Ecole Polytechnique

Project INF 573 Report

10/12/2022

Author: Omar El Khalifi
Dune Amr–Tardif

NeRF for modelisation in space using 2D pictures entries

Abstract

This report describes the results of our project about 3D representation of a scene based on a text prompt. We were first interested in the NeRFs (Neural radiance fields) as a way to represent 3D space. They are usually constructed using images of a scene taken from all its sides. Our project consists in constructing a coherent synthetic data set, using a text to image diffusion model as well as methods we were shown in the labs.

Introduction

This report will first provide an overview of Neural Radiance Fields (NeRF), a deep learning technique for 3D scene reconstruction and rendering that is trained using a data set of images presenting the same object from different angles. Additionally, this report will cover our attempt to create a fully synthetic dataset to be used for training a Nerf model, using a combination of open source diffusion models and image processing techniques.

1 State of art and motivation

1.1 NeRF

Neural Radiance Fields (NeRF) is a technique for rendering highly realistic 3D scenes using deep learning. Developed by researchers at UC Berkeley and NVIDIA [1], NeRF uses a neural network to model the complex relationship between light and geometry in a 3D scene, allowing it to generate photorealistic images from arbitrary viewpoints. This technique has been shown to produce images of unprecedented quality, with fine details and accurate lighting effects that rival those of traditional ray tracing methods.

Neural Radiance Fields present a promising approach to 3D reconstruction, it is a powerful deep learning model that can generate high-fidelity 3D representations from a single image. Recent advances in NeRF have demonstrated remarkable results in 3D reconstruction, with the potential to improve the accuracy, efficiency, and scalability of existing 3D reconstruction techniques. NeRF has also been successfully applied to a wide range of applications, including object detection, scene understanding, and autonomous navigation. The current research in NeRF is focused on further improving its accuracy and robustness, as well as exploring its potential applications to a wide range of real-world scenarios. NeRF has the potential to revolutionize the field of 3D rendering, enabling the creation of highly realistic virtual environments for applications such as gaming, film, and architectural design.

1.2 Motivation

Observing an object with different angles is something that could already be done wit**morphing**. It consists in interpolating a movement of a lots of points between a "begin" image and an "end" image (here, it would have been possible to get the wanted result with "begin" and "end" images as different views of the same object). The algorithm does an interpolation between the position of points between the two images in order to get an animate result. This is advantageous because we got as an output a nice animation with only 2 images.

Nevertheless, this isn't very precise : if we apply this to a face, the nose and the rest of the face will "move" with the same speed... And it's not what happen in reality. Then, it became necessary (and it's possible in the Autodesk Combustion software often used to do that kind of operations) to define "control" points, then it's the responsibility to the user to chose the right emplacement of each point at different moments of the animation, which can be a very long job for a quite short animation. Moreover, the output is only an animation, if we don't want to recompile for each wanted movement around the object, it's necessary to guarantee the reversibility of the operations. Bad luck, it's often not the case.

The main idea that comes is to keep in memory the information about an object from different points of view without having to recompile for each rotation around it is to stock the output of the algorithm in a sort of 3D object. To do that, a simple interpolation isn't possible, as the resolution method for Bezier curves and other interpolations depends on the direction it's guided with. NeRFs guarantee here to "coordinate" the information gotten from the different images and

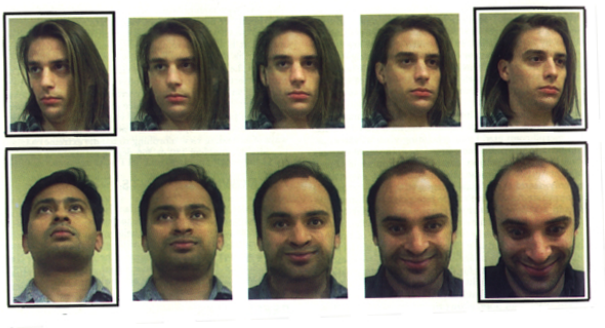
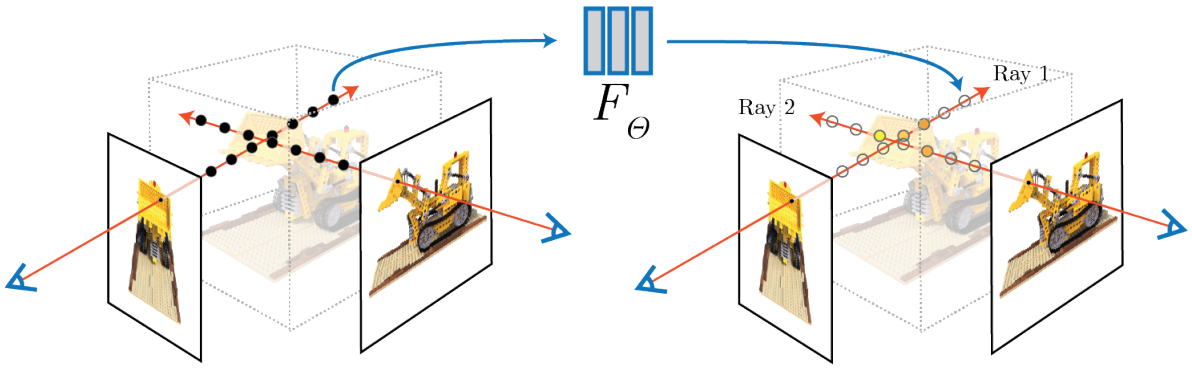


Figure 1: Example of interpolated images obtained by morphing between first, third and fifth images of each prompt (the computed images are the second and the fourth of each line)

combine them to generate a 3D model. It's precise way to work is what we'll study in the next paragraph.

2 NeRF algorithm : principle and using in context

2.1 How does NeRFs function ?



2.1.1 Overview

NeRF is doing a reconstruction of scenes by using multiple images as input for a scene. the input is fed to an MLP (Multi Layer Process) neural network that learns to transform a 5D (x; y; z; θ; ϕ) input into RGBα. After that, this output is given to a rendering program that generates the scene view.

2.1.2 Positional encoding

The positional data (x, y, z) is transformed into a larger, more meaningful vector thanks to frequency encoding. It allows the network to be less biased towards learning high frequency function

2.1.3 The loss function

The color seen from an input point and direction is encoded in our model as the integration of our density and color value along the ray plus a term that is added to take occlusion into account. The color is given by :

$$C(r) = \int_{t=0}^{t=f} T(t) * \sigma(r(t)) * c(r(t), d)\, dt, \text{where } T(t) = \exp\left(-\int \sigma(r(s))\, ds \text{ from } t_n \text{ to } t\right)$$

That integral is later approximated with classical methods. Each pixel of our images is therefore a data point and the integrated color is then compared to the the real value of the pixel is used as a loss (summed over all the pixels of all the images)

2.2 Practical use

Even given a large number of input views, NeRF can still be time-consuming to train; it often takes takes up to 2 days to run the original model on a good Gpu which is just not affordable at our scale. We started to look toward tinyNerf that is an implementation that skip the hierarchical sampling optimisation step. It was still too much for our computers and produced blurry images (which was to be expected as it is way less capable). Later we learned about more recent techniques such as plenoxels that use the same principles as Nerf but replaces the multilayered perceptron with an optimisation on a 3D grid. That approach reduced the computation time by orders of magnitude but we sadly didn't manage to make it work.

3 Our idea and implementation

We decided to combine diffusion models's potential and the methods studied in the labs (homography matrices and image convolutions) to generate a data set of different views from the same object exploitable by the NeRF. Using warped versions of our image alone wouldn't make much sens as the information contained on the original image wouldn't be enough to train a Nerf. Our idea was to use the information contained in the diffusion model's latent space in order to enrich our base image, the main challenge we faced was maintaining a reasonably good coherence between the pictures

We used the open source algorithm "Stable Diffusion", as it allowed us to work locally (first we tried with DallE but the number of queries was to small). Diffusion models are used to generate images from an entry text, which would allows a user to simply prompt a text of the object he wants to see in 3D and obtain what he wants (automatisation of the process). How does it works : diffusion models like Stable Diffusion are algorithms trained on a very large dataset of labelled images. For each image, a first "mini model" adds noise in order to get an unrecognizable picture, then a second "mini model" is trained to find again the input image with Markov chain models principles. The text input is taken in count for both the generation of noise and the denoising.
The advantage of this method is the creation of a full semantic into the machine, we can then ask for very complex things (for exemple a catzilla, which is not available in real life to take photographs, or would demand much time to a graphist to draw).

Figure 2: Catzilla with stable diffusion

3.1 First attempt

It is possible to explore the latent space of the algorithm by adding tiny inputs to the text embedding in order to navigate small steps at a time, this allows us to have some continuity between the outputs which is what we want on paper to generate a different perspective of the same object. Sadly it is too difficult to explore as it is a 77 * 768 dimensional space in our case, and finding the

direction that could produce a rotated view of the object is a hard task, in addition we have no guaranty that it even exists in the latent space. We had to find another way.

3.2 Second attempt

The second idea we had was to use the in-painting feature of the stable diffusion model, We began with a picture of the Eiffel Tower found on the Internet in order to test the guided image-completion on a diffusion model (here stable diffusion). We draw an in-painting mask to complete with hopes to see a new perspective from the AI completion. Nevertheless the result wasn't what we expected (the point of view was the same).

Figure 3: a. Before inpainting b. After inpainting

3.3 Third attempt

We decided to use techniques we learned during the labs such as homography projections and edge detection to improve the way we inpainted the image.

3.3.1 The basic heuristic

First we thought of homography matrices to change the perspective of the image to simulate a subtle rotation around the subject. Our basic assumption was that when looking at most objects while moving to the right, their right edges unveil what is hidden behind them while they shrink and disappear along their left edge. Following this assumption that holds only for isolated objects we could create a thick custom mask that covered the right edges of our object. We then feed it back to the diffusion model that infers either what is hidden behind or the side of the object using the general knowledge it learned about the world through it's massive training set.

3.3.2 Creating the masks

We are assuming that we are moving in spherical coordinates (theta, phi) centered on the object with a fixed radius. To create the mask we first had to get the information about the direction we are rotating about (up, down, right or left), then according to the direction we apply a corresponding convolution with a signed sobel filter to the image (it assumes that the background color is white) that extracts the edges in the corresponding direction. We then denoise the result and thicken the edges to create the mask with multiple convolutions based on an other filter. The result is still not perfect as not all edges are due to depth discontinuity and texture or surface normal discontinuities get picked up.

3.3.3 Homography matrices

We tried to find a well chosen homography matrix that simulated a small rotation around the subject. The problem is that even with the standard decomposition we learned, it is pretty hard

to visualise and predict it's precise effect on a single image as there is no real unit or even frame of reference, we recalled that during the labs we only used them to link between multiple images and it seemed pretty hard to create a frame of reference based on a single image. We decided to settle for a perspective changer in an image editing tool (for us : Krita, french and free software) in order to get a shape near the ones we wanted.

Figure 4: a. Before inpainting after manual modification b. After inpainting

3.3.4 Feeding the results to the diffusion model

The inpainting protocol was quite efficient to complete the "environmental" part of the picture, but for the Eiffel tower in itself, its not that powerful. When we add "Eiffel tower" as a text entry to guide the algorithm it seems to be more efficient but the added part is always a part of a Eiffel tower seen from the front of it. What we thought was that as the Eiffel tower is often photographed from a front view, and that in the diffusion model semantic used, the legs of the Eiffel tower are (almost) only a front view of them.

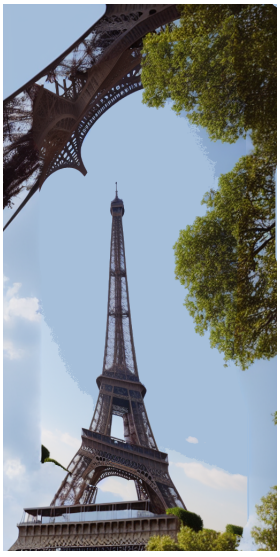


Figure 5: The result of the guided inpainting with entry text "Eiffel tower"

4 Conclusion and ouverture

To conclude, we didn't manage to fully connect our pipeline and the results weren't that impressive, even with the guided masks obtained with edge detection. Even if we had a couple coherent images, the diffusion model didn't give results that were reliable enough and the accumulated variations meant it was unlikely we would have a stable data set that went around the object. We therefore couldn't achieve the main goal we set at the beginning : creating a fully synthetic data set for training a Nerf. Nevertheless, we learned a lot about NeRFs to get 3D scenes from images and diffusion models. We spent a lot of time prompt engineering to get the best results but at the end, it seemed to us that the pre-trained diffusion models were biased a lot towards the dataset they were trained on (teddybears always seen from face, then even given the prompt "left side" they were from a front view but thinner than the result without specifying the orientation). A way we could improve our results would be to train our own diffusion model with "calibrated" images : blank background, different views with labels precising the view angles, views from above. To expand on our exploratory work we could mention DreamFusion which is a text to Nerf algorithm based on an image diffusion model much like the one we tried to make. It uses Imagen and the Nerf model as building blocks to make complex interconnected network but it's implementation goes beyond the scope of this project.

Bibliography

- [1] **A Style-Based Generator Architecture for Generative Adversarial Networks**, Tero Karras & Al., 2019
- [2] **https://www.matthewtancik.com/nerf**
- [3] **https://github.com/bmild/nerf**
- [4] **https://github.com/yenchenlin/awesome-NeRF**
- [5] **State of the Art on Neural Rendering**, A. Tewari & O. Fried & J. Thies et al., 2020
- [6] **Nerfs : https://arxiv.org/pdf/2003.08934.pdf https://arxiv.org/abs/2112.05131**
- [7] **Plenoxels : https://arxiv.org/abs/2112.05131**
- [8] **Latent space exploration : https://keras.io/examples/generative/random**$_{w}alks_{w}ith_{s}table_{d}iffusio$
- [9] **3d diffusion based on a text to image models : https://dreamfusion3d.github.io/**
  - **Morphing image :**https://old.cescg.org/CESCG97/penkler/tom8.jpg