2D Image to 3D Model

Author

Lam Nguyen**,** University of Central Florida

Abstract

There are a few different tools to convert 2D images into 3D models. They are still a work in progress, however we will be discussing a few tools that are available and their limitations and capabilities. The tools that will be discussed are:

1. Kaolin
2. Pytorch3d
3. Pifuhd
4. Colmap

Introduction

In this project, we will be converting a 2D image or series of 2D images into a 3D Model. There are various reasons for doing this. 3D Models/ Assets are used in art, movies, architecture, scientific purposes, video games, etc… However, they are very time consuming to create. A 3D scene can have hundreds or even thousands of objects that need to be created by hand on a computer. Everything from creating a human, building architecture… To tedious things like making leaves, trash, and cups.. They all must be created. An artist, designer or draftsman can greatly speed up the production process of these assets by merely taking a picture and inputting it in an application to produce at least a rough 3D Model output that can then be modified.

So far, there are still a lot of limitations with this type of technology. 3D objects require much more calculations and resources then a 2D image. Also a lot of information is lost when turning a 3D scene or a real life scene into a flat image. A trained machine learning algorithm can make up for this lost information. However, due to current algorithm limits and/or hardware limits… It is often only practical to train a model on a particular class of object/being. For example, you only want to train a model to recreate humans. Or to recreate kitchen utensils. Or to recreate jellyfish, etc.. This is to work within the limits of computing power and costs as of this time.

Also, these algorithms are still in their stage of infancy. So a lot of the technologies available at this time are still pretty buggy and unstable. And with developers constantly changing things in the operating systems or the dependencies… Bugs are often an issue with using these algorithms… Especially when they require GPU drivers. Admittedly, I was forced to spend a lot of time building the environment to run the algorithms. Some of these algorithms work better on Windows… Others on the Linux Kernel. And other algorithms only work for a particular linux kernel… But not another. The type of GPU and drivers also matters as well… For example, running an AMD or Intel branded discrete GPU is not an option for such a technology in its infancy.

There are a couple of technologies that were created to do this task over the past couple of years. In this paper we will be going over four of them that work with the current packages as of 2024 for the Ubuntu 22.04 and Windows 11 operating systems.

These four packages are:

1. **Pytorch3D:** a package developed by Meta(Facebook) for working with 3D Data
2. **Kaolin:** A package and gui developed by Nvidia used to create synthetic 2D image datasets and also can train on those datasets to reproduce a 3D model
3. **PifuHD:** Developed by Canadian researchers funded by Meta(Facebook) that specializes in converting a single 2D Image of of a person into a 3D model.
4. **ColMap:** A program developed by Swiss and American researchers that can take a series of images of buildings or whatever and then convert them to a 3D Model.

The following section will provide a quick outline of how to set up the environments for each of these programs that I've found to work as of the year 2024 with Ubuntu 22.04 and Windows 11. The reason for creating this section is because I've found out the hard way that even after 2-4 years… The developers have changed so many things between packages that incompatibilities develop and the programs are not able to work. Also the documentation is not always clear about this or is outdated..

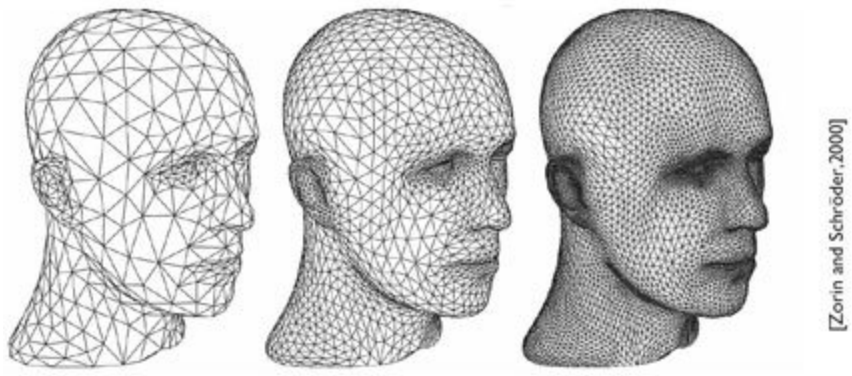
Theoretical Introduction to the Technologies

Before discussing anything, we need to cover some of the basics of how 3D models are represented. We know that 2D images are basically represented as a grid of pixels. Each pixel will have varying levels of Red, Green and/or Blue (RGB). This is the generally accepted manner in which 2D images are represented.

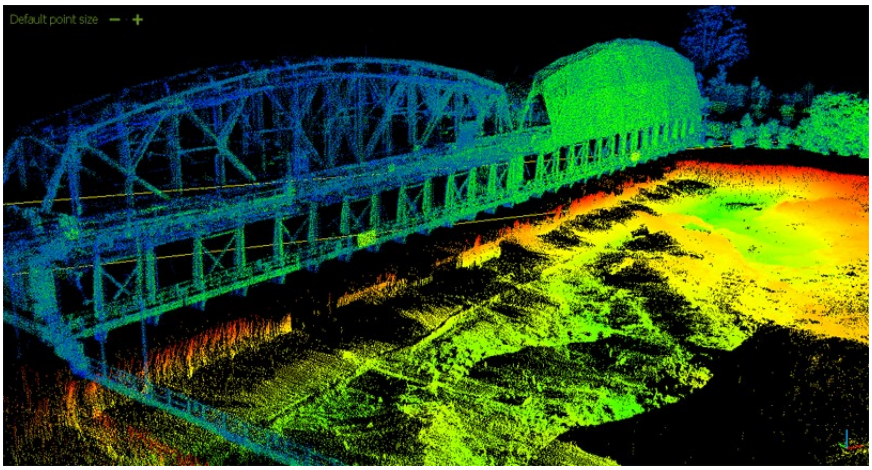
However 3D representations of objects, have a couple of ways of being represented. They will be listed below:

Voxel: 2D images are shown in pixels which are just a collection of tiny squares that make up a larger image. A Voxel representation of a 3D model is similar to a 2D image, but instead of a 2D pixel, it is a 3D cube. The 3D model is then made up of these blocks stacked together. It's not only the outside of the model that is shown, but the inside volume as well.

Polygonal Mesh: A collection of points, edges and faces that define an object's surface in 3 dimensions. The main difference between a voxel representation and a polygonal mesh is that a 3D voxel representation is like a solid brick with an outside surface and a dense inner volume. Whereas a Polygonal mesh is like an empty aluminum soda can that has a defined shape, but nothing inside.



Point Cloud: A point cloud is collection of 2D points in a 3D coordinate system. The more points, the greater the detail of the object. The difference between a point in a point cloud and a voxel in a voxel representation is that a point stores much less information then a voxel. So the point cloud representation of data is much more compact and takes up less space on a hard disk.



Introduction to Pytorch3D

Pytorch 3D is an extensive package built by the company Meta (Facebook) to work with 3D data and deep learning. We will limit the scope of the discussion to convert 2D Images to a 3D object. Pytorch3D is built on top of the Pytorch Module and was created for accelerating research in the application of deep learning to 3D. The original Pytorch module had issues with 3D data representation, batching and speed. So Pytorch3D was developed to address these issues. Some of the features of Pytorch3D are:

1. Data Structure for Storing and Manipulating Triangle Meshes
2. Efficient Operations on Triangle Meshes such as:
    a. Projective Transformations
    b. Graph Convolution
    c. Sampling
    d. Loss Functions
3. Differentiable Mesh Renderer
4. Implicitron: Framework for new-view synthesis via implicit representations

For the example shown in this project, the process of building and training the model requires these steps:

1. Create a synthetic dataset of 2D images from an existing 3D model. You could also just have multiple images of the object you want to create a 3D model from. These multiple images will be taken and the algorithm will use those images as a guide to sculpt a bash mesh to a 3D model. There is no need for a 3D mesh for the actual training of the machine learning model. We will create a series of RGB images and then a series of silhouettes that the model then can be trained on.

2. To create a synthetic dataset for our code, we need to:
    a) Import 3D model into 3D Program called Blender
    b) UV Unwrap the model
    c) Add a Texture and Material
    d) Export the Texture as a PNG File
    e) Export the mesh as .OBJ file.
    f) At the end of the 3D Model Preprocessing, we should have a: .OBJ file, a .MAT file and a .PNG file.
    g) These 3 files will be used by our code to produce synthetic training data that the model will use to reconstruct a model.

3. First the algorithm will try to sculpt the mesh based on just the silhouette images.
4. Next we will try to sculpt the mesh based on the RGB images.
5. Then we save the results.

Introduction to Kaolin

Like Pytorch3d, Kaolin is also a package built on top of the regular Pytorch package in order to work with 3D data. Whereas Pytorch3d was built by the company Meta, Kaolin was built by the company Nvidia. Here are some features of Kaolin:

1. Modular Differentiable Rendering
2. Fast Conversions Between Representations
3. Data Loading
4. 3D Checkpoints
5. Differentiable Camera API
6. Differentiable Lighting with Spherical Harmonics and Spherical Gaussians
7. Structured Point Clouds
8. Interactive 3D Visualization for Jupyter Notebooks
9. Batched Mesh Container

For the particular example we are creating in Kaolin, the following steps will be done to convert 2D Images to a 3D Model:

1. Create a synthetic dataset by taking a preexisting 3D model and "taking photos" of the 3D object from different viewpoints. We use the Omniverse Kaolin Application GUI for this.
2. The input for the Synthetic dataset creation has to be in the file format called .USD. To create a .USD file, you can import 3D data into Blender and convert it to .USD.
3. Load the Mesh Sphere Template. This will be the "clay" that the algorithm will sculpt into the object based on the pictures we feed the algorithm.
4. Set up Loss Function, Regularizer and Optimizer for Training
5. Set up Differentiable Renderer which outputs a 3D model at each training step.
6. Train using a a chosen Neural Architecture.
7. Visualize output and save to a format for use in other 3D programs.

Introduction to Pifuhd

Pifuhd is an acronym for "Multi-Level Pixel-Aligned Implicit Function for High Resolution 3D Human Digitization." This module was created by a partnership between the University of Southern California and the company Meta (Facebook). This model specializes in inputting high resolution 1K and up images of humans to produce more detailed 3D models. Unlike Pytorch3D and Kaolin, this module is more specialized, more open source and is trainable from end to end. This module comes with a pretrained model that can be downloaded and used by the network so that the user doesn't have to train their own network and can also understand what data is used for training.

Also unlike the other software modules, this one only needs a single frontal image to work whereas the other ones required multiple images at different angles. So the model had to learn how to infer the backside of human beings… Which is another reason why Pifuhd is so specialized as it wouldn't be feasible to do this backside inference with too many different types of objects or beings.

The General Steps for the conversion of a 2D image to a 3D model is given below:

1. Input 2D image into network
2. The 2D image has to first be preprocessed through a software called OpenPose. OpenPose is a software that determines where the joints, hands and face locations are on a human image. Essentially it creates a stick figure superimposed over the human image and outputs the coordinates for that stick figure in a .JSON file.
    a) So the input into PifuHD is actually the image of the person and the .JSON file.
    b) Make sure that the image is a full body frontal photo and that the image has the background grayed out to reduce the chance of the algorithm being confused.
    c) It is easier to install the OpenPose binary in the Windows Operating System. The binaries aren't yet available for Linux.

3. Images passes through a Network to produce the global 3D structure for the 3D Model.
4. Also Image is passed through another network to predict the backside of an image.
5. The front and backside of the image as well as the rough global 3D model are passed through another network to add fine details.
6. Output model is then produced.

Introduction to Colmap

Colmap is a general purpose Structure-from-Motion and Multi-View Stereo pipeline. These are the steps for the Colmap Structure-from-Motion algorithm:

1. Input multiple images.
2. Feature Detection and Extraction from images.
3. Match the features and geometric verification. Basically means that you check whether the same item or feature found in different images is actually the same feature or item.
4. Structure and Motion Reconstruction
5. The output of the image is a Point Cloud stored in .PLY file. To convert the file to a Mesh, use the free software called Meshlab. The Mesh can then be used for 3D Art, video games or movies.

Setting up the Environments:

1. Pytorch3D

According to the Documents, Pytorch3D works with Windows, Linux and MacOS. However, at least with CUDA support, I was only able to get the package to successfully install on Linux, specifically Ubuntu 22.04.

1. Use Linux. I used Ubuntu 22..04
2. Install Anaconda
3. In the terminal: conda create -n pytorch3d python==3.9
4. In terminal: conda activate pytorch3d
5. In terminal: conda install pytorch=1.13.0 torchvision pytorch-cuda=11.6 -c pytorch -c nvidia
6. In terminal: conda install -c fvcore -c iopath -c conda-forge fvcore iopath
7. In terminal: conda install -c bottler nvidiacub
8. In terminal: conda install jupyter
9. In terminal: pip install scikit-image matplotlib imageio plotly opencv-python

10. In terminal: conda install pytorch3d -c pytorch3d

Maybe using a more up to date python and pytorch package will work… But there were issues on the installation. So to reduce the headache, just use the package versions specified in the steps.

2. Kaolin

1. Ubuntu 22.04 or Windows 11 both work.
2. Install Anaconda
3. In the terminal: conda create -n kaolin python==3.10
4. In terminal: conda activate kaolin
5. In terminal: conda install pytorch=2.1.1 torchvision pytorch-cuda=12.1 -c pytorch -c nvidia
6. In terminal: pip install kaolin==0.15.0 -f https://nvidia-kaolin.s3.us-east-2.amazonaws.com/torch-2.1.1_cu121.html
7. In terminal: conda install matplotlib
8. Download the Omniverse SDK from Nvidia. You will have to register an email. Install Omniverse. For Windows, this is a pretty straightforward process. For Ubuntu, you will get the Omniverse Application stored in the format called an AppImage.
9. The AppImage file needs to be made executable to be installed. You can google how to do this on Linux. However, when you try to do this on at least the Ubuntu Distribution of Linux, you will get a warning that a packaged named Fuse needs to be installed. And you will be given a terminal command to install it… **DON'T USE THIS COMMAND.** You will install a version of Fuse that will break your Ubuntu Installation if you are on in Ubuntu 22.04. You have to install a newer version of Fuse called LibFuse2. Use this command instead: sudo apt install libfuse2
10. The Libfuse2 package will allow you to execute the Omniverse AppImage file without breaking your Ubuntu 22.04 OS.

3. Colmap

1. Colmap works on both Linux and Windows. However, I can only seem to use Colmap with GPU compute capabilities reliably on the Windows OS. I'm not able to compile Colmap on Ubuntu 22.04 with GPU capabilities.
2. On Windows, you can download the Colmap Executable file and run it. To build Colmap from source, follow the documentation.
3. On Linux Ubuntu 22.04, to install Colmap, use the terminal commands:
    1. sudo apt-get update
    2. sudo apt-get -y install colmap

Note that installing Colmap using the terminal command will only install a version of Colmap that uses the CPU. GPU Compute won't be available for this installation.

Source: https://colmap.github.io/install.html

4. PifuHD

1. Pifuhd only works on Linux as far as I know. The steps are for Ubuntu 22.04.
2. Install OpenPose Binaries in order to get a simplified pose structure or rig of the human in the image or video.
3. Install Anaconda
4. In terminal: conda create -n pifuhd python==3.7
5. In terminal: conda activate pifuhd
6. In terminal: conda install pytorch==1.5.0 cudatoolkit=10.2 -c pytorch
7. In terminal: pip install torchvision==0.6.0
8. In terminal: conda install anaconda::pillow
9. In terminal: conda install anaconda::scikit-image
10. In terminal: conda install conda-forge::tqdm
11. In terminal: conda install conda-forge::opencv
12. In terminal: conda install conda-forge::trimesh
13. In terminal: conda install anaconda::pyopengl
14. In terminal: conda install conda-forge::ffmpeg
15. In terminal: pip install matplotlib
16. In terminal: pip install numpy --upgrade

Conclusion

If you watch the demonstration video, you will see that even with these massive companies backing the research behind AI generated 3D Views, there is still a lot of work to do. The 3D models are still rough. And the software required to do the task is still very buggy, and it's easier to run different steps in the process on different operating systems and different versions of the same package. There is still a lot work to be done in streamlining the process.

Also the results leave something to be desired. They could be improved by tweaking parameters or the input data.

Sources:

Kaolin Video Tutorial: https://www.youtube.com/watch?v=BwZ7s7DqbFs&t=9s

Kaolin Github: https://github.com/NVIDIAGameWorks/kaolin

Pytorch3d Video Tutorial:
https://www.youtube.com/watch?v=v3hTD9m2tM8&list=PL3OV2Akk7XpBPBEw1jekpxDJYIRPEHbUi

Pytorch3d Github: https://github.com/facebookresearch/pytorch3d

Pifuhd Research Paper: https://arxiv.org/pdf/2004.00452.pdf

Pifuhd Github: https://github.com/facebookresearch/pifuhd

Colmap Github: https://github.com/colmap/colmap

Colmap Documentation: https://colmap.github.io/

Colmap Tutorial: https://www.youtube.com/watch?v=Zm1mkOi9_1c

Point Cloud to Mesh Tutorial: https://www.youtube.com/watch?v=N-Yk8sIOu_w&t=250s