Reconstructing the Past: A Novel Dataset for Single-View 3D Modeling of Cultural Relics

Xinyuan Xia[1], Yidan Cui[1], Jinyu Cai[1], Yuanyi Song[1], Yuxin Zhang[1], Xueying Li[1], Yutong Huang[1], Yingjun Shang[1], QinYi Jia[1], Haomin Ma[1], Xiang Liao[1], Xingzhi Zhong[1]
[1]The Experimental Class of Artificial Intelligence, Shanghai Jiao Tong University

Abstract

Single-view 3D reconstruction is a significant problem in the field of machine learning with broad application prospects. Its basic principle involves deriving the complete 3D structure from an image of a given perspective using some prior knowledge and geometric assumptions. Due to the lack of depth and parallax information in single-view images, single-view 3D reconstruction poses considerable challenges. In this project, we used web crawlers to collect a vast amount of data models of cultural relics and curiosities from multiple 3D model websites. These models were then subjected to cleansing and rendering processes, ultimately yielding over 8,500 model data points. We constructed a dataset of cultural relics and curiosities that conforms to the input format of pre-trained models. After surveying multiple papers, we adopted the One-2-3-45 model and leveraged neural radiance fields for 3D reconstruction, achieving promising results.

Code: https://github.com/MinstrelsyXia/3D-dataset-builder/tree/main

1. Introduction

In recent years, the field of computer vision has seen significant progress. Among them, Single-view 3D reconstruction stands as a critical challenge within machine learning, holding vast potential for diverse applications. This capability is crucial for a wide range of applications, from virtual reality and robotics to autonomous systems and computer-aided design. The task essentially revolves around the recovery of three-dimensional structures from two-dimensional images by leveraging inherent knowledge and geometric principles.

The core challenge in this area involves extracting detailed 3D information from just one 2D image. Unlike traditional approaches that rely on multiple views or depth sensors, single-view reconstruction must infer depth, structure, and texture from limited data. This poses unique difficulties but also opens up opportunities for more accessible and cost-effective solutions across various industries. By the absence of depth cues and the lack of multiple viewpoints in single-image scenarios, this significantly complicates the endeavor to accurately reconstruct 3D models from just one perspective.

In the domain of single-view 3D reconstruction, significant strides have been made by leveraging deep learning and multimodal approaches. Early methods primarily relied on heuristic techniques and manually designed features [7], which were limited in their ability to capture the complex structures of objects. The introduction of neural networks marked a turning point, allowing for more sophisticated feature extraction and representation learning [2]. Notably, advancements in generative models and the integration of geometric principles have further pushed the boundaries of what can be achieved with single-view 3D reconstruction [10].

Researchers have explored various strategies to address the inherent challenges of reconstructing 3D models from a single image. For instance, some studies have employed class-specific priors by training 3D generative networks on labeled shape datasets [3], although these methods often fail to generalize well to unseen categories. Others have utilized neural radiance fields (NeRF) [13] and diffusion models [12] to enhance the quality and efficiency of reconstructions.

A notable trend is the use of large-scale datasets and pretrained models to facilitate transfer learning and improve generalization. For example, the Gibson Env [12] provides realistic environments for embodied agents, while Habitat [8] offers extensive resources for training navigation systems. Moreover, frameworks like GOAT-Bench [3] benchmark lifelong learning capabilities in multi-modal settings, highlighting the importance of adaptability.

Despite these advancements, existing methods still face common dilemmas such as time-consuming optimizations, memory intensiveness, and inconsistencies in 3D results [13]. To overcome these limitations, recent works have proposed novel solutions that aim to streamline the reconstruction process while maintaining or improving accuracy [9]. For instance, One-2-3-45 [4] leverages view-conditioned 2D diffusion models and SDF-based neural surface reconstruction to generate high-quality 360-degree meshes efficiently.

Overall, the field continues to evolve with ongoing research focusing on enhancing the robustness, efficiency, and generalizability of single-view 3D reconstruction methods. These efforts are crucial for advancing applications in areas such as robotics, augmented reality, and cultural heritage preservation.

To address the challenges and advance research in this field, we have initiated a project aimed at building a high-quality dataset for training and optimizing single image 3D generation models. In this project, we utilized web crawlers to collect a large number of digital models of cultural relics and treasures from multiple 3D modeling websites. Then, these models were cleaned and rendered, resulting in a dataset containing over 8500 models that meet the input format requirements of the pretrained model, providing a solid foundation for subsequent model training.

After reviewing multiple papers, we chose the One-2-3-45 model [4] and used neural radiation field (NeRF) [5] technology for 3D reconstruction, achieving promising results. In addition, we implemented an automatic rendering process using the Blender API, ensuring that each. glb object can be accurately rendered into an image from multiple angles, including ground truth color images, standardized depth maps, and millimeter depth maps, further improving the quality of the dataset.Finally, we will summarize the research findings and discuss potential future research directions and technological improvements.

Through the implementation of this project, we not only provided valuable resources for single view 3D reconstruction, but also verified the feasibility and effectiveness of the new technology in practical applications, providing reference for future research.

2. Dataset Setup

In the task of single-image 3D generation, the quality of the dataset greatly affects the performance of the 3D generation model. A high-quality, diverse, and representative dataset can provide the model with rich learning materials, enabling it to infer the 3D structure more accurately from a single image. We crawled approximately 10,000 model data from various websites, and combined the use of GPT methods and human methods to clean the raw data and translate the model descriptions into English. We rendered the cleansed model files and generated the required dataset.

2.1. Spider

In the data collection task, we need to employ web scraping techniques to extract the required 3D models and associated information from several 3D model websites. All content will be consolidated into a JSON file for subsequent model rendering and training.

First, we will discuss the fundamental concepts of web scraping. Web scraping involves simulating user behavior to search for and extract the source code of web pages, thereby obtaining the required nodes. The simulation of user behavior encompasses all interactions with the browser, including opening and closing the browser, mouse clicks, scrolling, and so on. The source code of a webpage can be accessed through the inspection menu available via a right-click, revealing that the code is composed of numerous nested nodes, such as and , each representing an element on the page, such as images or links. Moreover, each node contains relevant attributes, such as class and id, which can be utilized for retrieval. The method for retrieving a specific node is accomplished by constructing the path to that node within the source code of the webpage, known as XPath, which establishes a unique path for the node within that webpage. In other words, by specifying the XPath in the script, we can leverage web scraping libraries to automatically locate the corresponding nodes within the webpage's source code.

In practice, we utilized the Selenium web scraping library, along with the XPath Helper plugin to assist in retrieving the XPath paths of nodes. The crawling scripts and data extraction were primarily performed on three 3D model websites: https://3d.si.edu, https://sketchfab.com, and https://poly.cam. The theme of the models we aimed to retrieve was "wenwan" or cultural artifacts, referring to objects that have cultural, artistic, and historical value. Classic types of cultural artifacts include the Four Treasures of the Study, ancient currency, tea utensils, seals, and prayer beads, among others.

Next, we will take the https://poly.cam website as an example to discuss the process of writing a web scraping script in detail. First, we need to choose the caption and construct corresponding URL. Then, we will open the browser and set the browser preferences. The scraping script will facilitate actions such as loading the page, logging in to the account, and filling in the email address. Once we access the model download

interface, we will first obtain a list of all available models on the current page. We will then iterate through each model, performing several button clicks and waiting for the page to load. For each model, the script will extract the model name, textual description, download link, and UID from the source code. Finally, we will click the download button to download the model to the local system.

During the retrieval process, we experimented with numerous keywords, including ceramic, bronze, dynasty, ornaments, statue, tradition, religion, clay, pottery, mythology, sculpture, artifact, teapot, vase, ancient, antique, culture, skull, Chinese, and so on. We found that some inappropriate samples were mixed in; for instance, while scraping for "vase," we encountered not only antique-style flower vases but also modern artistic vases. Additionally, due to the nature of certain websites, some of the models available for free download were rendered and uploaded by individual users, which meant that the quality of these models could not be guaranteed. We will remove these exceptions during the subsequent model cleaning process.

During the implementation of the script, we encountered several challenges and overcame them one by one. For example, some web pages required scrolling to load additional models, but the scrollbar could not be explicitly obtained from the page source code. Therefore, we opted for an implicit approach by adjusting the offset of the model traversal list for automatic adjustment. Additionally, regarding the saving of model files, we used UUIDs to assign each model a unique identifier for subsequent retrieval.

Through our relentless efforts, we managed to crawl nearly 10,000 models in total till the end.

2.2. Data Cleansing

Data cleansing refers to the process of processing collected data to eliminate errors, duplicates, and irrelevant information, ensuring the quality and consistency of the data. This process can reduce noise during model training, eliminate data that may cause model bias, and ensure consistent data formats and measurement standards within the dataset, thereby improving the accuracy of the model. By combining the use of GPT methods and manual screening, we have greatly enhanced the efficiency of data cleansing.

2.2.1 GPT-based Method

Prompt

According to the following descriptions and object names, categorize each as: human bone or its model, decorative items, artworks, sculptures, cultural relics or antiques(1), real plants(2), real animals or animal skeletons(3), buildings or architectural models(4), or none of them(5). For example, masks, badges and stones should be categorized as 1. Any ancient items or those from museums should also be categorized as 1, souvenirs should be categorized as 1. Objects made of bronze, stone or clay should be categorized as 1. Clothes, caps, chairs should be categorized as 5: \n{batch_request}\n. Please provide the category index for each model in the form like Model 2: 1.

Due to the large volume of data, manual cleaning is time-consuming. Therefore, we propose using GPT to pre-screen unqualified data based on the model's descriptions, and then humans can decide whether to retain the data flagged as unqualified by GPT. This significantly reduces the workload.

GPT categorizes the 3D files based on their object names and descriptions into five classes: Artifacts, Animals, Plants, Buildings, Others.

This classification approach is designed to select Class 1 (cultural Artifacts) and exclude Classes 2, 3, and 4, while leaving Class 5 (uncategorized) for human judgment.

Since the goal is to filter out cultural items, we provide GPT with detailed prompts about what objects fall into this category, including:"human bone or its model, decorative items, artworks, sculptures, cultural relics, or antiques."

Additionally, examples of objects that might confuse GPT during the screening process are included in the prompt to further improve its accuracy.

2.2.2 Human-based Method

After checking the cleased data, we've found that some captions doesn't necessarily describe the object, thus confusing the GPT while determining its label, so doubling checking by visual features is necessary.

Therefore, we've presented a toy display webpage where you can load the local glb files, rotate the object by your mouse to check its features, and click the button to judge whether it is ancient antique or not.

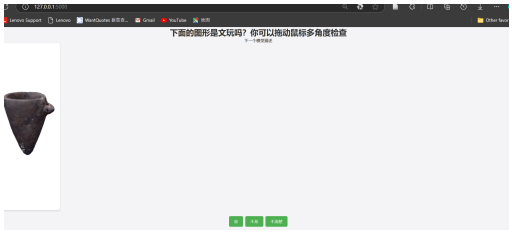


Figure 1. A toy display of models

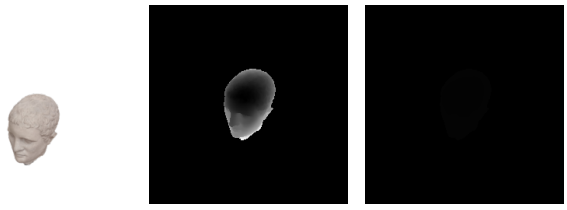
Then, the backend automatically receives the choice of user and move the original glb file to different folders. We mainly use python lib Flask and route mechanism as backend, javascript as frontend rendering and css as page management.

2.2.3 Caption Translation and Data Gathering

2.3. Blender Rendering

We automated our rendering process using Blender Python API. We finally got 120 pictures for one .glb object: totally 40 views, and each view contains a ground truth picture, a depth map (in meters) and a depth map in millimeters. The implementation details are as follows.

1. Initialization: Set the camera in the scene. The resolutions of x and y axis are both 256. Set the render device to CYCLES and choose a relatively low sample rate for efficiency.
2. Normalization: Normalize the object, and move it to the center.
3. Setting Camera: For each camera to world matrix in the given One2345_training_pose.json, change the camera pose. Set the focal length to 28 according to the given intrinsic matrix.
4. Rendering: For different picture types to render (ground truth, normalized depth map, millimeter depth map), set the corresponding compositor and render the picture.



(a) ground truth (b) normalized depth map (c) millimeter depth map

Figure 2. The three pictures rendered at camera view 0_0_10

2.4. Summary

We construct the dataset for single-view 3D reconstruction with an emphasis on scale, diversity, and meticulous cleansing, ensuring its suitability for high-quality 3D reconstruction tasks. Approximately 10,000 3D models of cultural artifacts ("wenwan") were collected from various online sources. These include categories such as ancient tools, ornaments, sculptures, and pottery, ensuring a rich variety of object types.

We employ web scraping techniques to gather the data, utilizing libraries like Selenium and tools like XPath Helper for precise extraction of model information including names, descriptions, download links, and unique identifiers (UIDs). The scraping process was carried out across multiple websites, including Smithsonian's 3D Explorer, Sketchfab, and Polycam, using a wide range of keywords to capture diverse artifacts. To better demonstrate the scale and high quality of our dataset, we visualized it using a word cloud as shown in figure 3

Figure 3. Word Cloud

In terms of data quality, we conducted rigorous cleansing, combining GPT-based methods and manual verification to filter out irrelevant, low-quality, or inappropriate models. After an initial cleansing by GPT, we followed up with a detailed manual inspection, ensuring that only high-quality models remained in the dataset. Each model was rendered to verify usability, and unique identifiers were assigned for efficient management. Additionally, we standardized and translated model descriptions into English to maintain linguistic consistency and clarity across the dataset.

This combination of large-scale collection, diverse category inclusion, and careful cleansing ensures a comprehensive dataset capable of enhancing the performance and reliability of single-image 3D generation models. Figure 4 demonstrates some examples of the 3D models we obtained from various websites.

Figure 4. Dataset Overview

3. Method

One-2-3-45 utilizes a 2D diffusion model combined with cost volume-based 3D reconstruction technology to achieve 3D reconstruction from a single image, boasting astonishing processing speed and exceptional output quality. We have studied the related papers, replicated the model's code, and trained it on our cultural artifacts dataset, obtaining favorable results.

3.1. NERF

NeRF, or Neural Radiance Fields, is a powerful method in the field of computer graphics and computer vision for synthesizing highly detailed and realistic 3D scenes by Mildenhall et al [6].

3.1.1 The implementation of NeRF

The core idea of the paper is to train a multilayer perceptron (MLP) to map the spatial coordinates (x, y, z) and viewing direction (θ, ϕ) of a 3D point to its radiance field properties: density and color (r, g, b).This representation is termed the Neural Radiance Field (NeRF). Using the volume rendering equation, NeRF can implicitly model a 3D scene and generate high-quality images from novel viewpoints.

1. Generation of light: Through the internal and external parameters of the camera, the pixel coordinates of a picture can be converted into coordinates under the unified world coordinate system, we can determine the origin of a coordinate system, and each pixel of a picture can be calculated according to the origin and the position of the picture pixel of the pixel relative to the origin of the unit direction vector d. According to origin o And direction t , change different depth d By building a series of discrete points, you can simulate a line of light passing through the pixel $r(t) = o + td$.The coordinates and orientation of these points are NeRF's MLP input, and the output value rendered by the volume is equivalent to the value loss of the pixel the light passed through.
2. Position embedding: Input x(x, y, z) and d(x, y, z) on a ray, output $\gamma(x), \gamma(t)$, where

   $$\gamma(p) = (sin(2^0\pi p), cos(2^0\pi p), ..., sin(2^{L-1}\pi p), cos(2^{L-1}\pi p))$$

3. MLP prediction: For each point on a ray, it is necessary to pass an MLP. it is mentioned that a ray coarse sampling 64 points, then these 64 points will pass through the MLP, that is, 64 σ will be output And then add $\gamma(d)$. Notice that for each of these 64 points they're all in the same ray, so $\gamma(d)$ for each

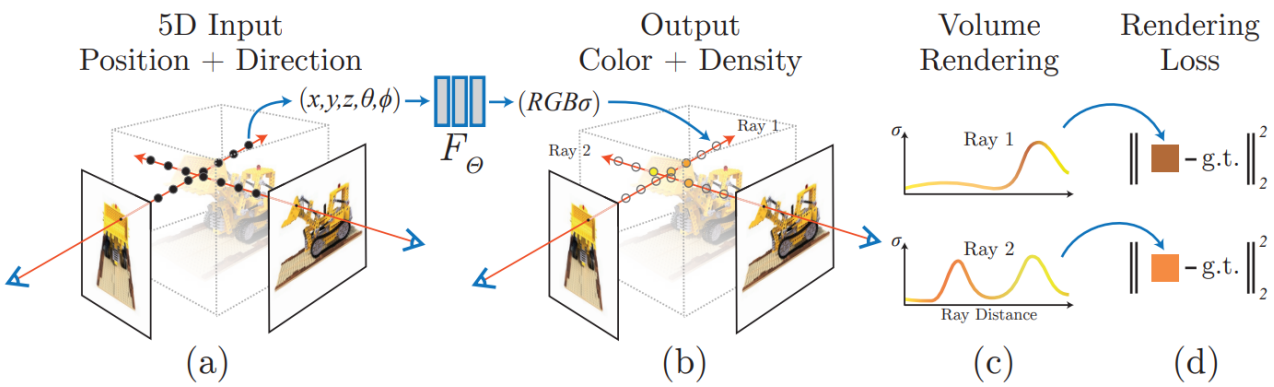


Figure 5. An overview of our neural radiance field scene representation and differentiable rendering procedure. We synthesize images by sampling 5D coordinates(location and viewing direction) along camera rays (a), feeding those locationsinto an MLP to produce a color and volume density (b), and using volume rendering techniques to composite these values into an image (c). This renderingfunction is differentiable, so we can optimize our scene representation by minimizing the residual between synthesized and ground truth observed images (d).

point it's all the same, and then you get 64 points that correspond to the predicted rgb values.

4. Volume rendering and optimization: NeRF uses volume rendering to calculate the color values along the light,The radiance along the ray is accumulated based on volume density, yielding the pixel color:

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t), d)dt$$

$$T(t) = exp(-\int_{t_n}^{t} \sigma(r(s))ds$$

we sample multiple points on the ray using predefined strategies, forming a discrete approximation that provides the expected pixel color:

$$C(r) = \sum_{i=1}^{N} T_i(1 - exp(-\delta_i\sigma_i))c_i$$

$$T_i = exp(-\sum_{j=1}^{i-1} \sigma_j\delta_j)$$

where $\delta_i = t_{i+1}t_i$ represents the distance between adjacent sample.

This is the forward process of NeRF and the overall model is shown in Figure 5.

3.2. One-2-3-45

One-2-3-45 is a general solution that transforms images of any category into a high-quality 3D textured mesh[4]. Given an image, One-2-3-45 first uses a view-conditioned 2D diffusion model, Zero123, to generate corresponding multi-view images and then lift them into 3D space. Due to inconsistent multi-view predictions in traditional reconstruction methods, it builds the 3D reconstruction module based on an SDF-based generalized neural surface reconstruction method and propose several key training strategies to achieve 360° mesh reconstruction, producing better geometry and more 3D-consistent results.

3.2.1 Neural Surface Reconstruction from Defective Multi-View Images

As shown in 6, the reconstruction module takes m source images with different poses as input. It first extracts m feature maps using a 2D feature network. Then, a 3D cost volume is constructed by projecting each 3D voxel onto m 2D feature planes and computing the variance of the corresponding features. This cost volume is processed by a sparse 3D CNN to generate a geometric volume that encodes the underlying geometry of the input shape.

To predict the SDF at any 3D point, an MLP network uses 3D coordinates and interpolated features from the geometric volume as input. Another MLP network predicts the color of the 3D point by combining 2D projection features, interpolated geometry features, and the viewing direction relative to the source image. The network predicts the mixing weights for each source view and computes the weighted sum of the projected colors. Finally, SDF-based rendering is applied to these two MLP networks for RGB and depth rendering[11].

The reconstruction model is trained on a 3D object dataset while freezing Zero123. We use a spheri-

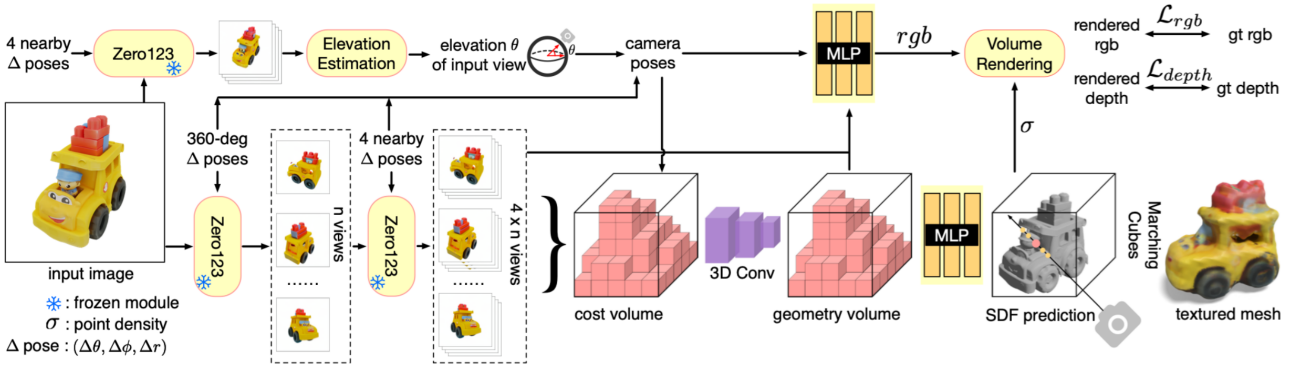


Figure 6. One-2-3-45 consists of three parts: a) Multi-view synthesis: Using the view-conditioned 2D diffusion model Zero123, the input is a single image and relative camera transformation, which is parameterized by the relative spherical coordinates $(\Delta\theta, \Delta\varphi, \Delta r)$, to generate multi-view images. b) Pose estimation: The elevation angle of the input image is estimated based on four nearby views generated by Zero123. Then, the pose of the multi-view images is obtained by combining the specified relative pose with the estimated pose of the input view. c) 3D reconstruction: The multi-view pose images are fed into an SDF-based generalizable neural surface reconstruction module for 360° mesh reconstruction.

cal camera model to normalize the training shapes and render n Ground-Truth RGB and depth images from n camera poses uniformly distributed on the sphere. For each view, Zero123 is used to predict four nearby views. During training, all $4 \times n$ predictions with Ground-Truth poses are input into the reconstruction module, and one view is randomly selected from the n Ground-Truth RGB image views as the target view, following a two-stage source view selection strategy.

This strategy effectively handles the inconsistent predictions from Zero123, enabling the reconstruction of a consistent 360° mesh. The two-stage view selection is critical to reduce the distance between camera poses, as larger distances can affect local correspondences. Ground-Truth renderings are used for depth supervision during training, while during inference, Zero123 predictions replace the Ground-Truth renderings, with no depth input required. We find that this mixed training strategy is crucial for model performance.

To export the textured mesh, we use marching cubes to extract the mesh and query the colors of the mesh vertices. Although the model is trained on a 3D dataset, it generalizes well to unseen shapes, primarily relying on local correspondences.

3.2.2 Camera Pose Estimation

Our reconstruction module requires the camera poses of $4 \times n$ source view images. Note that we use Zero123 for image synthesis, which parameterizes the camera in a standard spherical coordinate frame (θ, ϕ, r), where θ, ϕ, and r represent the pitch angle, azimuth angle, and radius, respectively. While we can adjust the azimuth angle ϕ and radius r of all source view images simultaneously, leading to rotations and scaling of the reconstructed object, this parameterization requires knowledge of the absolute pitch angle θ of a camera to determine the relative poses of all cameras in the standard XYZ coordinate system. Specifically, even if $\Delta\theta$ and $\Delta\phi$ are the same, the relative pose between a camera (θ_0, ϕ_0, r_0) and a camera $(\theta_0+\Delta\theta, \phi_0+\Delta\phi, r_0)$ will vary depending on θ_0. Therefore, changing the pitch angle of all source images (e.g., shifting up or down by 30 degrees) will result in distortions in the reconstructed shape (see example in Figure 10).

Therefore, we propose a pitch angle estimation module to infer the pitch angle of the input image. First, we use Zero123 to predict four neighboring views of the input image. Then, we enumerate all possible pitch angles in a coarse-to-fine manner. For each pitch angle candidate, we compute the camera poses corresponding to these four images and calculate the reprojection error of the camera poses to measure the consistency between the images and the camera poses. The pitch angle with the minimum reprojection error will be used to generate the camera poses for all $4 \times n$ source views by combining the poses of the input views and their relative poses. For details on how to compute the reprojection error for a set of posed images, please refer to the supplementary materials.

3.3. Code Analysis

We have reproduced the [NeurIPS 2023] official code of One-2-3-45: Any Single Image to 3D Mesh in 45 Seconds without Per-Shape Optimization, using our new datasets to finetune and test. The core structure of the code[1] is based on the main function in run.py, which defines the generation of multiview pictures and recon-

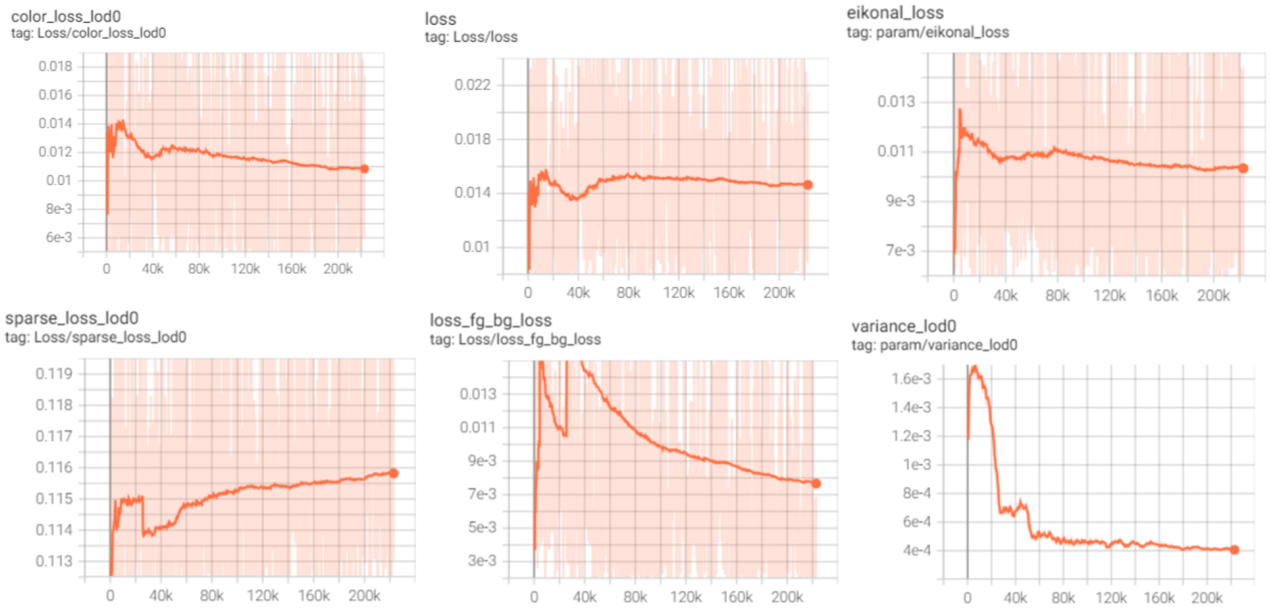


Figure 7. Training Loss Curve

struction. The inputs of main() are a single picture and other optional parameters. predict_multiview() preprocesses the input images, and then generates 32 pictures of different views. reconstruct creates the 3D models based on the generated images.

preprocess() scales the image to at most 512 × 512 and transforms it to the RGB color space with transparency. Then we utilize the pretrained large segmentation model SAM to generate the mask by sam_out_nosave(), which distinguishes between the foreground and background. Additional preprocessing steps, including contrast adjustment, value normalization and so on, are also employed. stage1_run() utilizes the pretrained large 3D reconstruction model Zero123 to generate multiview images. stage1_run() outputs views at the same elevation through predict_stage1_gradio(). stage2_run() infers 4 nearby views for an image of stage 1 to estimate the polar angle. Take the first picture produced by stage 1 for an example. We apply zero123_infer() to create the images of nearby views and predicted polar angles.

The support functions invoked above are defined in utils. sam_utils.py includes the functions of SAM such as sam_out_nosave(). It calls a more fundamental function pred_bbox() to delineate the segmentation box. Then we use predictor.predict() to generate images with segmentation masks: A four-channel(RGBA) image, where the first three channels store the original RGB image, and the fourth channel(Alpha channel) is used to store the mask result. The Alpha value of the foreground is 255, and the Alpha value of the background is 0. zero123_utils.py contains several functions about multiview images. predict_stage1_gradio() defines different angles: $delta\_x\_1\_8$ =[0,0,0,0,30,30,30,30,-30,-30,-30,-30], $delta\_y\_1\_8$ =[0, 90, 180, 270, 30, 120, 210, 300, 30, 120, 210, 300]. sample_model_batch() invokes get_learned_conditioning() to extract image features. Then we calculate angle features using degrees and trigonometric function values. Concatenate image features and angle features together and perform a projection transformation to get the condition vector cond['c_crossattn'], which consists of semantic and spatial information. Furthermore, we call model.encode_first_stage() to encode the image feature cond['c_concat']. Then, employ the DDIMSampler to gradually denoise based on the conditional features obtained before, generating a high-dimensional feature map $(n\_samples, 4, h//8, w//8)$. Finally, model.decode_first_stage() is used to restore the actual image at that angle. zero123_infer() defines two lists of rotation angles: $delta\_x\_2$ = [-10, 10, 0, 0], $delta\_y\_2$ = [0, 0, -10, 10], and uses infer_stage_2() to generate images from viewpoints near the original image.

reconstruction is the main structure of training. The main function of exp_runner_generic_blender_val.py read a series of parameters from the command line, and call different runner functions based on the mode. The difference between modes lies in the different modes of

calling forward(). train() invokes train_step(), which obtains the image feature, the light information, and the camera parameters information. Then we use the image features and the camera's projection information to generate conditional volumes and render the image using ray tracing. Compare with sample_rays to calculate the weighted sum of color loss, depth loss, sparsity loss, foreground-background loss and so on. The return value of the loss is the total loss value at two levels, a detailed set of loss values, and depth data, used for back propagation. Then call validate_mesh() to generate the 3D model in .ply format. val_step gets the feature information of the image, the light information, and the camera parameters information to create the depth map. Render the image using ray tracing, followed with validate_mesh() to generate the 3D model in .ply format.

3.4. Result Analysis

We fine-tuned the One2345 model on part of dataset over our collecting 7,000 samples, using approximately 4,500 objects for fine-tuning. Figure 7 shows the loss curve during our training process. As observed, each loss initially increases at the beginning of training. This is because our dataset focuses on artifacts, a category with limited representation in the original model's training dataset. Therefore, the initial rise in loss is expected.

It can be seen that after fluctuating for a short period, each loss curve begins to converge. Except for the sparse loss, all losses show a downward trend. The total loss stabilizes around 80K iterations and reaches its lowest point at approximately 40K iterations. Upon analysis, it is noted that due to the limited availability of artifact images and the restricted quality of the dataset, these factors may affect the fine-tuning results.

Figure 8 shows some of the rendering results. In comparison with the model without finetuning, it can be observed that our fine-tuned model achieves better performance in the reconstruction and rendering of artifact details. For instance, our model can reproduce the faded patterns on vases and jars caused by their long history, infer the overall style based on the input perspective, and reconstruct sculptures with more accurate spatial relationships.

However, for more complex objects, such as the reconstruction of portrait sculptures, the results before and after fine-tuning are both unsatisfactory. This is partly due to the low resolution of the input images and the training dataset for this category of objects. Also the effect of multi-view image generation model limits the reconstruction result.

Overall, our fine-tuning model has achieved an improved result. We hope to address the problems mentioned above in the future, which may enable better reconstruction of objects with more elements and intricate details.

References

[1] GitHub. One-2-3-45 repository, 2023. Accessed: 2025-01-16. 7

[2] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, and Weilin Zhao. Minicpm: Unveiling the potential of small language models with scalable training strategies. 2024. 1

[3] Mukul Khanna, Ram Ramrakhya, Gunjan Chhablani, Sriram Yenamandra, Theophile Gervet, Matthew Chang, Zsolt Kira, Devendra Singh Chaplot, Dhruv Batra, and Roozbeh Mottaghi. Goat-bench: A benchmark for multi-modal lifelong navigation. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 1

[4] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization, 2023. 2, 6

[5] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. 2020. 2

[6] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. CoRR, abs/2003.08934, 2020. 5

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, and Jack Clark. Learning transferable visual models from natural language supervision. 2021. 1

[8] Santhosh K. Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alex Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, and Angel X. Chang. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. 2021. 1

[9] Sethian and A. J. Fast marching methods. SIAM Review, 41(2):199–235, 1999. 2

[10] Junchi Wang and Lei Ke. Llm-seg: Bridging image segmentation and large language model reasoning. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). 1

[11] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. CoRR, abs/2106.10689, 2021. 6

[12] Fei Xia, Amir Zamir, Zhi Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. IEEE, 2018. 1

Figure 8. Comparison of Reconstruction Results

[13] Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. Navid: Video-based vlm plans the next step for vision-and-language navigation. 2024. 1, 2

Work Division

The twelve authors have the same contribution.

1. Xinyuan Xia: Distributing the job, finishing human-based method coding, reorganizing the code. Collected about 700 models, cleaned about 700 models, rendered about 300 models
2. Yidan Cui: Finishing the web scraper coding for https://3d.si.edu and https://poly.cam, collecting over 3500 models, cleaning 233 models, rendering around 1100 models.
3. Jinyu Cai: Finishing the web scraper coding for https://sketchfab.com/feed, collecting about 5000 models, cleaning about 500 models, rendering around 1100 models. Supporting the training.
4. Yuanyi Song: Conduct the whole experiment, including reading code, configuring environment, fine-tuning model and evaluating result. Help with rendering around 200 models.
5. Yuxin Zhang: Clean about 1200 models, render about 1000 models. Read and analysis the code. Configure the environment.
6. Xueying Li: Finishing GPT-based data cleansing method with Haomin Ma, helping with the rendering pipeline code, cleaning about 900 models, rendering about 600 models.
7. Yutong Huang: Collect about 800 models, clean 800 models, render about 540 models. Complete the abstract part in paper.
8. Yingjun Shang: Completing the analysis of the paper One-2-3-45 and the related report. Collecting about 1200 models, Cleaning about 900 models, rendering about 700 models.
9. QinYi Jia: Collect about 800 models, clean about 800 models, render about 900 models, complete the part about Nerf in paper.
10. Haomin Ma: Clean about 2700 models, render about 1200 models. Complete the part about GPT-based Method of data cleansing in paper.
11. Xiang Liao: Finish the coding for translation of descriptions, finish the coding for integration of all the cleaned JSON files, clean about 300 models, render about 300 models.
12. Xingzhi Zhong: Clean about 800 models, render about 900 models, complete the introduction part in paper.