Shape-from-Silhouette using Visual Hull and Deep Image Prior

Gokhan Egri
gokhaneg@mit.edu

Xinran (Nicole) Han
xinranha@mit.edu

Abstract

Visual hull construction is a preliminary step for a majority of 3D shape reconstruction tasks and as such poses an important problem for many sub-fields of computer vision. In this work, we first implement and evaluate a familiar voxel-based visual hull construction algorithm which serves as the baseline for our proposed method. For our proposed method, we extend the original Deep Image Prior method by Ulyanov et al. *to the problem of visual-hull construction by viewing the $3D \rightarrow 2D$ projection as a corruption. We find that our proposed method is both capable of converging on viable visual hulls for an array of different objects and resilient to noise and various occlusions. We also present some preliminary results for our method on 3D denoising and 3D inpainting.*

1. Introduction

At a glance, from close or afar, shape is what makes us identify the fundamental properties of an object: what it is and what it is for. From the early work on shape from shading [4][16], texture [9] and silhouette [6] to recent deep learning based approaches [3], a considerable portion of researches in vision have targeted recovering 3D shape using one or multiple images. Unlike approaches in shape from shading/texture that require knowledge of the material property, shape from silhouette methods such as visual hull can be applied to various types of objects as long as the segmentation and camera parameters are known. The basic principle of visual hull is to create a 3D representation of an object using its silhouettes from various viewpoints. Each of the silhouette from a camera view constrains the object within a visual cone. The intersection of all those cones gives an approximation to the object shape as seen in 1.

Despite the robustness of the classical visual hull algorithm, learning based approaches rarely leverage ideas from this method. For our project, we hope to bridge the gap between geometry-based visual hulls and learning based reconstruction algorithms, which will shed light on the ability of neural networks to perform shape reconstruction. Our contribution is three-fold [1]:

- We provide a voxel-based visual hull implementation in Python, which is previously not available on open source platforms (*i.e.* Github).
- We extend the Visual Hull algorithm and combine it with the architecture of Ulyanov *et al.*'s Deep Image Prior to investigate learning-based shape reconstruction. We motivate Deep Visual Hull Prior through evaluations on 3D inpainting and 3D denoising.
- We also provide a Python script to generate synthetic multi-view stereo data from custom objects in Blender.

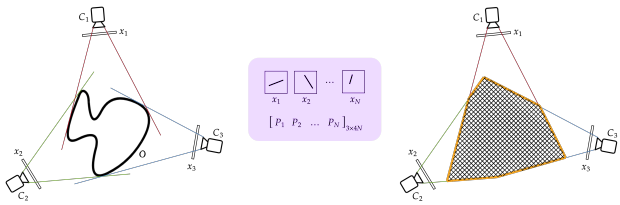

Figure 1. In a 2D example, the intersection of visual cones from different camera views form the Visual Hull of object $\mathcal{O}$.

2. Related Work

Shape from Silhouette. The Visual Hull concept is closely related to the 3D reconstruction method Shape-from-Silhouette (SfS), which was introduced by Baumgart in 1974 [2] and Laurentini in 1994 [6]. As we show later in this report, voxel-based visual hull SfS is easy to implement and can provide an upper bound for object shape without knowing the reflectance and texture properties of the object. The output of visual hull can be later used for a variety of tasks, such as navigation and obstacle avoidance in robotics. However, there are also limitations of this method. For instance, camera calibration error may be present and limit the accuracy of the back-projection in SfS. The reconstruction also does not work well for concave objects. The method also requires an accurate segmentation of object silhouette

[1] The code for our project is available at https://github.com/egrigokhan/deep-visual-hull-prior

from its background, while in real life this task can be challenging and and result may be noisy.

Deep Image Prior. Ulyanov *et al*. [7] states that Deep Convolutional Networks (DCNs) which are widely-used in contemporary computer vision, are inherently equipped, due to their structure, to act as a prior for natural images which has high impedance for high-frequency content, *i.e*. noise. Deep Image Prior (DIP) exploits this impedance-to-noise for carrying out "standard inverse tasks" such as denoising, superresolution, and inpainting which can be characterized as Energy Minimization Tasks of the form

$$x^* = \min_x E(x, x_0) + R(x) \tag{1}$$

where $E(x, x_0)$ denotes a task related metric, x and x_0 are the original and corrupted images, and $R(x)$ is the image prior which is often reconstructed from the data. DIP assumes that $R(x)$ is inherently embedded within the structure of the DCNN and as such formats the task in terms of a model optimization as

$$x^* = f_{\theta^*} \text{ where } \theta^* = \arg\min_x E(f_\theta(z), x_0) \tag{2}$$

where, f is the CNN model, and z is a fixed matrix sampled from a normal distribution $\mathcal{N}(\mu, \sigma^2)$.

The CNN is then trained to overfit on the original corrupted image, however due to the network's high impedance to noise, the network first overfits onto an uncorrupted, low-frequency version of the image and only then proceeds to overfit the corruptions, *i.e*. the high-frequency components. Ulyanov *et al*. finds that using early-stopping in the middle of the training process allows for state-of-the-art results in many inverse task (*e.g*. inpainting, denoising, superresolution) benchmarks.

3. Methodology

Problem Statement. Given N binary masked images $[x_1, x_2 \dots x_N]$ of an object $\mathcal{O}$ from N cameras views with projection matrices $[P_1, P_2, \dots P_N]$, we want to recover an upper-bound hull-approximation $\mathcal{S}$ of the object. (For computational and evaluation purposes, we represent the 3D shape as a voxel grid instead of an intersection of cones.)

Input. For $i \in N$ viewpoints, we have image x_i with associated projection matrix P_i that is computed from:

(1) Camera calibration matrix (same for all images)

$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$ where f is the principal distance and p_x, p_y are the principal point offsets.

(2) orthonormal rotation matrix R_i

(3) vector representation translation $t_i = (t_{ix}, t_{iy}, t_{iz})^T$

The projection matrix is given by the 3 by 4 matrix $K[R_i|t_i]$. Here, the $[R_i|t_i]$ component converts the object position in world coordinates to camera coordinates, and camera intrinsic matrix K further project the object onto the 2D image plane.

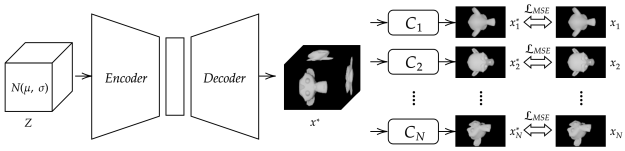

Figure 2. Proposed Deep Visual Hull architecture.

Output. Voxel grid V of dimension $M \times M \times M$. (For our project, since all the models are on the scale of $5 - 10$ centimeters, we choose $M = 90$ which we find gives a good approximation of the object shape.) For the deterministic method, each voxel stores a value $v_{x,y,z} \in [0, N]$. For the learning-based method, we have $v_{x,y,z} \in [0, 1]$ as our output becomes a confidence score which is explained in further detail momentarily. We then use Marching Cubes [1] [8] algorithm to convert it to a mesh form. During the conversion, we compute a threshold t to determine whether a particular voxel gets filled (binary 1) or not (binary 0) in the final reconstruction.

3.1. Deterministic Visual Hull

The Voxel Based Visual Hull is computed with the following algorithm:

1. For all M^3 voxels, initialize each with count 0.
2. For each silhouette $x_i, i = 1$ to N:
   For each voxel $v_j = (x, y, z), j = 1$ to $M \times M \times M$
   (a) Project the voxel onto the i-th image plane by $P_i(x, y, z, 1)^T$, where P_i is the projection matrix and $(x, y, z, 1)^T$ is the voxel v_j's homogenous coordinates.
   (b) If the projection lies within the object mask x_i, add 1 to v_j's total count.
3. Set a threshold t s.t. all voxels with count $\geq t$ are considered filled and within the object. Other voxels are considered in the background and are left empty.
4. Store the computed voxel grid $\mathcal{V}$ and threshold t. Further convert it to a mesh (.dae) object using Marching Cubes and obtain mesh representation $\mathcal{S}$.

3.2. Deep Visual Hull Prior

For our proposed method, our key observation is that the projection of the visual hull onto the N image planes it was constructed from can be seen as a corruption. We subsequently formulate visual-hull reconstruction as an inverse task and extend the Deep Image Prior architecture.

Our proposed Deep Visual Hull Prior (DVHP) network seen in 2 takes as input a matrix z sampled from a normal distribution $\mathcal{N}(\mu, \sigma)$ where μ and σ are parameters of the underlying distribution. Through a series of 3D convolutions and deconvolutions (we choose a U-Net architecture

as in the original Deep Image Prior paper), the network outputs a voxel-grid x^* with values for each voxel representing a confidence score between 0 and 1 for whether the voxel is part of the final visual hull or not.

We then project the approximate voxel-grid onto the N camera views using the known camera projection matrices to obtain N camera-views $\{x_1^*, x_2^*, \dots x_N^*\}$. Finally, the network is trained end-to-end using back-propagation over the MSE-loss between the reconstructed camera-views $\{x_1^*, x_2^*, \dots x_N^*\}$ and the original camera-views $\{x_1, x_2, \dots x_N\}$.

4. Experiments & Discussion

In this section, we perform experiments to evaluate our 3D reconstruction performance based on the visual hull algorithms: the deterministic method and the learning based method.

4.1. Dataset

Based on the visual hull algorithm, a desired dataset needs to contain images taken from a variety of views, together with the camera intrinsics and extrinsics. Therefore, we decide to perform our experiments on the Middlebury Multi-view Stereo dataset [11]. This dataset contains two objects, the DinoRing model and Temple model, each with 48 views taken on a semi-hemisphere around the object and are captured using the Stanford spherical light field gantry. However, the original authors have not released the ground truth model for those objects therefore we use this dataset mainly for qualitative evaluation.

To overcome this limitation, we use the Blender graphics software to generate synthetic multi-view stereo data. Similar to the original set-up in the Middlebury dataset, we sample a ring at 10 different heights and take photos of the same object from 12 angles, giving a total of 120 viewpoints on a spherical rig. The variety of viewpoints can provide more information for the learning-based model than the sparse data from Middlebury dataset.[2]

4.2. Evaluation Metric

In both algorithms, the generated output is a voxel grid. Since the ground truth model is a mesh object, we can evaluate our reconstruction performance by either converting the output prediction to a mesh or converting the ground truth to a voxel grid. Inspired by prior work [12][3], we align the model and ground truth and evaluate our reconstruction performance using the following metrics .

Voxel Intersection over Union (IoU). We use the conversion tool from [14], which converts the ground truth mesh into a voxel grid with specified dimension. When computing the **intersection-over-union (IoU)**, we divide the number of voxels that are filled by both models by the number of voxels filled in either one. A higher IoU indicates better reconstruction result.[3]

Surface Distance. We also evaluate based on mesh surface distance following the idea of Jensen *et al*. in [5]. For this metric, we convert the predicted voxel output to a mesh file and then sample 10k points to compute the distance. Using the software from [13], we obtain the **accuracy** and **completeness**. Accuracy is the distance of the reconstruction to the ground truth and completeness is the distance from ground truth to reconstruction. Lower accuracy/completeness indicates better reconstruction quality.[4]

4.3. Performance Evaluation

4.3.1 Deterministic Visual Hull

From preliminary experiments, we found that the voxel-based algorithm achieves good performance in reconstructing simple objects (*e.g*. cube, sphere, torus). Therefore, for evaluation we highlight its performance on complex objects: the DinoRing model from Middlebury dataset and Suzanne (Monkey) model from our synthetic dataset. The results are shown in 3.

For the **DinoRing** model, since we do not have the ground truth mesh, we use the 48-view reconstruction as a relative benchmark. From the performance plots, we see that after around 16 viewpoints, the performance plateaus.

For the **Suzanne (Monkey)** model, we are able to achieve an IoU of 0.847 with the ground truth when using all 120 viewpoints. We also provide a qualitative visualization of the reconstruction results in **Appendix D.**

4.3.2 Deep Visual Hull Prior for Reconstruction

We evaluate our proposed Deep Visual Hull Prior architecture on a synthetic dataset consisting of a cube, sphere and a torus as well as the more complex Suzanne (Monkey) model from before.

For the evaluation, we fix the normal distribution for the noise input z to $\mathcal{N}(0, 1)$ and the output voxel-grid x^* dimension to $90 \times 90 \times 90$. For the Marching Cube algorithm, we again use the public implementation from [1] with iso-value ψ as $\psi = 0.95 \max(x^*_{\{ijk\} \in [1,90]^3})$.

As we are not carrying out inverse tasks, which will be discussed momentarily, but evaluating whether or not our proposed architecture is capable of fitting to a viable visual hull, for this part of our evaluation we train for 1000 iterations and report the best results between the 600 and 1000 iterations.

[2]For completeness, we also provide a method to simulate the environment of the Middlebury dataset using the same camera parameters.

[3]The code to voxelize the mesh ground truth is available at https://github.com/xrhan/mesh-voxelization.

[4]The code for mesh distance evaluation is available at https://github.com/xrhan/mesh-evaluation.

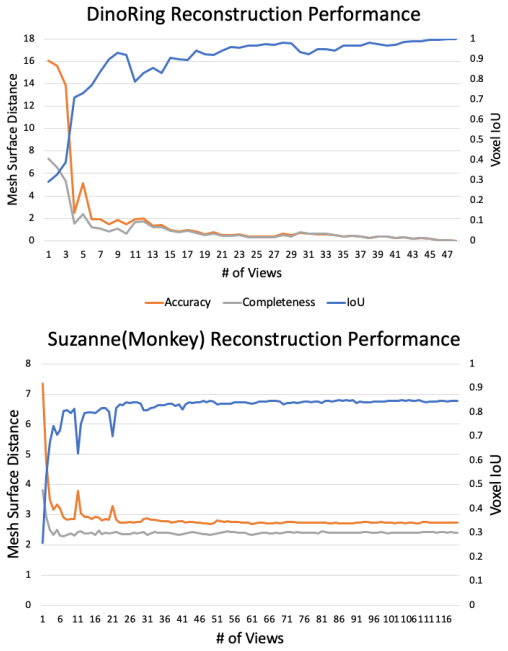

Figure 3. 3D reconstruction performance using deterministic voxel-based visual hull

Both our qualitative from 4 and quantitative results from 1 show that our proposed architecture is capable of converging on a viable visual hull representation for all three basic object classes as well as to an upper-bound approximation for the more complex Suzanne (Monkey) model from before.

The performance gap between polygonal objects (*e.g.* cube) and spherical objects (*e.g.* sphere, torus) is not surprising and arises from neural networks' well-documented spectral bias towards low-frequency content [10][15] which makes it difficult for CNNs to map high-frequency features (*e.g.* vertices, straight edges).

5. Conclusion

Our conclusions from this project can be summarized as follows:

- We find that **3D CNNs are capable of converging on viable visual hulls for objects supervised by multi-view images**. This further motivates the use of CNNs for more complex 3D reconstruction tasks as they are inherently capable of creating intrinsic visual hull representations of objects.
- The performance gap between reconstructions of polygonal and spherical objects can be attributed to deep neural networks' well-documented inability to map high frequency content addressed by [10][15].
- We find that this **impedance to high-frequency content enables DVHP to work under noise** and various other distortions which further motivates our proposed method (see **Appendix A**).
- Finally, this work can be extended to experiment with various inversion tasks such as (1) 3D noise removal, (2) 3D super-resolution and (3) 3D inpainting in compliance with the original Deep Image Prior paper.[5]

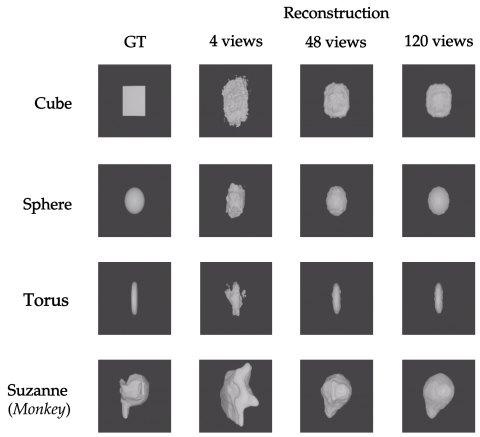

Figure 4. Qualitative results for the evaluation of the proposed DVHP architecture on Cube, Sphere, Torus and Suzanne (Monkey) models

Object	Metric	# of views		
		4	48	120
Cube	IoU (↑)	**0.2763**	0.2438	0.2407
	Accuracy (↓)	4.6658	**4.5218**	4.6082
	Completeness (↓)	**3.5595**	6.9307	7.0719
Torus	IoU (↑)	0.4356	0.5133	**0.5227**
	Accuracy (↓)	7.6789	**4.68561**	4.7144
	Completeness (↓)	2.3941	2.20575	**2.2000**
Sphere	IoU (↑)	0.6567	**0.7304**	0.7261
	Accuracy (↓)	2.6801	0.8107	**0.4738**
	Completeness (↓)	2.1231	0.7658	**0.4418**
Suzanne (Monkey)	IoU (↑)	0.3247	0.45170	**0.4633**
	Accuracy (↓)	11.5083	6.6413	**6.2728**
	Completeness (↓)	6.0204	5.7641	**4.5193**

Table 1. Quantitative results for the evaluation of the proposed DVHP architecture on Cube, Sphere, Torus and Suzanne (Monkey) models

[5]For preliminary results on inverse tasks not included in the scope of the original project, please see **Appendix A**.

References

[1] Pymcubes. https://github.com/pmneila/PyMCubes. 2, 3

[2] Bruce G Baumgart. Geometric modeling jor computer vision. *Computer Science Dept. of Stanford Univ*, 1974. 1

[3] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016. 1, 3

[4] Berthold KP Horn and Michael J Brooks. *Shape from shading*. MIT press, 1989. 1

[5] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413, 2014. 3

[6] Aldo Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on pattern analysis and machine intelligence*, 16(2):150–162, 1994. 1

[7] Victor Lempitsky, Andrea Vedaldi, and Dmitry Ulyanov. Deep image prior. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 2

[8] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 163–169, New York, NY, USA, 1987. Association for Computing Machinery. 2

[9] Jitendra Malik and Ruth Rosenholtz. Computing local surface orientation and shape from texture for curved surfaces. *International journal of computer vision*, 23(2):149–168, 1997. 1

[10] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks, 2019. 4

[11] S.m. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR06)*. 3

[12] Daeyun Shin, Charless C Fowlkes, and Derek Hoiem. Pixels, voxels, and views: A study of shape representations for single view 3d object shape prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3069, 2018. 3

[13] David Stutz. Learning shape completion from bounding boxes with cad shape priors. http://davidstutz.de/, September 2017. 3

[14] David Stutz and Andreas Geiger. Learning 3d shape completion from laser scan data with weak supervision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2018. 3

[15] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020. 4

[16] Ying Xiong, Ayan Chakrabarti, Ronen Basri, Steven J Gortler, David W Jacobs, and Todd Zickler. From shading to local shape. *IEEE transactions on pattern analysis and machine intelligence*, 37(1):67–79, 2014. 1

A. Preliminary Results for Inverse Tasks

As an extension on our original project, we also include an evaluation of the proposed DVHP model on inverse tasks, namely (1) 3D denoising and (2) 3D inpainting.

3D Denoising. For 3D Denoising, we add a radial particle field to the Sphere model from **Section 4.3.2** in Blender to simulate uniform ambient noise.

We find that while the classical voxel-based visual hull construction is able to remove the ambient noise from the reconstruction, it does so at the cost of deforming the sphere into an oblique ellipsoid (see $x^*_{baseline}$ from bottom row in 5). DVHP on the other hand is able to both remove the ambient noise as well as largely preserve the geometry of the sphere.

3D Inpainting. For 3D Inpainting, we apply a Boolean difference modifier on the Sphere model from **Section 4.3.2** with an oblique plane which removes a cross-section of the object.

We find that while the classical voxel-based visual hull construction is unable to recover the uncorrupted Sphere, DVHP's high-impedance to noise allows our proposed model to converge to an uncorrupted, low-frequency model of the sphere which allows us to inpaint the removed cross-section.

B. Deep Visual Hull Prior Detailed Architecture

We hereby include a detailed breakdown of the proposed DVHP architecture for completeness in 6.

C. Group Member Contributions

Gokhan Egri. For the project, I wrote the first implementations of the voxel-based visual hull method and the Deep Visual Hull Prior which we then developed and fine-tuned with Nicole. I was also responsible for the Blender script for generating the synthetic datasets. For the presentation and the report, we also had an even division of labour where I focused mostly on the DVHP-half of the write-up.

Xinran (Nicole) Han. For the project, I experimented with the Middlebury dataset as well as exploring different hyperparameters/loss functions for the Deep Visual Hull Prior. I wrote the voxel IoU script and adapted a C++ based mesh evaluation/voxelization tool to include scaling and re-centering for performance evaluations. I wrote a Blender script to generate photos from user-specified camera parameters. For the presentation and report, I focused mainly on the deterministic method.

D. Reconstruction Results for Deterministic Method

Here in 7, we provide a visualization of the experiment results for deterministic visual hull in **Section 4.3.1**.

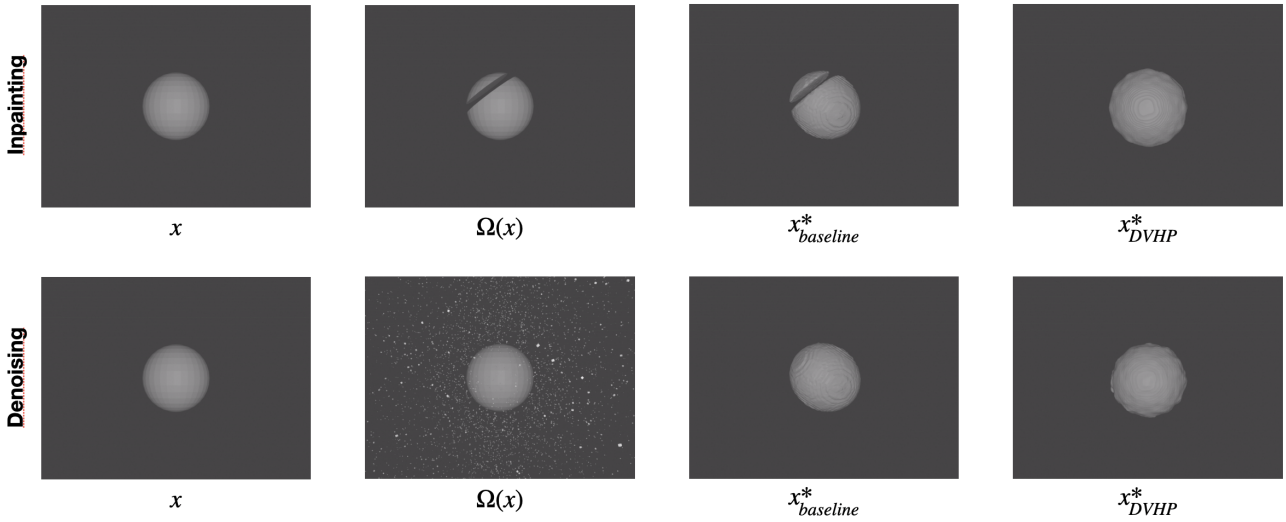

Figure 5. Qualitative results for the evaluation of the proposed DVHP model on 3D Inpainting (top) and 3D Denoising (bottom) tasks

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv3d-1    [-1, 16, 164, 164, 164]             448
         AvgPool3d-2       [-1, 16, 82, 82, 82]               0
       BatchNorm3d-3       [-1, 16, 82, 82, 82]              32
         LeakyReLU-4       [-1, 16, 82, 82, 82]               0
            Conv3d-5       [-1, 16, 80, 80, 80]           6,928
       BatchNorm3d-6       [-1, 16, 80, 80, 80]              32
         LeakyReLU-7       [-1, 16, 80, 80, 80]               0
   DownsampleBlock-8       [-1, 16, 80, 80, 80]               0
            Conv3d-9       [-1, 32, 78, 78, 78]          13,856
        AvgPool3d-10       [-1, 32, 39, 39, 39]               0
      BatchNorm3d-11       [-1, 32, 39, 39, 39]              64
        LeakyReLU-12       [-1, 32, 39, 39, 39]               0
           Conv3d-13       [-1, 32, 37, 37, 37]          27,680
      BatchNorm3d-14       [-1, 32, 37, 37, 37]              64
        LeakyReLU-15       [-1, 32, 37, 37, 37]               0
  DownsampleBlock-16       [-1, 32, 37, 37, 37]               0
      BatchNorm3d-17       [-1, 32, 37, 37, 37]              64
  ConvTranspose3d-18       [-1, 16, 39, 39, 39]          13,840
      BatchNorm3d-19       [-1, 16, 39, 39, 39]              32
        LeakyReLU-20       [-1, 16, 39, 39, 39]               0
  ConvTranspose3d-21       [-1, 16, 41, 41, 41]           6,928
      BatchNorm3d-22       [-1, 16, 41, 41, 41]              32
        LeakyReLU-23       [-1, 16, 41, 41, 41]               0
          Upsample-24       [-1, 16, 82, 82, 82]               0
    UpsampleBlock-25       [-1, 16, 82, 82, 82]               0
      BatchNorm3d-26       [-1, 16, 82, 82, 82]              32
  ConvTranspose3d-27        [-1, 1, 86, 86, 86]           2,001
      BatchNorm3d-28        [-1, 1, 86, 86, 86]               2
        LeakyReLU-29        [-1, 1, 86, 86, 86]               0
  ConvTranspose3d-30        [-1, 1, 90, 90, 90]             126
      BatchNorm3d-31        [-1, 1, 90, 90, 90]               2
    UpsampleBlock-32        [-1, 1, 90, 90, 90]               0
================================================================
Total params: 72,163
Trainable params: 72,163
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 17.45
Forward/backward pass size (MB): 1491.62
Params size (MB): 0.28
Estimated Total Size (MB): 1509.35
----------------------------------------------------------------
```

Figure 6. Deep Visual Hull Prior Detailed Architecture

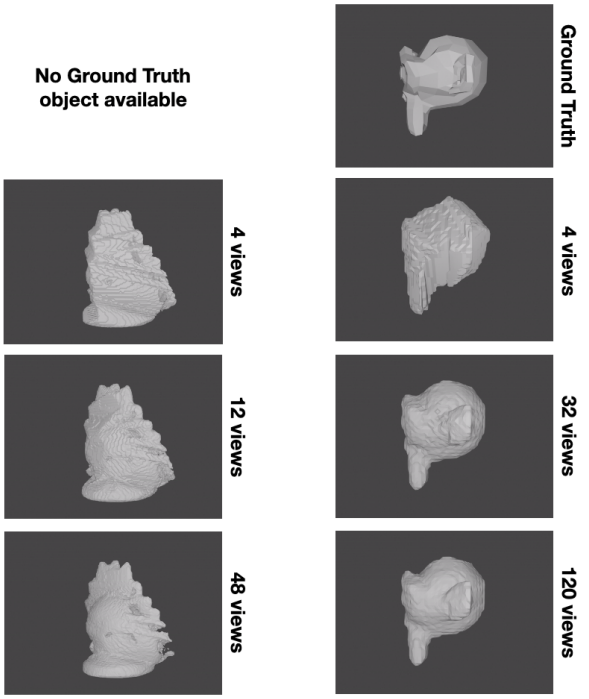

Figure 7. Voxel-based deterministic visuall hull reconstruction results of DinoRing model (left, using 48-view reconstruction as a relative benchmark) and Suzanne(Monkey) model (right).