Eurographics Symposium on Geometry Processing 2022
M. Campen and M. Spagnuolo
(Guest Editors)

Volume 41 (2022), Number 5

Supplementary Material for "MendNet: Restoration of Fractured Shapes Using Learned Occupancy Functions"

N. Lamb, S. Banerjee, and N. K. Banerjee

Clarkson University, USA

In this supplementary, we provide a derivation of the feasibility loss used in the paper in Section 1. We discuss our sampling method and implementation details in Section 2. We give an overview of the hardware we used and the runtimes for our approach and baseline approach using DeepSDF [PFS*19] in Section 3. Section 4 gives the number of hard plastic and complex ceramic fractures we generate for testing the generalizability of our approach. In Section 5 we give additional implementation details for 3D-ORGAN [HS18]. In Section 6 we show restoration shapes generated using the baseline approach of DeepSDF discussed in Section 7.5 of the main paper with Boolean subtraction in mesh space, and artifact cleanup using Euclidean distance thresholding. We show complete, fractured, and restoration meshes reconstructed by our approach and complete meshes reconstructed by the baseline approaches, corresponding to examples shown in the paper, in Section 7.

1. Expression for Feasibility Loss

In this section, we provide the derivation for the feasibility loss $\mathcal{J}_{\text{feas}}$ provided in Equation (8) in the main paper. As discussed in Section 3 of the paper, the equations

$$o_{\mathcal{S}}(\mathbf{x}) = f_{\mathbf{\Theta}}(\mathbf{z}_{\mathcal{S}}, \mathbf{x}), \tag{1}$$

$$\mathbf{z}_{\mathcal{R}} = \mathbf{g}_{\mathbf{\Phi}_{\mathcal{R}}}(\mathbf{z}_{\mathcal{F}}, \mathbf{x}), \text{ and} \tag{2}$$

$$\mathbf{z}_{\mathcal{C}} = \mathbf{g}_{\mathbf{\Phi}_{\mathcal{C}}}(\mathbf{z}_{\mathcal{F}}, \mathbf{x}) \tag{3}$$

provide the functional representation for occupancy values $o_{\mathcal{S}}$ for a shape $\mathcal{S} \in \{\mathcal{F}, \mathcal{R}, \mathcal{C}\}$, and the transformations for obtaining $\mathbf{z}_{\mathcal{R}}$ and $\mathbf{z}_{\mathcal{C}}$ from $\mathbf{z}_{\mathcal{F}}$, where $\mathcal{F}$, $\mathcal{R}$, and $\mathcal{C}$ are fractured, restoration, and complete shapes respectively. The functional form for $o_{\mathcal{F}}$ is expressed using Equation (1) as

$$o_{\mathcal{F}}(\mathbf{x}) = f_{\mathbf{\Theta}}(\mathbf{z}_{\mathcal{F}}, \mathbf{x}). \tag{4}$$

The forms for $o_{\mathcal{R}}$ and $o_{\mathcal{C}}$ can be expressed as

$$o_{\mathcal{R}}(\mathbf{x}) = f_{\mathbf{\Theta}}(\mathbf{z}_{\mathcal{R}}, \mathbf{x}), \text{ and} \tag{5}$$

$$o_{\mathcal{C}}(\mathbf{x}) = f_{\mathbf{\Theta}}(\mathbf{z}_{\mathcal{C}}, \mathbf{x}). \tag{6}$$

By substituting the expression for $\mathbf{z}_{\mathcal{R}}$ from Equation (2) into Equation (5), we obtain

$$o_{\mathcal{R}}(\mathbf{x}) = f_{\mathbf{\Theta}}(\mathbf{g}_{\mathbf{\Phi}_{\mathcal{R}}}(\mathbf{z}_{\mathcal{F}}, \mathbf{x}), \mathbf{x}). \tag{7}$$

Similarly, we re-write the expression for $o_{\mathcal{C}}$ by substituting Equation (3) into Equation (6), as

$$o_{\mathcal{C}}(\mathbf{x}) = f_{\mathbf{\Theta}}(\mathbf{g}_{\mathbf{\Phi}_{\mathcal{C}}}(\mathbf{z}_{\mathcal{F}}, \mathbf{x}), \mathbf{x}). \tag{8}$$

The following equation

$$h_{\text{feas}}(o_{\mathcal{F}}, o_{\mathcal{R}}, o_{\mathcal{C}}) = ((1 - o_{\mathcal{F}})(1 - o_{\mathcal{R}})(1 - o_{\mathcal{C}})) + (o_{\mathcal{F}}(1 - o_{\mathcal{R}})o_{\mathcal{C}}) + ((1 - o_{\mathcal{F}})o_{\mathcal{R}}o_{\mathcal{C}}). \tag{9}$$

provides the expression for the feasibility function h_{feas} in terms of the occupancy values for $\mathcal{F}$, $\mathcal{R}$, and $\mathcal{C}$ as discussed in Section 4 of the paper. The loss $\mathcal{J}_{\text{feas}}$ forces h_{feas} to 1 to encourage feasible occupancy values, as

$$\mathcal{J}_{\text{feas}}(\mathbf{z}_{\mathcal{F}}, \mathcal{W}) = \sum_{\mathbf{x} \in \mathcal{X}} l(h_{\text{feas}}(o_{\mathcal{F}}(\mathbf{x}), o_{\mathcal{R}}(\mathbf{x}), o_{\mathcal{C}})(\mathbf{x}), 1), \tag{10}$$

where $\mathcal{X}$ represents the point sampling, l represents the cross-entropy function, and $\mathcal{W}$ represents the weights of the networks. During optimization, the feasibility loss is aggregated over $\mathbf{z}_{\mathcal{F}}$ estimates for all training samples. We substitute Equations (4), (7), and (8) into Equation (10) to obtain

$$\mathcal{J}_{\text{feas}}(\mathbf{z}_{\mathcal{F}}, \mathcal{W}) = \sum_{\mathbf{x} \in \mathcal{X}} l(h_{\text{feas}}(f_{\mathbf{\Theta}}(\mathbf{z}_{\mathcal{F}}, \mathbf{x}), f_{\mathbf{\Theta}}(\mathbf{g}_{\mathbf{\Phi}_{\mathcal{R}}}(\mathbf{z}_{\mathcal{F}}, \mathbf{x}), \mathbf{x}), f_{\mathbf{\Theta}}(\mathbf{g}_{\mathbf{\Phi}_{\mathcal{C}}}(\mathbf{z}_{\mathcal{F}}, \mathbf{x}), \mathbf{x}), 1). \tag{11}$$

Equation (11) provides the expression for the feasibility loss as used in the paper.

2. Implementation Details

To obtain the set of sample points $\mathcal{X}$, discussed in Section 5 of the main paper, for a given shape $\mathcal{S}$ during training and testing, we sample n points where $\frac{4n}{5}$ points lie on or near the surface of the shape and $\frac{n}{5}$ are uniformly sampled around the shape. We ensure that of the n points, at least m points are from the interior of the shape and m points are from the exterior. During training, we combine the point sets for the fractured shape $\mathcal{F}$, the restoration shape $\mathcal{R}$, and the complete shape $\mathcal{C}$, to obtain $\mathcal{X}$, whereas during inference $\mathcal{X}$ is obtained only from $\mathcal{F}$. We set the value of n to be 5,461 for $\mathcal{F}$ and $\mathcal{R}$ and 5,462 for $\mathcal{C}$ during training, and 8,000 for $\mathcal{F}$ during inference. We set m to $\frac{n}{6}$ during training and $\frac{n}{2}$ during inference. To initialize gradient descent, we randomly set the value

of the latent fracture code, $\mathbf{z}_{\mathcal{F}}$, according to a Gaussian distribution centered at zero mean with standard deviation of 0.01 during training and 10^{-4} during inference. During training we set λ_{feas}, the scalar multiplier for the training feasibility loss term, to 1.0 and λ_{reg}, the scalar multiple for the training regularization loss term, to 10^{-4}. During inference we set λ_{feas}, the scalar multiplier for the inference feasibility loss term, to 1.0, λ_{reg}, the scalar multiple for the inference regularization loss term, to 10^{-4}, and λ_{nzr}, the scalar multiple for the non-zero loss term, to 0.5. We use the Adam optimizer to estimate network weights during training, and latent codes during training and inference, and perform optimization by running gradient descent for 2,000 epochs with a learning rate of 5×10^{-3}. We set the size of the latent codes to be $p = 256$ for jars, bottles, and mugs, and $p = 400$ for the remaining classes.

3. Hardware and Runtime

We train our approach on 40-core Intel Xeon servers with 2 NVIDIA RTX 3090s, with 3 NVIDIA RTX 3090s, and with 4 NVIDIA Volta v100s. Training our approach takes approximately 56 hours per 1000 training samples on a 40-core Intel Xeon server with 2 RTX 3090s. Training the baseline approach using DeepSDF takes approximately 32 hours per 1000 training samples on a 40-core Intel Xeon server with 2 NVIDIA RTX 3090s. Performing latent code inference using the baseline approach of DeepSDF takes approximately 21 seconds per sample, performing mesh reconstruction using an inferred code takes approximately 5.1 seconds per sample, and discarding disconnected components from a reconstructed mesh takes approximately 1.5 seconds per sample.

4. Shape Counts for Hard Plastic and Complex Ceramic Fractures

As discussed in Section 7 of the main paper, to evaluate the generalizability of our approach we generate synthetic fractures that simulate hard plastic fractures and complex ceramic fractures with concavities for testing. Our ShapeNet test set contains 278, 661, 1,106, 295, 220, 797, 612, and 1,123 shapes for the bottles, cars, chairs, jars, mugs, planes, sofas, and tables before fracturing. Of the shapes we retain 275, 653, 1,100, 294, 257, 791, 612, and 1,123 hard plastic fractured shapes after fracturing for the bottles, cars, chairs, jars, mugs, planes, sofas, and tables classes from ShapeNet. We retain 279, 664, 1,106, 297, 258, 798, 611, and 1,122 complex ceramic fractured shapes after fracturing for the bottles, cars, chairs, jars, mugs, planes, sofas, and tables classes from ShapeNet.

5. Implementation Details for 3D-ORGAN

We compare our approach to 3D-ORGAN [HS18], the only existing approach to generate complete shapes given fractured shapes as input, in Section 7.4 of the paper. To create fracture samples, we use the approach of Hermoza and Sipiran [HS18], which generates novel fractures at training time. We trained three variations of the generative adversarial network (GAN) architecture proposed by Hermoza and Sipiran including the GAN architecture with and without skip connections, and the architecture with skip connections and with Squeeze-and-Excite [HSS18] layers after each 3D convolutional layer. For each training class we trained an additional variation of the network using the architecture with skip connections and Squeeze-and-Excite layers where half of the fractured shapes for the training set were selected from the fractures generated using our fracturing approach, and half were generated randomly at runtime using the approach of Hermoza and Sipiran. We trained an additional variation where we combined the bottles, jars, and mugs classes and used the architecture with skip connections and Squeeze-and-Excite layers. For all variations we use a gradient penalty of $\lambda = 10$, and a completion loss coefficient of $k = 100$ as suggested by Hermoza and Sipiran. Each variation we tested performed comparably to the results shown in Section 7 of the paper.

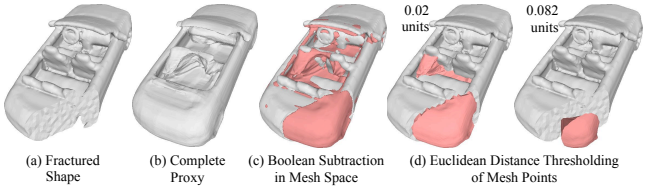

Figure 1: *Given a fractured shape, we show a complete proxy generated using the shape completion approach of DeepSDF [PFS*19] on an incomplete shape without the fracture, a restoration generated by Boolean subtraction in mesh space, and a restoration generated by removing vertices in the restoration mesh obtained by subtraction in SDF space that are close to the fractured shape. Note the surface artifacts generated on the restoration that protrude onto the fractured shape and the non-smooth join caused by thresholding the Euclidean distance.*

6. Restoration Generation using Boolean Subtraction and Artifact Cleanup using Euclidean Distance Thresholding

As discussed in the main paper, to compare our approach to a restoration generated by subtracting the fractured shape from a complete proxy, we implement subtraction in occupancy space rather than use traditional Boolean subtraction in mesh space due to its computational infeasibility. In Figure 1(c) we show an example restoration mesh generated by performing Boolean subtraction in mesh space of a complete proxy, shown in Figure 1(b) from an input fractured mesh, shown in Figure 1(a), where the complete proxy is generated from an incomplete input using the baseline approach with DeepSDF discussed in Section 7.5 of the main paper. Similar to the artifacts generated by occupancy-space subtraction, exact Boolean mesh-space subtraction also produces protrusion artifacts that are challenging to clean up.

In addition to the baseline artifact removal technique described in section Section 7.5 of the main paper, one may attempt to remove artifacts using an approximate subtraction in mesh space. In Figure 1(d) we show a restoration mesh generated by thresholding the Euclidean distance between points on the fractured mesh and the complete mesh generated using DeepSDF, retaining points with their corresponding mesh elements where the distance is higher than a threshold, and manually remeshing the resulting mesh components. As shown in Figure 1(d), a low threshold, e.g., 0.02 units for the car, does not eliminate the artifacts completely, as shown by

the remaining component in the middle of the car. The restoration mesh also does not join smoothly to the fractured mesh, as the gap between the exterior of the predicted restoration mesh and the fracture region caused by removing close vertices has to be remeshed. While a high threshold, e.g., 0.082 units, eliminates artifacts, it generates a restoration that is too far away from the fractured mesh and shows a large gap at the intersection of the restoration mesh and fractured mesh. As shown by the result in the top right of Figure 16 of the main paper, our approach automatically produces restorations devoid of artifacts that demonstrate a close join to the original shape.

7. Complete set of Inputs and Outputs for Results in Paper

We show the complete set of inputs and outputs for various results shown in the paper. In Figure 2, we show the ground truth (GT) complete, fractured, and restoration meshes and the complete, fractured, and restoration meshes reconstructed using our approach corresponding to Figure 1 in the paper. In Figure 3 we show the complete and restoration meshes reconstructed by our approach and by the shape completion approaches using PoinTr [YRW*21] with ConvONet [PNM*20] and using DeepSDF, and fractured meshes reconstructed by our approach. Meshes shown correspond to the meshes in Figure 16 of the main paper. In Figure 4 we show complete, fractured, and restoration meshes reconstructed using various loss combinations, corresponding to Figure 12 of the main paper. From left to right we show meshes reconstructed using $\mathcal{J}_{\text{data}}$, where we use only the data loss during inference, $\mathcal{J}_{\text{data}}, \mathcal{J}_{\text{nzr}}$, where we use weighted combination of the data loss and the non-zero loss, $\mathcal{J}_{\text{data}}, \mathcal{J}_{\text{feas}}$, where we use a weighted combination of the data loss and the feasibility loss, and $\mathcal{J}_{\text{data}}, \mathcal{J}_{\text{nzr}}, \mathcal{J}_{\text{feas}}$, where we use a weighted combination of the data loss, the non-zero loss, and the feasibility loss. In Figure 5 we show the GT complete, fractured, and restoration meshes and the complete, fractured, and restoration meshes reconstructed using our approach corresponding to Figure 8 in the paper.

References

[HS18] HERMOZA, RENATO and SIPIRAN, IVAN. "3D reconstruction of incomplete archaeological objects using a generative adversarial network". *Proceedings of Computer Graphics International*. New York, NY: ACM, 2018, 5–11 1, 2.

[HSS18] HU, JIE, SHEN, LI, and SUN, GANG. "Squeeze-and-excitation networks". *Proc. CVPR*. Piscataway, NJ: IEEE, 2018, 7132–7141 2.

[PFS*19] PARK, JEONG JOON, FLORENCE, PETER, STRAUB, JULIAN, et al. "Deepsdf: Learning continuous signed distance functions for shape representation". *Proc. CVPR*. Piscataway, NJ: IEEE, 2019, 165–174 1, 2.

[PNM*20] PENG, SONGYOU, NIEMEYER, MICHAEL, MESCHEDER, LARS, et al. "Convolutional occupancy networks". *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Berlin, Germany: Springer, 2020, 523–540 3.

[YRW*21] YU, XUMIN, RAO, YONGMING, WANG, ZIYI, et al. "Pointr: Diverse point cloud completion with geometry-aware transformers". *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, 12498–12507 3.

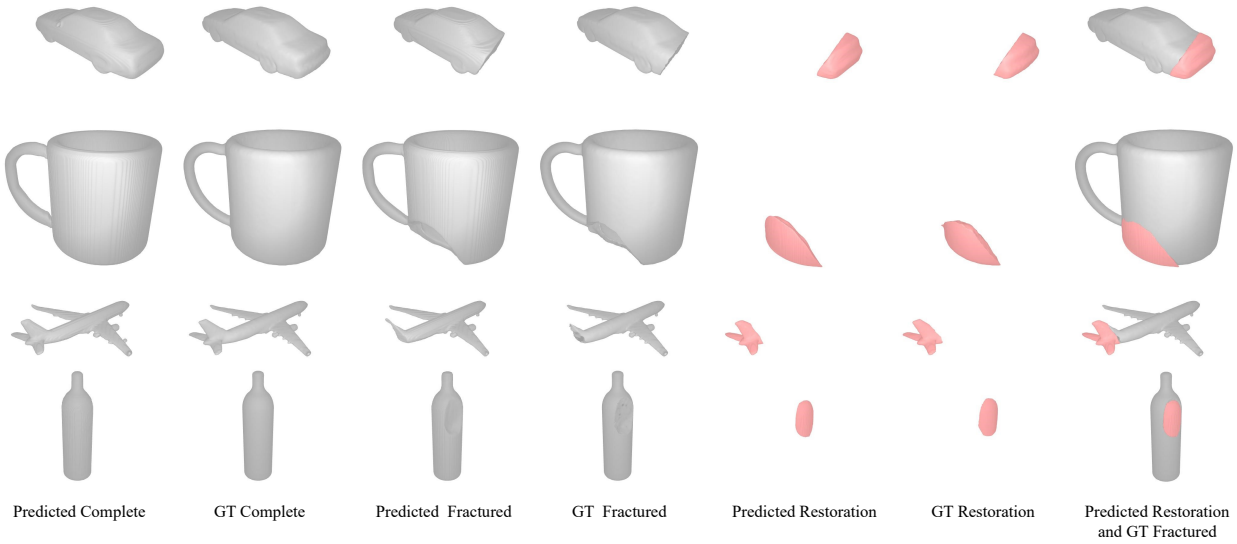

Figure 2: *Predicted complete, fractured, and restoration meshes using our approach, shown with their corresponding ground truth (GT) meshes, and the predicted restoration mesh from our approach shown joined with the ground truth fractured mesh. Meshes shown correspond to Figure 1 in the paper. Complete and fractured meshes are shown in gray, restoration meshes are shown in red.*

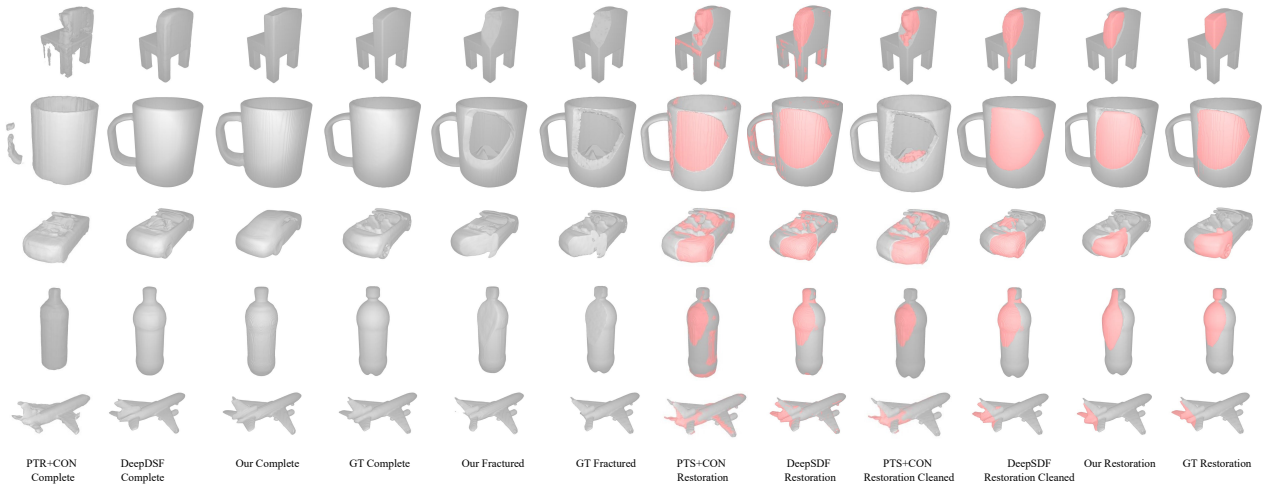

Figure 3: *Predicted and ground truth complete (GT) meshes shown for the baseline subtraction approaches using PoinTr with ConvONet (PTR+CON), using DeepSDF, and using our approach for the complete, fractured and restoration shapes. Meshes shown correspond to Figure 16 of the paper. Complete and fractured meshes are shown in gray, restoration meshes are shown in red. Restoration meshes are shown joined to the GT fractured shape.*

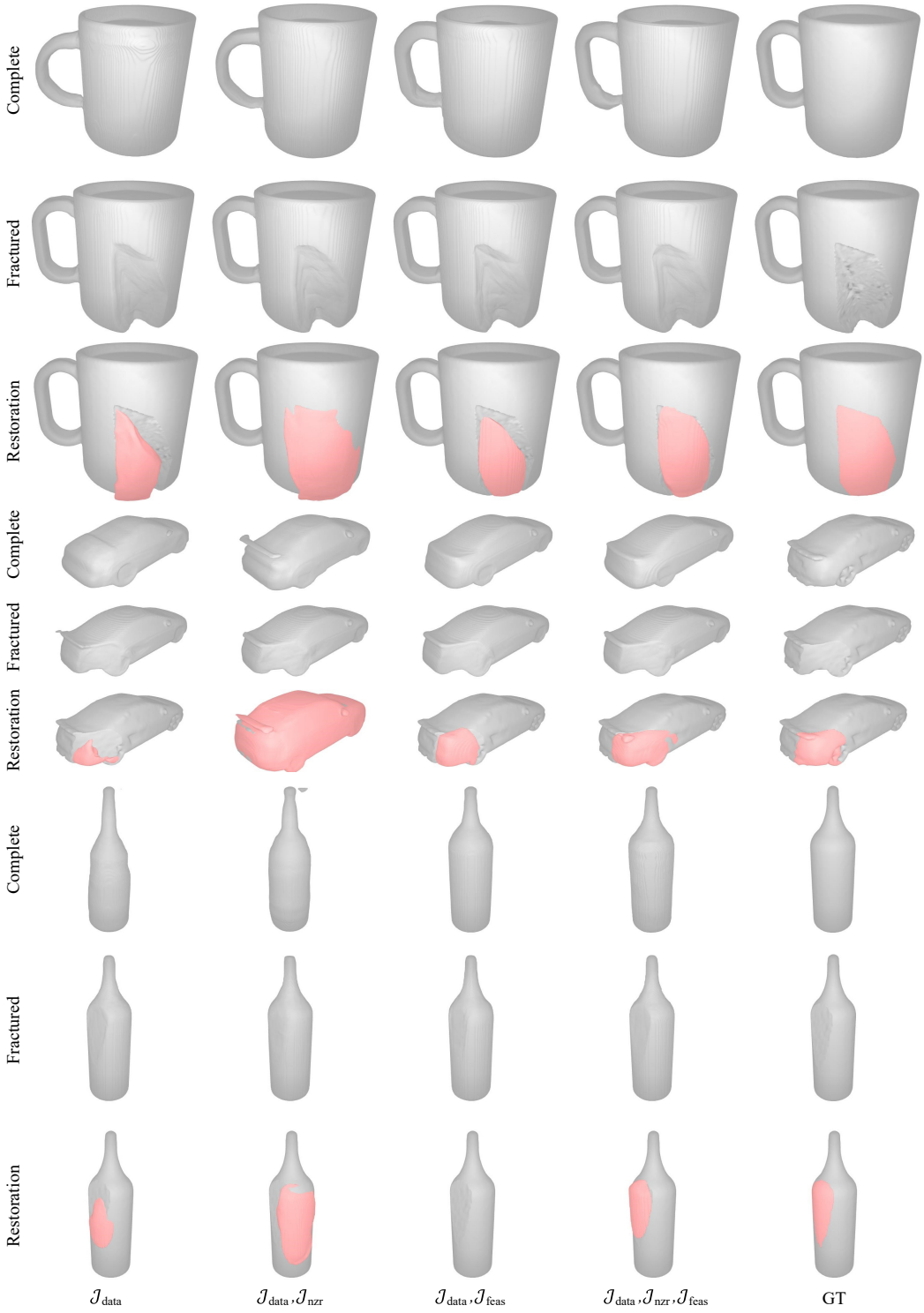

Figure 4: *Predicted and ground truth (GT) complete, fractured, and restoration meshes using our approach with just data loss ($\mathcal{J}_{data}$), using data loss and non-zero loss ($\mathcal{J}_{data}, \mathcal{J}_{nzr}$), using data loss and feasibility loss ($\mathcal{J}_{data}, \mathcal{J}_{feas}$), and using data loss, non-zero loss, and feasibility loss ($\mathcal{J}_{data}, \mathcal{J}_{nzr}, \mathcal{J}_{feas}$) for a mug, a car and a bottle. Meshes shown correspond to Figure 12 of the paper. Complete and fractured meshes are shown in gray, restoration meshes are shown in red. Restoration meshes are shown joined to the GT fractured shape.*

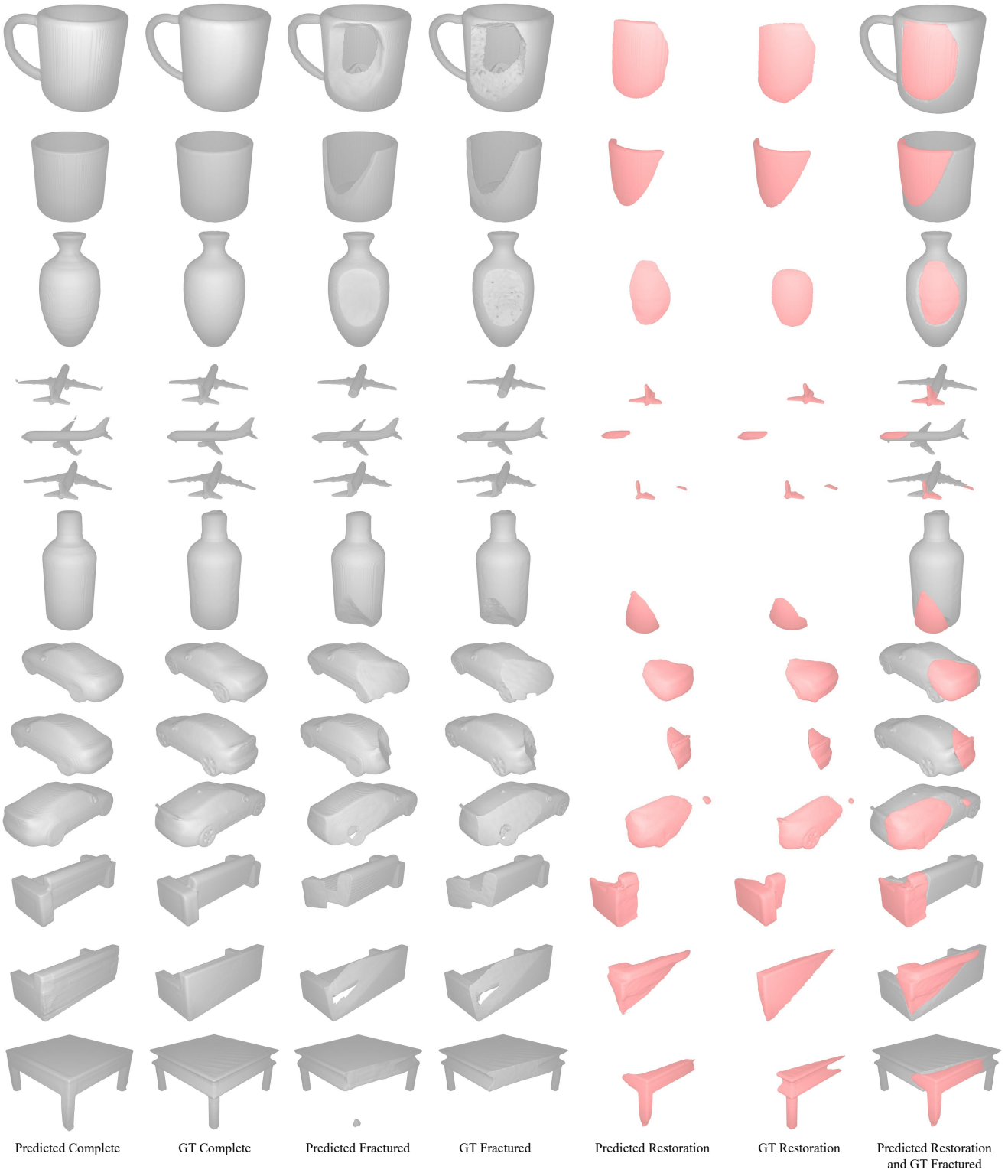

Figure 5: *Predicted complete, fractured, and restoration meshes using our approach, shown with their corresponding ground truth (GT) meshes, and predicted restoration meshes from our approach shown joined with the GT fractured mesh, corresponding to meshes shown in Figure 8 in the paper. Complete and fractured meshes are shown in gray, restoration meshes are shown in red. Restoration meshes are shown joined to the GT fractured shape.*