Dreamfusion in Nerfstudio

Abhik Ahuja, David McAllister

Abstract
In this project, we aim to develop a system that translates text prompts into coherent 3D model. Building on recent advancements in 3D reconstruction and text-to-image synthesis models, we plan to use the Nerfstudio and Stable Diffusion projects as the foundation for our implementation.

University of California, Berkeley

Contents

Introduction

Text-to-3d synthesis is an emerging field in 3d vision. DreamFusion[1] attracted major attention to the topic and demonstrated the possiblity of adapting existing 2D generative models to supervise 3D synthesis. As a new research topic, there are still many improvements to be made. The current implementations of text-to-3d generative models have large computing requirements and exist mostly in very technical contexts, putting them out of reach of the average user. Contributing our own implementation to the Nerfstudio project gives us the opportunity to make this generative technique accessible to more users. Furthermore, we will leverage all of the powerful features built into Nerfstudio to enable the user to view, profile, and export generated models.

1. Background

1.1 Neural Radiance Fields

Neural radiance fields, or NeRFs[2] (Tancik, et al.), are neural implicit representations of a radiance field. In other words, they are models that predict outgoing light in every direction across a scene. This values can then be integrated using volumetric rendering techniques to produce photo-real renders.

1.2 Dreamfusion

Dreamfusion[1] (Poole et al.) is a recent text-to-3D model that combines 2D generative models and NeRFs to generate 3D models from a text prompt. In particular, it uses Imagen (Saharia et al.) to generate images from a text prompt. Those generated images are then used to supervise a NeRF model based on mip-NeRF 360 (Barron et al.) to create a 3D model.

1.3 Nerfstudio

Nerfstudio[3] (Tancik et al.) is a recently-launched open source repository for neural radiance fields. Its key goal is to power NeRF development, testing, and distribution. The models are designed to be

highly modular, which proved to save a lot of time in our implementation. This project was developed at UC Berkeley, and we had experience in its codebase, so it was a natural choice to use it as the base for our project. In the future, we plan to distribute this project in open source through Nerfstudio.

2. Methods

In the planning stages of this project, we outlined a series of steps necessary to build a minimum viable implementation. These steps were as follows:

Table 1. Implementation Steps

Steps	Difficulty
Generative pipelines in Nerfstudio	4
Dynamic random cameras for training	1
Geometry normals	5
Lambertian shading	4
Stable Diffusion	2
Memory management	4
NeRF model adaptation	3
SDS loss	4
Background color prediction	3

2.1 Establishing a Training Loop

There was significant background work before our implementation could be effectively tested. The first major step was to implement generative pipelines in Nerfstudio. In contrast to normal scenes, generative pipelines do not have training image and camera pose ground truth pairs. Instead, they must generate random camera poses and supervise the resulting NeRF renders in a custom loss function. This required significant modifications to Nerfstudio's pipelines. We tried to make our implementation as modular as possible, but this is something we plan to refine before final release.

2.2 Normal Prediction

To shade models and supervise geometry, we need to be able to extract normal vectors from the NeRF geometry. Luckily, this is also needed for quality mesh extraction, so other Nerfstudio contributors

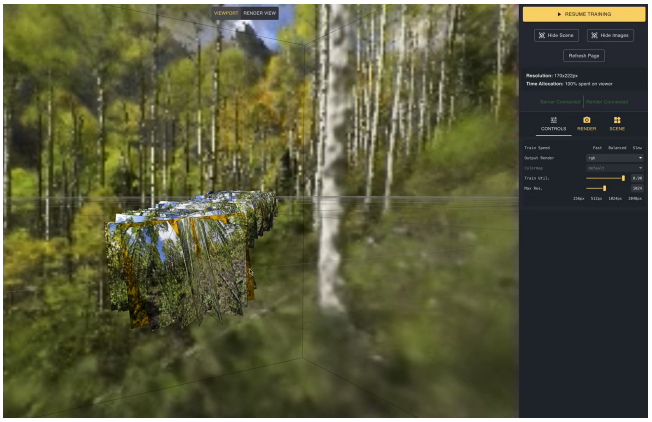


Figure 1. Classic Nerfstudio Training Cameras

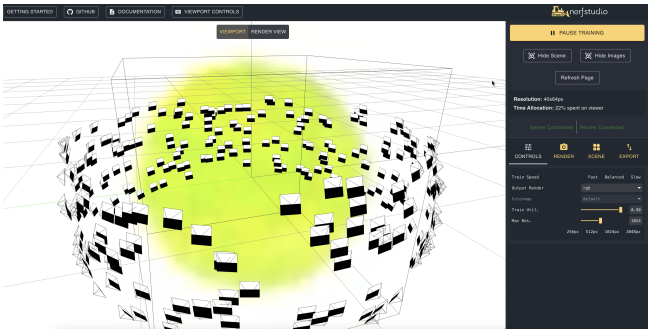


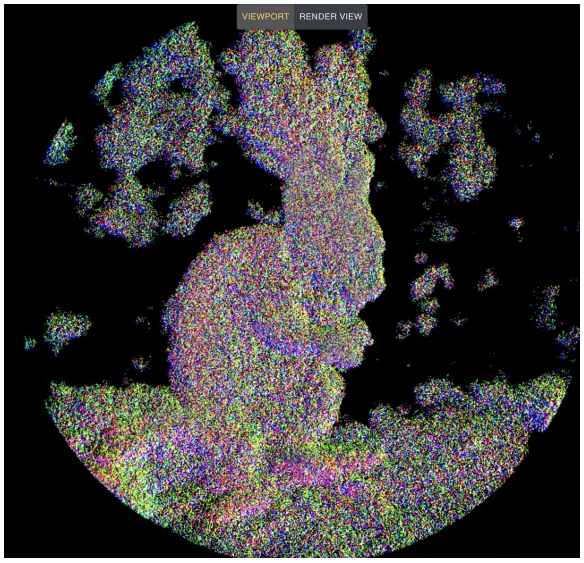
Figure 2. New Generative Training Cameras

were developing the feature in parallel. Implementing the normals in the generative pipeline required some modification, and we are still tuning it to create smoother normal gradients.

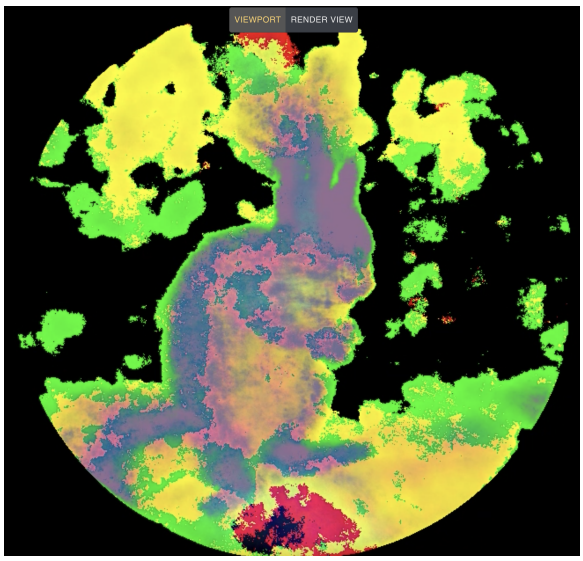
In line with Ref-NeRF (Verbin et al.), we fit a small MLP to the analytical normals in order to smooth out high frequency detail and include a loss term that incentivizes normals to point toward the camera. We use these predicted normals for our lambertian shading.

2.3 Score Distillation Sampling

For our Dreamfusion implementation, we needed to implement SDS loss using Stable Diffusion. SDS stands for Score Distillation Sampling, and it was a key discovery in the Dreamfusion paper. In essence, it follows the diffusion model principle of adding noise then diffusing through a U-Net to denoise. The key difference is that the noise we predict is compared to the noise we added. We then treat this difference as the gradients for our loss. In this way, we use Stable Diffusion to tell the NeRF



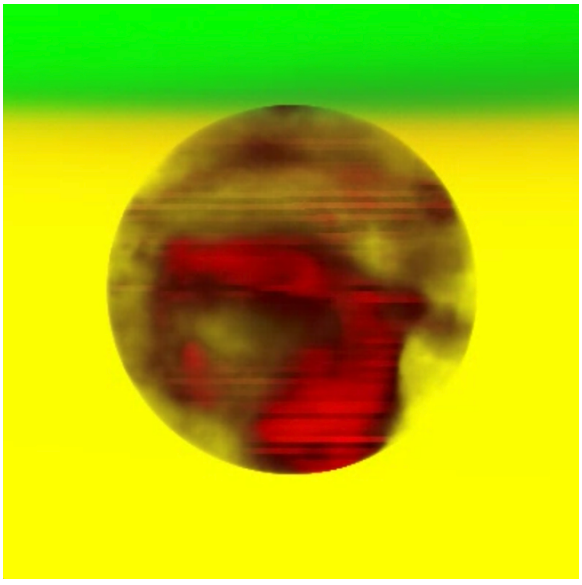
(a) Analytical Normals



(b) Predicted Normals

(c) SDS Noise Difference

(d) Lambertian Shading



(e) Background Color MLP

Figure 3. Figures demonstrating methods

which direction to push its weights to make the noise difference smaller.

In addition, the Dreamfusion authors found that it was important to use a high classifier-free guidance scale. This value sacrifices diversity of outputs for strength of conditioning. A lower diversity of outputs is beneficial in our case since it's more likely to encourage multiple iterations of the diffusion loss to have similar outputs.

3. Challenges

3.1 Stable Diffusion vs. Imagen

Since Imagen, the text-to-image model used by Dreamfusion, is not open-source, we used Stable Diffusion, an open-source alternative, instead. However, there are some key differences between Imagen and Stable Diffusion:

1. Imagen diffuses its images in pixel space, while Stable Diffusion does so in latent-space. This means that we must introduce a frozen Variational Autoencoder to transform our images from pixel-space into latent-space.

2. Imagen produces higher quality images than Stable Diffusion, limiting the potential quality of our model.

4. Results and Discussion

The model performs well on rotationally symmetric scenes such as pineapples. We believe this is because pictures of symmetric scenes generated by stable diffusion will be more similar, creating a more consistent signal for the NeRF to learn.

4.1 Difficulties

4.1.1 Coherence

Our implementation struggles with more complex geometries. Even with location based prompting as implemented in DreamFusion, stable diffusion tends to generate images of scenes from the same viewpoint, leading to inconsistent geometries. While the model sometimes succeeds in creating a coherent result, it often falls into nonsensical geometries or creates multiple copies of the target.

4.1.2 Color Mapping

Colors are also matched improperly between stable diffusion and the NeRF, creating cartoony textures without much detail. While we expect colors to be oversaturated due to the nature of SDS loss, our model struggles to create high frequency changes in color, often preferring to color the entire scene with 2-3 colors.

4.1.3 Convergence

Our implementation also fails to converge on occasion, either falling too quickly into a degenerate solution or placing density throughout the bounding sphere. We believe this can be addressed with further hyperparameter tuning, as relatively little work has been done so far to address it.

Acknowledgments

Thank you to Terrance Wang and Jake Austin for implementing the predicted normals and generative datamanager, respectively. Thank you also to Matthew Tancik and Ethan Weber, who have guided the direction of our model.

References

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

[2]

[3] Evonne Ng* Ruilong Li Brent Yi Terrance Wang Alexander Kristoffersen Jake Austin Kamyar Salahi Abhik Ahuja David McAllister Angjoo Kanazawa Matthew Tancik*, Ethan Weber*. Nerfstudio: A framework for neural radiance field development, 2022.

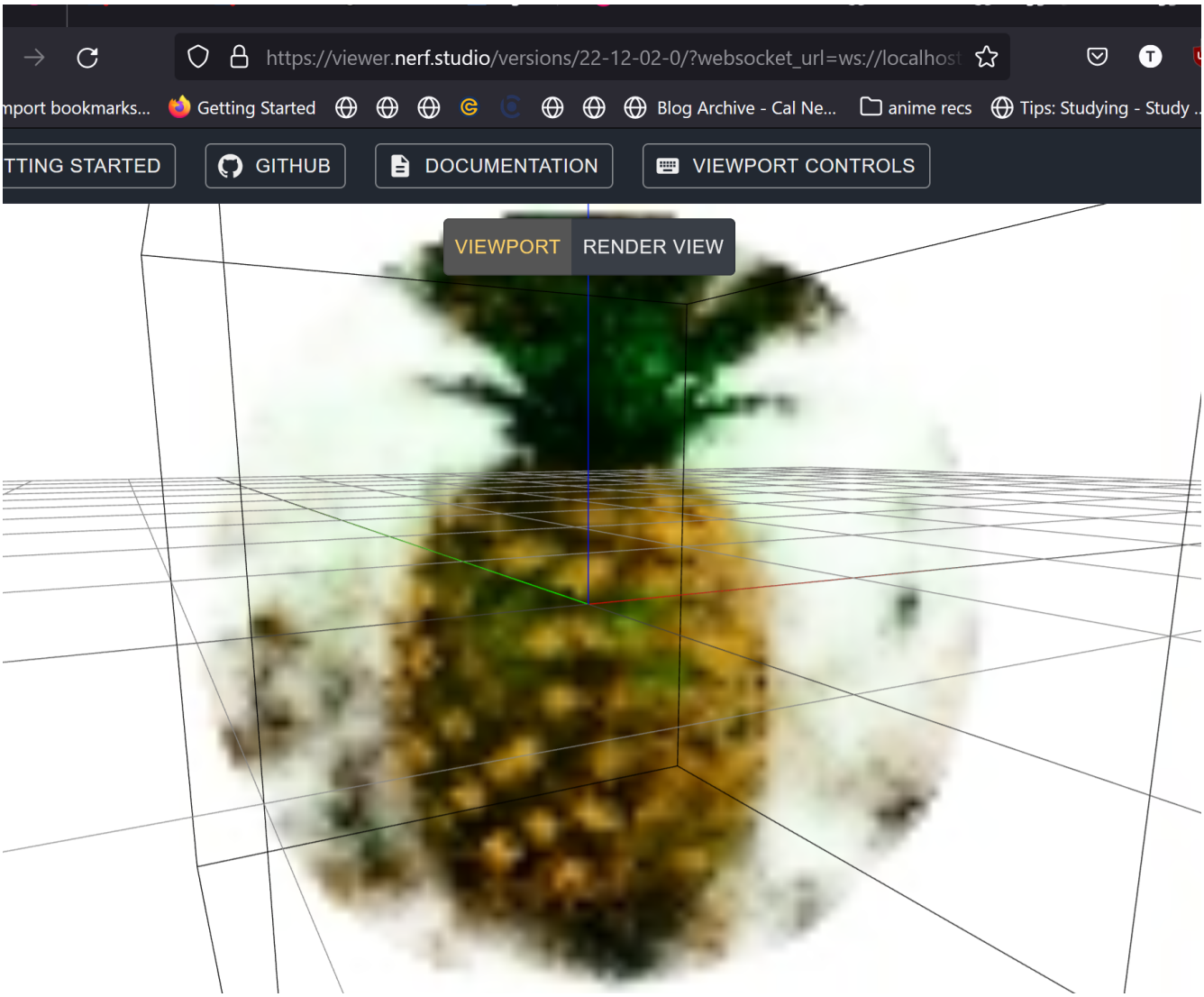


Figure 4. Selected Result for Prompt: “A high quality photo of a pineapple.”

Figure 5. Result for Prompt: ”A beautiful orange house in a forest.”

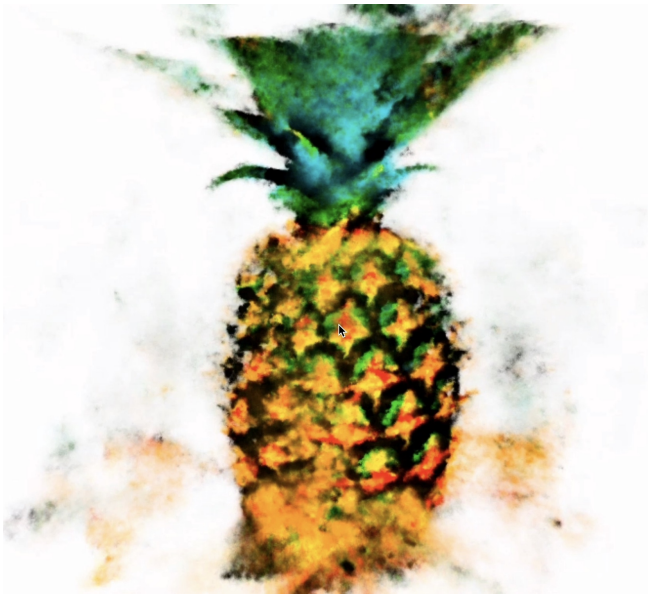


Figure 6. Another selected result for Prompt: ”A high quality photo of a pineapple.”

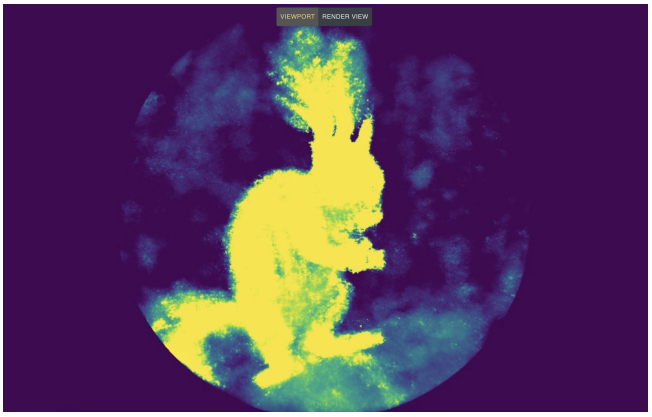


Figure 7. Accumulation of ”A high quality photo of a squirrel.”

Figure 8. Multiple copies of a squirrel generated for ”A squirrel sitting at a desk typing on a laptop.”

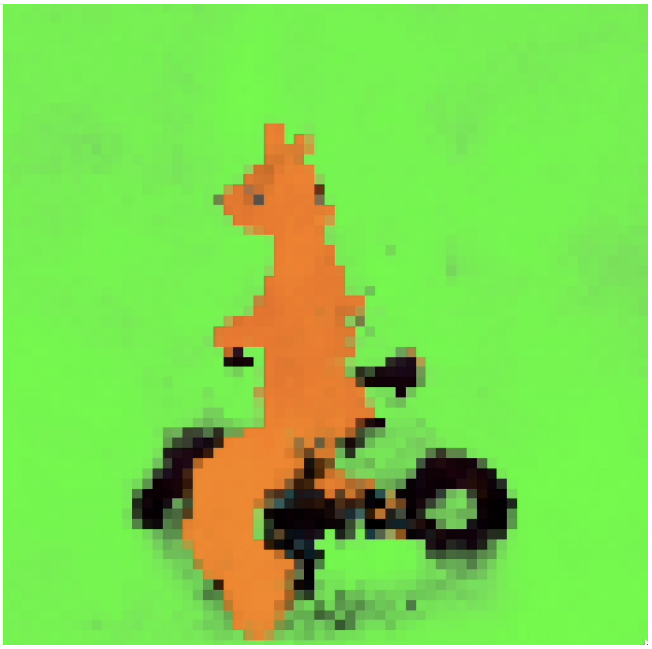


Figure 9. A back view of: ”A squirrel riding a motorbike.” Note the extra wheels on the sides, created to match a side view of the scene.