Speech Driven 3D Modeling

A Major Qualifying Project (MQP) Report
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements
for the Degree of Bachelor of Science in

Electrical and Computer Engineering,

By:

Ryan Antes
Jacob Hand

Project Advisors:

Dr. Ziming Zhang

Date: April 2024

Abstract

This project introduces an innovative model that transforms textual prompts into detailed 3D meshes. Utilizing advanced Neural Radiance Field (NeRF) techniques alongside a new corner view image generation model, our project seamlessly converts verbal descriptions into 3D forms. This simplification of the modeling process enhances accessibility for everyone, including individuals with disabilities.

At the heart of this innovation is the integration of speech input with our newly developed text-to-3D model, facilitating a direct transition from textual descriptions to 3D meshes. This breakthrough has immediate applications in educational settings, providing visually impaired students with the means to physically interact with and understand complex subjects through 3D printed models. Extending beyond educational applications, this technology can significantly impact virtual reality and various forms of media design, areas where 3D modeling is becoming increasingly essential. Moreover, this tool broadens the inclusion of diverse groups in 3D design, pushing the boundaries of traditional 3D modeling.

Please find the code to our project at:

https://github.com/JHand11/Speech-Driven-3D-Modeling

Acknowledgements

Our team would like to thank Professor Ziming Zhang for his support and guidance throughout the entirety of this project. We would also like to thank the WPI department of Electrical and Computer Engineering for access to resources and further support along the way.

Contents

List of Figures

List of Tables

Authorship

Section	Author(s)	Editor(s)
1. Introduction	All	All
2. Background		All
2.1 Instant 3D	Ryan Antes	All
2.2 Dreamfusion	Ryan Antes	All
2.3 Shap-E	Ryan Antes	All
3. Methodology		All
3.1 Stage One: Image Grid Generation	Jacob Hand	All
3.2 Stage Two: 3D Mesh Reconstruction	Jacob Hand	All
4. Results and Discussion		All
4.1 4 View Image Generation	All	All
4.2 3D Model Generation	Jacob Hand	All
4.3 Model Comparison	Jacob Hand	All
4.4 Future Work	All	All
5. Conclusion	Ryan Antes	All

1 Introduction

The intersection of text-driven 3D modeling and Artificial Intelligence (AI) technology allows for an impressive future in engineering and computer science. Recently, there has been a rapid acceleration in research and development areas hoping to realize the vision of "automatic" 3D generation from textual prompts. This increase in interest has attracted input from numerous major companies, each trying to build the most advanced and efficient models in this area.

The applications of 3D models cover many industries and fields, ranging from 3D printing and video game design to architectural visualization and interactive reality experiences. As technology continues to evolve, 3D modeling is becoming effective and important in various sectors. In STEM (Science, Technology, Engineering, and Mathematics) curriculums, 3D models are becoming increasingly popular.

In recent years, there has been an effort to integrate STEM material into educational programming at every level, from elementary schools to colleges. This act aims to enact critical thinking, problem-solving skills, and creativity among students by involving them in hands-on, experimental learning. 3D modeling plays a pivotal role in this educational shift giving teachers and professors a powerful tool to visualize abstract concepts, explore challenging content, and engage in live learning experiences.

In science education, 3D models can depict molecular structures, astronomical phenomena, and biological organisms, allowing students to explore these concepts in depth and better understand our world. In math, 3D models facilitate the visualization of geometric shapes, spatial relationships, and math concepts, making abstract theories more tangible and accessible to students. In engineering and technology education, 3D modeling allows students to make prototypes, simulate real-world scenarios, and experiment with engineering principles in a simulated environment.

The array of immersive technologies, like augmented realities (AR) and virtual reality (VR), enhances the educational potential of 3D modeling by providing immersive, interactive learning experiences. With AR and VR, students can explore virtual environments, interact with 3D models in real-time, and collaborate with peers in collaborative spaces, pushing the limits of traditional classroom settings.

The ability to interact with and visualize objects plays a crucial role in understanding concepts. This can create a significant barrier for those with visual impairments. The education system relies heavily on visual ability. Within the STEM fields, this is even more essential. The challenge is allowing visually impaired individuals the ability to interact with objects in a different way to gain understanding.

Recognizing this challenge, our initial idea was to explore innovative solutions that could bridge the gap between visual and tactile learning experiences. Inspired by the concept of electronic tactile pin arrays, familiar to many as toys that allow the creation of shapes by pressing into a grid of needles, we envisioned a more advanced solution. We considered the potential of creating shape-shifting objects but eventually focused on a more practical approach: leveraging recent advancements in 2D image generation to produce 3D models.

The objective was to enable the creation of mesh files for 3D printing, thereby providing visually impaired learners with a tangible way to explore and understand educational content without the need to directly create the 3D models themselves. For example, a teacher when preparing their lesson plans could use this to quickly create and 3D print models for demonstrations and learning aids as they saw fit. There is no precursor of needing to deeply understand 3D modeling software and spend hours working on one model. Within 2-3 minutes they would have a model ready to be printed and used the next day in class.

In this paper we produce a model that can produce a 3D mesh file from a simple text prompt. This model can generate from very diverse prompts without categorical definition. It is built upon a large-scale 2D image generation model in combination with a 3D reconstruction model (NeRF).

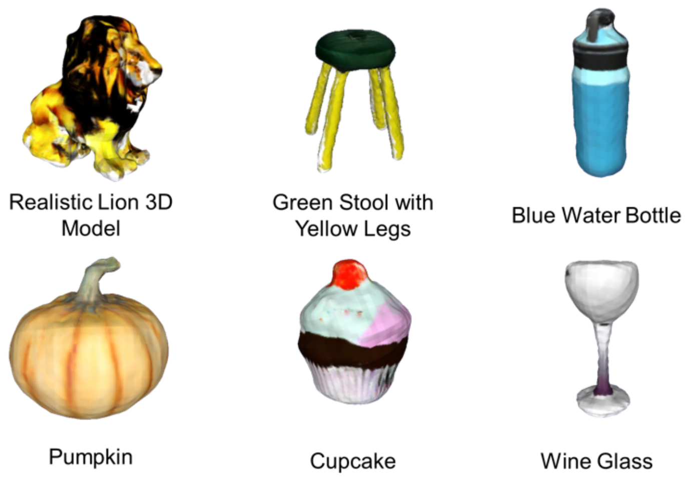


Figure 1: Model results from our model with prompts shown below

It is important to recognize the great impact such technology could have on access within educational environments. By successfully connecting the gap between visual and tactile learning, text-driven 3D modeling allows the potential to revolutionize accessibility for young people with visual impairments. By being freed from limitations, these young people would be able to independently explore and understand difficult STEM concepts in a way that was previously inaccessible.

The benefits reach further beyond the visually impaired community. All young people can benefit from the incorporation of text to 3D modeling into the educational curriculum. By presenting problems in different ways teachers can better help students understand complex issues. The use of text-3D modeling in the classroom can foster further interest in STEM fields for all students as well, especially if there are objects that students only have ever seen in a 2D space. By giving them the capability to view objects in different dimensions the possibilities are endless for exploration. The implications of this technology can extend to many other fields within the industry. Occupations such as design, gaming, and architecture can leverage text-3D modeling to quicken the prototyping and visualization cycle, reducing the costs commonly associated with 3D design tools and enhancing the innovation timeline. The integration of AI technology and an emphasis on accessibility has sparked a shift towards a more flexible and inclusive educational environment. We have the opportunity to reshape the landscape of learning with text-3D modeling, allowing people of all ages and abilities to interact with and understand complex concepts in innovative ways.

2 Background

2.1 Impact on Accessibility

This project was created and developed with accessibility in mind. The idea came from the discussion on the great impact 3D objects can have on the education process for visually impaired children. Here we share some background on this pursuit and the provide evidence for the impacts it can have on different types of individuals.

2.1.1 Education with Visually Impaired Students

Visually Impaired Students suffer from multiple disadvantages in the education system. The main one is the deprivation of a full scale of sensory data. Audial and Tactile data can only provide so much in the current education system through tools like audiobooks or braille. Yes, a subject or diagram can be described and interpreted, but what about objects that may no longer exist physically or audial descriptions just simply don't do them justice. An article written by Dr Radhika Kapur discusses in depth the challenges faced by visually impaired learners. They look into the large gap that is created when a student cannot understand concepts such as figures or objects that typically largely rely on visual ability to understand. They recommend the use of tactile materials such as braille images or diagrams[2]. This idea that visual access can be made up for through physical or tactile representations further supports the use case for our work. Creating objects as diverse as the imagination of students and educators is possible with our model in combination with 3D printing, which offers a whole new level of sensory input for visually impaired students.

2.1.2 Simplifying 3D Model Creation

The use of 3D modeling software has typically been closed off to most of the population through a few barriers. Most of the existing software is quite expensive for the average individual. A few examples of these prices are 3DS Max $1,505 per year, Revit $2,250 per year[3]. There are examples of free software that can bypass this issue including Blender. Having enough experience to use the software effectively remains another major barrier. Working with this type of software at the production level requires a great deal of experience and practice. These two major points prevent many from being able to realize the benefits of 3D modeling. This is where our model can streamline the process for many. There is no experience necessary for using our model and it is completely open source and free.

2.2 Review of Current Models

In computer-generated content, the introduction of artificial intelligence technology, especially deep learning models, has led to a creative boom. A branch of artificial intelligence called generative AI has taken a large step in the area of 3D modeling, offering new possibilities for automated content production. Although it is still in its early phases, the creation of models that can translate written specifications into complex three-dimensional models is an incredible development.

The field of generative 3D modeling is changing quickly as scientists and engineers look to produce models that improve on the previous. This field-wide innovation propels the creation of great solutions. However, there are few models that can quickly, cheaply, and effectively create 3D models from textual inputs, despite the advances.

In order to tackle this obstacle, our project began with a thorough investigation of current models and approaches within the field of text-driven 3D modeling. Using information from several carefully examined sources including well-known projects like Shap-E, Dreamfusion, and Instant 3D we carried out research to understand the capabilities of existing models.

Understanding the current models and strategies, we developed a strategy to address some of the difficulties in this area of modeling. Our approach made use of advanced deep-learning models as well as existing frameworks for a more complete final result.

Our project aims to advance text-driven 3D modeling by combining knowledge from previous studies, utilizing AI technology, and taking an interdisciplinary and cooperative approach. We hope that our work will open up new avenues for artistic expression, expedite the process of creating material, and provide a wider audience with more equitable access to 3D modeling tools.

2.2.1 Instant 3D

We leveraged the "Instant3D: Fast Text-to-3D with Sparse-View Generation and Large Reconstruction Model"[4] paper to support our project of creating a speech-to-3D model. The Instant 3D paper was an important resource in aiding our project because it is one of the few published papers examining the complexities of creating a true text-to-3D model. The instant 3D paper shows a groundbreaking approach to converting textual information into three-dimensional models. Its central idea is the incorporation of sparse-view generation and a large-scale reconstruction model.

How these new techniques were presented was a major part of the Instant 3D paper and the complicated text-to-3D conversion provides valuable solutions for researchers to tackle new problems. Instant 3D emphasizes speed, accuracy, and scalability and by successfully implementing these techniques it has cemented its place as an important work in the realm of 3D model generation from textual data. The paper's primary contribution is its ability to generate 3D models rapidly while maintaining a high level of accuracy. The integration of sparse-view generation lets the model optimize resources without compromising the quality of the output model.

The use of a very large database and reconstruction model ensures that the finer details are captured, bettering the overall quality of the 3D-generated models. These key methodologies have set a benchmark for future research in the field of 3D model generation, with the potential to influence subsequent projects such as ours which seeks to harness the power of textual information for 3D model creation. Our project hinged on the development of a speech-to-3D model, and leveraged support in the principles and techniques used by Instant 3D.

By using the details from the Instant 3D paper, we were able to enhance our text-3D conversion. The similarities between the challenges addressed in the Instant 3D paper and those encountered in our project reinforce the significance of using proven methodologies to overcome challenges associated with projects in this area. With inspiration from Instant 3D, our project incorporated similar ideas to speed up the conversion process and improve the accuracy of the generated 3D models. Applying these similar approaches is key to further understanding and development within these types of models.

The Instant-3D paper is a groundbreaking publication that has the opportunity to inspire innovation for years to come.

2.2.2 Dreamfusion

In the pursuit of advancing 3D model generation from textual data, Dreamfusion[5] is a very important factor. Dreamfusion is a latent diffusion model, that diffuses in a latent space rather than the original image space. This literature review explores the innovative concepts that are introduced by Instant 3D and considers the paper's implementation of Dreamfusion for 3D image generation. This literature review aims to discern the key ideas derived from Instant 3D and examine how these concepts have been used with Dreamfusion.

Dreamfusion, with its aim to advance 3D model generation, uses the principles and techniques from the Instant 3D paper. Our project recognizes the significance of Instant 3D's approach and its potential to address challenges associated with converting spoken and textual information into 3D models. The similarities between the challenges addressed in the Instant 3D paper and those encountered in Dreamfusion underscore the importance of using tried methodologies to overcome challenges in diverse data conversion.

Instant 3D's approach, which focuses on sparse-view generation and a large-scale reconstruction model, finds resonance in Dreamfusion's efforts to make the text-to-3D conversion process easier. The Dreamfusion method can incorporate similar ideas to enhance the efficiency and accuracy of the generated 3D models, and we used similar ideas to build our very own model. By incorporating Instant 3D's concepts into Dreamfusion, it not only validates the efficacy of the former's approach but also opens lanes for more exploration and improvement in text-to-3D conversion.

Dreamfusion's utilization of proven methods taken from Instant 3D emphasizes that continuous research drives innovation. To conclude, Instant 3D has left an indelible mark on the landscape of 3D model generation from text data. The Dreamfusion method derived in part from Instant 3D shows the ongoing influence and relevance of the seminal work in text to 3D generation. The collaboration between innovative projects like Dreamfusion and Instant 3D leads the way for continued advancements in the ever-changing field of text-to-3D conversion, inspiring future research and innovation for the foreseeable future.

2.2.3 Shap-E

Shape-E[6] is the latest frontier of conditional generative models for 3D assets. Shap-E differentiates itself from newer work in 3D generative models by emphatically generating parameters of implicit functions. Different from models that produce a single output representation, Shap-E's approach allows for the generation of textured meshes and neural radiance fields. The training of Shap-E unfolds in two stages: first, an encoder is trained to determine the map of 3D assets needed to make the parameters of an implicit function. Then the training of a conditional diffusion model on the outputs of the encoder.

The utilization of a large dataset featuring paired 3D and text data lets Shap-E generate complicated and diverse 3D assets quickly. Open AI has two models in this area. Shap-E and Point-E Shap-E has the ability to generate 3D meshes while Point-E generates simple point clouds. They are both effective models and reference and strategy can be inferred from both.

Shap-E's direct contribution lies in its ability to directly generate implicit parameters, enabling the rendering of textured meshes and neural radiance fields. This cutting-edge approach adds a layer of versatility to 3D asset generation, allowing for different representations with efficiency. The two-stage training process, which involves an encoder and a conditional diffusion model, reflects a strategy that contributes to Shap-E's quick convergence and noteworthy sample quality. Shap-E's performance comparison with Point-E highlights its capability and effectiveness, showcasing the potential for improvements in multi-representative output spaces. The rollout of model weights, inference code, and samples underscores Shap-E's mission of fostering collaboration and technical breakthroughs in the field giving resources and practitioners valuable resources to advance model generation. In the malleable space of 3D models, Shap-E is now a notable player, challenging the boundaries of what is achievable in terms of speed, quality, and diversity in 3D asset generation.

3 Methodology

Our model adopts a comprehensive two-stage approach to address the intricate challenge of changing textual prompts into 3D models. In figure 2 we have a graphic that shows the basic flow of our methodology. To start, we use a complex image grid generation process, which involves the arrangement of multiple images to show various perspectives of the intended object. This first stage sets the foundation for the following 3D reconstruction stage.

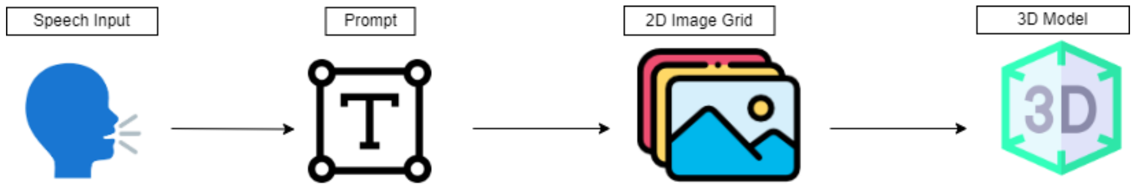


Figure 2: Methodology: Speech Input-Text Prompt-2D Image Grid- Output 3D Model

After the image grid generation, we start on the refinement of a Neural Radiance Field (NeRF) generator, a deep learning architecture known for its ability to reconstruct detailed 3D scenes from little input data. In this phase, we mold the One-2-3-45[1] NeRF model to our specific application, fine-tuning its parameters and architecture to optimize performance.

By integrating these two stages, our strategy of a combination of image-processing techniques and advanced neural network architectures. This approach enhances the accuracy of 3D model generation and ensures application across many textual inputs. Through experimentation and refinement, we aim to push the bounds of text-driven 3D modeling and pave the way for advancements in this ever-growing field.

3.1 Stage One: Image Grid Generation

The first stage was to take a text prompt and create an array of images taken from different viewpoints around an object. The Instant3D paper[4] references the ability to finetune the

SDXL[7] model with this capability.

The first step in finetuning the SDXL model for our purpose was to create the training data. For this we needed to render images of objects within the objaverse dataset[8]. The Objaverse dataset was used as it was the largest available dataset with caption to model pairs. We were able to manipulate code from the “Objaverse-rendering-main” repository on github[9]. This code was designed to render objaverse objects in Blender and save pictures around the objects. The script was run on the entire dataset in order to prepare all the data for relevant filtering and training. This produced 4 512x512 image files at 90-degree angles around each object from a viewing angle of 20 degrees looking down towards the center.

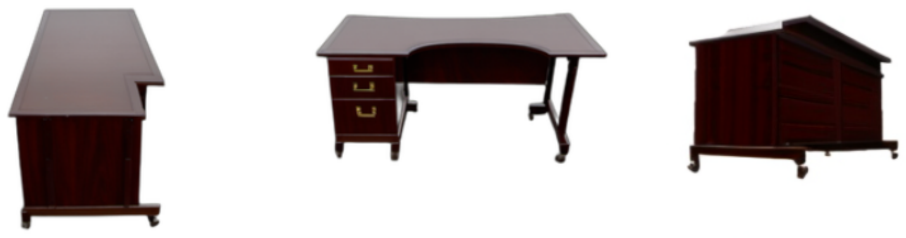
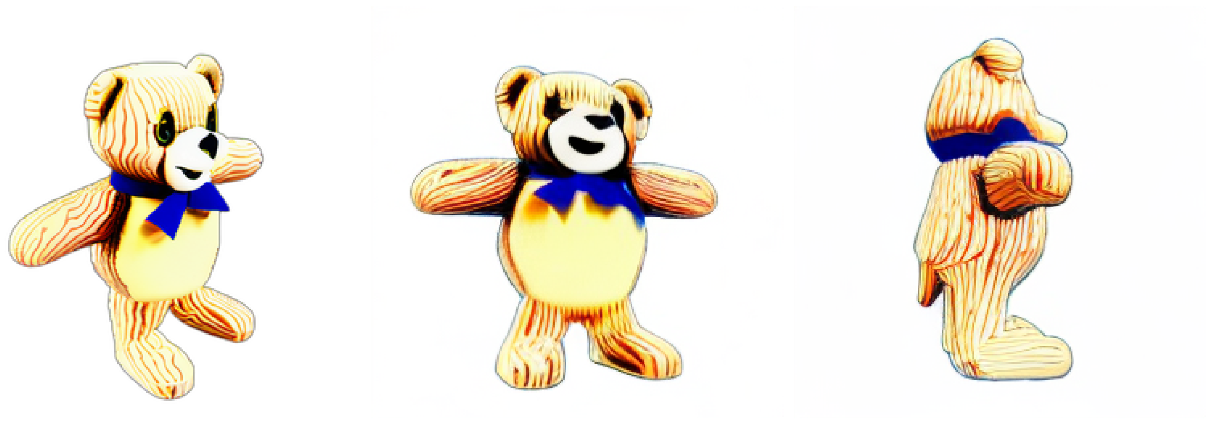


Figure 3: Ground truth versus inferred generation example. Left: True Front View, Middle: True Right View, Right: Inferred Right Corner View

Our major change in this area was taking the images at 45-degree offsets instead of the 4 azimuth angles. This provided more detail for the second stage of the model. Pictures from the azimuth can leave much up to the creativity of the generation model because it will largely only give information on one side of the object. This type of result can be seen in figure 3 as the inferred view from the corner is much different than the truth on the other side. The model will create its own ground truth when provided with little inference. When pictures are taken at the “corners”, the 2 sides are now “understood” by the generation model. This type of result can be seen in figure 4. The corner view leads to two more views that are highly accurate to the ”truth”. This is so important because in order to produce a good quality mesh the images have to be both clear and highly similar in the different views that overlap.



(a) Corner Generated View (b) Inferred Front View (c) Inferred Left View

Figure 4: Example of Corner View leading to consistent side views

The objaverse[8] dataset while extremely useful has the issue of containing noisy data. There are many objects within the set that would harm the training we were trying to perform. In order to classify images for better curation of training data we used ResNet50[10] feature extraction. The details behind ResNet50 can be found in appendix ??. The product of feature extraction was an array of features produced from the deep learning network tied to each object id.

The next step comes from Instant 3D[4] procedure where we manually labeled 2000 random objects as good or bad for our classification training data. This was completed through the use of a simple python script to display and allow labeling input stored into a csv tying object identification to label. Good objects had a combination of both consistent and complex geometry. There were also 3D files in the dataset that were very 2D-looking or messy in their modeling. This classification also allowed us to filter out those objects.

An SVM was then ran on the 2000 labeled objects as training data. This trained model was then applied to all of the objects using their extracted features as input. Scoring was applied through the probability output of the model and we took the top 20,000 probability scores for being “good” objects for SDXL[7] finetuning.

Once the 20,000 objects were selected for training the 4 images of each object were appended into a single 1024x1024 grid image. This was done through a simple python script on only the 20,000. Within this script the object labels (from the caption 3D objaverse file[11]) were matched through object ids so the resulting grid images were named with the object caption.

With the data prepared we used the built in Finetuning feature described through the SDXL[7] github page. This training took 2 days on 2 NVIDIA RTX A5000 GPUs. The output of this training was a model ckpt that could be used to generate image grids shown in figure6. At this point there was now a functional model to go from text to 4 views around an object that started to create a 3D understanding of the generated scene.

As seen some prompts and generations perform better than others. The important part was that the model was able to generate 4 views around a mostly consistent object. This would create the input for our second stage of the design.

3.2 Stage 2: 3D Mesh Reconstruction

In the second phase of our project, we began on the utilization of a cutting-edge Neural Radiance Field (NeRF) model to make complex 3D models from a limited set of input images. Delving into the area of existing models within this domain, we encountered the ”One-2-3-4-5 Project”[1] discussed further in Appendix B. This project characterized by its innovative approach, revolves around elevating a single image into a comprehensive 3D environment.

A common challenge encountered in the area of generating 3D models from restricted input data is what is referred to as the Janus Problem. This issue discusses the challenge of correctly reconstructing unseen regions of the model based only on input data.

To illustrate, considering a picture of a person captured from the front might result in a reconstructed model with a second, unanticipated face on the back of their head due to the lack of information regarding the obscured regions. Our method avoids this challenge by using an input consisting of four images uniformly captured from different angles around the object. This mitigates the Janus Problem and improves the overall generation process by providing a more complete set of visual data. A result of this 4 view input implementation can be seen in figure 5. The single view input model infers certain views with extra or missing parts of the model, where the 4 view input generally is much more accurate to the ”truth” of the object. It is generally in objects like humanoids, animals, and geometrically complex objects, where our model is much more effective in consistency.

The NeRF model implementation starts with two stages of image generation. The first stage uses a finetuned Zero123 model[12] to infer a new image of the object from the initial four input images, rotating the object at 30-degree increments around both the x and y axes. We use the pretrained Zero123 model checkpoint from the One-2-3-45[1] project. This is trained through pairs of images with the camera transformation information. This idea is similar to the SDXL finetuning principle where the 2D generation model has the ability to generate views around one consistent object. This produces eight images, significantly increasing the available information about the object's appearance. Next, to enhance the data set even more, the second stage generates an additional four images by shifting the object picture by 15 degrees around the primary image. This new set of images is then projected to onto the 3D plane, using estimations of the various camera poses. Finally, a Convolutional Neural Network (CNN) is used to extract the intricate shape mesh from this visual data, This is inspired by the original SparseNeus project[13].

In essence, our changed approach to text-driven 3D modeling not only addresses the inherent challenges posed by limited input data but also accelerates the generation process to new heights of accuracy and intricacy. Through the integration of updated neural network architectures and careful data processing techniques, our project has unlocked the potential

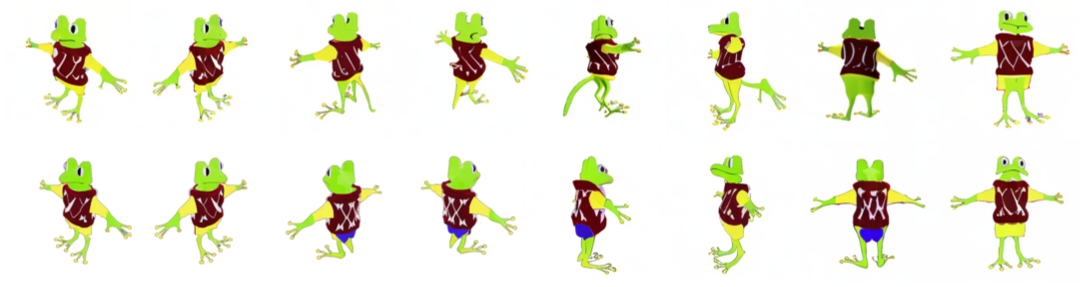


Figure 5: A comparison of the 8 Base images. One-2-3-45 Base model results on top. Our Model Below.

to make detailed and realistic 3D models from textual descriptions, heralding a new era of innovation in the field of computer-generated models.

4 Results and Discussion

4.1 4 View Image Generation

Through the development of our project, we ran the SDXL[7] finetuning model 3 times, each time improving certain aspects that we found were lacking in the previous version. The first training produced good results however we did not curate input data and simply took the first 10,000 objects from the dataset. We also did not remove backgrounds, so some of these generations had full backgrounds or were undesirable colors for the next stage generation. We found these results to be more cartoonish and have less consistency as the objaverse dataset[8] contains many objects that are not desirable for this type of model. These can be grouped as objects that are either too simple or not useful for the type of generation we desire.

Some results from our initial training phase are shown in figure 6. These images show four different views and offer insight into the initial capabilities of the model and highlight areas for improvement in the different methods we used. It was clear that these images still generally contained blurry background and were prone to mistakes in correctly generating all 4 of the azimuth views. Despite the limitations mentioned above, these results gave a foundational understanding of the model's performance and guided our changes moving forward. In our other model training attempts, we recognized these issues and implemented a stricter data validation process to ensure the quality of training images was at a higher level. By fine-tuning our data set to include objects more conducive to our required image generation outcomes, we tried to heighten the model's ability to construct visually appealing and contextually relevant images and models.

In the second stage of SDXL[7] finetuning we applied the changes as discussed in Methodology Stage One. With this curated input dataset our output results were much

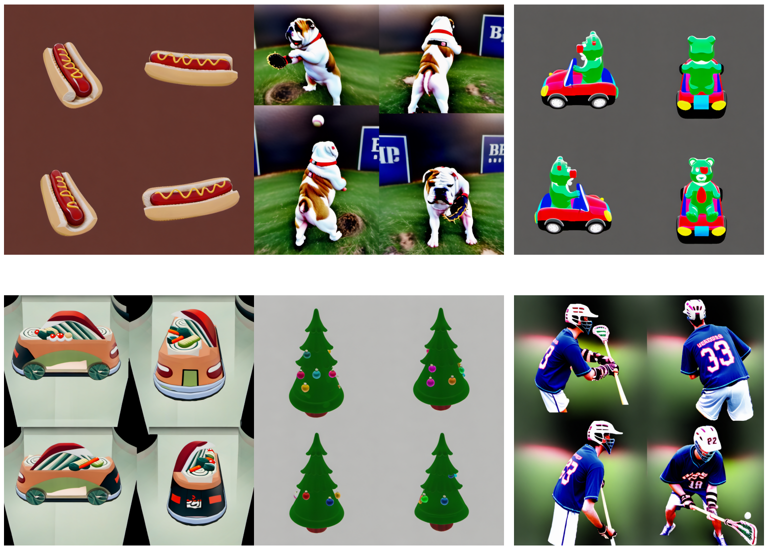


Figure 6: Examples of Original First Stage Generation Results.

more clear and consistent.

The results shown in figure 7 were much more promising and were generally good enough quality to be input to the One-2-3-45[1] 3D generation model. As seen in the figure background's were consistently blank and the views were clearly showing in 90 degree increments.

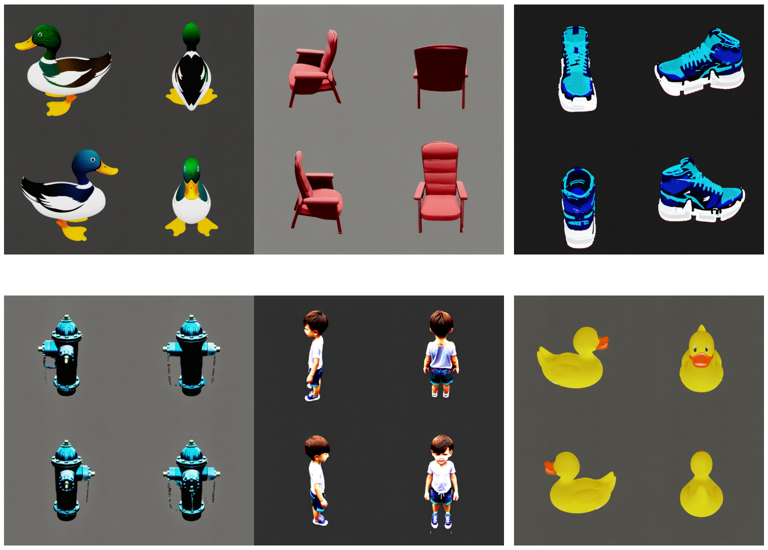


Figure 7: Examples of Adjusted First Stage Generation Results.

4.2 3D Model Generation

In the original stage of the 3D reconstruction we just used the base One-2-3-45[1] model. We would generate a model for each of the 4 views with some success. Some of these meshes produced from the objects in figure 7 can be seen in figure 8. From these results it was clear that one picture of an object can only provide so much detail for a regeneration model. The model could predict and perform with simple objects but with the more complex the model could not quite understand what it could not directly visualize.

Some examples of typical errors can be seen in figure 8. For the rubber duck model, we see good generation on the side that was fed to the model, but the other side is flat and lacks the correct geometry. On the chair, we see good shape and color in the model. However, The arm that is opposite to the input image bows out and does not align correctly.

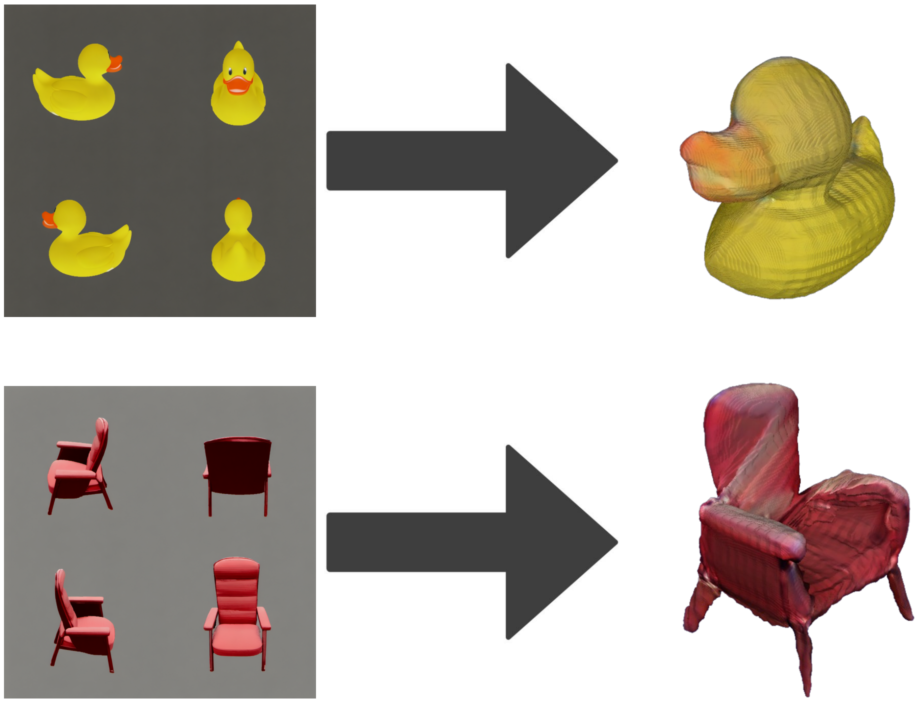


Figure 8: Examples of Original 3D Mesh generation results from first stage grids.

This is where we decided the 3D generation strategy needed to be adjusted. We rewrote the One-2-3-45[1] code to take our 4 images as input and generate off of these to create the output mesh. With these 4 input images the model would have clear information detailing all sides and angles of the object. The results of this new model can be seen in figure 9.

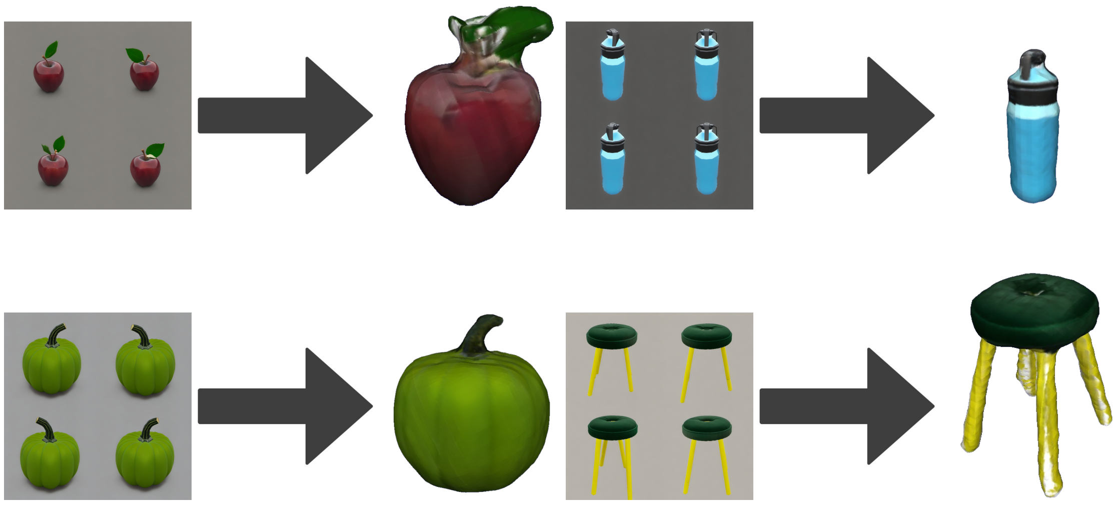


Figure 9: Examples of Models produced from the 4 View input implementation.

This stage is where we were able to produce passable results from our original requirements. We were still interested in upgrading the model for better quality and higher consistency. This is where we implemented the Corner input generation as seen through the grids seen in figure 10

The strategy of 2D 4 corner view generation combined with using all 4 of these images as input to our newly innovated NeRF model is where the best results so far were seen. We were solving some of the problems we set out to remedy in earlier devlopments.

4.3 Model Comparison

To augment the depth of our model evaluation, we conducted a comprehensive analysis of the intricate nuances in the generated 3D models. These comparisons can be seen in figure 11. The first of which being against the base One-2-3-4-5 model [1]. This involved a microscopic examination of various attributes such as the smoothness of the surface, geometric

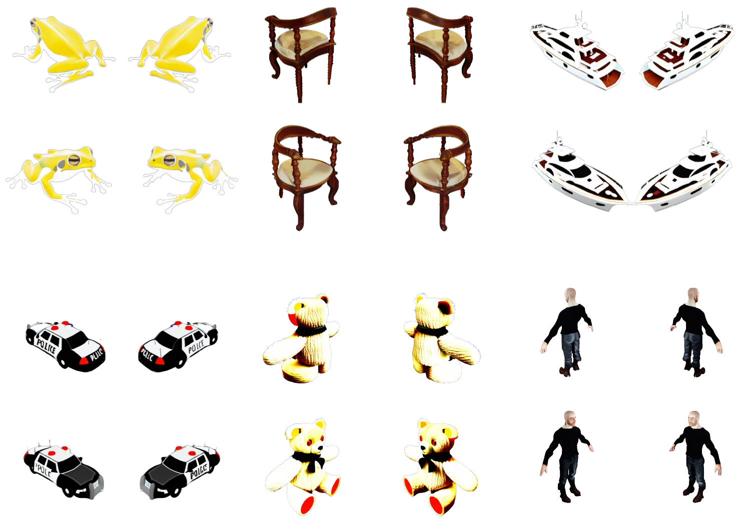


Figure 10: Examples of Grids Produced From the Corner View SDXL Checkpoint.

accuracy, and structural fidelity. By subjecting the models to rigorous examination across these dimensions, we aim to gain a complete understanding of their perceptual quality and functional efficacy.

Furthermore, to provide a more holistic perspective on the performance of our model, we used feedback feedback from people within the educational community. Their insights and help, informed by real-world experience, served to enrich our evaluation criteria and shed light upon the real-world applicability of the generated 3D models in educational contexts.

In addition to qualitative assessments, we used quantitative metrics to evaluate the performance of our model objectively. Metrics such as Intersection over Union (IoU), Mean Squared Error (MSE), and Structural Similarity Index (SSI) were used to assess the geometric accuracy, semantic segmentation fidelity, and perceptual similarity of the generated 3D models compared to reference models.

By integrating both qualitative and quantitative evaluation methodologies, we tried to offer a total and multi-faceted assessment of our text-driven 3D modeling approach. This evaluation framework not only provides a complex understanding of the capabilities and limitations of our model but also facilitates targeted refinement and optimization efforts to further improve its metrics and utility across diverse applications and domains.

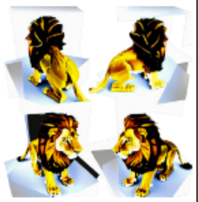
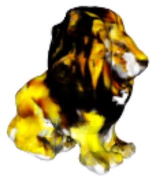
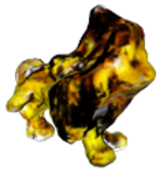
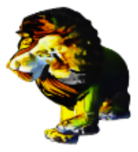
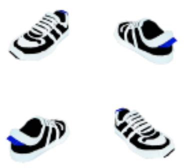
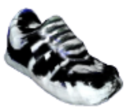
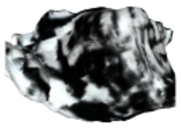
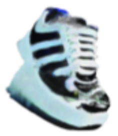
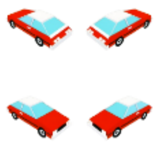
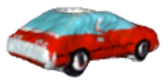
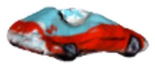
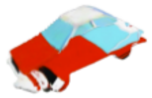
Input	Resulting Image Grid	Our Model	One-2-3-45 (Using One Image From Grid)	Meshy.ai (Using One Image From Grid)
"Realistic Lion 3D Model"				
"Realistic Sneaker 3D Model"				
"Realistic Simple Car 3D Model"				

Figure 11: Comparison of Models generated by our model vs the One-2-3-45 model

4.4 Challenges During Project Work

The development of this project went through many stages, and there were many obstacles to overcome in order to produce a working model up to the pre-determined standards.

4.4.1 Multi-View Generation

The process of 2D image generation models being adapted to multi-view was detailed in previous literature. When curating the input data for training we ran into multiple problems involving the rendering and capturing of 3D mesh files. We used a python script on the remote Linux Machine in the VISLAB. When working thorugh a remote machine it requires a certain setup to allow rendering in the headless environment. When we began running the script we were failing in getting the blender application to load in each iteration because it was not able to open headlessly. We originally tried to use screens which create a standalone terminal session that lasts after the local machine disconnects from the remote. This allowed continual running of our script but screens still do not provide the graphical interface needed for the rendering task. Our solution was using an X server in combination with a terminal screen. We used the Xvfb extension to run the rendering script. Xvfb is an X server implentation that runs the graphical processes on virtual memory with no screen output.

With the script now rendering objects we had to balance speed and processing capability. While trying to maximize GPU capability we would often run out of space on the two GPUs we had access to on the machine. We settled on a final number of 6 workers on each GPU and were able to successfully render the 4 desired views of each of the objects in the Objaverse dataset[8].

4.4.2 3D Reconstruction

With the base model from the multi-view generation stage. We knew we needed a fast reconstruction model that used a small number of views to generate a 3D mesh. We went through experimentation with projects like pixelNeRF[14], however we were not able to adapt the projects to our specific implementation. Many of these NeRF projects were Single Image to 3D model and used various algorithms that did not allow for the adaptation of a multi-view input.

We then discovered the One-2-3-45[1] project which through reading through their paper and published code found evidence for the ability for adaptation and innovation. With a project as complex as this our first barrier was fully understanding the code behind the project. This took a lot of research and analysis work on their code base. We were able to target a few areas for change involving the original processing of images. This is where we made the discovery through failed generations, that if we began with 4 corner views we believed the following images for input to the NeRF would be much more accurate.

4.4.3 Current Model Challenges

We created an end to end text to 3D model process as was our goal. We also were able to generate very promising and successful results with this framework. With these successful generations, we also have had many problem cases and failures. There are a few specific areas where we do not see great success with our process.

When objects are majorly white in color, the model struggles to reconstruct any type of accurate mesh. Many of these white areas are often left as empty space. This is because the background of these images is white and the model understands this color as the empty space when shaping the mesh. A result where we see a specific instance of this can be seen in figure 12.

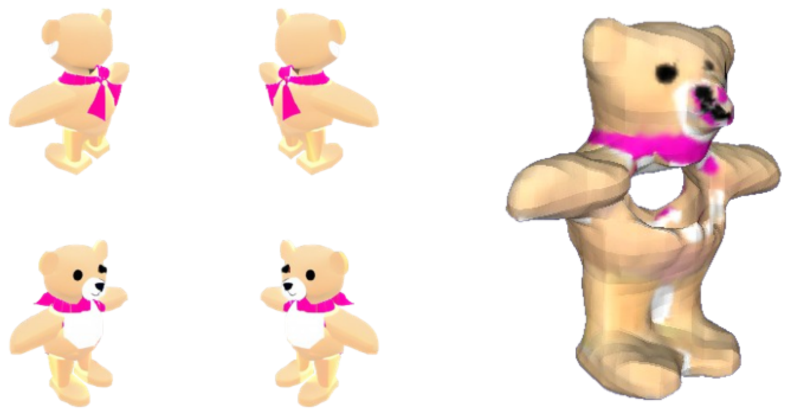


Figure 12: A Teddy Bear Model with a major defect in the white area of its chest

There are some instances with much more complex objects where the multiview generation is inconsistent. Typically when we see this error there is one view that is repeated twice. Some examples of this type of error can be seen in figure 13. This leads to poor rendering as there is incorrect information being fed to the second stage that it cannot directly fix. These types of generation models are never going to be perfect. However, it is about training them to be as accurate as possible. To attempt to fix this problem we would have to conduct a data review and look further into different training strategies to improve consistency.

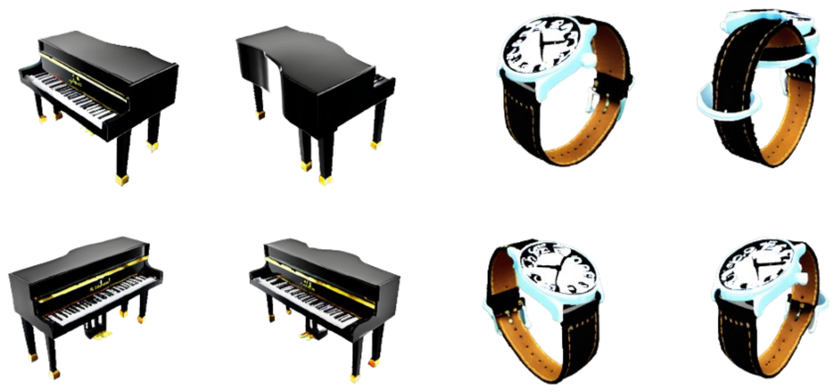


Figure 13: A Grid Result of a Watch and a Piano, both with one view repeated

Through our experimentation we have found that prompting is important for successful generation. The 2D image generation model is trained on a highly diverse dataset, and as such is extremely creative. We have found that being more specific in the desired model is much more effective. For example, when running most of our prompts we found the strategy of using the format ””Realistic” ”object description” ”3D Model””. When these two terms are left out there can be generations that are slightly two-dimensional or cartoon-like. This is something we would want to remedy in future work because in application it is not desirable to have to use specific keywords when prompting the model.

One issue that we currently run into is that with our GPU capacity, we cannot use a high-level mesh resolution. The framework of the reconstruction model is built to have an adjustable resolution, but our hardware only allows us to go up to a resolution of about

700. Simple objects can be represented accurately with this model, but objects with complex geometries are more difficult to represent, so typically these surfaces are smoothed.

4.5 Future Work

This project was developed through the exploration of many different options in both of the two stages of generation. In this section we discuss what we see as further areas for improvement and innovation in relation to this area of work.

When developing our image grid generation model, we used the Objaverse 1.0 dataset [8]. When looking at areas to improve we discovered the Objaverse XL dataset [15]. The Objaverse XL dataset is over 10 times larger than the 1.0 set. Using our strategy to curate the 1.0 data, we would also be able to filter the XL dataset down to its best objects. We see this as having higher quality of data which would lead to increased accuracy and ability in the image grid generation. Through our work and experiments with the model when the second stage is fed clean data in the 4 corner images, it is highly successful at constructing an accurate and consistent 3D mesh.

This work has brought up the topic if it is possible to directly model from text to 3D models. Currently, the projects in this area majorly use a 2 stage approach with the first doing some form of image generation. Text straight to 3D would involve a model that could map the textual information to planes and vertices in the 3-dimensional space. With the expansion of Artificial Intelligence and Machine Learning, it is not out of the picture for a model like this to be developed. The major missing piece right now is the available data for a model like this. For example the SDXL model [7] was trained on over 100 million text image pairs. The text to 3D dataset would need to be created and filtered in order to attempt this type of model training.

5 Conclusion

This project represents the power of technology in fostering inclusive education and advancing accessibility initiatives. Through the integration of Neural Radiance Fields (NeRF) and corner view generation methods, we have not only pushed the boundaries of 3D modeling but have also opened new avenues for educational innovation. The significance of our work is not only in its technical ability but also in its potential to address longstanding shortcomings in education. By using the capabilities of text-driven 3D modeling, we have provided opportunities for enhancing learning experiences for all students.

One of the most interesting aspects of our approach is its potential to equalize access to a full education. For visually impaired students, who face significant barriers to traditional learning materials, our model represents an opportunity to interact with a world of knowledge previously inaccessible. By providing tactile learning aids taken from textual prompts, we empower these students to understand complex concepts in a meaningful and immersive way. Our model has far-reaching implications beyond education. The ability to make detailed 3D models from textual descriptions has the potential to change how we create and interact with digital content.

Our project also shows the power of collaboration in pushing innovation. By bringing together experts from fields as different as computer science, and education, we can create a culture of creativity and collaboration that will lead to success for all. We will continue to understand the potential applications of technology as it evolves. Research can contribute to a future where technology is a force for positive change in people's lives by pushing the boundaries of what is possible.

To conclude, our project represents more than just a technical achievement. It is a symbol of progress and possibility. We believe that through this type of technology education can truly become accessible to all.

Appendices

A ResNet50

ResNet50 is a type of model that allows complex understanding of images. This specific model is made up of 50 different layers that each have separate extraction purposes such as edges colors or shapes. The model is able to extract very quickly due to the use of skip connections. These are basically shortcuts for the model that allow it to learn quicker. It was used specifically to extract these features and these features were then input into an SVM for a binary classification task for SDXL data curation. ResNet50 was derived from ResNet34 which is comprised of 34 weighted layers. ResNet34 provides a way to add more convolutional layers to a CNN, without hitting the vanishing gradient problem, using the method of shortcut connections. Shortcut connections skip over some layers, converting a regular network to a residual network. ResNet50 on the other hand has an architecture based on a bottleneck for building blocks. A bottleneck residual block uses 1x1 convolutions, known as a "bottleneck", which reduces the number of parameters and matrix multiplications. This enables faster training of each layer as it uses a stack of three layers rather than two layers. This comparison highlights the fact that innovation is ongoing in the space of computer generation and ResNet50 is only a stepping stone for faster neural networks.

ResNet34	**ResNet50**
34 Layers 3x3 Filter	50 Layers 1x1 Convolutions

Table 1: ResNet34 vs ResNet50

Script for Extracting Image Features

```
    import os
import pandas as pd
import torch
from torch.utils.data import DataLoader, Dataset
```

```python
import torchvision.models as models
import torchvision.transforms as transforms
from PIL import Image
from tqdm import tqdm

class ImageDataset(Dataset):
    def __init__(self, root_dir, transform=None):
        print("initialize dataset")
        self.root_dir = root_dir
        self.transform = transform
        self.folders = [os.path.join(root_dir, f) for f in os.listdir(
    root_dir)]
        print("Done initializing")

    def __len__(self):
        return len(self.folders)

    def __getitem__(self, idx):
        folder = self.folders[idx]
        images = os.listdir(folder)
        if not images:
            return None, None  # No images in folder
        image_path = os.path.join(folder, images[0])  # Load the first
    image
        image = Image.open(image_path).convert('RGB')
        if self.transform:
            image = self.transform(image)
        return image, folder

# Define transformations
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
    0.225]),
])

# Initialize dataset and dataloader
dataset = ImageDataset(root_dir='views', transform=transform)
dataloader = DataLoader(dataset, batch_size=32, shuffle=False, num_workers
    =4)

# Load a pre-trained model
model = models.resnet50(pretrained=True)
model = model.to('cuda')
model.eval()

# Function to extract features
def extract_features(model, dataloader):
    features = []
    for inputs, paths in tqdm(dataloader, desc="Extracting Features"):
        if inputs is None:
            continue
        inputs = inputs.to('cuda')
        with torch.no_grad():
            outputs = model(inputs)
        features.extend(zip(paths, outputs.cpu().numpy()))
    return features

# Extract features
features = extract_features(model, dataloader)

# Convert to DataFrame
df = pd.DataFrame(features, columns=['path', 'features'])

# Save to CSV
```

```
df.to_csv('extracted_features.csv', index=False)
```

B One-2-3-45

One-2-3-45[1] is a project designed and published in 2023 that could produce 3D models from a single input image. The project implements a 2D generation model which can take input images and produce new images of the same object with shifted camera view. Through this implementation combined with a NeRF model, the project is able to generate accurate meshes from a single image in less than 60 seconds. The code for this project is public on github, so it was easier for us to adapt it to our needs. The One-2-3-45 method as shown in Figure 10 gives a good graphical representation of what they did to generate 3D models. To start they implemented a Multi-view synthesis which uses a view conditioned 2D diffusion model, Zero123[12], to generate multi-view images in a two stage manner. The input of the model includes a single image and a relative camera transformation, which is parameterized by relative spherical coordinates. Second they used Pose Estimation to estimate the elevation angle theta of the input image based on four nearby views. Then the poses of the multi-view generation are obtained by combining the specific relative poses with the estimated pose of the input view. Finally we complete 3D reconstruction. The model feeds the multi-view posed images to an SDF-based generalized neural surface reconstruction module for 360 mesh reconstruction. This method although a bit different from what we did helped us understand different ways that text-3D model generation could be completed.

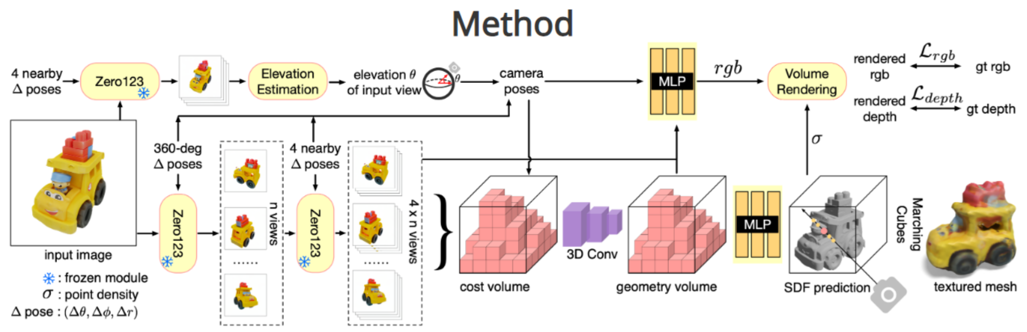


Figure 14: One-2-3-45 Method (Graphic taken from the One-2-3-45 Paper) [1]

C SDXL Finetuning Details

We use SDXL[7] as the base for our first stage generation. We take most of the specifics of this finetuning from the Instant 3D[4] implementation. We use a fixed learning rate of 10^{-5}, with a batch size of 6. We finetune the model on 2 NVIDIA RTX A5000 GPUs. We train for 40,000 steps, and this took about 40 hours. Table 3 below shows the recommended and possible image resolutions for training. As stated we were only able to access two GPU's so it was hard to get the best quality. However, with a multitude of resolutions available, we were able to complete training.

Recommended	**Possible**
1024 x 1024	512 x 512
1236 x 832	768 x 1344

Table 2: SDXL Generation Resolutions

D Full Script Code

```python
#!/home/vislab-002/anaconda3/envs/One2345/bin/python
import os
import subprocess
import argparse
from datetime import datetime
import paramiko
from scp import SCPClient
import shutil

def create_scp_client(host, port, username, password):
    ssh = paramiko.SSHClient()
    ssh.load_system_host_keys()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(host, port, username, password)
    return SCPClient(ssh.get_transport())

def transfer_directory_with_paramiko(source_path, destination_path, host,
    username, password):
    scp = create_scp_client(host, 22, username, password)
    try:
        scp.put(source_path, recursive=True, remote_path=destination_path)
        print("Directory successfully transferred.")
    except Exception as e:
        print(f"Failed to transfer directory: {e}")
    finally:
        scp.close()

def create_unique_directory(base_path):
    """Creates a unique directory based on the current timestamp."""
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    directory = os.path.join(base_path, f"run_{timestamp}")
    os.makedirs(directory, exist_ok=True)
    return directory


def get_recent_files(directory, pattern, count):
    command = f'find "{directory}" -maxdepth 1 -name "{pattern}" -printf
    "%T@ %p\\n" | sort -n | tail -n {count}'
    process = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE
    , stderr=subprocess.PIPE)
    stdout, stderr = process.communicate()

    if process.returncode != 0:
        print("Error:", stderr.decode('utf-8'))
        return []

    files = [line.split(' ', 1)[1].strip() for line in stdout.decode('utf
    -8').splitlines()]
    return files

def split_images_function(images, output_dir):
    for image in images:
        os.makedirs(output_dir, exist_ok=True)

        split_command =(
            f'cd ~/local/HandAntesMQP; '
            f'source Hand/bin/activate; '
            f'"/media/vislab-002/SP2 4TB/OpenLRM/split_image.py" "{image}"
     "{output_dir}"'
        )
        process = subprocess.Popen(split_command, shell=True, stdout=
    subprocess.PIPE, stderr=subprocess.PIPE)
        stdout, stderr = process.communicate()

        if process.returncode != 0:
            print(f"Error splitting image: {image}", stderr.decode('utf-8'
    ))
        else:
            print(f"Image split successfully: {image}")

def process_split_images(images, command_base_dir, resolution, unique_dir)
    :
    if len(images) != 4:
        print("Error: Exactly four images are required.")
        return
    python_script_path = os.path.join(command_base_dir, 'run.py')
    python_executable = "/home/vislab-002/anaconda3/envs/One2345/bin/
    python"
    command = (
        f'{python_executable} "{python_script_path}" '
        f'--img_path "{images[0]}" '
        f'--supp_img1 "{images[0]}" '
        f'--supp_img2 "{images[1]}" '
        f'--supp_img3 "{images[2]}" '
        f'--supp_img4 "{images[3]}" '
        '--half_precision '
        f'--mesh_resolution {resolution}'
    )

    process = subprocess.Popen(command, shell=True, cwd=command_base_dir,
    stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    stdout, stderr = process.communicate()

    if process.returncode != 0:
        print("Error processing images:", stderr.decode('utf-8'))
    else:
        print("Images processed successfully.")
        # Parse stdout for mesh file path
        output = stdout.decode('utf-8')
        mesh_file_path = None
        for line in output.splitlines():
            if line.startswith("Mesh saved to:"):
                mesh_file_path = line.split(":")[1].strip()
                break

        if mesh_file_path:
            print(f"Mesh file path: {mesh_file_path}")
            # Copy the mesh file to the unique directory
            shutil.copy(mesh_file_path, unique_dir)
            print(f"Mesh file copied to {unique_dir}")

            # Determine the directory of the mesh file and locate the stage1_8 directory
            mesh_dir_path = os.path.dirname(mesh_file_path)
            stage1_8_dir = os.path.join(mesh_dir_path, "stage1_8")
            if os.path.exists(stage1_8_dir):
                for file in os.listdir(stage1_8_dir):
                    if file.endswith(".png"):
                        src_file_path = os.path.join(stage1_8_dir, file)
                        shutil.copy(src_file_path, unique_dir)
                        print(f"Copied {file} to {unique_dir}")
            else:
                print("stage1_8 directory does not exist.")
        else:
            print("Mesh file path not found in output.")

        return output

def main(args):
```

```python

    unique_dir = create_unique_directory(r'/media/vislab-002/SP2 4TB/One-2-3-45/Uniques')
    generate_image_command = (
        f'cd ~/local/HandAntesMQP; '
        f'source Hand/bin/activate; '
        f'cd kohya_ss; '
        f'python3 sdxl_gen_img.py '
        f'--ckpt "/home/vislab-002/local/HandAntesMQP/32724SDXL.ckpt" '
        f'--outdir "{unique_dir}" '
        f'--xformers --bf16 --W 1024 --H 1024 --scale 12.5 --sampler k_euler_a --steps 256 '
        f'--batch_size 8 --images_per_prompt 1 --prompt "{args.prompt}"'
    )
    subprocess.run(generate_image_command, shell=True, executable='/bin/bash', check=True)

    recent_images = get_recent_files(unique_dir, "*.png", 1)
    print("Recent Image:", recent_images)

    split_images_function(recent_images, unique_dir)

    split_images = get_recent_files(unique_dir, "*.png", 4)

    One2345dir = r'/media/vislab-002/SP2 4TB/One-2-3-45'

    output = process_split_images(split_images, One2345dir, args.mesh_resolution, unique_dir)
    print(output)
    transfer_directory_with_paramiko(unique_dir, 'File Destination', 'IP Address', 'UserName', 'Password')
    return output

```

```python
if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='Process images based on a prompt.')
    parser.add_argument('prompt', type=str, help='A prompt for image generation')
    parser.add_argument('mesh_resolution', type=int, help='The resolution of the exported mesh')  # Changed type to int
    args = parser.parse_args()

    main(args)
```

E Major Changes to One-2-3-45 Framework

Changes to the input arguments to the main loop.

```python
    if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='Process some integers.')
    parser.add_argument('--img_path', type=str, default="./demo/demo_examples/01_wild_hydrant.png", help='Path to the input image')
    parser.add_argument('--gpu_idx', type=int, default=0, help='GPU index')
    parser.add_argument('--half_precision', action='store_true', help='Use half precision')
    parser.add_argument('--mesh_resolution', type=int, default=256, help='Mesh resolution')
    parser.add_argument('--output_format', type=str, default=".ply", help='Output format: .ply, .obj, .glb')
    parser.add_argument('--supp_img1', type=str, required=True, help='Path to the supplementary image 1')
    parser.add_argument('--supp_img2', type=str, required=True, help='Path to the supplementary image 2')
    parser.add_argument('--supp_img3', type=str, required=True, help='Path to the supplementary image 3')
    parser.add_argument('--supp_img4', type=str, required=True, help='Path to the supplementary image 4')
```

Changes to structuring of file setup.

```python
def stage1_run(model, device, exp_dir,
               input_im, supp_imgs, scale, ddim_steps):
    # folder to save the stage 1 images
    stage1_dir = os.path.join(exp_dir, "stage1_8")
    os.makedirs(stage1_dir, exist_ok=True)

    output_ims = []
    output_ims_2 = []
    for i, img in enumerate(supp_imgs):
        # Check if img is a path or an already opened image
        if isinstance(img, str):  # img is a path
            img = Image.open(img).convert("RGB")
        elif not img.mode == "RGB":  # img is an opened image but not in RGB mode
            img = img.convert("RGBA")
        img = img.resize((256, 256), Image.Resampling.LANCZOS)
        white_bg = Image.new('RGBA', img.size, (255,255,255,255))
        white_bg.paste(img, mask=img)
        white_bg = white_bg.convert('RGB')
        # Assuming img is now an opened Image object in RGB mode
        white_bg.save(os.path.join(stage1_dir, f"{i}.png"))
        output_ims.append(white_bg)
```

```python
def infer_shifted_images_for_folder(model, input_dir_path, save_path_stage2, delta_x, delta_y, device, pic_start, ddim_steps=75, scale=3.0):
    # Assume input_dir_path contains Stage 1 images to process
    # save_path_stage2 is where the shifted images will be saved

    stage1_image_paths = [os.path.join(input_dir_path, f) for f in os.listdir(input_dir_path) if f.endswith('.png')]
    sampler = DDIMSampler(model)
    image_counter = pic_start  # Start naming images from number 4

    for idx, stage1_image_path in enumerate(stage1_image_paths):
        raw_im = Image.open(stage1_image_path)
        input_im_init = np.asarray(raw_im, dtype=np.float32) / 255.0  # Normalize the image
        input_im = transforms.ToTensor()(input_im_init).unsqueeze(0).to(device)
        input_im = input_im * 2 - 1  # Scale to [-1, 1]

        # Generate one new image with specified shift
        x_samples_ddims_stage2 = sample_model_batch(model, sampler, input_im, [delta_x], [delta_y], n_samples=1, ddim_steps=150, scale=scale)

        # Save the generated image with naming from 4 onwards
        x_sample_stage2 = 255.0 * rearrange(x_samples_ddims_stage2[0].cpu().numpy(), 'c h w -> h w c')
        output_filename = f"{image_counter}.png"  # Constructs filename as "4.png", "5.png", etc.
        Image.fromarray(x_sample_stage2.astype(np.uint8)).save(os.path.join(save_path_stage2, output_filename))

        image_counter += 1  # Increment for next image's filename

        del input_im, x_samples_ddims_stage2
        torch.cuda.empty_cache()
```

References

[1] M. Liu, C. Xu, H. Jin, L. Chen, M. V. T, Z. Xu, and H. Su, “One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization,” 2023.

[2] R. Kapur, “Challenges experienced by visually impaired students in education,” 03 2018.

[3] Easy Render, “Most popular 3d modeling software and how much it costs.” https://www.easyrender.com/a/most-popular-3d-modeling-software-and-how-much-it-costs, 2024. Accessed: 2024-04-24.

[4] J. Li, H. Tan, K. Zhang, Z. Xu, F. Luan, Y. Xu, Y. Hong, K. Sunkavalli, G. Shakhnarovich, and S. Bi, “Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model,” 2023.

[5] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, “Dreamfusion: Text-to-3d using 2d diffusion,” 2022.

[6] H. Jun and A. Nichol, “Shap-e: Generating conditional 3d implicit functions,” 2023.

[7] P. von Platen, S. Patil, A. Lozhkov, P. Cuenca, N. Lambert, K. Rasul, M. Davaadorj, D. Nair, S. Paul, S. Liu, W. Berman, Y. Xu, and T. Wolf, “Diffusers: State-of-the-art diffusion models.”

[8] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi, “Objaverse: A universe of annotated 3d objects,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13142–13153, 2023.

[9] Allen Institute for AI, “Scripts for rendering objaverse.” https://github.com/allenai/objaverse-rendering, 2023.

[10] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015.

[11] T. Luo, C. Rockwell, H. Lee, and J. Johnson, “Scalable 3d captioning with pretrained models,” *arXiv preprint arXiv:2306.07279*, 2023.

[12] R. Liu, R. Wu, B. V. Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick, “Zero-1-to-3: Zero-shot one image to 3d object,” 2023.

[13] X. Long, C. Lin, P. Wang, T. Komura, and W. Wang, “Sparseneus: Fast generalizable neural surface reconstruction from sparse views,” 2022.

[14] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, “pixelnerf: Neural radiance fields from one or few images,” 2021.

[15] M. Deitke, R. Liu, M. Wallingford, H. Ngo, O. Michel, A. Kusupati, A. Fan, C. Laforte, V. Voleti, S. Y. Gadre, E. VanderBilt, A. Kembhavi, C. Vondrick, G. Gkioxari, K. Ehsani, L. Schmidt, and A. Farhadi, "Objaverse-xl: A universe of 10m+ 3d objects," *arXiv preprint arXiv:2307.05663*, 2023.