Introduction

This project aims to reconstruct accurate 3D models of room interior based on photographic images. This has a wide range of applications in fields such as real estate, architecture and engineering. The process of 3D reconstruction is as follows:

1. Process the photographic image to account for curvature of lens
2. Identify the visible edges of the floor, ceiling, walls and columns from the image
3. Extrapolate the visible edges to form cuboid geometries
4. Adjust the edges to account for perspective
5. Create an isometric 3D model based on the edges

The main challenge in the process highlighted above is step 2. It is extremely challenging to write an algorithm that considers all the possibilities of room layout and structure, as well as the furniture and decorations in the room. Therefore, machine learning is identified as a possible alternative to filter out the edges from the noise in photographic images.

Machine Learning Model

**Convolutional Neural Network (CNN)**

The machine learning model chosen for this application is CNN. This technique is widely used in image processing and is demonstrated by the Google DeepDream experiment to be a powerful tool in feature extraction. In our case, the feature that we wish to extract is the room edges. The neural network is built using Tensorflow and based on the concept used by SegNet. The detailed structure is visualised below:

Conv2D Layer

BatchNormalization Layer

LeakyReLu Layer

Dropout Layers

Conv2DTranspose Layer

Downsampling Block

Downsampling Block

Upsampling Block

Upsampling Block

Intermediate Hidden Layers

The neural network takes in an input RGB 256 x 256 image and generates an output of grayscale 256 x 256 image. The hidden layers composes of a series of convolutional layers that eventually reduce the image size to a minimum of 64 x 64. This is then run through a series of convolutional upsampling layers to generate an output image of 256 x 256. The final layer is normalised to a value between 0 and 1. The model is compiled using Mean Squared Error as the loss function.

**Data Processing**

The first step in developing a machine learning model is to create a dataset to train and test the model with. For typical image-related neural networks, the size of the dataset should be on the order of 104. For our model, the images must meet the following criteria:

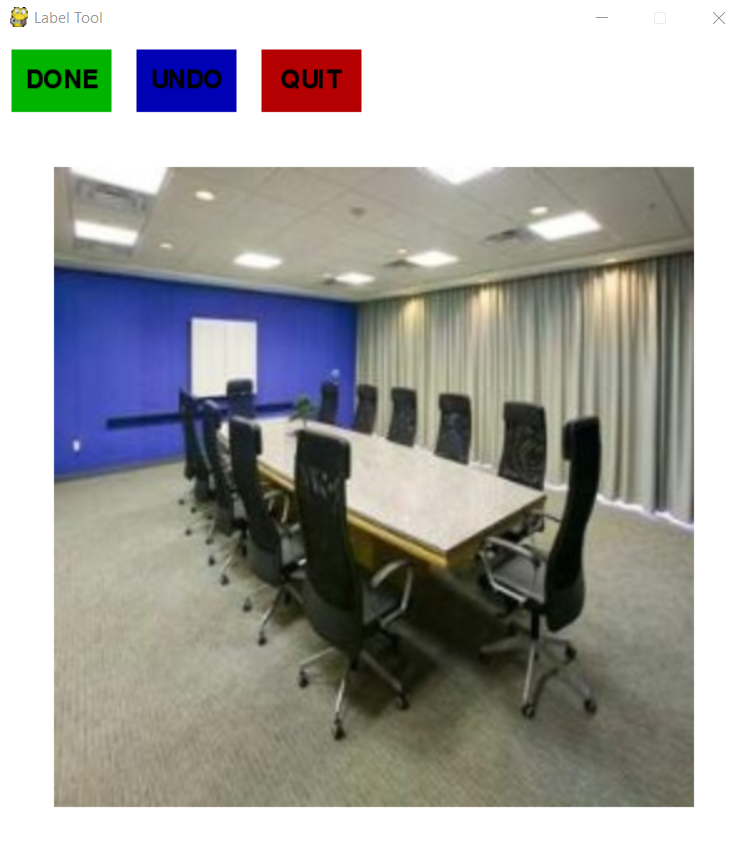
1. It must show the interior of a room
2. It must have at least one visible edge
3. It must not be overly warped
4. It must not have curved wall edges

There are 2 main difficulties associated with generating such a large dataset for our specific purposes:

1. Sourcing for a sufficiently large dataset that is suitable for our purposes
2. The time and effort required to manually clean and label each image

In order to tackle the first issue, we first sourced for images of empty rooms on the internet, but it returned very few appropriate results. We then expanded the search to include all images of rooms. The database that we eventually used was LSUN which was originally created for the Scene Classification Challenge. While the database contained a sufficiently large collection of room images (more than a million images), most images are not suitable as they do not fulfil the criteria highlighted above.

This brings us to our second difficulty, which is the time taken to process each image. In order to reduce the amount of manual effort required, we decided to use data augmentation to quadruple the labelled images. By flipping images horizontally, adjusting the brightness and a combination of both, we can generate 20,000 labelled images from just 5,000.

The last challenge remaining is to source for a labelling tool that can achieve the task of identifying edges from an image. After looking through popular online tools such as MIT LabelMe and Amazon Mechanical Turk, we realised that none of them are suitable for our purposes. As such, we used Python to create our own labelling tool that could create suitable labels based on edges (Fig. 2).

**Optimisation**

Given that edges make up very few pixels in the image, one concern is that the optimised model might generate blank images to minimise the mean squared error. In order to bypass that eventuality, we created a “heatmap” around each edge as shown in the array below:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0 | 0 |
| 0 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0 | 0 |
| 0 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0 | 0 |
| 0 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0 | 0 |
| 0 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0 | 0 |
| 0 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0 | 0 |

Since the labelling tool was designed by us, we could include this “heatmap” feature without much hassle.

**Model Output**

Since the output generated by the model is similar to a heatmap, we require a threshold value to decide if a pixel is part of an edge. This is arbitrarily set to be 0.8.

**Future Plans**

There are still many tasks that remain to be completed.

Firstly, all the remaining images must be labelled and augmented for model training.

Secondly, using the training data, the model must be optimised in terms of layer configurations, batch size and number of epochs.

Lastly, an algorithm to extrapolate disjointed line segments to form a 3D drawing must be done.