One-2-3-45: Any Single Image to 3D Mesh in 45 Seconds without Per-Shape Optimization - Supplementary Materials

Anonymous Author(s)
Affiliation
Address
email

1 Overview

We first show more qualitative comparison in Section 2, which is followed by a demonstration of additional examples on real-world images and the text-to-3D task in Sections 3 and 4 respectively. Furthermore, we present the details of our elevation estimation module in Section 5, training and evaluation details in Section 6. We finally show the failure cases and discuss the limitations in Section 7. **In addition, we offer three HTML files that will be introduced in the following sections, showcasing videos of 360° meshes. We recommend opening these HTML files using Chrome or Firefox browsers, since Safari users may encounter loading issues.**

2 More Qualitative Comparison

Figure 1: We compare One-2-3-45 with Point-E [7], Shap-E [3], Zero123 (Stable Dreamfusion version) [4], 3DFuse [10], and RealFusion [6]. In each example, we present both the textured and textureless meshes. As 3DFuse [10] and RealFusion [6] do not natively support the export of textured meshes, we showcase the results of volume rendering instead.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

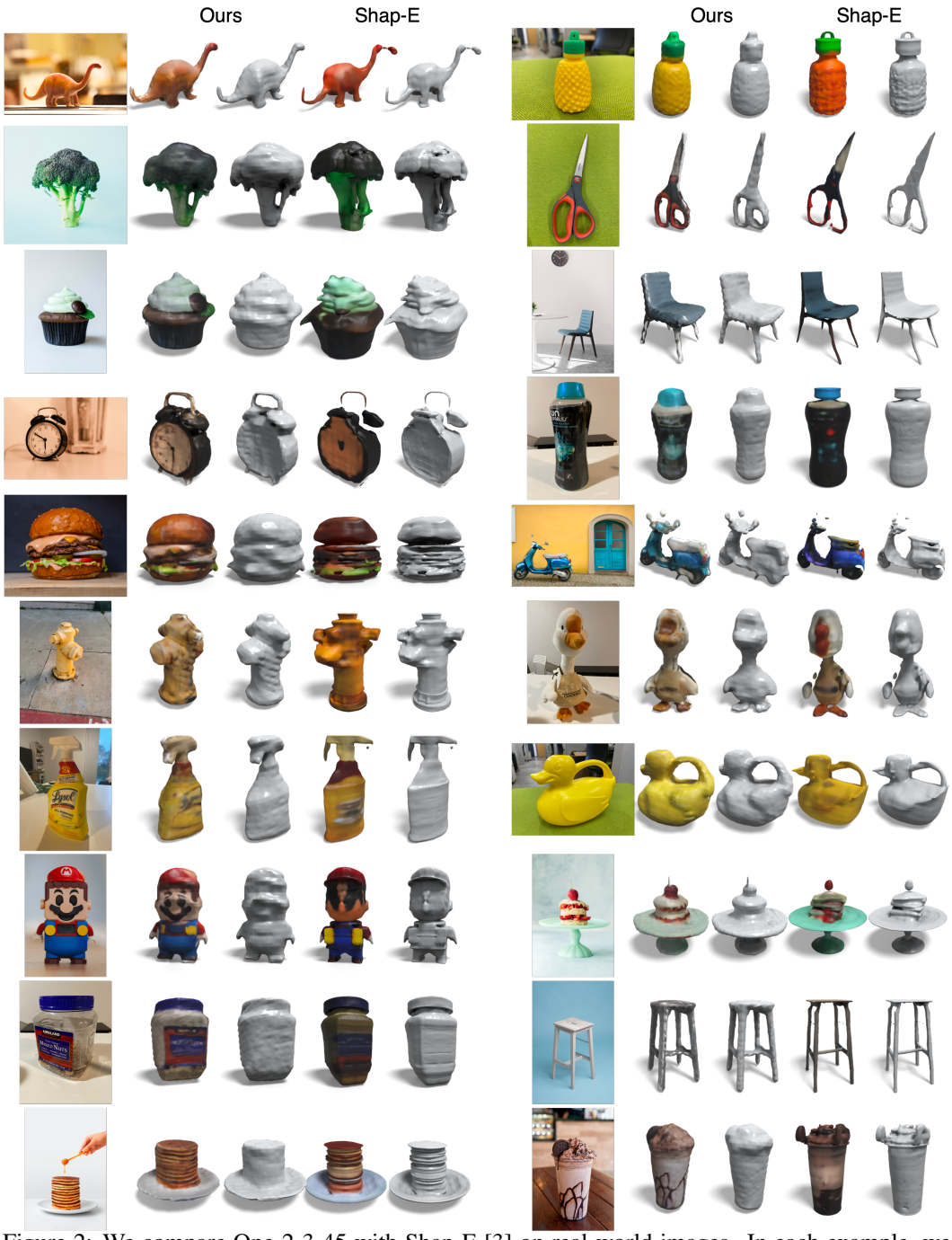

Figure 2: We compare One-2-3-45 with Shap-E [3] on real-world images. In each example, we present the input image, generated textured and textureless meshes.

In Figure 1, we demonstrate more qualitative comparison on Objaverse [1] and GoogleScannedObjects (GSO) [2] datasets. Note that all test shapes are not seen during the training of our 3D reconstruction module. **See comparison.html for videos of** 360° **meshes.**

3 More Examples on Real-World Images

In Figure 2, we showcase more examples on real-world images and compare our method with the concurrent method Shap-E [3]. The input images are from `unsplash.com` or captured by ourselves. Note that our results exhibit a closer adherence to the input image. **See real_world.html for videos of** 360° **meshes.**

Figure 3: Text-to-3D: We compare our method against two native text-to-3D approaches Stable DreamFusion [8] and 3DFuse [10]. To enable text-to-3D, our method first uses a pretrained text-to-image model DALL-E 2 [9] to generate an image from input text (prompted with "3d model, long shot"), and then uplifts the image to a 3D textured mesh.

4 More Examples on Text-to-3D

In Figure 3, we present additional examples for the text-to-3D task. It is evident that existing approaches struggle to capture fine-grained details, such as a tree hollow, or achieve compositionality, as seen in examples like an orange stool with green legs, a pineapple-shaped Havana hat, or a rocking horse chair. In contrast, our method produces superior results that adhere more closely to the input text. We hypothesize that controlling such fine-grained attributes in the 3D space using

existing optimization strategies is inherently challenging. However, by leveraging established 2D text-to-image diffusion models, our method becomes more effective in lifting a single 2D image to a corresponding 3D textured mesh. **See text_to_3d.html for videos of 360° meshes.**

5 Details of Elevation Estimation

To estimate the elevation angle θ of the input image, we first utilize Zero123 [4] to predict four nearby views (10 degrees apart) of the input view. With these predicted views, we proceed to enumerate all possible elevation angles and compute the re-projection error for each candidate angle. The re-projection error assesses the consistency between camera poses and image observations, akin to the bundle adjustment module employed in the Structure-from-Motion (SfM) pipeline.

Specifically, we enumerate all candidate elevation angles in a coarse-to-fine manner. In the coarse stage, we enumerate elevation angles with a 10-degree interval. Once we have determined the elevation angle e^* associated with the smallest re-projection error, we proceed to the fine stage. In this stage, we enumerate elevation angle candidates ranging from $e^* - 10^\circ$ to $e^* + 10^\circ$ with a 1-degree interval. This coarse-to-fine design facilitates rapid estimation, completing the elevation estimation module in under 1 second for each shape.

Given a set of four predicted nearby views, we perform feature matching to identify corresponding keypoints across each pair of images (a total of six pairs) using an off-the-shelf module LoFTR [11]. For each elevation angle candidate, we calculate the camera pose for the input image by employing the spherical coordinate system with a radius of 1.2 and an azimuth angle of 0. Note that the azimuth angle ϕ and the radius r can be arbitrarily adjusted, resulting in the rotation and scaling of the reconstructed object accordingly. Subsequently, we obtain the camera poses for the four predicted views by incorporating the specified delta poses.

Once we have the four posed images, we compute the re-projection error by enumerating triplet images. For each triplet of images (a, b, c) sharing a set of keypoints P, we consider each point $p \in P$. Utilizing images a and b, we perform triangulation to determine the 3D location of p. We then project the 3D point onto the third image c and calculate the reprojection error, which is defined as the $l1$ distance between the reprojected 2D pixel and the estimated keypoint in image c. By enumerating all image triplets and their corresponding shared keypoints, we obtain the mean projection error for each elevation angle candidate.

6 Details of Training and Evaluation

Training We train the reconstruction module using the following loss function:

$$\mathcal{L} = \mathcal{L}_{rgb} + \lambda_0 \mathcal{L}_{depth} + \lambda_1 \mathcal{L}_{eikonal} + \lambda_2 \mathcal{L}_{sparsity} \tag{1}$$

where $\mathcal{L}_{rgb}$ represents the $l1$ loss between the rendered and ground truth color, weighted by the sum of accumulated weights; $\mathcal{L}_{depth}$ corresponds to the $l1$ loss between the rendered and ground truth depth; $\mathcal{L}_{eikonal}$ and $\mathcal{L}_{sparsity}$ are the Eikonal and sparsity terms, respectively, following SparseNeuS [5]. We empirically set the weights as $\lambda_0 = 1$, $\lambda_1 = 0.1$, and $\lambda_2 = 0.02$. For λ_2, we adopt a linear warm-up strategy following SparseNeuS [5]. To train our reconstruction module, we utilize the LVIS subset of the Objaverse [1] dataset, which consists of 46k 3D models across 1,156 categories. The reconstruction module is trained for 300k iterations using two A10 GPUs, with the training process lasting approximately 6 days. It is important to note that our reconstruction module does not heavily rely on large-scale training data, as it primarily leverages local correspondence to infer the geometry, which is relatively easier to learn and generalize.

Evaluation We evaluate all baseline approaches using their official codebase. Since the approaches take only a single image as input, the predicted mesh may not have the same scale and transformation as the ground-truth mesh. To ensure a fair comparison, we employ the following process to align the predicted mesh with the ground-truth mesh. First, we align the up direction for the results generated by each approach. Next, for each generated mesh, we perform a linear search over scales and rotation angles along the up direction. After applying each pair of scale and z-rotation, we utilize the Iterative Closest Point (ICP) algorithm to align the transformed mesh to the ground-truth mesh. Finally, we select the mesh with the largest number of inliers as the final alignment. This alignment process

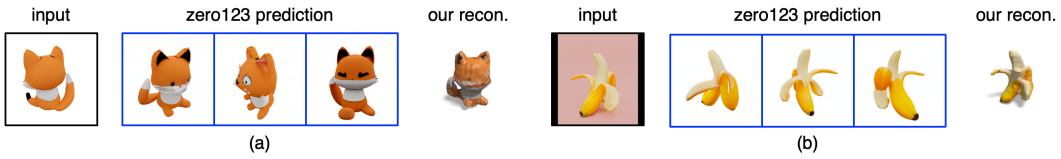

Figure 4: Failure cases. Our method relies on Zero123 to generate multi-view images, and we encounter challenges when Zero123 generates inconsistent results. (a) The input view lacks sufficient information. (b) The input view contains ambiguous or complicated structures.

helps us establish a consistent reference frame for evaluating the predicted meshes across different approaches.

7 Failure Cases and Limitations

Our method relies on Zero123 for generating multi-view images, which introduces challenges due to its occasional production of inconsistent results. In Figure 4, we present two typical cases that exemplify such inconsistencies. The first case involves an input view that lacks sufficient information, such as the back view of a fox. In this scenario, Zero123 struggles to generate consistent predictions for the invisible regions, such as the face of the fox. As a consequence, our method may encounter difficulties in accurately inferring the geometry for those regions. The second case involves an input view with ambiguous or complex structures, such as the pulp and peel of a banana. In such situations, Zero123's ability to accurately infer the underlying geometry becomes limited. As a result, our method may be affected by the inconsistent predictions generated by Zero123. It is important to acknowledge that these limitations arise from the occasional scenarios, and they can impact the performance of our method in certain cases. Addressing these challenges and refining the reliability of Zero123's predictions remain areas for further investigation and improvement.

We have also noticed slight artifacts on the back side of our generated results. As one of the first works in combining view-conditioned 2D diffusion models with generalizable multi-view reconstruction, we believe that there is still ample room for exploring more advanced reconstruction techniques and incorporating additional regularizations. By doing so, we expect to significantly mitigate the minor artifacts and further enhance results in the future.

References

[1] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. *arXiv preprint arXiv:2212.08051*, 2022.

[2] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2553–2560. IEEE, 2022.

[3] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023.

[4] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. *arXiv preprint arXiv:2303.11328*, 2023.

[5] Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. Sparseneus: Fast generalizable neural surface reconstruction from sparse views. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 210–227. Springer, 2022.

[6] Luke Melas-Kyriazi, Christian Rupprecht, Iro Laina, and Andrea Vedaldi. Realfusion: 360 {\deg} reconstruction of any object from a single image. *arXiv preprint arXiv:2302.10663*, 2023.

[7] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022.

[8] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[9] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[10] Junyoung Seo, Wooseok Jang, Min-Seop Kwak, Jaehoon Ko, Hyeonsu Kim, Junho Kim, Jin-Hwa Kim, Jiyoung Lee, and Seungryong Kim. Let 2d diffusion model know 3d-consistency for robust text-to-3d generation. *arXiv preprint arXiv:2303.07937*, 2023.

[11] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8922–8931, 2021.