Adversarially-Guided 3D Shape Deformation via Differentiable Rendering and 2D Supervision

Andrea Gevasio[1], Christian Napoli[1] and Katarzyna Nieszporek[1]

[1]Department of Artificial Intelligence, Czestochowa University of Technology, Czestochowa, Poland

Abstract

Recovering 3D geometry from 2D observations is a fundamental challenge in computer vision, with applications in animation, virtual reality, and robotics. Recent advances in differentiable rendering have enabled gradient-based optimization of 3D shapes using only image supervision. In this work, we propose a novel adversarial framework that enhances 3D mesh deformation by integrating a differentiable renderer into a Generative Adversarial Network (GAN). The generator deforms an initial mesh and optimizes textures to match 2D supervision from target images, while the discriminator—featuring dense connections and self-attention—learns to distinguish between real and synthesized renderings. Our method improves upon baseline differentiable renderers both quantitatively and qualitatively, achieving lower Chamfer distance and higher Intersection over Union (IoU) across a variety of object categories. The results demonstrate that adversarial training effectively guides mesh deformation, producing reconstructions that are more accurate and visually consistent with target images.

Keywords

differentiable rendering, shape deformation, 2D guidance, adversarial training

1. Introduction

Reconstructing 3D geometry from 2D images is a long-standing goal in computer vision, with applications in virtual and augmented reality, medical imaging, robotics, and digital content creation. Accurate 3D models enable realistic simulations, enhanced diagnostics, and immersive experiences. However, inferring 3D structure from limited 2D information remains a fundamentally ill-posed problem, particularly when dealing with complex shapes, partial occlusions, or diverse object categories.

Traditional 3D reconstruction methods include point cloud processing, voxel grids, and mesh-based optimization. While effective in constrained settings, these approaches often struggle with generalization and scalability. Voxel-based methods are limited by memory and resolution constraints, point cloud methods require dense input data, and mesh optimization techniques often need handcrafted objectives and careful initialization. These limitations are further exacerbated in dynamic or unconstrained environments.

Recent advances in deep learning have enabled significant progress, particularly with the advent of differentiable rendering. By making the rendering process differentiable, neural networks can be trained end-to-end to optimize 3D shape and appearance directly from 2D images. Frameworks such as Soft Rasterizer [1] and PyTorch3D [2] allow backpropagation of image-space losses to 3D geometry, opening the door to more flexible and generalizable reconstruction pipelines.

Despite these advances, current differentiable methods often produce over-smoothed or inaccurate shapes, particularly when only limited views are available [3, 4, 5]. To address this, we propose augmenting differentiable rendering with adversarial supervision. Our method integrates a Generative Adversarial Network (GAN) [6] into the mesh deformation pipeline, where the generator deforms an initial template mesh to match reference images, and the discriminator learns to distinguish real from generated renderings. The discriminator architecture includes dense connections and self-attention to effectively capture fine-grained spatial features [7, 8, 9, 10].

Our contributions are as follows:

- We introduce an adversarially-guided 3D shape deformation method that leverages differentiable rendering and 2D supervision.
- We design a discriminator with dense blocks and self-attention to improve shape fidelity and detail preservation.
- We integrate texture optimization and silhouette supervision to refine appearance and geometry simultaneously.
- We demonstrate quantitatively and qualitatively that our approach outperforms baseline differentiable rendering methods across diverse object categories.

The remainder of the paper is structured as follows. Section 2 reviews related work in differentiable rendering and adversarial mesh optimization. Section 3 presents the architecture and training pipeline. Section 4 describes the experimental setup and results. Section 6 concludes with a discussion of limitations and future work.

SYSTEM 2025: 11th Sapienza Yearly Symposium of Technology, Engineering and Mathematics. Rome, June 4-6, 2025

christian.napoli@pcz.pl (C. Napoli);
katarzyna.nieszporek@pcz.pl (K. Nieszporek)

2. Related Work

The problem of reconstructing 3D geometry from 2D observations has been studied extensively, with a range of techniques proposed over the years. These methods can be broadly categorized into classical reconstruction pipelines, deep learning-based approaches, and differentiable rendering frameworks. Recent efforts have also explored the integration of adversarial training to improve reconstruction fidelity.

2.1. Classical 3D Reconstruction

Traditional methods rely on structured representations such as point clouds, voxel grids, or explicit meshes. Point cloud-based methods require dense and accurate data, which is often impractical to obtain without expensive scanning equipment. Voxel-based approaches discretize space into uniform grids [**?**], but suffer from memory and resolution limitations. Mesh-based methods, while efficient in representing surfaces, often require manual initialization and lack robustness in unconstrained scenarios.

2.2. Learning-Based Mesh Reconstruction

Deep learning has significantly advanced mesh-based reconstruction. Pixel2Mesh [11] introduced a framework that deforms an initial ellipsoid mesh using graph convolutional networks guided by 2D image features. It demonstrated the effectiveness of learning-based deformation but struggled with fine-grained topology and texture detail. The 3Deformer model [12] further improved mesh deformation by incorporating image features into a neural mesh deformation pipeline, achieving high fidelity in geometry and structure. However, these methods are still sensitive to initialization and object complexity.

2.3. Differentiable Rendering

Differentiable renderers provide a powerful tool for optimizing 3D representations directly from image-space losses. Soft Rasterizer (SoftRas) [1] introduced a probabilistic rendering function that enables gradient-based optimization through occlusions and visibility. PyTorch3D [2] extended this idea into a flexible rendering framework for 3D deep learning. Wen et al. [13] built upon differentiable rendering to jointly reconstruct shape and appearance from single-view images using an encoder-decoder architecture, demonstrating improved color and surface detail. However, these methods often suffer from oversmoothing and limited detail recovery, especially under sparse supervision.

Nicolet et al. [14] proposed improving the stability of gradient-based optimization in differentiable rendering using sparse Cholesky factorization. While effective, such techniques are computationally intensive and remain sensitive to initialization and viewpoint ambiguity.

2.4. Adversarial Training for 3D Shape Generation

Adversarial learning has recently been applied to 3D tasks to improve realism and detail preservation. For instance, GANs have been employed to refine volumetric reconstructions or to hallucinate missing geometry. However, applying adversarial training in the context of differentiable mesh deformation remains underexplored. Our work addresses this gap by introducing a discriminator tailored to rendered images, combining dense blocks and self-attention mechanisms to provide fine-grained feedback to the generator during training.

2.5. Summary

In summary, while differentiable rendering has significantly improved 3D reconstruction from 2D supervision, challenges remain in achieving high-quality, generalizable mesh deformations. Adversarial training offers a promising solution by encouraging visual realism and better structural consistency. Our work builds upon this direction by integrating adversarial loss into a differentiable mesh optimization pipeline, enabling the reconstruction of more detailed and accurate 3D shapes.

3. Method

We propose an adversarial training pipeline for 3D mesh deformation, guided by differentiable rendering and 2D supervision. The system consists of a generator that deforms a base mesh to match a target image, and a discriminator that evaluates the visual realism of rendered outputs. The generator is optimized using a combination of reconstruction and regularization losses, while the discriminator provides adversarial feedback based on rendered RGB images and silhouettes.

3.1. Overview

Given a target image of a 3D object, our goal is to deform a source mesh (initialized as a sphere) and optimize its texture to match the target. The mesh is rendered from multiple viewpoints using a differentiable renderer, and the rendered images are compared to the ground truth using a composite loss function. The system alternates between optimizing the generator (mesh and texture) and training the discriminator to distinguish between real and synthetic renderings.

3.2. Data Preparation and Normalization

To ensure generalization across diverse shapes, we construct a dataset using freely available 3D models in the OBJ format [15]. Each model includes geometry (.obj), materials (.mtl), and texture maps.

Meshes are normalized by translating them to the origin and scaling them to fit inside a unit sphere. This ensures consistent scale and positioning, which simplifies optimization and stabilizes training. Meshes are loaded using PyTorch3D's `load_objs_as_meshes` function, which constructs batched `Meshes` objects for downstream processing.

3.3. Differentiable Rendering Pipeline

We render each mesh from multiple viewpoints using PyTorch3D [2]. The rendering setup includes:

3.3.1. Camera Configuration

We use multiple perspective cameras placed at uniformly sampled viewpoints around the object. Camera transformations are computed using `look_at_view_transform`, and projection is performed using `FoVPerspectiveCameras`.

3.3.2. Lighting and Shading

Lighting is modeled with a single `PointLights` source positioned above and to the side of the object. For RGB rendering, we employ a `SoftPhongShader`, which models ambient, diffuse, and specular reflections. For silhouette rendering, we use a `SoftSilhouetteShader` with a thresholded alpha channel to extract binary object contours.

3.3.3. Rasterization Settings

Rasterization is configured with a fixed image resolution and blur radius. We adjust the number of faces per pixel to trade off quality and rendering speed.

3.4. Loss Functions

The generator is trained to minimize a composite loss comprising multiple terms:

- **RGB Loss:** $\mathcal{L}_{\text{RGB}}$ — L2 loss between rendered and target RGB images.
- **Silhouette Loss:** $\mathcal{L}_{\text{sil}}$ — L2 loss between rendered and target silhouettes.
- **Edge Loss:** $\mathcal{L}_{\text{edge}}$ — Encourages preservation of mesh edge lengths to prevent distortion.
- **Normal Consistency Loss:** $\mathcal{L}_{\text{norm}}$ — Promotes smooth surfaces by enforcing normal alignment between adjacent faces.
- **Laplacian Smoothing Loss:** $\mathcal{L}_{\text{lap}}$ — Penalizes large deviations from the mean vertex position.
- **Adversarial Loss:** $\mathcal{L}_{\text{adv}}$ — Binary cross-entropy loss from the discriminator, encouraging realism in rendered outputs.

The total generator loss is defined as:

$$\mathcal{L}_G = \lambda_{\text{RGB}}\mathcal{L}_{\text{RGB}} + \lambda_{\text{sil}}\mathcal{L}_{\text{sil}} + \lambda_{\text{edge}}\mathcal{L}_{\text{edge}} + \\ + \lambda_{\text{norm}}\mathcal{L}_{\text{norm}} + \lambda_{\text{lap}}\mathcal{L}_{\text{lap}} + \lambda_{\text{adv}}\mathcal{L}_{\text{adv}}$$

where the weights λ_i are empirically set (see Section 4).

3.5. Adversarial Discriminator

The discriminator is a CNN designed to assess the realism of rendered RGB images. It integrates dense connections and self-attention to better capture spatial patterns and long-range dependencies.

3.5.1. Architecture

The network consists of an initial convolutional block followed by two dense blocks with growth rate 32 and intermediate channels of 64 and 192, respectively. Each dense block is followed by a self-attention layer with scaled dot-product attention. A final convolutional layer reduces the feature map to a scalar output passed through a sigmoid activation function. Spectral normalization is applied to all convolutional layers to stabilize adversarial training.

3.6. Training Procedure

Training proceeds in alternating steps:

1. **Generator step:** A batch of viewpoints is sampled. The generator deforms the mesh and optimizes texture to minimize the total loss $\mathcal{L}_G$.
2. **Discriminator step:** The discriminator receives real target images and generated renderings. It is trained using binary cross-entropy to maximize classification accuracy.

The generator is optimized using stochastic gradient descent with momentum, while the discriminator uses the Adam optimizer. Training is performed for a fixed number of iterations, with periodic visualizations to track progress.

3.7. Implementation Details

Our implementation uses PyTorch and PyTorch3D, with training conducted on Google Colab using an NVIDIA GPU runtime. All meshes are batched for efficient parallel processing. Code modules are structured for data loading, rendering, loss computation, and model optimization.

4. Experiments

We evaluate our method on a diverse set of 3D objects and compare it against a baseline differentiable rendering pipeline using PyTorch3D. Both quantitative metrics and qualitative visualizations are used to assess reconstruction accuracy, mesh quality, and generalization capability.

4.1. Experimental Setup

Due to restricted access to the ShapeNet dataset, we constructed a custom dataset using freely available 3D models in OBJ format [15]. Each model includes geometry, material, and texture information. All meshes were normalized to fit inside a unit sphere to ensure consistency across objects.

We evaluate five object categories: *cow*, *humanoid*, *fish*, *sword*, and *hand*. Each object is rendered from multiple viewpoints to simulate realistic 2D supervision.

Training was conducted on Google Colab using an NVIDIA GPU runtime. The generator was optimized with SGD (learning rate = 1.0, momentum = 0.9), while the discriminator was trained with the Adam optimizer (lr $= 0.0002$, $\beta_1 = 0.5$, $\beta_2 = 0.999$).

4.2. Evaluation Metrics

We use the following metrics to assess performance:

- **Reconstruction Loss:** The total loss defined in Section 3, combining RGB, silhouette, and regularization terms.
- **Chamfer Distance:** Measures point-wise similarity between predicted and target meshes.
- **Intersection over Union (IoU):** Measures volumetric overlap between the generated and ground truth meshes.
- **Visual Quality:** Qualitative comparisons of mesh renderings across viewpoints.

Unless otherwise noted, all reported values correspond to 2000 training iterations. Extended results for 10000 iterations are provided in the appendix.

4.3. Quantitative Results

Table 1 reports the reconstruction losses after 2000 iterations. Our method achieves comparable or better performance than PyTorch3D across all object categories.

Chamfer distances are shown in Table 2. Our model achieves lower or comparable distances, with significant improvement for the *hand* object, which contains fine structural features.

Table 3 presents IoU values. Our method improves upon the baseline in four out of five cases, especially for *humanoid* and *hand*, indicating better volumetric overlap and shape coverage.

Table 1
Total reconstruction loss (2000 iterations). Lower is better.

Model	Cow	Humanoid	Fish	Sword	Hand
PyTorch3D	0.0250	0.0143	0.0102	0.0105	0.0235
Ours	**0.0250**	0.0163	0.0125	**0.0096**	**0.0189**

Table 2
Chamfer Distance (2000 iterations). Lower is better.

Model	Cow	Humanoid	Fish	Sword	Hand
PyTorch3D	0.00222	0.00095	0.00046	0.00028	0.00157
Ours	**0.00221**	**0.00091**	0.00046	**0.00028**	**0.00102**

Table 3
Intersection over Union (2000 iterations). Higher is better.

Model	Cow	Humanoid	Fish	Sword	Hand
PyTorch3D	0.9720	0.8652	**0.9566**	0.6184	0.8490
Ours	**0.9762**	**0.9184**	0.9290	**0.6347**	**0.9417**

4.4. Qualitative Results

Figure 1 illustrates side-by-side renderings of the deformed meshes from PyTorch3D and our method, compared to the target image. Our model more accurately captures subtle features, such as the curvature of the horns and positioning of the ears.

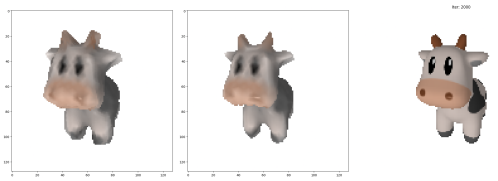

Figure 1: Visual comparison (left to right): PyTorch3D, our method, target image.

In Figure 2, we observe that PyTorch3D fails to accurately reconstruct the finger geometry of the hand object, while our model preserves the detailed articulation more effectively.

4.5. Extended Training Analysis

Training the models for 10000 iterations improves both reconstruction loss and geometric fidelity. Full tables and visualizations are provided in Appendix A. Notably, our method consistently outperforms the baseline in Chamfer distance and IoU with longer training, especially for high-frequency shapes such as *hand* and *sword*.

Table 4
Total reconstruction loss after 10,000 iterations.

Model	Cow	Humanoid	Fish	Sword	Hand
PyTorch3D	0.0226	0.0134	0.0102	0.0107	0.0211
Ours	**0.0214**	**0.0125**	0.0104	**0.0106**	**0.0189**

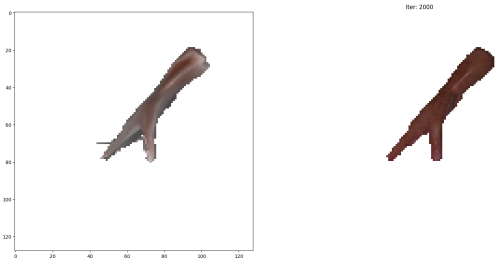

Figure 2: PyTorch3D-rendered hand showing deformation artifacts in the fingers.

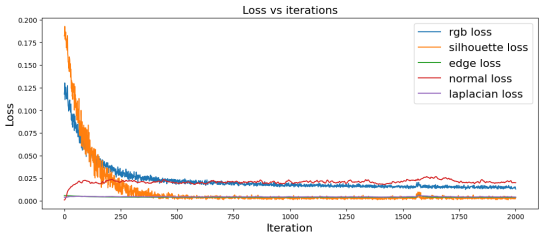

Figure 3: Training loss evolution for the *cow* object over 2000 iterations.

5. Extended Results

This appendix presents additional experimental results obtained by training the models for 10,000 iterations, along with loss progression plots and extended visual comparisons for both training durations.

5.1. Training Loss Evolution

Figure 3 illustrates the training loss components over 2000 iterations for the *cow* object. This plot is representative of the convergence behavior observed across all categories.

5.2. Quantitative Results at 10,000 Iterations

Tables 4, 5, and 6 present total loss, Chamfer distance, and IoU, respectively, after 10,000 training iterations. Our model shows continued improvement with more training, especially for high-detail categories like *hand* and *sword*.

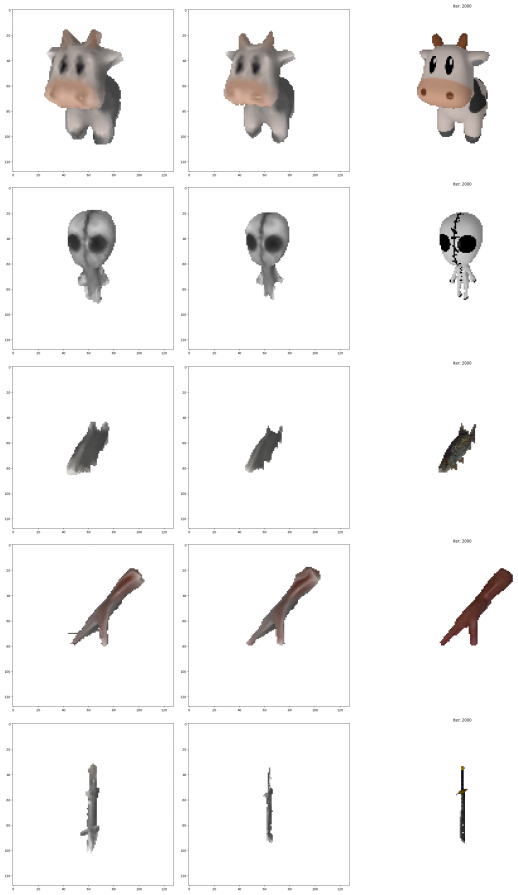

Figure 4: Qualitative comparison of reconstructions after 2000 iterations. Left: PyTorch3D, Center: our method, Right: target image.

5.3. Qualitative Comparisons

Figures 4 and 5 show side-by-side comparisons of mesh reconstructions after 2000 and 10,000 iterations, respectively. Each row includes results from PyTorch3D (left), our method (center), and the target image (right).

As shown, our method consistently produces more

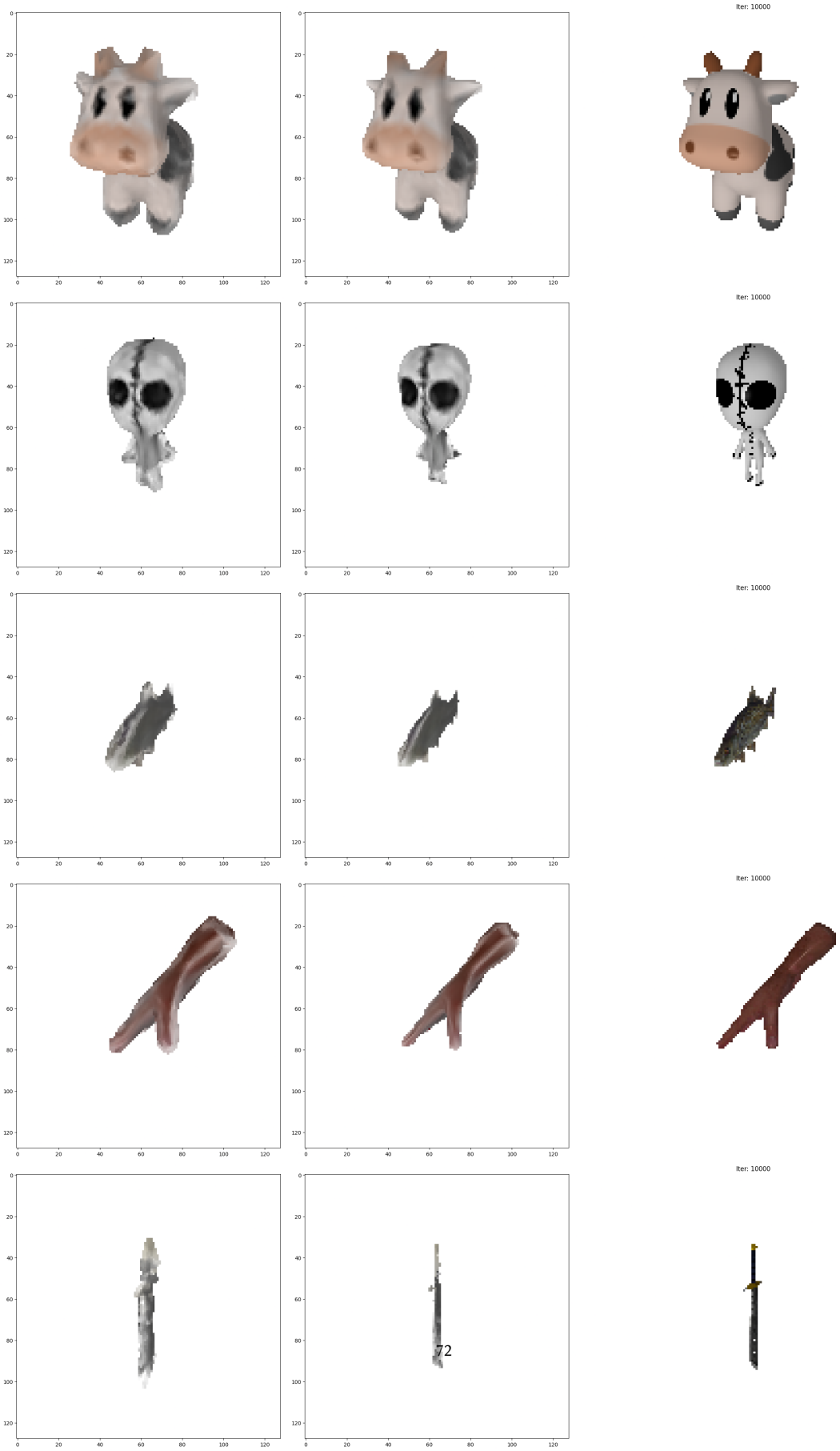

Figure 5: Qualitative comparison of reconstructions after 10,000 iterations. Left: PyTorch3D, Center: our method, Right: target image.

Table 5
Chamfer Distance after 10,000 iterations. Lower is better.

Model	Cow	Humanoid	Fish	Sword	Hand
PyTorch3D	0.00221	0.00084	**0.00044**	0.00028	0.00130
Ours	**0.00220**	**0.00075**	0.00045	**0.00028**	**0.00095**

Table 6
Intersection over Union after 10,000 iterations. Higher is better.

Model	Cow	Humanoid	Fish	Sword	Hand
PyTorch3D	0.9735	0.8941	0.9343	0.6262	0.8885
Ours	**0.9884**	**0.9230**	**0.9322**	**0.6779**	**0.9600**

accurate deformations, especially in object regions with complex structure or fine detail (e.g., *hand*, *sword*). These improvements validate the benefit of adversarial supervision for guiding mesh optimization under weak 2D supervision. Compared to the baseline, which often fails to preserve sharp boundaries or introduces artifacts in regions with occlusion or high curvature, our approach maintains geometric consistency and enhances fidelity to the silhouette and inner contours observed in the target images. Notably, at 10,000 iterations, the refinement introduced by our method leads to significant alignment not only in the external silhouette but also in internal features such as joint articulation and surface topology, confirming the progressive advantage of adversarial cues over traditional loss-only optimization strategies.

Furthermore, the visual quality improvements observed in later iterations indicate that the adversarial discriminator plays a crucial role in discouraging unrealistic deformations and encouraging plausible mesh structures even when direct pixel supervision is limited. This qualitative evidence complements the quantitative results reported in Section **??**, and supports the hypothesis that leveraging learned priors from adversarial training leads to more robust and semantically coherent reconstructions, particularly when only sparse or partial supervision is available.

6. Conclusion

We presented an adversarial framework for 3D shape deformation guided by differentiable rendering and 2D image supervision. By integrating a mesh generator with a self-attention-based discriminator, our method improves the visual quality and geometric accuracy of reconstructed 3D meshes from sparse image inputs.

Our results demonstrate that adversarial training can enhance mesh fidelity over standard differentiable rendering pipelines. Quantitatively, our method achieves lower Chamfer distances and higher Intersection over Union scores across multiple object categories. Qualitatively, it produces more realistic deformations, especially in regions with fine-grained geometry such as limbs or object extremities.

This approach contributes to the broader goal of building generalizable, high-fidelity 3D reconstruction systems that operate under weak supervision. Our design remains simple and modular, leveraging widely available toolkits such as PyTorch3D and standard GAN components.

Future work will explore extending this framework to dynamic or articulated objects, learning category-specific priors, and incorporating temporal consistency for video-based shape reconstruction. Additionally, improving texture fidelity and integrating semantic segmentation into the adversarial loss are promising directions.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT, Grammarly in order to: Grammar and spelling check, Paraphrase and reword. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] S. Liu, T. Li, W. Chen, H. Li, Soft rasterizer: A differentiable renderer for image-based 3d reasoning, 2019. URL: https://arxiv.org/abs/1904.01786. arXiv:1904.01786.

[2] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, G. Gkioxari, Accelerating 3d deep learning with pytorch3d, arXiv:2007.08501 (2020).

[3] N. Brandizzi, A. Fanti, R. Gallotta, S. Russo, L. Iocchi, D. Nardi, C. Napoli, Unsupervised pose estimation by means of an innovative vision transformer, in: Lecture Notes in Computer Science

(including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 13589 LNAI, 2023, p. 3 – 20. doi:10.1007/978-3-031-23480-4_1.

[4] N. Boutarfaia, S. Russo, A. Tibermacine, I. E. Tibermacine, Deep learning for eeg-based motor imagery classification: Towards enhanced human-machine interaction and assistive robotics, in: CEUR Workshop Proceedings, volume 3695, 2023, p. 68 – 74.

[5] G. De Magistris, R. Caprari, G. Castro, S. Russo, L. Iocchi, D. Nardi, C. Napoli, Vision-based holistic scene understanding for context-aware human-robot interaction, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 13196 LNAI, 2022, p. 310 – 325. doi:10.1007/978-3-031-08421-8_21.

[6] S. Russo, S. Ahmed, I. E. Tibermacine, C. Napoli, Enhancing eeg signal reconstruction in cross-domain adaptation using cyclegan, in: Proceedings - 2024 International Conference on Telecommunications and Intelligent Systems, ICTIS 2024, 2024. doi:10.1109/ICTIS62692.2024.10894543.

[7] D. Połap, M. Woźniak, C. Napoli, E. Tramontana, R. Damaševičius, Is the colony of ants able to recognize graphic objects?, Communications in Computer and Information Science 538 (2015) 376 – 387. doi:10.1007/978-3-319-24770-0_33.

[8] M. Woźniak, D. Połap, M. Gabryel, R. K. Nowicki, C. Napoli, E. Tramontana, Can we process 2d images using artificial bee colony?, in: Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science), volume 9119, 2015, p. 660 – 671. doi:10.1007/978-3-319-19324-3_59.

[9] C. Napoli, G. Pappalardo, E. Tramontana, Z. Marszalek, D. Polap, M. Wozniak, Simplified firefly algorithm for 2d image key-points search, in: IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - CIHLI 2014: 2014 IEEE Symposium on Computational Intelligence for Human-Like Intelligence, Proceedings, 2014. doi:10.1109/CIHLI.2014.7013395.

[10] G. Lo Sciuto, G. Capizzi, R. Shikler, C. Napoli, Organic solar cells defects classification by using a new feature extraction algorithm and an ebnn with an innovative pruning algorithm, International Journal of Intelligent Systems 36 (2021) 2443 – 2464. doi:10.1002/int.22386.

[11] N. Wang, Y. Zhang, Z. Li, Y. Fu, H. Yu, W. Liu, X. Xue, Y.-G. Jiang, Pixel2mesh: 3d mesh model generation via image guided deformation, IEEE Transactions on Pattern Analysis and Machine Intelligence 43 (2021) 3600–3613. doi:10.1109/TPAMI.2020.2984232.

[12] H. Su, X. Liu, J. Niu, J. Wan, X. Wu, 3deformer: A common framework for image-guided mesh deformation, 2023. URL: https://arxiv.org/abs/2307.09892. arXiv:2307.09892.

[13] Y. Zhu, Y. Zhang, Q. Feng, Colorful 3d reconstruction from a single image based on deep learning, in: Proceedings of the 2020 3rd International Conference on Algorithms, Computing and Artificial Intelligence, ACAI '20, Association for Computing Machinery, New York, NY, USA, 2021. URL: https://doi.org/10.1145/3446132.3446157. doi:10.1145/3446132.3446157.

[14] B. Nicolet, A. Jacobson, W. Jakob, Large steps in inverse rendering of geometry, ACM Trans. Graph. 40 (2021). URL: https://doi.org/10.1145/3478513.3480501. doi:10.1145/3478513.3480501.

[15] Free3D.Com, 3d models for free, 2024. URL: https://free3d.com/3d-models/obj.