction. Current renderers use losses based on pixels between a rendered image of some 3D reconstructed object and ground-truth images from given matched viewpoints to optimise parameters of the 3D shape.
These models require a rendering step, along with visibility handling and evaluation of the shading model. The main goal of this paper is to demonstrate that we can avoid these steps and still get reconstruction results as other state-of-the-art models that are equal or even better than existing category-specific reconstruction methods. First, we use the same CNN architecture for the prediction of a point cloud shape and pose prediction like the one used by Insafutdinov & Dosovitskiy. Secondly, we propose the novel effective loss function that evaluates how well the projections of reconstructed 3D point clouds cover the ground-truth object's silhouette. Then we use Poisson Surface Reconstruction to transform the reconstructed point cloud into a 3D mesh. Finally, we perform a GAN-based texture mapping on a particular 3D mesh and produce a textured 3D mesh from a single 2D image. We evaluate our method on different datasets (including ShapeNet, CUB-200-2011, and Pascal3D+) and achieve state-of-the-art results, outperforming all the other supervised and unsupervised methods and 3D representations, all in terms of performance, accuracy, and training time.

Keywords: 3D Reconstruction · Single-View 3D Reconstruction

1 Introduction

One of the main problems in 3D Computer Graphics and Vision is the ability of a model to learn 3D structure representation and reconstruction [9]. Supervised 3D Deep Learning is highly efficient in direct learning from 3D data representations [1], such as meshes, voxels, and point clouds. They require a large amount of 3D data for the training process, and also, their representation is sometimes

* Work performed while the author was Research Intern Apprentice under the supervision of professor Pietro Liò.

complex for the task of direct learning. These factors lead to the abandonment of this approach because of its inefficient performance and time consumption. Unsupervised 3D structural learning learns 3D structure without 3D supervision and represents a promising approach.

Differentiable rendering is a novel field that allows the gradients of 3D objects to be calculated and propagated through images [11]. It also reduces the requirement of 3D data collection and annotation, while enabling a higher success rate in various applications. Their ability to create a bond between 3D and 2D representations, by computing gradients of 2D loss functions with the respect to 3D structure, makes them a key component in unsupervised 3D structure learning. These loss functions are based on differences between RGB pixel values [13]. By rendering the predicted 3D structure from a specific viewpoint and then evaluating the loss function based on pixel-wise loss between rendered and ground-truth image, model parameters are optimised to reconstruct the desired 3D structure.

However, these evaluation techniques are very time-consuming. They don't contribute at all to an accurate 3D structure reconstruction. Here, we propose a novel idea for fast 3D structure reconstruction (in the form of a point cloud silhouette) and then we convert it to a 3D mesh and transfer the object's texture from a 2D image onto the reconstructed 3D object. Hence, unlike in loss functions that are based on pixels, our approach has an effective loss function that arises exclusively from the 2D projections of 3D points, without interpolation based on pixels, shading and visibility handling.

2 Related work

2.1 3D representations

Previous works [23,29] have concentrated on mesh reconstruction by using the full 3D supervision approach. The main problem with these approaches, besides inefficiency, is the usage of ground-truth 3D meshes, and they are mostly available in a limited number of datasets. Some approaches [24,25] solved this problem by using 2D supervision from multiple-scene images based on voxels.

2.2 Differentiable rendering

Prediction of 3D models from single images while achieving high-quality visual results is possible by using the differentiable renderer. A differentiable rendering framework allows gradients to be analytically (or approximately) computed for all pixels in an image. Famous frameworks include: RenderNet [19] and OpenDR [17].

2.3 Unsupervised learning of shape and pose with differentiable point clouds

The work that inspired us addresses the learning of an accurate 3D shape and camera pose from a collection of unlabeled category-specific images [8]. It uses a

specific convolutional neural network architecture to predict both model's shape and the pose from a single image. However, it is still time-consuming since it uses differentiable point cloud projection.

3 Proposed method

3.1 Intuitive overview

In order to overcome the problems of structural 3D learning, unsupervised methods introduced different differentiable renderers [8,9,11,3,16,17,19] to first render the reconstructed 3D shape into 2D images from different view-angles and then portray them as what got obtained through complete supervision. After this, we can calculate the pixel-wise losses between those 2D images from different view-angles and real (ground-truth) images from the dataset. Since the renderer is differentiable, the loss between these images back-propagates through the network to train it.

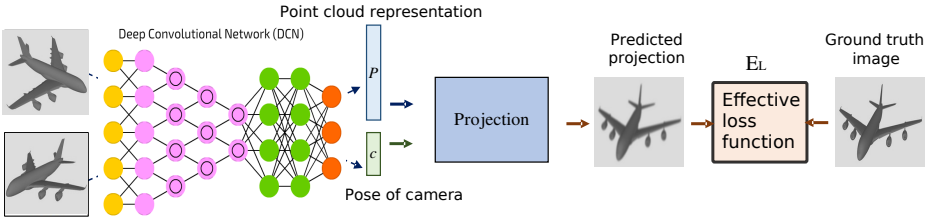

Fig. 1. Our method removes the rendering process and requires only 2D projections of 3D point clouds. During the generation of 3D shapes using multiple silhouette images (from different viewing angles), 2D projections of all points on the shape should uniformly cover the silhouette from each viewing angle. We implement this using two key ideas (that together form effective loss function). (1) For 3D shapes formed by 3D points, their projections for each view should locate within the silhouette. (2) All projections for each silhouette should distribute uniformly. We achieve this by maximising the loss between each of the pairs of these 2D projections. **P** - Point cloud representation, **c** - Pose of camera.

To evaluate the pixel-wise loss, previous differentiable renderers rendered the images by taking into account some form of interpolation [3] of the reconstructed 3D structure over each pixel, such as rasterisation and visibility handling.

We train a network that learns to generate a 3D point cloud based on a single image using the images from a dataset (from different view-angles) as supervision which is opposed to those that use ground-truth point clouds as supervision.

Current methods render based on differentiable renderers that render images of the reconstructed 3D shape and actual images and then minimise the pixel-wise loss to optimise the reconstructed 3D shape.

Total effective loss informs us how well the projected points cover the objective silhouette. The process includes two terms, one that forces all the projections

into the silhouette where the projections initialise randomly, and the other term moves the projections such that the distance between every two of them is the maximum possible, which allows the projections to cover the silhouette uniformly. Starting from some point cloud (randomly initialised), we can force all the projections in the silhouette using the first term, and then using the second term, we can uniformly distribute projections to cover the whole silhouette.

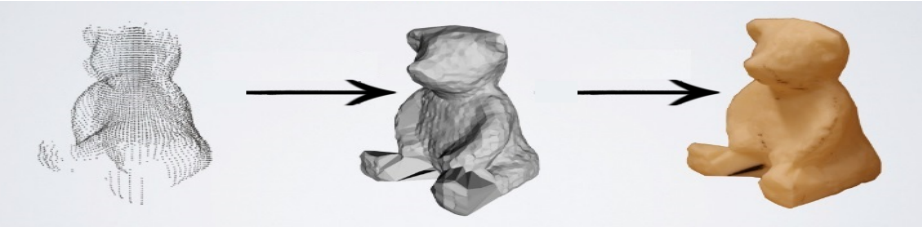

Fig. 2. Generated 3D point cloud is transformed into 3D mesh and then textured.

After completing the process shown in Figure 1, we generate 3D point cloud for a desired image. After this, we apply Poisson Surface Reconstruction [12] to generate 3D mesh from given 3D point cloud and then we use GAN for texture mapping on a particular 3D mesh and produce a textured 3D mesh based on the input image texture, which is shown in the Figure 2. Main paper deals with implementation details of an Effective Loss Function E_L which is our novelty, and compares the results with other approaches. More details and case study is available in Appendix A.

3.2 Implementation details

Our goal is to learn the structure of 3D point clouds (P) formed by N points n_j only from G_t ground-truth images of the silhouette S_i, where $j \in [1, N]$ and $i \in [1, G_t]$. Current differentiable renderers rely on point clouds (P) rendering into raster images S_i' from i-th viewing angle, which are used to produce a loss by comparing S_i' and S_i pixel by pixel. These steps are not necessary to get a precise solution.

Let the projection of the point n_j in view i be p_j^i. The error evaluates how well the sets of projected points $\left\{\boldsymbol{p}_j^i \mid j \in [1, N]\right\}$ cover the silhouette of the object. So, the loss is composed of two parts.

If we have a predicted 3D point cloud and a binary image of the silhouette, the loss calculates as follows: First, we project the points n_j and get projections p_j (we write abbreviated without i, this is $\boldsymbol{p}_j^i$) on the images of the silhouette S_i, where the pixel value of the projection p_j is denoted by $\boldsymbol{\pi_i}$. The first term penalises points outside of the foreground by calculating the difference $\mathbf{1} - \boldsymbol{\pi_i}$, assuming that the foreground in the binary silhouette image has a value of 1. Minimising this loss will force all projections into the foreground. Additionally,

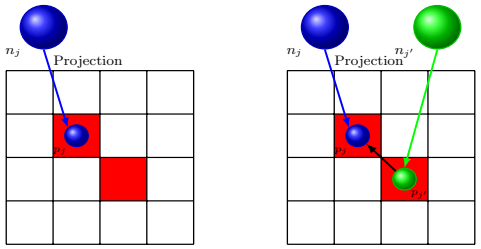

Fig. 3. The left and right grids represent two ground-truth silhouette images S_i. n_j point projects onto an image, and its projection is p_j. **Left:** For 3D shapes formed by 3D points, their projections for each view should locate within the silhouette, where the whole white grid is a silhouette, and its pixel values are 1 (red square). So, we are minimising differences between pixel values of projections and 1 for every projection. **Right:** Besides that, we must not only minimise the first term loss but also the second term loss which maximises the distance between two different point projections from a 3D point cloud: p_j and $p_{j'}$.

the second term adjusts the spatial distribution of the projected points. It forces the pairs of projections in the foreground to be as far apart from each other as possible (right grid shown in the Figure 3). Thus, such a system arranges the 3D locations of the points n_j through their projections p_j by simultaneously optimising these two losses.

The first term is calculated as the difference between 1 and the pixel value π_i of each projection p_j^i on the silhouette image S_i. We make use of bilinear interpolation to calculate the value π_i using the binary pixel values of the nearest pixels around p_j^i. All projections are forced to the foreground for all silhouette images by minimising the following L_1 norm:

$$L_1\left(\boldsymbol{\pi}_i\right) = \left\|1 - \boldsymbol{\pi}_i\right\|_1 \tag{1}$$

However, it is impossible to force all projections into the foreground by minimising this $\boldsymbol{L1}$ norm. If we optimise point cloud according to some silhouette image (a) and start from some randomly initialised points (b), then we will get inadequate point projections if we use $L1$ norm, as shown in the Figure 4.

There are two reasons why this problem occurs. One reason is the fact that $L1$ norm is non-differentiable. Even if we only look at the difference, the second reason is that we only examine the pixel intensity based on the difference between 1 and the interpolated pixel value π_i based on the four closest binary pixel values. This prevents training if the projections p_j^i are too far from the foreground.

Our goal is to produce non-zero gradients anywhere in the background part, while the pixel values in the foreground part do not require a modification. We will denote these processed silhouette images as S_i^G, to distinguish between the original silhouette image S_i and the processed image. For each pixel x on the background of the silhouette image S_i, we write:

$$\boldsymbol{S}_i^G(\boldsymbol{x}) = \begin{cases} 1, \boldsymbol{x} \in \mathcal{F} \\ 1 - d(\boldsymbol{x}, \partial\mathcal{F}), \boldsymbol{x} \in \bar{\mathcal{F}} \end{cases} \tag{2}$$

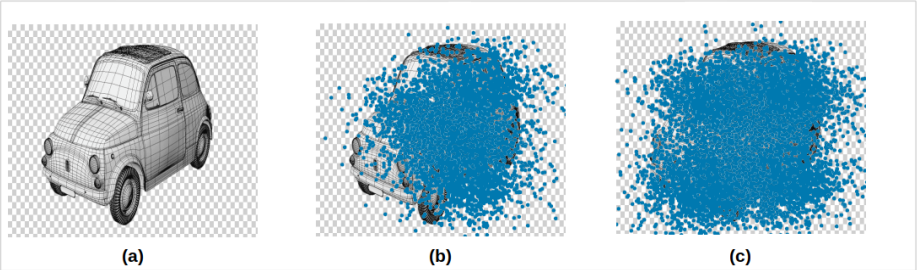

Fig. 4. We are given a silhouette image **(a)** and randomly initialized projections **(b)**. Then we cannot force projections into the foreground **(c)** because standard first term loss has a local minimum problem, which results in a non-uniform disposition of projections (blue dots). This problem is solved by smoothing the original silhouette to obtain pixel values of projections and calculate the difference.

where $\mathcal{F} = \{\boldsymbol{x} \mid \boldsymbol{\pi}_i(\boldsymbol{x}) = 1\}$ is the foreground, while $\bar{\mathcal{F}} = \{\boldsymbol{x} \mid \boldsymbol{\pi}_i(\boldsymbol{x}) = 0\}$ is a background, and $\partial\mathcal{F}$ is the foreground's boundary. $d(\boldsymbol{x}, \partial\mathcal{F})$ represents the L_2 distance between x and his closest $\partial\mathcal{F}$, which is normalised by the resolution of the S_i.

Normalisation is also performed on the processed pixel values in the background for them to lie in the interval $(0, 1)$. Min-max normalisation is used for this sub-task: $\boldsymbol{S}_i^G(\bar{\mathcal{F}}) = \text{minmax}\left(\boldsymbol{S}_i^G(\bar{\mathcal{F}})\right)$. Finally, the modified first term loss function is:

$$\mathcal{M}_1\left(\boldsymbol{\pi}_i\right) = \left\|1 - \boldsymbol{\pi}_i^G\right\|_1 \tag{3}$$

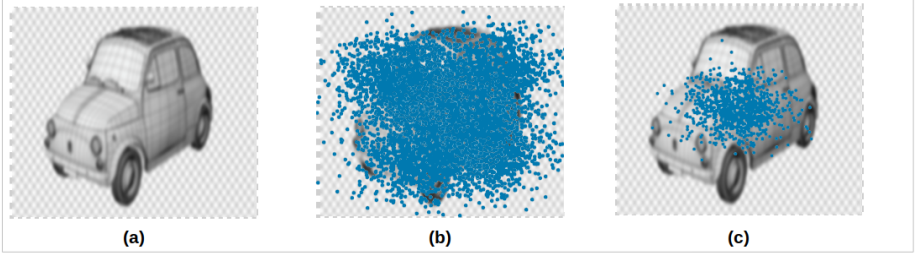

Fig. 5. With modified first term loss function, it is possible to force all projections of randomly initialised points **(b)** from a smoothed silhouette image **(a)** into the foreground **(c)** by minimising the (3). Blue dots represent the projections.

According to Figure 5, using only the first term loss leads to non-uniform point projections in the foreground. To accurately represent the 3D shape and cover the silhouette, we will use a second term loss. Through this loss, we will model

the spatial relationship between every two pairs of projections. That loss should force projections inside the foreground. They should be as far away from each other as possible.

To solve this problem, we propose a second-term loss function that increases the distance between projection pairs that are deeper within the foreground and reduces it for projection pairs around the foreground's boundary. It skips projections within the background.

For every projection pair p_j^i and $p_{j'}^i$, the $L2$ distance is calculated by the formula:

$$d\left(p_j^i, p_{j'}^i\right) = \left\|p_j^i - p_{j'}^i\right\|_2, \tag{4}$$

which we then normalise according to the resolution of the silhouette image.

This approach tends to maximise the distance $d\left(p_j^i, p_{j'}^i\right)$. We use the Gaussian function [14] to obtain a loss based on the invariance of the structure which decreases with increasing the distance. So, we can essentially minimise the loss of invariance along with the modified first term loss $\mathcal{M}_1$.

For each projection p_j^i, loss based on the invariance of the structure models its spatial relationship with all other projections $p_{j'}^i$:

$$\mathcal{L}_2\left(p_j^i, \{p_{j'}^i\}\right) = w_j^i \sum_{j'=1}^{N} \left[w_{j'}^i \cdot \exp\left(\frac{-d\left(p_{j'}^i p_{j'}^i\right)}{\theta} + \mu_j^i \right) \right], \tag{5}$$

where w_j^i and $w_{j'}^i$ are weights corresponding to the projections p_j^i and $p_{j'}^i$ respectively, $\theta > 0$ is the decay parameter, $\mu_j^i > 0$ is the boundary bias for the projection p_j^i.

w_j^i expresses to what level the projection p_j^i merges with the background. If that weight is set to zero, the projection p_j^i is such that the invariance of the structure is completely removed so that the modified first term loss $\mathcal{M}_1$ immediately forces p_j^i within the foreground. The decay (merge) parameter controls the merge interval (invariance of the structure intensity) of a given background 3D model. The projection boundary bias μ_j^i for projection p_j^i controls the distance to the foreground's boundary where the invariance over that projection reduces.

Weight w_j^i is calculated using bilinear interpolation based on the closest binary pixel values in the silhouette image S_i, as shown in the Figure 3. We use multi-scale gradients [21] to compute μ_j^i. Binary pixel values are extracted from adjacent points located at the vertices of the squares in the grid (around the projection p_j^i) - Figure 3. We perform interpolations over them, and we take the mean value of all of these interpolations to calculate μ_j^i. This approach progressively reduces invariance of the structure as p_j^i approaches the foreground's boundary.

Finally, the effective loss function E_L is calculated through simultaneous minimisation of the modified first term loss function $\mathcal{M}_1$ and the second term loss function $\mathcal{L}_2$ based on the invariance of the structure. The total error E_L is

obtained by the following formula (α and β are used for balancing the losses, we average over points N and views G_t):

$$E_L = \frac{\sum_{i=1}^{G_t} \sum_{j=1}^{N} \left(\alpha \mathcal{M}_1 \left(\pi_i\right) + \beta \mathcal{L}_2 \left(p_j^i, \left\{p_{j'}^i\right\}\right)\right)}{G_t \cdot N} \tag{6}$$

After this process, we have a 3D point cloud which is then transformed to a 3D mesh using Poisson Surface Reconstruction [12]. We use GAN [30] for texture mapping on a particular 3D mesh and produce a textured 3D mesh based on the input image texture, which is shown in Figure 2. The generator generates displacement maps and textures, and the discriminator discriminates between real/fake displacement maps and textures.

4 Results

In this section, we succinctly report the results, primarily through comparison with other approaches. More details and case study is available in Appendix A.

The quantitative results using Chamfer's distance [22] are shown in Table 1. Our point cloud output (Ours) outperforms its voxel equivalent (Ours-V) in all cases. Chamfer's distance improves with the increase of resolution. We also outperform the previous best method that used rendering [8] and DRC method [24].

Table 1. Quantitative results on shape prediction with known camera pose (on ShapeNet dataset). We report the Chamfer's distance between normalised point clouds, multiplied by 100 and use three categories: Airplanes, Cars and Chairs. Our point cloud output outperforms all other methods in terms of Chamfer's distance. Lower value is better; bold = best.

	Resolution 32				Resolution 64		Resolution 128	
	DRC [25]	DPC [8]	Ours-V	Ours	DPC [8]	Ours	DPC [8]	Ours
Airplane	8.35	4.52	4.49	**3.99**	3.50	**3.15**	2.84	**2.63**
Car	4.35	4.22	3.75	**3.79**	2.98	**2.86**	2.42	**2.37**
Chair	8.01	5.10	5.34	**4.64**	4.15	**3.99**	3.62	**3.46**
Mean	6.90	4.61	4.53	**4.14**	3.55	**3.33**	2.96	**2.82**

Our results outperform state-of-the-art differentiable renderers in the Volumetric IoU metric [20] while simultaneously being less time-consuming during the training phase, as shown in Table 2. For cars, our outcome is better than renderers based on voxels but very similar to renderers based on meshes because meshes represent a superior initial 3D representation for large areas of flat surfaces [10] (such as cars).

Table 2. Quantitative Volumetric IoU [20] comparison with differentiable renderers for different 3D representations and supervised methods (on ShapeNet dataset). We use three categories: Airplanes, Cars and Chairs. Bigger value is better; bold = best.

	Unsupervised learning			Supervised learning							
	SoftRas [16]	DIB-R [3]	Ours	P2M [27]	IN [15]	RN [4]	AN [6]	DSN [32]	3DN [28]	ON [18]	Ours
Airplane	58.4	57.0	**62.4**	51.5	55.4	42.6	39.2	57.5	54.3	57.1	**75.3**
Car	77.1	**78.8**	75.6	50.1	74.5	66.1	22.0	74.3	59.4	73.7	**75.1**
Chair	49.7	52.7	**58.3**	40.2	52.2	43.9	25.7	54.3	34.4	50.1	**57.8**
Mean	61.7	62.8	**65.43**	47.3	60.7	50.9	29.0	62.0	49.4	60.3	**64.97**

Table 3. Training time efficiency in hours.

	$3D$ representations	Rendering	32^2 image 2000 points/ 32^3 voxels	64^2 image 8000 points/ 64^3 voxels	128^2 image 16000 points/ 128^3 voxels
DRC [25]	Voxels	Yes	≈14h	≈60h	≈216h
DPC [8]	Point clouds	Yes	≈14h	≈24h	≈72h
Ours	Point clouds	No	≈**6.5**h	≈**11**h	≈**34.5**h

Also, FID scores on Mesh (produced from 3D point cloud), Texture (extracted by a GAN) and Both (final output - textured 3D mesh) produced state-of-the art results, which can be seen in Figure 6.

5 Datasets, metrics & code

Datasets We used the following datasets: ShapeNet [2] (train/test split from [8]), CUB-200-2011 [26] (train/test split from [10]), and Pascal3D+ dataset [31] (train/test split from [10]).

Metrics Numerical evaluation for point clouds is performed by using Chamfer's distance [22] between predicted and real (ground-truth) point clouds.

Volumetric IoU [20] comparison is used by comparing the 3D grid voxelised from the predicted point cloud with the one voxelised from the ground-truth point cloud.

Fréchet Inception Distance (FID) is widely used as an evaluation metric [7] (not only for 2D GANs but also for our task). FID scores will evaluate 2D projections of generated point clouds to meshes. 3D mesh and textures are evaluated separately in this process.

Code Implementation, data and trained models are available at:
https://github.com/NikolaZubic/2dimageto3dmodel

6 Possible extensions and limitations

Our work can be used as part of more complex software that deals with video games, animation, or any aspect where it is necessary to have base 3D mod-

els which can be additionally polished with more sculpting. The work can be extended by taking even more account of the smooth characteristics of the functions. Our work is the first one to tackle the challenging problem of Single-View 3D Reconstruction without Rendering. Results are impressive, but this task is far from being fully solved. Our model struggles to predict camera poses that are rare in the training dataset. Also, it captures the major shape characteristics of each instance but ignores some details. For example, legs of zebras, cows and horses are not separated (Figure **??** and Figure **??**).

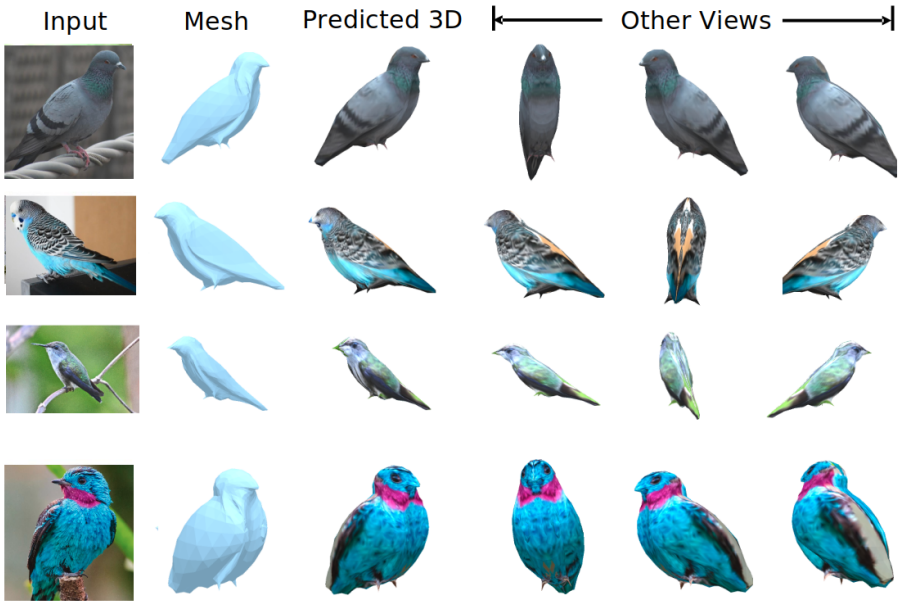

Fig. 6. We use real-world 2D bird images as input for generating a 3D model. In the first column is the input where we have images of the real birds, in the second column, there is a generated 3D mesh (obtained from 3D point cloud after Poisson Surface Reconstruction [12]), and in the next four columns, there is a predicted 3D model visible in 4 poses.

7 Conclusion

In this paper, we proposed a 3D reconstruction based on a single image, a method for learning the pose and shape of 3D objects given only their 2D projections, using the initial point cloud representation and then converting that representation to a 3D mesh. Mesh is textured using GANs to produce the final output. Extensive experiments have shown that point clouds compare well with the voxel-based representation, such as performance and accuracy. The proposed

framework learns to predict shape, texture, and pose from single images, without rendering step, based solely on 2D projections of 3D point clouds and their coverage of the ground-truth silhouette. While rendering requires exhaustive computation, our key finding is that it does not endow accuracy in 3D structure learning.

8 Acknowledgements

We would like to thank the anonymous reviewers for reviewing the paper before final submission and providing helpful and detailed comments.

References

1. Ahmed, E., Saint, A., Shabayek, A.E.R., Cherenkova, K., Das, R., Gusev, G., Aouada, D., Ottersten, B.: A survey on deep learning advances on different 3d data representations (2019)
2. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: Shapenet: An information-rich 3d model repository (2015)
3. Chen, W., Ling, H., Gao, J., Smith, E., Lehtinen, J., Jacobson, A., Fidler, S.: Learning to predict 3d objects with an interpolation-based differentiable renderer. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 32, pp. 9609–9619. Curran Associates, Inc. (2019), https://proceedings.neurips.cc/paper/2019/file/f5ac21cd0ef1b88e9848571aeb53551a-Paper.pdf
4. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction (2016)
5. Community, B.O.: Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam (2018), http://www.blender.org
6. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: Atlasnet: A papier-mâché approach to learning 3d surface generation (2018)
7. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium (2018)
8. Insafutdinov, E., Dosovitskiy, A.: Unsupervised learning of shape and pose with differentiable point clouds. CoRR **abs/1810.09381** (2018), http://arxiv.org/abs/1810.09381
9. Jimenez Rezende, D., Eslami, S.M.A., Mohamed, S., Battaglia, P., Jaderberg, M., Heess, N.: Unsupervised learning of 3d structure from images. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 29, pp. 4996–5004. Curran Associates, Inc. (2016), https://proceedings.neurips.cc/paper/2016/file/1d94108e907bb8311d8802b48fd54b4a-Paper.pdf
10. Kanazawa, A., Tulsiani, S., Efros, A.A., Malik, J.: Learning category-specific mesh reconstruction from image collections (2018)
11. Kato, H., Beker, D., Morariu, M., Ando, T., Matsuoka, T., Kehl, W., Gaidon, A.: Differentiable rendering: A survey (2020)

12. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the Fourth Eurographics Symposium on Geometry Processing. p. 61–70. SGP '06, Eurographics Association, Goslar, DEU (2006)
13. Kumar, T., Verma, K.: A theory based on conversion of rgb image to gray image. International Journal of Computer Applications **7**(2), 7–10 (2010)
14. Li, Z., Shafiei, M., Ramamoorthi, R., Sunkavalli, K., Chandraker, M.: Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image (2019)
15. Liu, J., Lu, H.: Imnet: A learning based detector for index modulation aided mimo-ofdm systems (2019)
16. Liu, S., Chen, W., Li, T., Li, H.: Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction (2019)
17. Loper, M.M., Black, M.J.: Opendr: An approximate differentiable renderer. In: European Conference on Computer Vision. pp. 154–169. Springer (2014)
18. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space (2019)
19. Nguyen-Phuoc, T., Li, C., Balaban, S., Yang, Y.L.: Rendernet: A deep convolutional network for differentiable rendering from 3d shapes (2019)
20. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision (2020)
21. Sreegadha, G.: Image interpolation based on multi scale gradients. Procedia Computer Science **85**, 713–724 (2016). https://doi.org/https://doi.org/10.1016/j.procs.2016.05.258, `https://www.sciencedirect.com/science/article/pii/S1877050916306081`, international Conference on Computational Modelling and Security (CMS 2016)
22. Sun, X., Wu, J., Zhang, X., Zhang, Z., Zhang, C., Xue, T., Tenenbaum, J.B., Freeman, W.T.: Pix3d: Dataset and methods for single-image 3d shape modeling (2018)
23. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs (2017)
24. Tulsiani, S., Efros, A.A., Malik, J.: Multi-view consistency as supervisory signal for learning shape and pose prediction (2018)
25. Tulsiani, S., Zhou, T., Efros, A.A., Malik, J.: Multi-view supervision for single-view reconstruction via differentiable ray consistency (2017)
26. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.J.: The caltech-ucsd birds-200-2011 dataset (2011)
27. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G.: Pixel2mesh: Generating 3d mesh models from single rgb images (2018)
28. Wang, W., Ceylan, D., Mech, R., Neumann, U.: 3dn: 3d deformation network (2019)
29. Wu, J., Wang, Y., Xue, T., Sun, X., Freeman, W.T., Tenenbaum, J.B.: Marrnet: 3d shape reconstruction via 2.5d sketches (2017)
30. Xian, W., Sangkloy, P., Agrawal, V., Raj, A., Lu, J., Fang, C., Yu, F., Hays, J.: Texturegan: Controlling deep image synthesis with texture patches (2018)
31. Xiang, Y., Mottaghi, R., Savarese, S.: Beyond pascal: A benchmark for 3d object detection in the wild. In: IEEE Winter Conference on Applications of Computer Vision (WACV) (2014)
32. Xu, Q., Wang, W., Ceylan, D., Mech, R., Neumann, U.: Disn: Deep implicit surface network for high-quality single-view 3d reconstruction (2019)

A Appendix - Network architecture details

The network we used for the $3D$ reconstruction (in a point cloud representation) based on a single image is composed of a $2D$ encoder and a $3D$ point cloud decoder. The $2D$ encoder represents a 7-layer *CNN*. The first layer consists of a 5×5 kernel with 16 channels and a stride of 2. Each of the remaining layers has three kernels and comes in pairs, where the given layer in pairs has a stride of 2, while the second one has a stride of 1. The number of channels increases by a factor of 2 after each layer with a stride. These convolutional layers are followed by two fully connected layers whose dimensions are 1024. The $3D$ point cloud decoder has one fully connected layer whose dimensions are 1024, and it then predicts the point cloud representation. The point cloud that is consisted of N points gets predicted as a vector whose dimensions are $3N$ (point coordinates).

We have chosen this architecture because it achieved state-of-the-art results for the problem of Single-View 3D Reconstruction, but the process of differentiable rendering was unnecessary, and we obtained a more precise solution without it. This architecture represents an optimal solution because it can firmly reconstruct real-world data, despite the absence of accurate ground-truth camera poses. Also, it can be used as a basis to learn colors and textures, but that would require explicit reasoning about lighting and shading.

B Appendix - More results and discussion

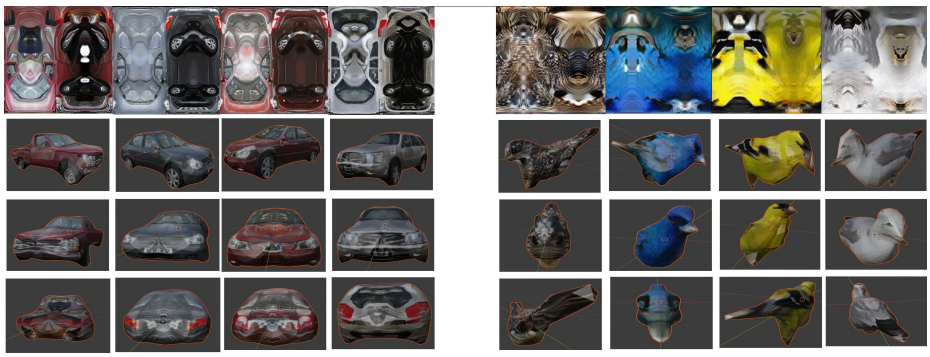

Fig. 7. Qualitative results on Pascal3D+ (**left**) and CUB-200-2011 (**right**) dataset. Each object has been rendered from three views (in Blender [5]), and the top row represents the texture learned by GAN.

Figure 7 shows a few generated, textured meshes rendered from the multiple views in Blender [5], and also their corresponding textures. Results on the CUB-200-2011 dataset have high resolution, but the back of the cars in the Pascal3D+ dataset has some irregularities. After further analysis, we found that the dataset

is very imbalanced, with only 20% of the images showing the back of the car and the majority of them showed the frontal part. So, this issue could be solved by using more training data.

C Appendix - Ablation study and efficiency

We carried out ablation studies to justify our claims in terms of the effectiveness of each element of our method under airplanes at a resolution of 2000 points in Table 1. In Table 4, we report results with only some losses, fewer views (like $G_t = 3$ and $G_t = 2$), and without weights and biases.

Table 4. Ablation studies in terms of Chamfer's distance (CD).

	$\mathcal{M}_1$	$\mathcal{L}_2$	Pixel+$\mathcal{L}_2$	$\mathcal{M}_1$ + no w_j^i	$\mathcal{M}_1$ + no μ_j^i	$G_t = 2$	$G_t = 3$	$G_t = 4$
CD	19.50	139.10	24.59	4.58	4.41	4.79	4.54	**4.01**

This study shows that our loss E_L cannot learn the structure of shapes using only $\mathcal{M}_1$ or $\mathcal{L}_2$ loss, also not with standard L_1 loss because of the local minimum issue and non-differentiability. The second term loss and its hyperparameters (indicator weights and boundary bias) contribute to the reconstruction accuracy and efficiency of optimization. Parameters α and β contribute to the conflict and trade-off between modified first term loss and second term loss based on structure invariance. Using fewer views than our $G_t = 4$ views in training degenerates the structure learning performance.

As for efficiency, we compared our model's training time with state-of-the-art differentiable renderers for 3D shapes, as shown in Table 3. The voxel-based method (DRC) has a weakness in terms of a vast computational burden due to the cubic complexity of voxel grids, which limits it to work only in low resolutions such as 32^3 and 64^3 with a slow convergence rate. Although the point cloud-based method by Insafutdinov & Dosovitskiy [8] does not require 3D convolutional layers as DRC, the rendering procedure still requires intensive computation with discrete 3D grids. So, this method requires more time (6×10^5 mini-batch iterations) during training than our method (1×10^5 mini-batch iterations).

We used parameters learned in different steps during training to reconstruct a shape from a corresponding image in a test set. Additionally, by using an image from test rather than the training set we demonstrated the generalization ability learned in optimization, which strongly justifies our effectiveness. Also, it is shown that our model adapts well to real-world images.