Dhruvik Patel

Final Project – Progress Report

Part 1: This part is not completely done, but is close to being done because I was able to create multiple point clouds but still trying to figure out a way to merge all the point clouds together into a single point cloud that shows the 3D reconstruction. Multiple problems were faced when doing this part because in the new version of python, the SIFT() function and all of its function are not available for use, thus I was forced to in looking for a different approach which involved StereoSGBM method. This is a matching algorithm in OpenCV. Following the class implementation listed on OpenCV, I was able to create matching points between two images. After that was done, I was able to compute the disparity between those two images. For the first part I was able to get the disparity between multiple images but still working on how to put all the point clouds created by those disparities. The disparity is converted to 3D points using the function reprojectImageTo3D in the openCV library. Once we have those points we are able to print out the point cloud on the screen and also saves it as a .ply file which can be viewed in MeshLab. When viewing the point cloud in MeshLab, we are able to move it around to see all the different views of the 3D reconstruction. The goal is to figure out how to combine all the point clouds together so that the output point cloud looks similar to the inputted object. Obviously the test images were still basic. For example I am still using a match box. Once I am able to get the point cloud for the match box, which is a rectangle, then I can start working with complicated shapes. Along with this report I have attached the .ply file which can be checked using MeshLab. Also the code that is being uploaded will print a point cloud but it is better if the user uses MeshLab in order to view the point cloud. Below is the results of the part one so far just in case it does not compile and work on someone else’s laptop:

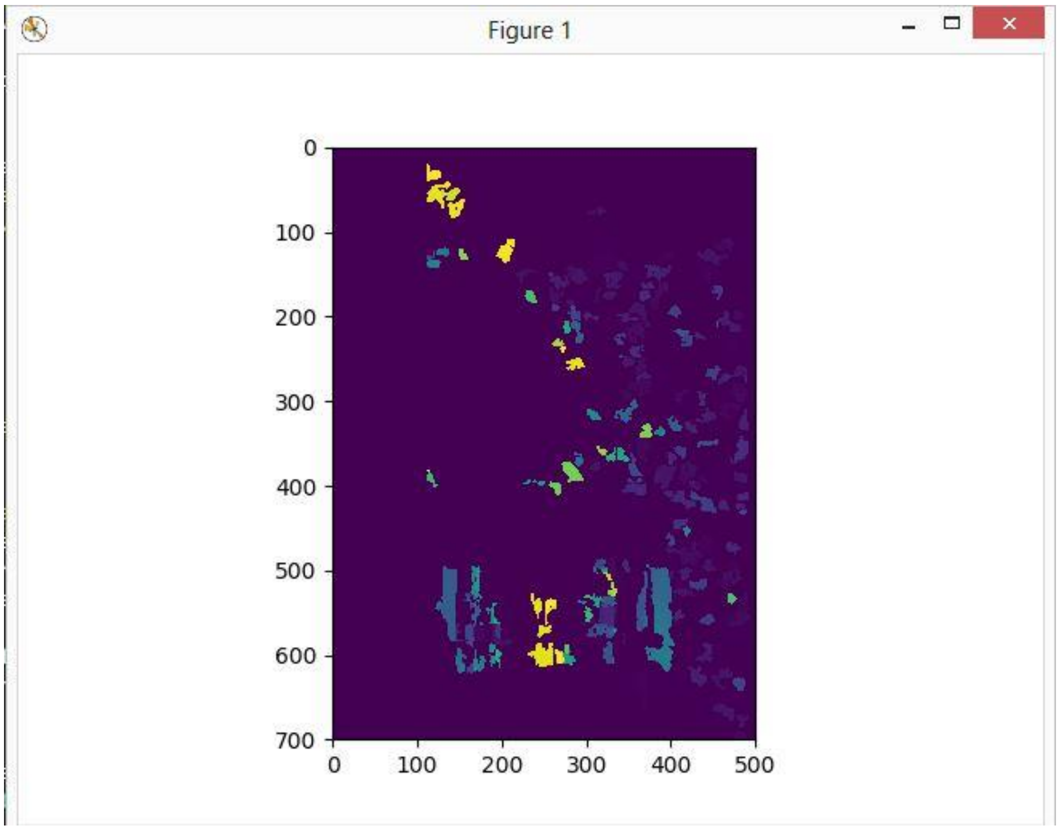
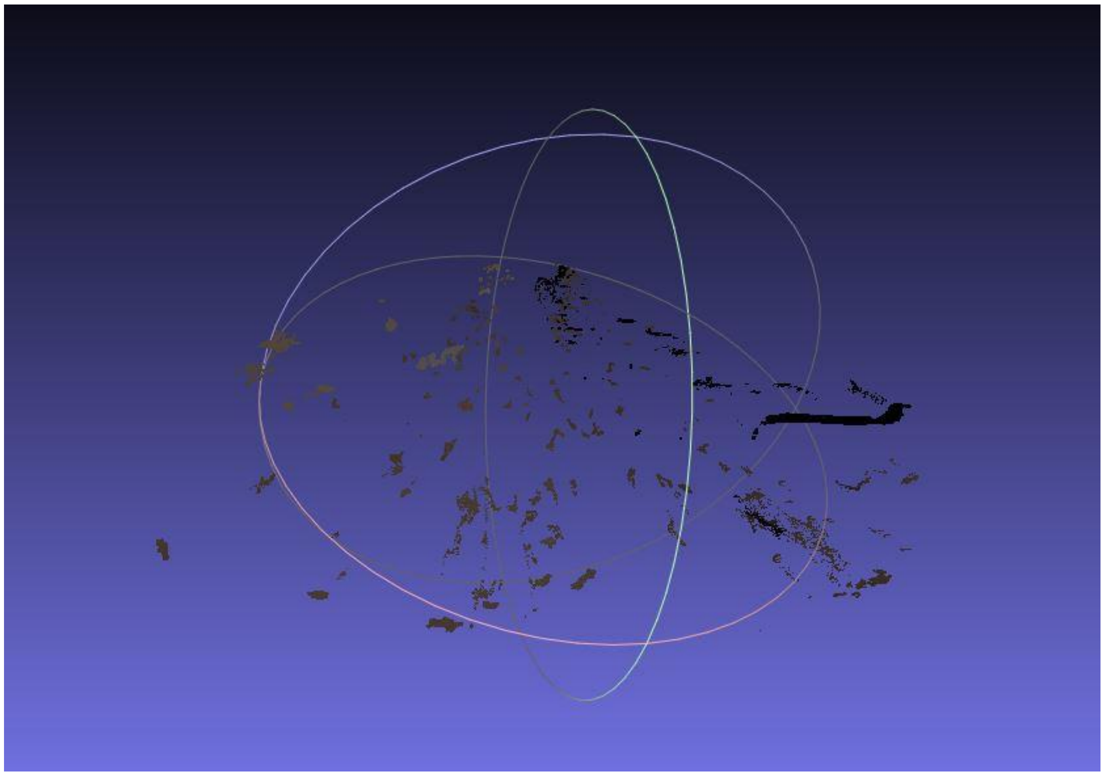
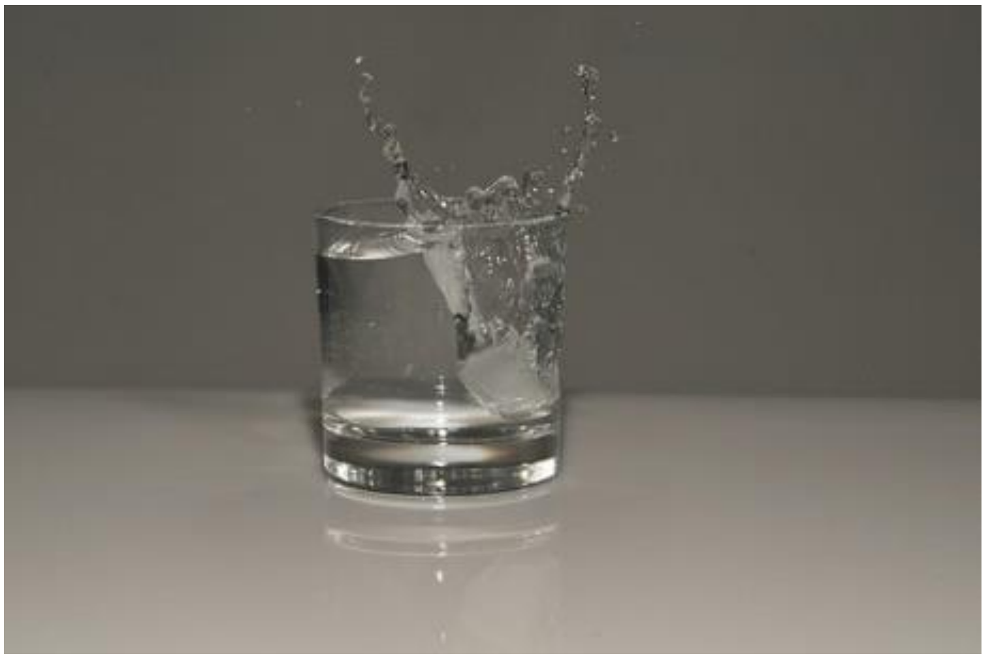


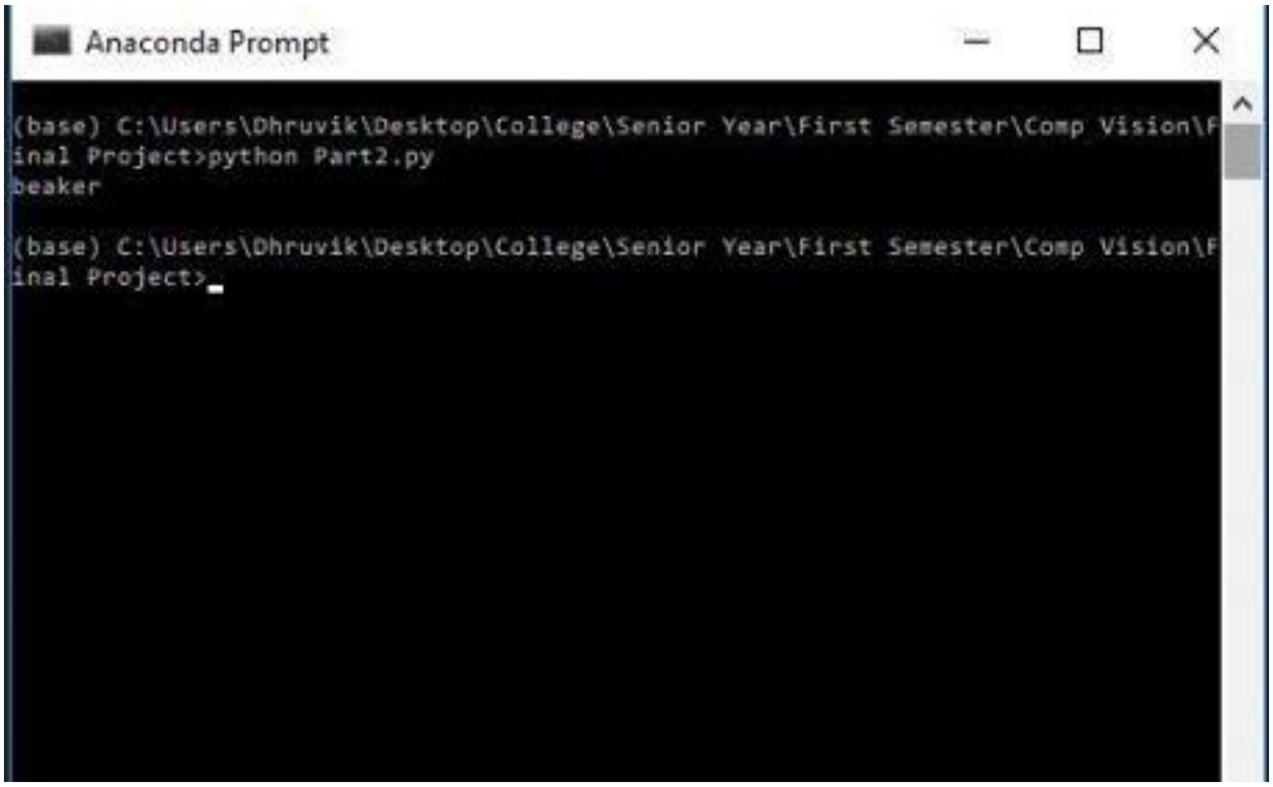
Figure 1
0
100
200
300
400
500
600
700
0
100
200
300
400
500



As you can see, the point clouds are coming together, but I just have to figure out how to combine all the point clouds into one so that the full 3D reconstruction is completed.

Part 2: Getting started with part 2 has been a little difficult because again of the different python version not supporting some things and supporting others. I was finally able to make some headway in part 2 where I just took the simplest photos and outputted a class for that for images. There are multiple different training models we are able to use for the first part where we have to select a pre-trained network to conduct a recognition experiment. I tried it with multiple different networks such as SqueezeNet and CIFAR10. Using Pytorch, I was able to import these training networks and then get a label at the end for an input image. Since we have to pick out our own images, I decided that I will be using better architecture that will enable me to do the recognition experiment better. For the SqueezeNet code, I selected an image that falls under one of the 1,000 ImageNet classes, then I initialized the model, with the pretrained attribute set to True. When I did this, this would have downloaded the weights for the SqueezeNet model. After that I needed to define the preprocessing transform, hence normalize the transform. Without this normalization, the classifier would not work properly. I then downloaded the image form the web using the requests library and created a pillow image. Then the image would be preprocesses using the parameters I set before when I was normalizing. To add dimension to the batch, we need to use .unsqueeze_(0) function. After that a forward pass with the network is ran and then the labels are downloaded. The output of the forward pass was torch Tensor, which basically gives us a numpy array. With that array, we are able to find the index of the maximum element and that would be our predicted class. Then we look in the labels with that key to get the class name and the finally the class name is printed on the screen. Below are the results that I currently have, with only one image:





```
(base) C:\Users\Dhruvik\Desktop\College\Senior Year\First Semester\Comp Vision\F
inal Project>python Part2.py
beaker

(base) C:\Users\Dhruvik\Desktop\College\Senior Year\First Semester\Comp Vision\F
inal Project>
```