Pixel2Mesh Analysis

Leo Sipowicz

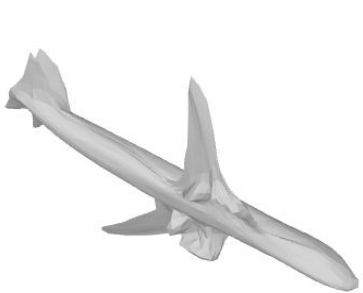
First thoughts:

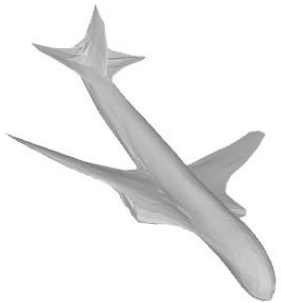
Pixel2Mesh is a Github Project created by Nanyang Wang and collaborators its purpose is to transform 1 png image into a 3D mesh model. It’s written in Python and uses Tensorflow. To use the package you must have the correct packages and a Cuda ready GPU, working on my M1 mac I don't have access to a Cuda gpu so my only option was to run the package on a google colab library created by Mathias Gatti. Once this was running I was able to import all packages necessary about 10 minutes and run the demo function which creates a mesh of a plane PNG. The input and output of this process are scene on the next page.

Plane photo to object

Plane.PNG

5KB

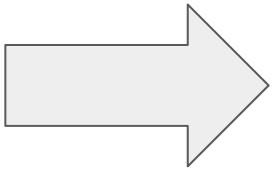




Screenshots of Plane.OBJ

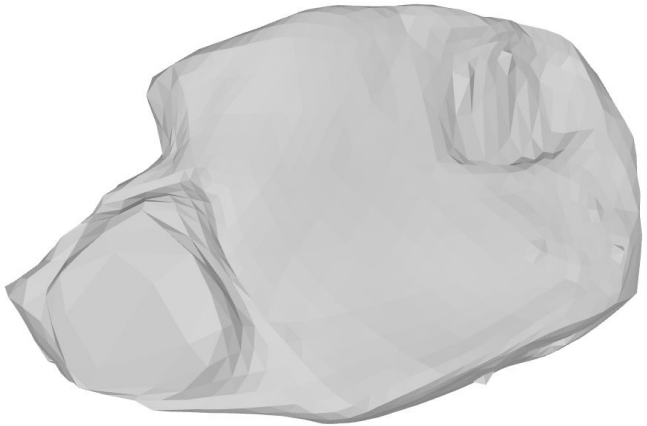
Cup input

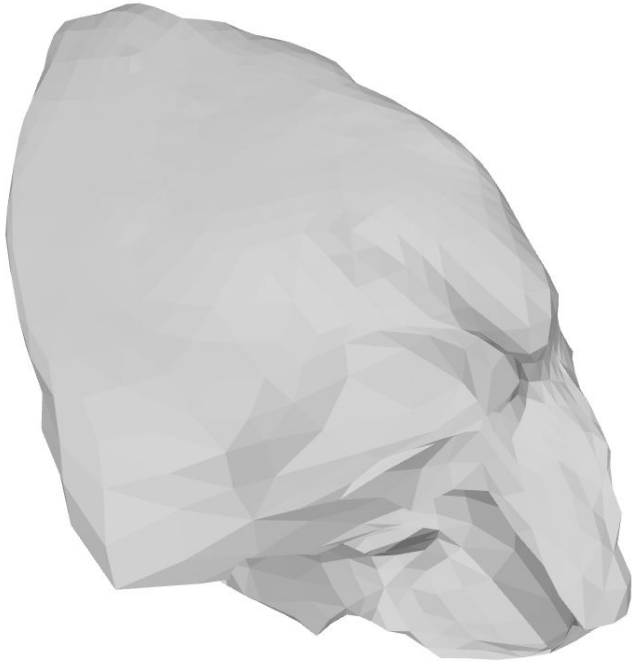
Cup.JPEG



CupNoBackground.PNG

Cup 3D model

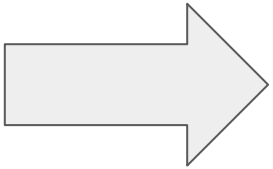




Handle is slightly visible and shape of cup is close

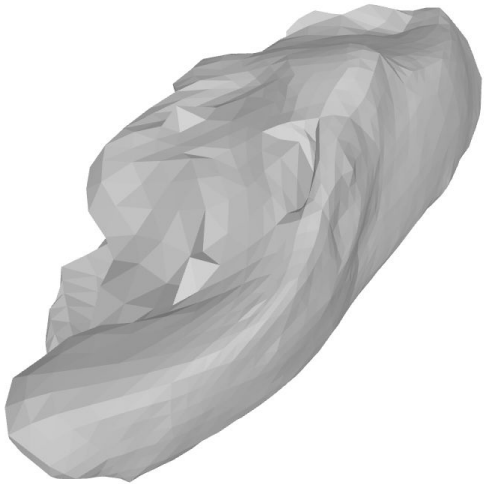
Shoe input

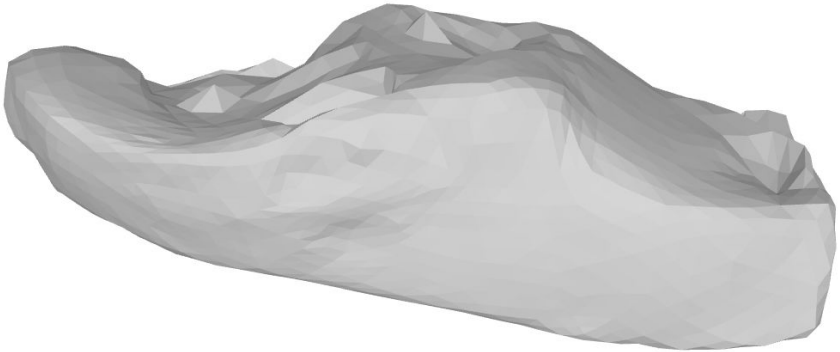
Shoe.JPEG



ShoeNoBackground.PNG

Shoe 3D model





Side profile is close to accurate, top is very rough.

Analysis of Test:

It seems like something is going wrong with my inputs as my models aren't nearly as accurate as the demo images providing. I suspect it’s something to do with the size of my images as they are much larger. It is also possible that the colors of my images are throwing the model off because it was trained on a more muted data set. I’ll now investigate the code and see if I can figure out how the images are being projected to create the models.

Things I have found:

In the demo file I have been working with images are reshaped to 224 by 224 using the skimage library. It's unclear if this could affect the accuracy of my models. But the plane demo PNG was already in 224 by 224 so it could be.

```
def load_image(img_path):
        img = io.imread(img_path)
        if img.shape[2] == 4:
                img[np.where(img[:,:,3]==0)] = 255
        img = transform.resize(img, (224,224))
        img = img[:,:,:3].astype('float32')

        return img
```

Things I have found:

Running the demo is a series of loading and the model which is created from the shapeNet dataset. Then passing the image though the model and seeing what it can predict

```
# Runing the demo
pkl = pickle.load(open('Data/ellipsoid/info_ellipsoid.dat', 'rb'))
feed_dict = construct_feed_dict(pkl, placeholders)

img_inp = load_image(FLAGS.image)
feed_dict.update({placeholders['img_inp']: img_inp})
feed_dict.update({placeholders['labels']: np.zeros([10,6])})

vert = sess.run(model.output3, feed_dict=feed_dict)
vert = np.hstack((np.full([vert.shape[0],1], 'v'), vert))
face = np.loadtxt('Data/ellipsoid/face3.obj', dtype='|S32')
mesh = np.vstack((vert, face))
pred_path = FLAGS.image.replace('.png', '.obj')
np.savetxt(pred_path, mesh, fmt='%s', delimiter=' ')

print 'Saved to', pred_path
```

Next Steps:

For the group project I think it would be better to take on a different library. While Pixel2Mesh is very interesting it doesn't use as many concepts from class as I hoped it would. I thought it would be more based on best fits and projections while instead its entirely based on machine learning and training and predicting based on a model.