

BTP Project

AR / VR based Furniture Marketplace

By - B21SK01

Sayam Kumar - S20180010158
Nitin Kumar Chauhan - S20180010119
Dasari Jayasree - S20180010047

Outline

- Objective
- Problem Definition
- Related Works
- Dataset
- Model and Loss functions
- AR Core Functioning
- Live demo
- Future tasks

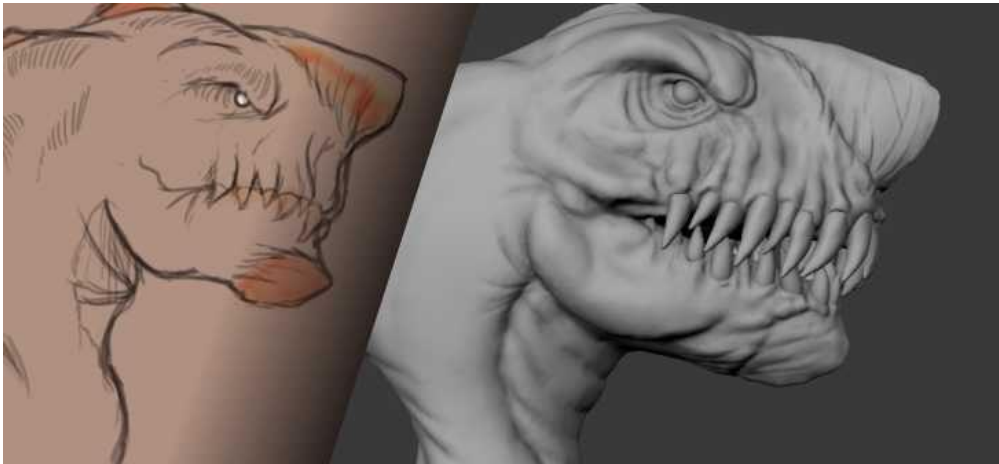


Objective

- Use Augmented Reality / Virtual Reality to simplify the work of furniture showrooms and Interior designers.
- Scan furniture and store them in central place.
- Improve customer engagements by allowing them project furniture in rooms.

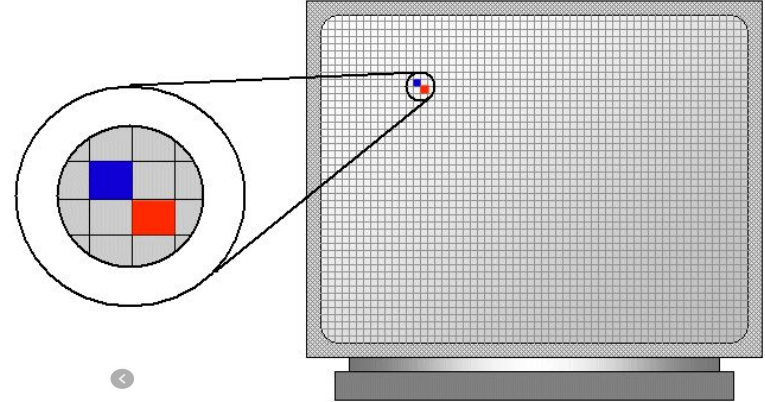
This evaluation -

- Created a novel end-to-end pipeline from 2D image to 3D model shape.
- Represented the generated 3D model in virtual space.

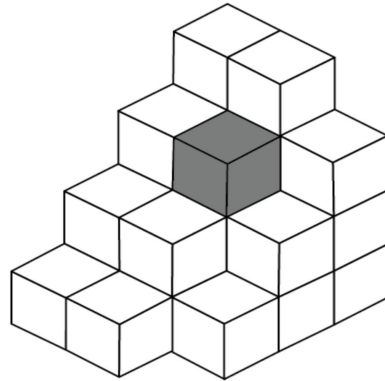


Terminology

Pixel - Smallest unit for a 2D image. All images are formed by stacking and arranging pixels.



Voxel - Smallest cubical unit for a 3d structure.



Problem Definition

- Given an input Image (I), generate a 3D model (M) for it.
- This can be seen as a Supervised Learning problem. Let's say, we make use of hypothesis function (h).
 - $M = h(I)$
- Further breaking it down,
 - We have ground truth Image I and it's true model M from the dataset.
 - Our network learns a transition from pixel representation of 2D image to voxel representation of 3D model.
 - We compare the generated 3D model with true 3D model and use sparse loss functions to train for visually appealing results.

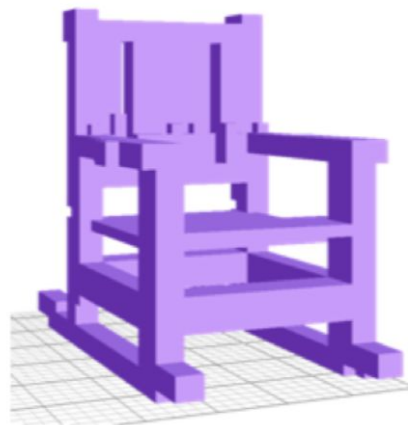
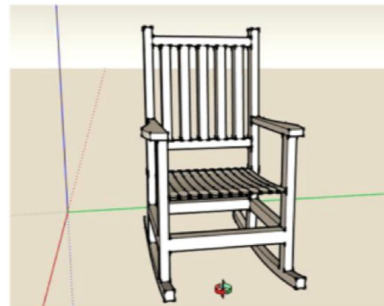
Pipeline Overview

2D image



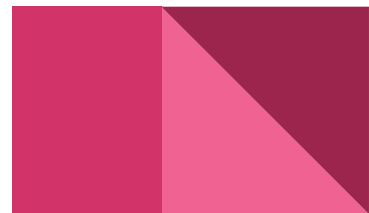
Model

3D model



Voxel
(volumetric pixel)

Loss



Related Works

- Geometry-based Reconstruction - Make 3D objects from 2D images by satisfying geometry constraints and adding prior knowledge. Eg - If I am modelling sphere, I make sure the 3D model has equal radius from center to surface ([Moll et al, 2015](#)).
- Learning-based Reconstruction - These methods use data driven approaches for 2D-3D mapping. Initial works are described in 3D Recurrent Neural Network (3D-R2N2) ([Choy et al, 2016](#)) paper which uses multiple views of images and Long Short Term Memory (LSTM).

Our method uses data driven approaches with Encoder-Decoder networks as hypothesis function.

Dataset

- Used Shapenet dataset. Original shapenet
 - Is built by generating 3D models using Computer Aided Design (CAD)
 - 270 categories of objects
 - 51k unique 3D models
- Our data pipeline -
 - 2 categories - chair and table
 - 2200 chair models and 2539 table models
 - 12 2D views for the 3D model as images captured by 30° rotation around vertical axis.
 - Images in [png](#) format and 3D model in [obj](#) format.
 - Background from images is removed if any.

Learning Representation

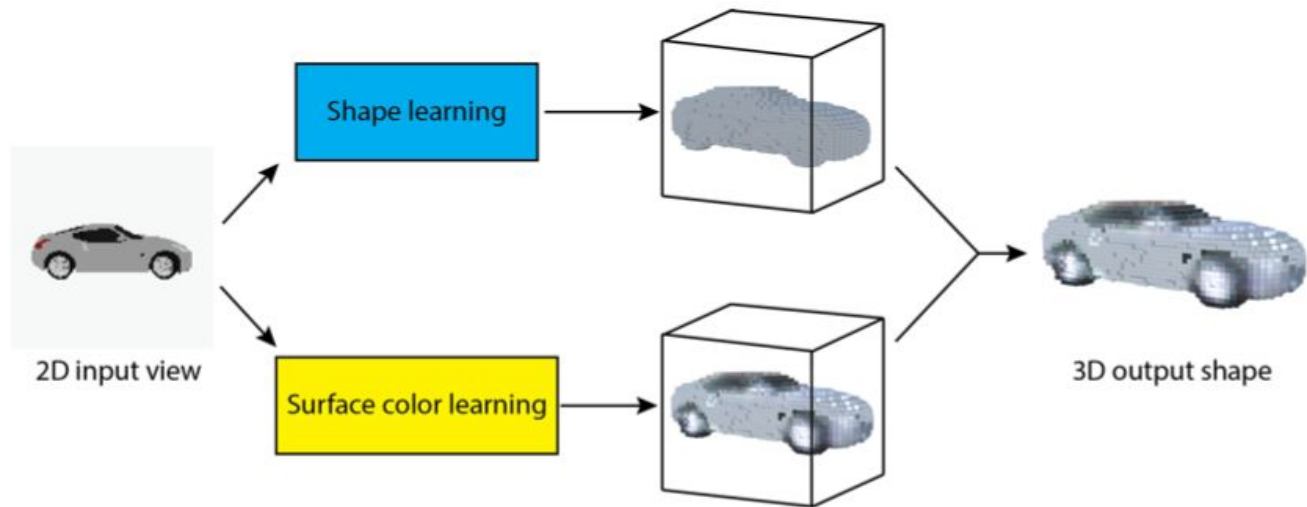
We need to learn two things from an image - 3D model shape and 3D model texture.

- Joint Learning
 - Learn both tasks simultaneously
 - Each occupied voxel is learnt by 0 or 1. 0 means unoccupied voxel and 1 means occupied voxel.
 - For color, the network learns three numbers for RGB for occupied voxels and a vector $[-1, -1, -1]$ for unoccupied voxels.
 - Learn 4 numbers in total.
 - A bad approach - For sparse models (like chair) and imbalanced voxels, the network pushes more towards learning unoccupied voxels better.

Learning Representation

Our Approach

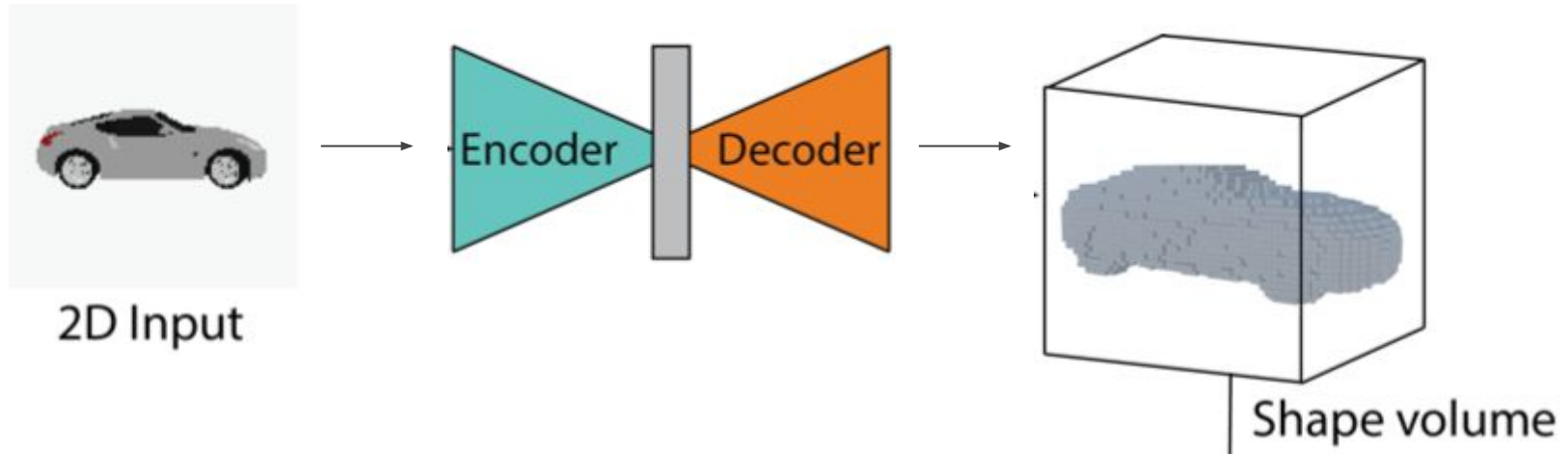
Learn both tasks separately.



This avoided the training difficulty due to shape color imbalance

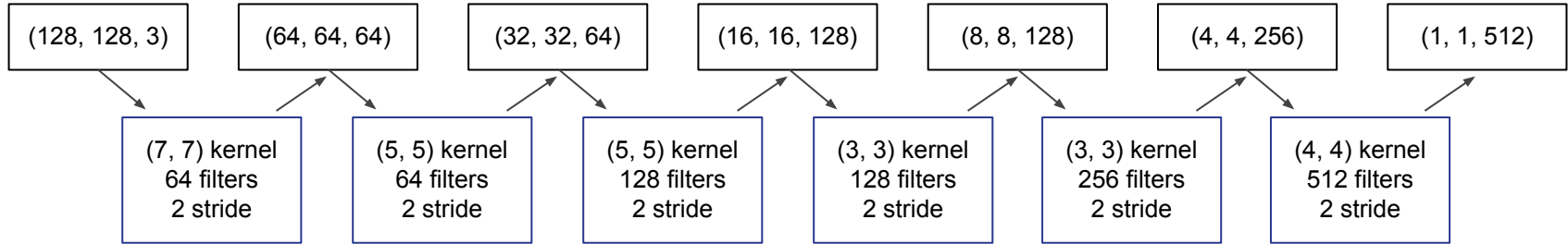
Shape Network

- Given an input image (I), the encoder network (e) first learns the latent representation and then is fed into a decoder network (d).
- The decoder generates a 1 channel shape volume $V' = d(e(I))$

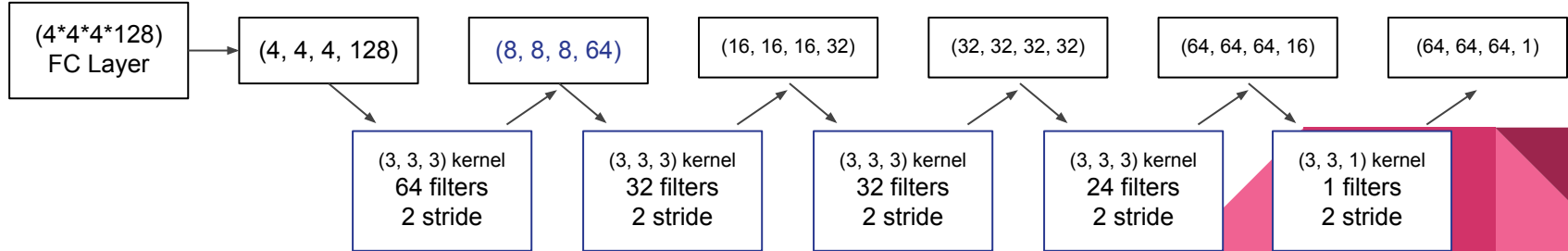


Shape Network

Encoder - 6 2D convolution layers (conv layers for weight sharing and focussing on local relations)



Decoder - 5 3D convolution layers



Shape Network Loss Functions

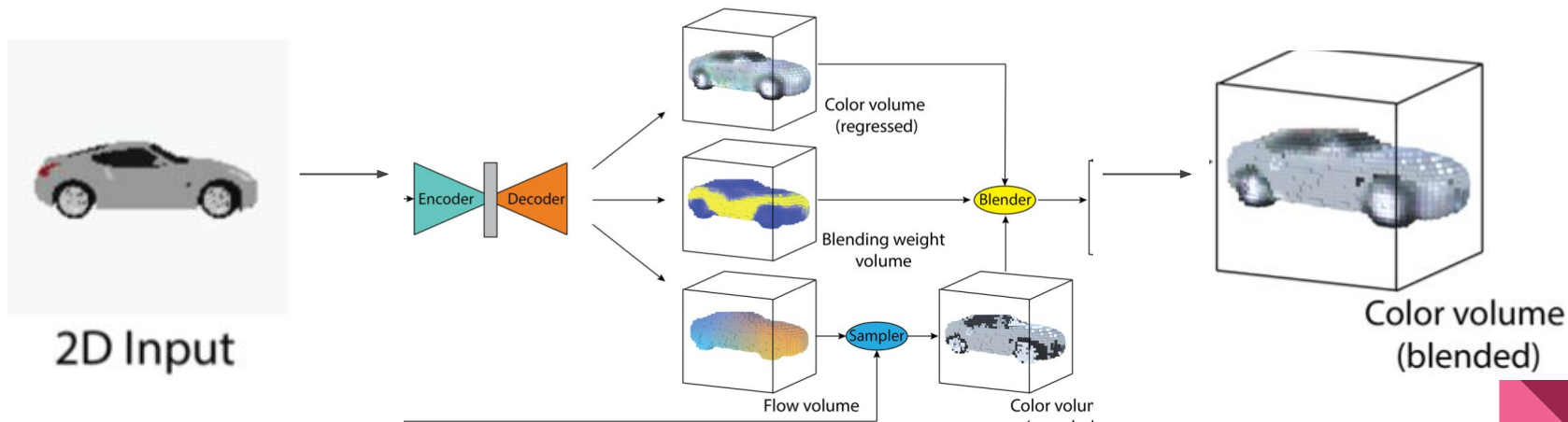
- L2 loss = $\sum_i (V_i - \hat{V}_i)^2$
where i is voxel index and V_i is state of voxel occupancy $\{0, 1\}$
- Moved to Cross Entropy loss
Loss = $-\sum_i [V_i \log(\hat{V}_i) + (1 - V_i) \log(1 - \hat{V}_i)]$
- To handle sparse relations better, we modified it to use two combined losses -
 - False Positive Cross entropy on unoccupied voxels
 - False Negative Cross entropy on occupied voxels

$$FPCE = -\frac{1}{N} \sum_{n=1}^N [V_n \log \hat{V}_n + (1 - V_n) \log (1 - \hat{V}_n)]$$

$$FNCE = -\frac{1}{P} \sum_{p=1}^P [V_p \log \hat{V}_p + (1 - V_p) \log (1 - \hat{V}_p)]$$

Color Network

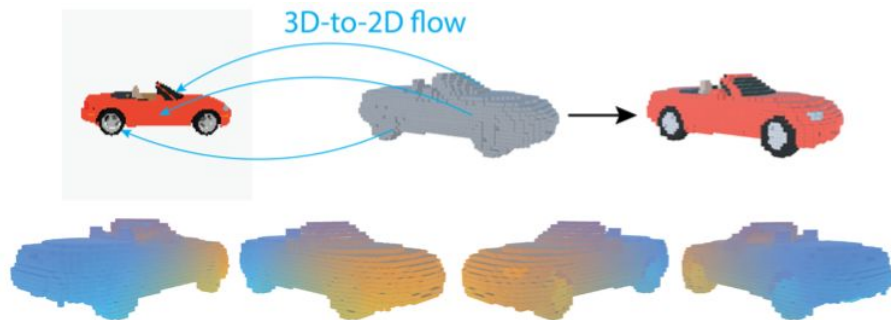
- For getting the colors, we build the same 3d structure. Each 3D voxel gets its shape from corresponding voxel index.
- Two combined strategies - sample color from 2d image and other regress colors directly.



Sampling Strategy

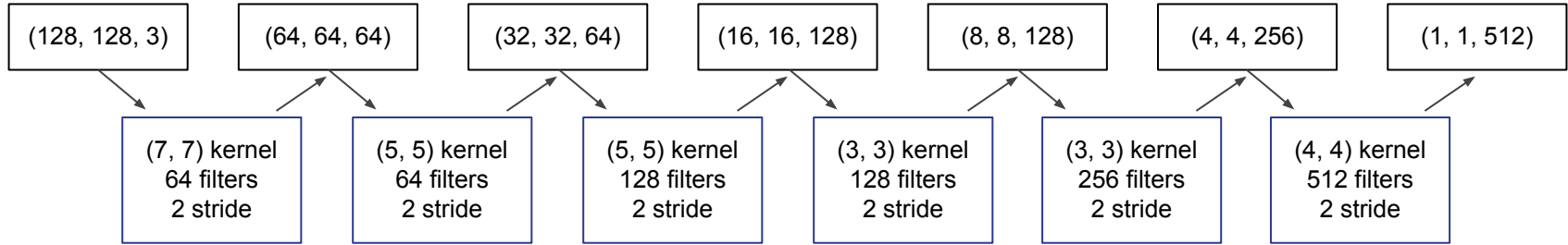
- A 2d to 3d view is created by aligning 2d image and its 3d structure.
- We make the color network learn a matrix $K[R \ T]$ to obtain target flow vectors.
- For hidden (occluded) points -
 - Make use of symmetries
 - Sample colors from nearest neighbour

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim K \begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

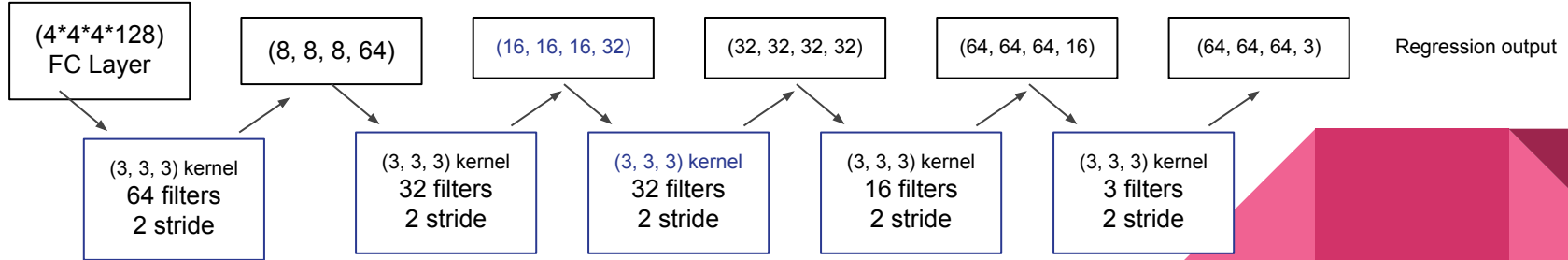


Color Network

Same Encoder - 6 2D convolution layers



Decoder - 5 3D convolution layers



Regression and blending

- We use another strategy of Regression to infer 3D surface colors. This will give better results for high rough regions.

Loss functions

- Target flow loss
$$L_{flow} = \frac{1}{S} \sum_{i=1}^S \|V_i^{flow} - \hat{V}_i^{flow}\|_2$$
- Regression loss
$$L_{clr_regress} = \frac{1}{S} \sum_{i=1}^S \|V_i^{color} - \hat{V}_i^{clr_regress}\|_2$$
- Weighted loss
$$\hat{V}_i^{clr_blend} = w \times \hat{V}_i^{clr_sample} + (1-w) \times \hat{V}_i^{clr_regress}$$
- Blend loss
$$L_{blend} = \frac{1}{S} \sum_{i=1}^S \|V_i^{color} - \hat{V}_i^{clr_blend}\|_2$$
- Final loss
$$L = L_{flow} + L_{clr_regress} + L_{blend}$$

We only use Target flow during training.

We only train for colored voxels.

Training

- Train / Test Split = 0.9 / 0.1
- Adam learning optimizer
- Learning rate = 0.0003
- Batch size = 60
- Training Epochs = 500



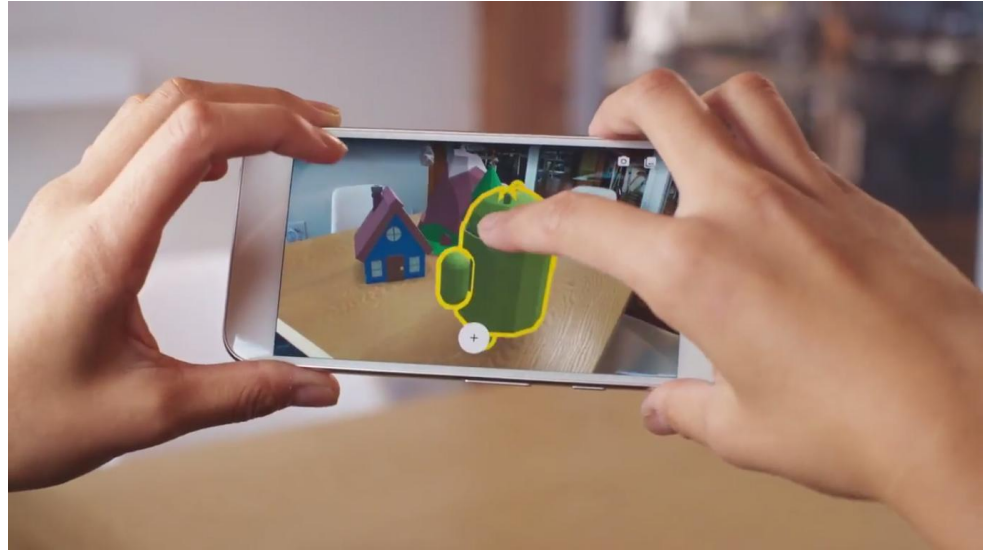
Tech Stack

- Tensorflow
- cv2
- Numpy
- Pillow
- Scipy
- Python




AR Core

- AR is a system that incorporates three basic features: a combination of real and virtual worlds, real-time interaction, and accurate 3D registration of virtual and real objects
- The generated 3D model can be downloaded and placed in real world after calibration of the motion and camera sensors with the surrounding.
- The placed model could be rotated from it's axis or zoomed in/out for perfect placement in the real world.



AR Core Functioning

ARCore uses three key capabilities to integrate virtual content with the real world as seen through your phone's camera:

- Motion tracking - ARCore detects visually distinct features in the captured camera image called feature points and uses these points to compute its change in location.
 - Depth Understanding - ARCore creates depth maps, images that contain data about the distance between surfaces from a given point.
 - Light estimation - ARCore can detect information about the lighting of its environment and provide you with the average intensity and color correction of a given camera image..
- 

App Integration

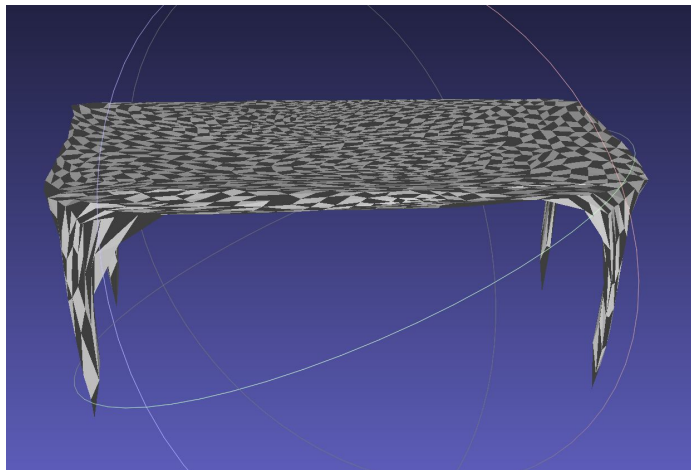
- obj to glb/glTF conversion : glTF (GL Transmission Format) is a specification for the efficient transmission. glTF minimizes the size of 3D assets, and the runtime processing needed to unpack and use them. This format is supported by AR Core.
- Hosting to 3D models : We are using github to host our models.
- Rendering in the App. : The models can be downloaded and rendered in real time.



Demo Output



Table



In real world

Generated by model

Complexity Explosion for color learning

- Number of images = $4.7k * 13 = 61k$ files
 - Only 30% extraction of files possible in drive in 13 hours
 - This is just 2 categories (chair and table)
- Input Dimensions * 3
- Network * 4
- Training will dramatically slow down.

This is what we plan to focus for next evaluation.



Contributions

- Sayam Kumar - S20180010158
 - Literature Review
 - Dataset Creation
 - Model designing and training
- Nitin Kumar Chauhan - S20180010119
 - Literature Review
 - 3D model conversion and rendering
- Dasari Jayasree - S20180010047
 - Literature Review



References

- Tensorflow docs https://www.tensorflow.org/api_docs
 - AR Core docs <https://developers.google.com/ar/reference>
 - Numpy docs <https://numpy.org/doc/>
 - Scipy docs <https://www.scipy.org/docs.html>
 - Im2Avatar paper <https://arxiv.org/pdf/1804.06375.pdf>
 - Wikipedia https://en.wikipedia.org/wiki/Main_Page
- 