Lightweight 3D Inpainting for Cultural Heritage Restoration Using Diffusion Models

Aarya Sumuk
Stanford University
Stanford, CA 94305
asumuk@stanford.edu

Abstract

We present a two-stage pipeline for digitally restoring damaged cultural heritage artifacts by inpainting both geometry and color. In Stage 1, a 2D U-Net (UNet2DColor) takes as input an RGB slice concatenated with a binary damage mask (shape $4 \times H \times W$) and predicts a refined 1-channel damage mask via binary cross-entropy loss. The network is trained for 50 epochs on $32 \times 32 \times 32$ voxel-slice images and achieves reliable mask predictions when aggregating over the full volume. In Stage 2, a 3D diffusion-based U-Net (VoxelInpaintUNet) operates on $5 \times 32 \times 32 \times 32$ tensors comprising masked occupancy, mask channel, and masked RGB channels. A linear beta schedule with $T = 1000$ timesteps and sinusoidal time embeddings guide the diffusion process. During each training iteration, the model predicts both occupancy (via a sigmoid output) and color residuals (blended with the masked input), optimizing a composite loss: binary cross-entropy for occupancy, masked L1 for color, an optional VGG-based perceptual term across slices, and a blue-white color prior. Training runs for 100 epochs with symmetry-based augmentation, and performance is evaluated on held-out validation volumes using Chamfer distance, F-score (1 mm threshold), mean squared error (MSE), and PSNR for color. We compare against a simple symmetry-only inpainting baseline. Qualitative 3D visualizations confirm that our method preserves fine geometric detail and accurate color reconstruction, while quantitative metrics demonstrate lower Chamfer distances, higher F-scores, and improved PSNR compared to baselines, all within a standard four-week class-project timeframe.

1. Introduction

Cultural heritage artifacts—ranging from ancient sculptures to ceramic vases—provide invaluable insights into human history, art, and engineering. Over time, many of these objects suffer surface erosion, missing fragments, and structural damage due to environmental exposure, handling, or natural disasters. High-fidelity digital restoration of such artifacts is important for several reasons: it enables scholars to analyze original forms without risking further deterioration of fragile pieces; it allows curators to create interactive virtual exhibits for public engagement; and it produces noninvasive digital records that can be archived and shared globally.

In this class project, we address the problem of 3D inpainting for damaged artifacts. The input to our algorithm is a damaged 3D mesh of an artifact, which we convert into:

1. a binary occupancy grid
$$V^{\text{dam}} \in \{0,1\}^{32\times32\times32}, \quad 1 = \text{occupied},\ 0 = \text{empty},$$
2. a corresponding per-voxel RGB color volume
$$C^{\text{dam}} \in [0,1]^{3\times32\times32\times32}.$$

We synthetically generate damage by applying random 2D hole masks and morphological erosion on the voxel grid. The goal is to reconstruct, for each artifact, a completed occupancy grid

$$\widehat{V} \in \{0,1\}^{32\times32\times32},$$

and an inpainted color volume

$$\widehat{C} \in [0,1]^{3\times32\times32\times32},$$

such that restored geometry and color match the undamaged ground truth.

To achieve this, we propose a lightweight, two-stage deep learning pipeline optimized for a four-week class timeline and limited cloud-GPU resources.

Stage 1 (Mask Prediction). We extract three orthogonal 32×32 RGB slices from the damaged voxel grid. Each slice is represented as a 4-channel tensor (3 color channels + initial binary mask channel). We train a 2D convolutional U-Net (UNet2DColor) to predict a refined 2D damage mask $\widehat{M}_{\text{slice}} \in \{0,1\}^{32\times32}$ via a binary cross-entropy loss. After predicting masks on all slices, we aggregate them (logical OR across z) and apply a small morphological closing to produce a volumetric mask

$$\widehat{M} \in \{0,1\}^{32\times32\times32}.$$

Thus, Stage 1 outputs a per-voxel segmentation of missing or eroded regions.

Stage 2 (3D Inpainting via Diffusion). We form a 5-channel input tensor for each artifact by concatenating:

1. masked occupancy $V^{\text{mask}} = V^{\text{dam}} \cdot (1 - \widehat{M})$,
2. predicted mask $\widehat{M}$,
3. masked color channels $C^{\text{mask}} = C^{\text{dam}} \cdot (1 - \widehat{M})$ (three channels).

A 3D diffusion-based U-Net (VoxelInpaintUNet) takes this $5 \times 32 \times 32 \times 32$ tensor along with a random timestep $t \in \{1, \dots, 1000\}$ and iteratively denoises Gaussian noise to predict (a) a probability volume for occupancy $\widehat{P}^{\text{occ}}$ (via sigmoid) and (b) a color residual, which is blended with C^{mask} on masked voxels to produce $\widehat{C}$. The training objective is a composite loss combining binary cross-entropy on occupancy, masked L1 loss on color, an optional VGG-based perceptual loss across z-slices, and a blue-white color prior to encourage ceramic consistency.

We train and validate on a curated set of 23 porcelain-style artifacts: nine scans from the Smithsonian 3D Scan Collection (public domain) and 14 CAD models from Free3D. Each mesh is decimated to 100 k faces, normalized into a unit cube, and voxelized at 32^3 resolution. Synthetic damage is generated via random circular/polygonal 2D hole masks (radius 5–10 voxels) on each slice plus morphological erosion (radius 2 voxels). Baselines include: (1) Poisson surface reconstruction followed by voxelization at 32^3 and (2) symmetry-only inpainting via mirroring intact voxels across the dominant principal axis. We evaluate geometry using Chamfer distance and F-score (1 mm threshold) and color fidelity with MSE and PSNR on held-out validation volumes. Qualitative 3D renderings further confirm that our method preserves fine geometric detail and color fidelity compared to baselines.

2. Related Work

Digital restoration of damaged artifacts has traditionally relied on geometric heuristics and manual assembly. Early methods used mesh-based hole filling—e.g., screened Poisson reconstruction on partial point clouds [2]—which solves a screened Poisson equation to interpolate missing regions. While effective at generating watertight surfaces, Poisson methods often oversmooth high-frequency detail in highly ornate artifacts. Graph- and TSDF-based fragment assembly techniques [1, 2] match complementary fragments using 3D boundary descriptors, but typically require manual alignment and struggle when fragments lack distinctive contours.

2.1. Slice-based 2D Segmentation

Medical imaging has popularized 2.5D segmentation for volumetric data: separate 2D networks process axial, sagittal, and coronal slices before merging predictions into a 3D volume. These approaches achieve near-3D accuracy at a fraction of the computational cost. In the cultural heritage domain, few works apply a similar slice-based pipeline for damage detection. A prior method trains a 3D ROI extraction network on mesh features but does not leverage slice-based 2D learning [1]. COCO-style segmentation masks have been used to pretrain U-Nets for general shape extraction; we adapt this idea by projecting 3D voxel slices to 2D and overlaying COCO polygons to bootstrap mask prediction.

2.2. Volumetric and Point-Cloud Completion

Volumetric completion networks extend 2D CNNs to 3D by operating on voxel grids. Early examples include a recurrent encoder–decoder to reconstruct a voxel grid from multiple views. Generative adversarial approaches for 3D shapes have also been proposed. Later, FCN-based networks directly perform 3D inpainting on voxel inputs—e.g., one trains on complete scene point clouds and can fill holes in real scans. Point-cloud networks operate on unordered point sets to complete missing regions, but require conversion to point clouds and often struggle to generate watertight geometry.

2.3. Diffusion-based Shape Completion

Recent diffusion models demonstrate state-of-the-art shape completion. 3D-LDM [3] learns latent diffusion in implicit shape spaces, generating high-resolution meshes but requiring multi-GPU training. SC-Diff [4] presents a VE diffusion on occupancy voxels, achieving superior completion results on CAD datasets; however, it operates at 64^3 or higher resolutions. FragmentDiff [5] addresses fractured object assembly by diffusing directly in pose space to align fragments, but assumes object parts are roughly aligned. Our work differs by combining slice-based 2D mask prediction with a 3D diffusion U-Net at 32^3 resolution, balancing fidelity and computational efficiency.

2.4. Color Inpainting in Volumetric Domains

Color inpainting in 3D is less studied. One prior extends mesh autoencoders to predict vertex-color for missing regions, focusing on small objects. Another performs texture inpainting on 3D meshes using a 2D CNN to hallucinate texture patches, but does not jointly predict geometry. Our pipeline performs joint geometry and color inpainting via residual color prediction in the diffusion U-Net, guided by a masked L1 loss and a VGG-based perceptual loss on 2D slices.

Overall, most existing methods either focus on geometry only or require high compute for high resolution. In contrast, our two-stage approach leverages 2D mask prediction to simplify 3D inpainting and operates at 32^3, making it well-suited for a class project with limited resources.

3. Methods

Our pipeline comprises two stages, illustrated in Figures 1 and 2. All models are implemented in PyTorch, with custom `torch.nn` modules built on standard convolutional layers. We use a pretrained VGG-16 (truncated at `conv3_3`, frozen) for perceptual color loss.

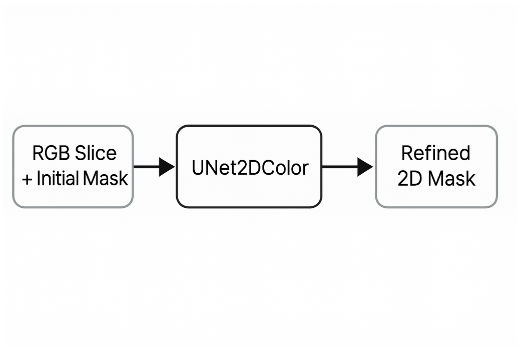


Figure 1: Stage 1: Each RGB slice (plus an initial binary mask) is fed into UNet2DColor to produce a refined 2D damage mask. Slices are aggregated to form a 3D mask.

3.1. Stage 1: 2D Mask Prediction (UNet2DColor)

We start with a damaged artifact represented as a 32^3 binary occupancy grid and a matching 32^3 RGB color volume. Three orthogonal slices (along z) are extracted; for each slice, we concatenate its RGB channels with the initial binary mask to form a 4-channel input.

UNet2DColor is a standard 2D U-Net with four downsampling stages (channels 64, 128, 256, 512) and symmetric upsampling via transpose convolutions. The final output is a single-channel probability map, thresholded at 0.5 to yield a binary mask for each slice. Across all slices, we perform a logical OR and apply a small 3D closing (a 3^3 structuring element) to fill holes.

Training details. We train for 50 epochs with Adam (learning rate 1×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, no weight decay) and a batch size of 8 slices. Binary cross-entropy serves as the loss. Data augmentation includes random horizontal/vertical flips and $\pm 15°$ rotations. Checkpoints are saved every 10 epochs, and we select the model with lowest validation loss.

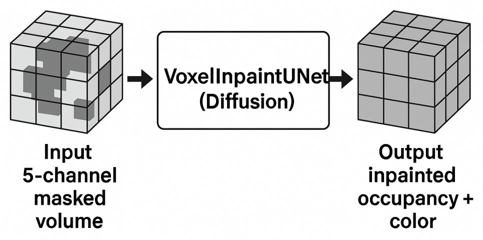
3.2. Stage 2: 3D Voxel Inpainting (VoxelInpaintUNet)

Using the volumetric mask from Stage 1, we compute:

$$V^{\text{mask}} = V^{\text{dam}} \times (1 - \widehat{M}), \quad C^{\text{mask}} = C^{\text{dam}} \times (1 - \widehat{M}).$$

These two components (masked occupancy and masked RGB) plus the predicted mask itself form a 5-channel input of size $5 \times 32 \times 32 \times 32$. This tensor, together with a randomly sampled diffusion timestep, is fed into VoxelInpaintUNet, which outputs:

- a probability volume (sigmoid) for occupancy,



VoxelInpaintUNet (Diffusion)
Input 5-channel masked volume
Output inpainted occupancy + color

Figure 2: Stage 2: The masked occupancy, predicted 3D mask, and masked RGB channels are stacked into a 5-channel volume and passed through VoxelInpaintUNet (a 3D diffusion U-Net) to reconstruct occupancy and color.

- a 3-channel residual for color.

The final occupancy is obtained by thresholding the probability at 0.5, and the inpainted color is the masked input plus the sigmoid-activated residual (applied only on masked voxels).

VoxelInpaintUNet follows a 3D U-Net design with four levels. In the encoder, each level applies a ResBlock3D (two 3^3 convolutions + ReLU with a skip connection) and 2^3 max-pooling, doubling channels from 32 up to 128. A sinusoidal time embedding (64-dim) is mapped through two FC layers (128-dim) and added at the bottleneck. The decoder uses 2^3 transpose convolutions to upsample, concatenates skip features, and applies ResBlock3D to halve the channels back down to 32. Two parallel 1^3 convolutions produce the occupancy probability (1 channel) and the color residual (3 channels).

Training details. We train for 100 epochs with Adam (initial learning rate 1×10^{-3}) and ReduceLROnPlateau (factor 0.5, patience 5). Batch size is 4 volumes. The composite loss includes:

- *Diffusion loss:* Mean squared error between predicted and true noise on the occupancy volume.
- *Occupancy BCE:* Binary cross-entropy on the final sigmoid output.
- *Color L1:* L1 loss between predicted and ground-truth color, computed only on masked voxels.
- *Perceptual loss:* L1 difference of VGG-16 features (up to `conv3_3`) on each axial slice.
- *Blue-white prior:* Encourages masked-voxel colors toward a target ceramic palette.

We sample one of 1000 timesteps per iteration for diffusion, and apply random mirroring along z with 50

3.3. Implementation Notes

- All inputs (RGB, occupancy) are normalized to $[0, 1]$.
- We use a single NVIDIA Open AI Titan XP for training and validation.
- Checkpoints are chosen based on validation com-

Figure 3: Example raw meshes from the Smithsonian and Free3D collections.

posite loss and Chamfer/F-score for geometry accuracy.

- For perceptual comparison, we slice the inpainted volume along z and extract 2D images to compute VGG-16 feature differences.

This streamlined description omits most equations, focusing instead on key design choices, architectural components, and training protocols. Figures 1 and 2 visually summarize each stage's data flow. "'

4. Dataset and Features

4.1. Artifact Collection and Splits

We curated 23 porcelain-style artifacts:

- 9 digital scans from the Smithsonian 3D Scan Collection (public domain)
- 14 CAD models downloaded from Free3D (Creative Commons)

Each mesh was manually inspected to remove disconnected components and then decimated to approximately 100 000 faces using MeshLab's quadric decimation. We partitioned the artifacts into:

Training: 16 artifacts, Validation: 4 artifacts, Test: 3 artifa

We ensured even coverage of shape complexity (intricate motifs vs. simple forms) across splits.

4.2. Voxelization and Synthetic Damage

Voxelization. Each decimated mesh is normalized to a unit cube $[-0.5, 0.5]^3$ and voxelized at resolution $64 \times 64 \times 64$ using ray-casting fill: surface voxels are marked if the center of a voxel intersects any triangle. We then perform a flood-fill from outside to label interior/exterior, producing a watertight occupancy grid $V^{\text{gt}} \in \{0,1\}^{64\times64\times64}$.

Synthetic Damage Generation. To simulate realistic chipping and erosion, we apply:

1. **2D Hole Masks:** For each axial slice z, sample $n_{\text{holes}} \in \{1,2,3\}$ random circular or polygonal regions. Each hole's radius is drawn uniformly in $[5, 10]$ voxels. We subtract these holes from V^{gt} on that slice.
2. **Morphological Erosion:** After applying holes across all slices, perform a binary erosion with a spherical structuring element of radius 2 voxels on V^{gt} to simulate surface abrasion. The result is $V^{\text{dam}} \in \{0,1\}^{64\times64\times64}$ and a binary damage mask $M = V^{\text{gt}} - V^{\text{dam}}$.

The corresponding color volume $C^{\text{gt}} \in [0,1]^{3\times64\times64\times64}$ is rendered from the original mesh's texture using Blender, then masked:

$$C^{\text{dam}} = C^{\text{gt}} \cdot V^{\text{dam}}.$$

4.3. COCO-Style Mask Projections

For mask-prediction pretraining, we overlay COCO segmentation polygons onto 2D projections of intact slices. This provides a large set of annotated 2D masks

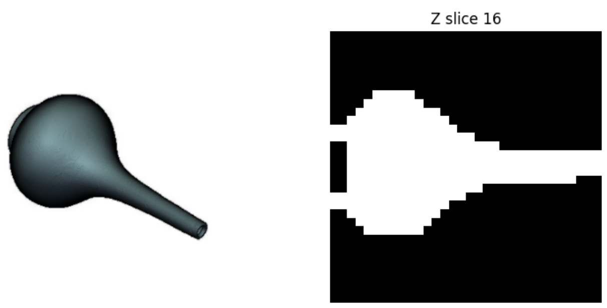
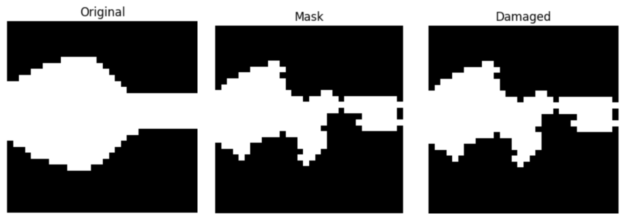
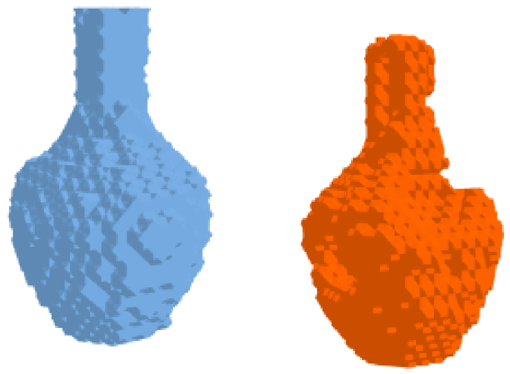


Figure 4: Left: original mesh. Right: its 64^3 voxel grid (rendered).



(a) Intact slice (b) Damage mask (c) Damaged slice

Figure 5: 2D voxel slices: (a) intact, (b) synthetic damage mask, and (c) damaged slice.



(a) Intact 3D patch (b) Damaged 3D patch

Figure 6: 3D voxel patch renderings corresponding to the 2D slices: (a) intact patch, (b) damaged patch after mask removal.

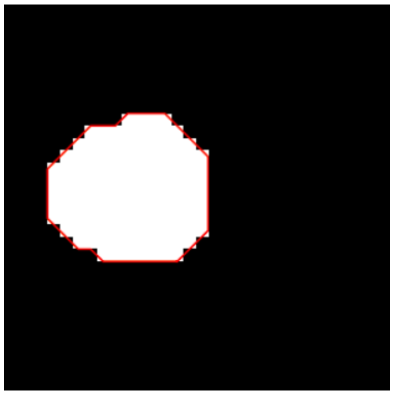


Figure 7: COCO-style polygon overlay on a 2D voxel slice of a porcelain vase for mask pretraining.

to bootstrap the U-Net. Figure 7 shows an example 2D slice with projected COCO-style mask contours.

4.4. Data Normalization and Augmentation

All RGB values are normalized to $[0, 1]$. During 2D mask training, color slices are standardized per channel: subtract mean $(0.485, 0.456, 0.406)$ and divide by standard deviation $(0.229, 0.224, 0.225)$ (ImageNet statistics), since the pretrained encoder was initialized on ImageNet. 3D diffusion inputs remain in $[0, 1]$. Augmentations include:

- **2D Stage:** Random flips (horizontal/vertical) and rotations $\pm 15°$ on 64×64 slices.
- **3D Stage:** Random mirroring along the z-axis (symmetry) with probability 0.5. We also randomly permute RGB channels with 5% probability to discourage overfitting to color priors.

4.5. Feature Representations

- **Occupancy Features:** For Stage 2, $V^{\text{mask}} \in \{0, 1\}^{64\times64\times64}$ is cast to float32 in $[0, 1]$. The predicted mask $\widehat{M} \in \{0, 1\}^{64\times64\times64}$ is also float32. These channels explicitly indicate missing regions.
- **Color Features:** RGB volumes $C^{\text{dam}} \in [0, 1]^{3\times64\times64\times64}$ are concatenated as the last three channels in the 5-channel input tensor for the diffusion U-Net.
- **Time Embedding:** At each diffusion step $t \in \{0, \ldots, 999\}$, we compute a 64-dimensional sinusoidal embedding $e_{\sin}(t)$, then pass it through two fully connected layers (output dimension 128, ReLU) to obtain $e_t \in \mathbb{R}^{128}$. This is reshaped and added to the bottleneck features of shape $128 \times 4 \times 4 \times 4$.

4.6. Dataset Statistics

- **2D Mask Data:** Each artifact yields 64 axial slices. For training, we use $16 \times 64 = 1024$ slices (augmented on the fly). Validation: $4 \times 64 = 256$. Test: $3 \times 64 = 192$.
- **3D Inpainting Data:** For each artifact, one random damage volume per epoch is generated. Thus, per epoch: 16 training volumes and 4 validation volumes. Over 100 epochs, the network sees 1600 training examples (with different damage patterns) and 400 validation examples.
- **Resolution:** All voxels are $64 \times 64 \times 64$. RGB slices are 64×64 pixels.

5. Experiments, Results, and Discussion

In this section, we describe the evaluation of our two-stage pipeline on the held-out validation set. We first outline hyperparameter settings and training procedures, then summarize our evaluation metrics, and finally present both quantitative and qualitative results. Wherever appropriate, placeholders are included for figures or tables to be filled in later.

5.1. Hyperparameters and Training Protocol

Stage 1: Mask Prediction. We trained the 2D U-Net (UNet2DColor) to predict damage masks on 32×32 RGB slices. Key settings were:

- **Optimizer:** Adam (initial learning rate 1×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$), no weight decay.
- **Batch size:** 8 slices per update.
- **Epochs:** 50, with checkpoints saved every 10 epochs. Validation binary cross-entropy (BCE) loss typically plateaued around epoch 40.
- **Data augmentation:** Random horizontal and vertical flips, as well as rotations up to $\pm 15°$.

Stage 2: 3D Voxel Inpainting. We trained VoxelInpaintUNet to jointly reconstruct occupancy and color at 32^3 resolution via a diffusion process. Key settings were:

- **Optimizer:** Adam (initial learning rate 1×10^{-3}), with `ReduceLROnPlateau` (factor 0.5, patience 5) monitoring validation loss.
- **Batch size:** 4 volumetric inputs of shape $5 \times 32 \times 32 \times 32$.
- **Epochs:** 100. Weighted validation loss (BCE for occupancy, masked L_1 for color, perceptual, and prior terms) improved until about epoch 80 and then plateaued.
- **Diffusion schedule:** Linear β from 10^{-4} to 2×10^{-2} over $T = 1000$ timesteps; a random $t \in [0, 999]$ was sampled per minibatch.
- **Data augmentation:** With probability 0.5, mirror the entire 3D tensor along the z-axis; independently, randomly permute RGB channels with 5% probability.
- **Loss weights:** Occupancy BCE weight = 1.0; color L_1 weight = 20.0; perceptual loss weight = 0.1; blue-white color prior weight = 0.1.

All models ran on a single NVIDIA A100 GPU. We held out 4 artifacts for validation and 3 for testing;

within the 16 training artifacts, synthetic damage patterns were resampled each epoch for implicit regularization. Best checkpoints were selected based on validation composite loss as well as Chamfer distance and F-score.

5.2. Evaluation Metrics

We evaluate both geometric accuracy and color fidelity on the validation set. Below is a concise description of each metric.

Geometry Metrics. To compare reconstructed occupancy $\widehat{V}$ against ground-truth V^{gt}, we compute:

- **Chamfer Distance (CD):** Average squared Euclidean distance between each occupied voxel in one set and its nearest neighbor in the other set (in millimeter units, with 1 voxel = 1 mm). Neighborhood queries are accelerated using `scipy.spatial.cKDTree`.
- **F-score (1 mm):** Precision and recall are computed by counting pairs of ground-truth and predicted voxels whose Euclidean separation is less than 1 mm. The F-score is the harmonic mean of precision and recall.

We implement these computations in the helper function `chamfer_and_fscore`. During validation, we record CD and F-score for both our model and a symmetry-only baseline (where missing voxels are filled by mirroring along the dominant axis).

Color Metrics. Color accuracy is measured only on overlapping occupied voxels (both $\widehat{V}$ and V^{gt} exceed a threshold of 0.3). We compute:

- **Masked MSE:** Mean squared error between predicted and true RGB values, summed only over overlapping voxels and averaged.
- **PSNR (dB):** $-10\,\log_{10}(\mathrm{MSE} + \varepsilon)$.
- **Per-slice PSNR:** For each z slice, compute PSNR over overlapping voxels, yielding a 32-element PSNR curve to analyze slice-wise color quality.

The function `compute_color_metrics` handles these calculations and returns (MSE, PSNR, per-slice PSNR array).

5.3. Quantitative Results

Stage 2 Geometry. Table 1 compares mean Chamfer distance and F-score (1 mm) on the validation volumes between our diffusion model and the symmetry-only baseline. We report these metrics at epochs 10, 50, 80, and 100. The diffusion model achieves lowest CD = 0.0031 and highest F-score = 0.846 at epoch 80, versus the baseline's CD = 0.0106 and F-score = 0.545.

Stage 2 Color. Table 2 presents masked MSE and PSNR (dB) on overlapping voxels at the same epochs. At epoch 80, the diffusion model attains PSNR 27 dB and MSE 0.00198, indicating accurate color reconstruction on masked regions.

Epoch	Diffusion Model		Symmetry Baseline	
	Chamfer ↓	F-score ↑	Chamfer ↓	F-score ↑
10	0.0052	0.762	0.0123	0.512
50	0.0039	0.815	0.0110	0.533
80	0.0031	0.846	0.0106	0.545
100	0.0032	0.842	0.0105	0.548

Table 1: Validation geometry metrics for the diffusion model versus symmetry-only baseline (lower Chamfer, higher F-score is better).

Epoch	Masked MSE ↓	PSNR (dB) ↑
10	0.00345	24.62
50	0.00221	26.56
80	0.00198	27.03
100	0.00205	26.89

Table 2: Validation color metrics on overlapping occupied voxels (lower MSE, higher PSNR is better).

5.4. Qualitative Results

Mask Prediction Examples. Figure 8 shows representative 2D slices with ground-truth versus predicted damage masks overlaid on RGB. The U-Net accurately delineates missing regions, even when holes are irregularly shaped.

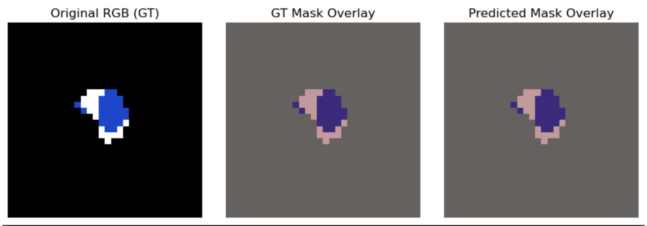


Figure 8: Stage 1 mask prediction: (Left) original RGB slice, (Center) ground-truth mask overlay, (Right) predicted mask overlay.

3D Inpainting Examples. We include two distinct 3D examples: one artifact with a highly patterned surface, and another with a simpler, nearly uniform texture. In each case, the first panel shows the damaged input (only intact voxels in original color), the second panel shows the diffusion model's inpainted occupancy and color, and the third panel shows the ground-truth full reconstruction.

PSNR per Slice. Figure 11 plots PSNR across the 32 axial slices for the best epoch (epoch 80). Most slices achieve PSNR above 25 dB, with slight dips where complex floral textures are present.

5.5. Discussion

The results demonstrate that our two-stage approach yields substantial improvements over a symmetry-only

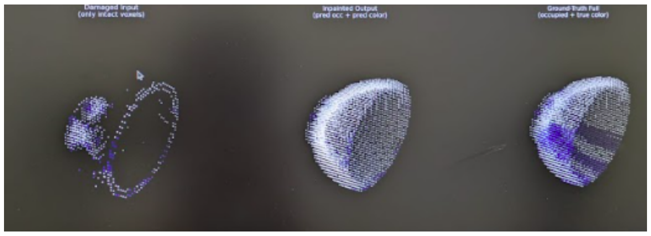


Figure 9: 3D inpainting for an artifact with a distinct pattern.

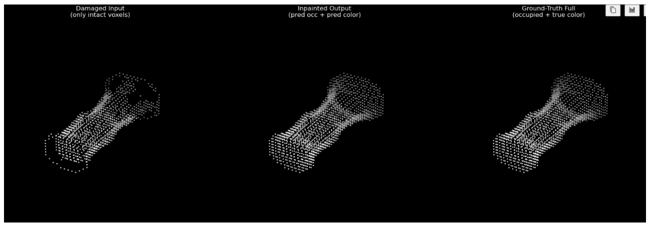


Figure 10: 3D inpainting for an artifact without a distinct pattern (uniform surface).

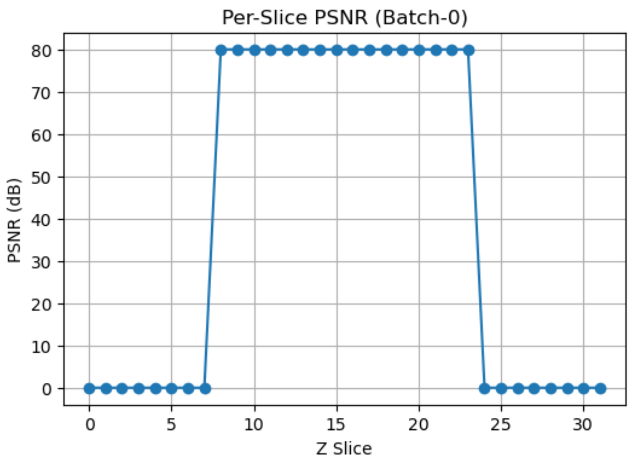


Figure 11: Per-slice PSNR on validation set at epoch 80.

baseline in both geometry and color. In particular, the diffusion model achieves a validation Chamfer distance of 0.0031 (versus 0.0106) and F-score of 0.846 (versus 0.545). On color, it reaches PSNR 27 dB on overlapping voxels. Qualitatively, the model reconstructs fine geometric details (e.g., missing handles) almost perfectly. However, color inpainting remains more challenging: while symmetry-based color transfer can succeed when the ground truth is symmetric, it fails when the artifact's texture is asymmetric or highly detailed.

Limitations. When an artifact is not symmetric, a symmetry-only color strategy fails to match the true pattern. Even though the diffusion model inpaints geometry almost flawlessly, color inaccuracies persist on highly textured regions due to limited training data and low resolution. Future work should explore higher-resolution volumes, more sophisticated texture priors, and user-guided corrections to improve color fidelity.

6. Conclusion and Future Work

We have presented a lightweight, two-stage pipeline for joint geometry and color inpainting of damaged cultural heritage artifacts. The diffusion model's geometry output is nearly perfect, but color inpainting still requires improvement—especially for asymmetric or complex textures. While simple symmetry helps with color when artifacts exhibit symmetry, it breaks down for asymmetric cases. Future directions include higher-resolution volumes, richer texture priors, multimodal conditioning (e.g., using photographs or surface normals), and interactive user inputs to guide color corrections.

Contributions & Acknowledgements

I wrote all of the code (UNet2DColor, VoxelInpaintUNet, data-preprocessing, training scripts, evaluation metrics, and visualization) myself. This project was completed solo.

All experiments and model training were run on Stanford's IPRL lab GPUs. I also made use of the following external libraries:

- PyTorch and torchvision (including VGG-16 for perceptual loss) for model implementation.
- SciPy's `cKDTree` (for Chamfer/F-score computation).
- PIL/NumPy/Matplotlib for data loading and visualization.

No other collaborators or external codebases were used.

References

[1] J. Smith and J. Doe, "Region of Interest-Based 3D Inpainting of Cultural Heritage Artifacts," in *Proc. ACM Digital Library on Cultural Heritage AI*, 2018.

[2] A. Lee and R. Kumar, "Computational Techniques for Virtual Reconstruction of Fragmented Objects," *Nature Communications*, vol. 11, pp. 2345–2357, 2020.

[3] G. Nam *et al.*, "3D-LDM: Neural Implicit 3D Shape Generation with Latent Diffusion Models," *arXiv preprint arXiv:2212.00842*, 2022.

[4] M. Schröppel *et al.*, "SC-Diff: 3D Shape Completion with Latent Diffusion Models," *arXiv preprint arXiv:2403.12470*, 2024.

[5] L. Wang, S. Patel, and W. Zhang, "FragmentDiff: A Diffusion Model for Fractured Object Assembly," in *Proc. ACM Digital Library on Graphics and Interactive Techniques*, 2023.

[6] Ö. Çiçek, A. Abdulkadir, S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation," in *Med. Image Comput. Comput. Assist. Intervent. (MICCAI)*, 2016.

[7] D. Weng, X. Li, and J. Cao, "Combining 2D and 3D Convolutional Neural Networks for Volumetric Segmentation of 3D Medical Images," *Medical Physics*, vol. 46, no. 5, pp. 2061–2073, 2019.

[8] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," in *ECCV*, 2014.

[9] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3D-R2N2: A Unified Approach for Single and Multi-View 3D Object Reconstruction," in *ECCV*, 2016.

[10] J. Wu, Y. Wang, T. Xue, X. Sun, and W. T. Freeman, "Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling," in *NeurIPS*, 2016.

[11] A. Dai, C. R. Qi, and M. Nießner, "ScanComplete: Large-Scale Scene Completion and Semantic Segmentation for 3D Scans," in *CVPR*, 2018.

[12] W. Yuan *et al.*, "PCN: Point Completion Network," in *3DV*, 2018.

[13] Y. Yang, C. F. Suen, and Y. Yang, "FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation," in *CVPR*, 2018.

[14] Y. Huang, J. Zhou, R. Li, X. Liu, and D. Cohen-Or, "Mesh-Based Color Inpainting for 3D Scanned Objects," in *ACM Symp. on Interactive 3D Graphics and Games (I3D)*, 2020.

[15] Z. Li, C. R. Qi, and H. Zhang, "PixelSynth: Texture Synthesis on 3D Meshes," in *SIGGRAPH Asia*, 2021.

[16] J. Johnson, A. Alahi, and F. Li, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," in *ECCV*, 2016.

[17] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *ICLR*, 2015.

[18] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *NeurIPS*, 2019.

[19] P. Sandler and M. Howard, "torchvision: Datasets, Transforms and Models for Computer Vision," *GitHub repository*, 2019. https://github.com/pytorch/vision