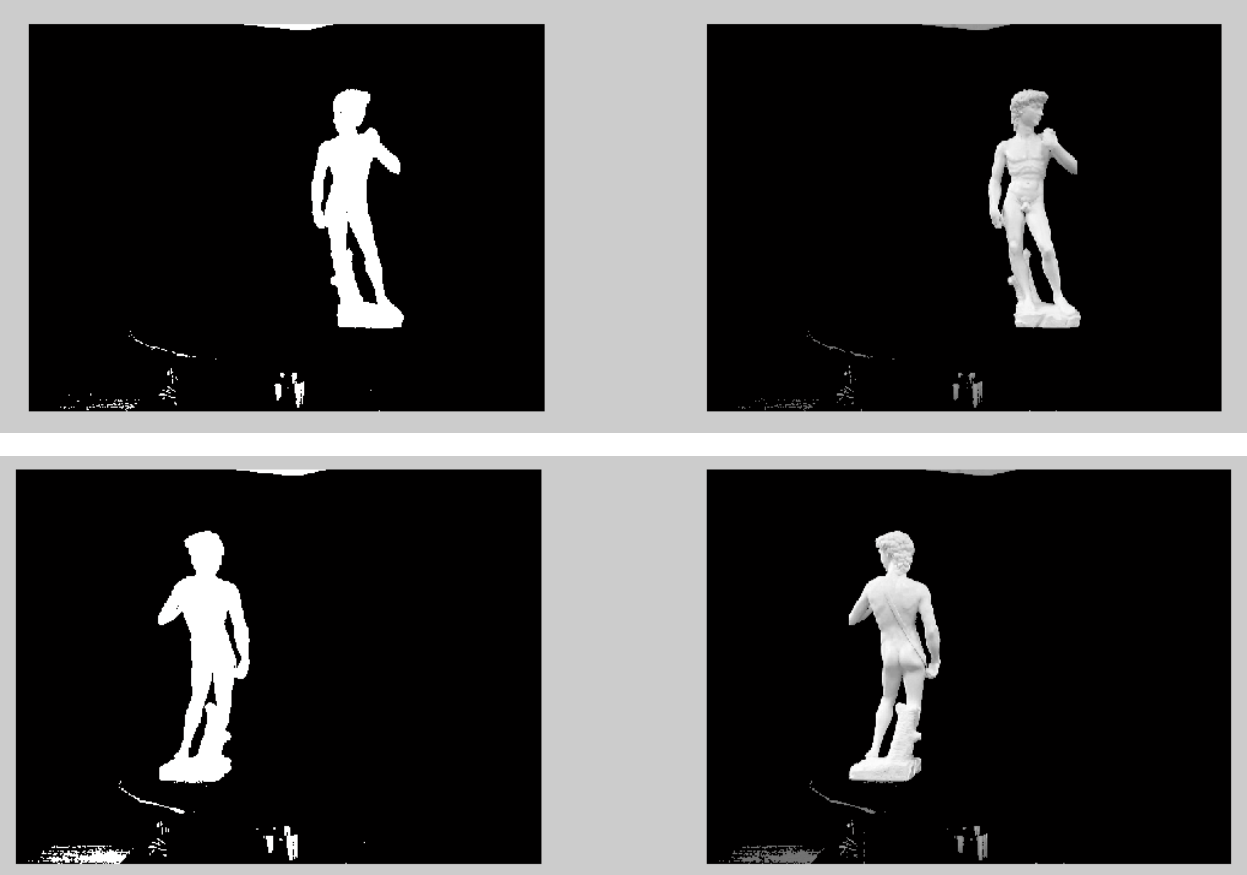
Computer Vision
Exercise 7

Amrollah Seifoddini

Task 1)

For this part, I just increased/decreased silhouetteThreshold gradually until the result silhouettes got clear and continues. With this approach, I reached 110 as the optimal value for silhouetteThreshold parameter. Here are two samples of silhouettes and corresponding image pixels extracted from images.

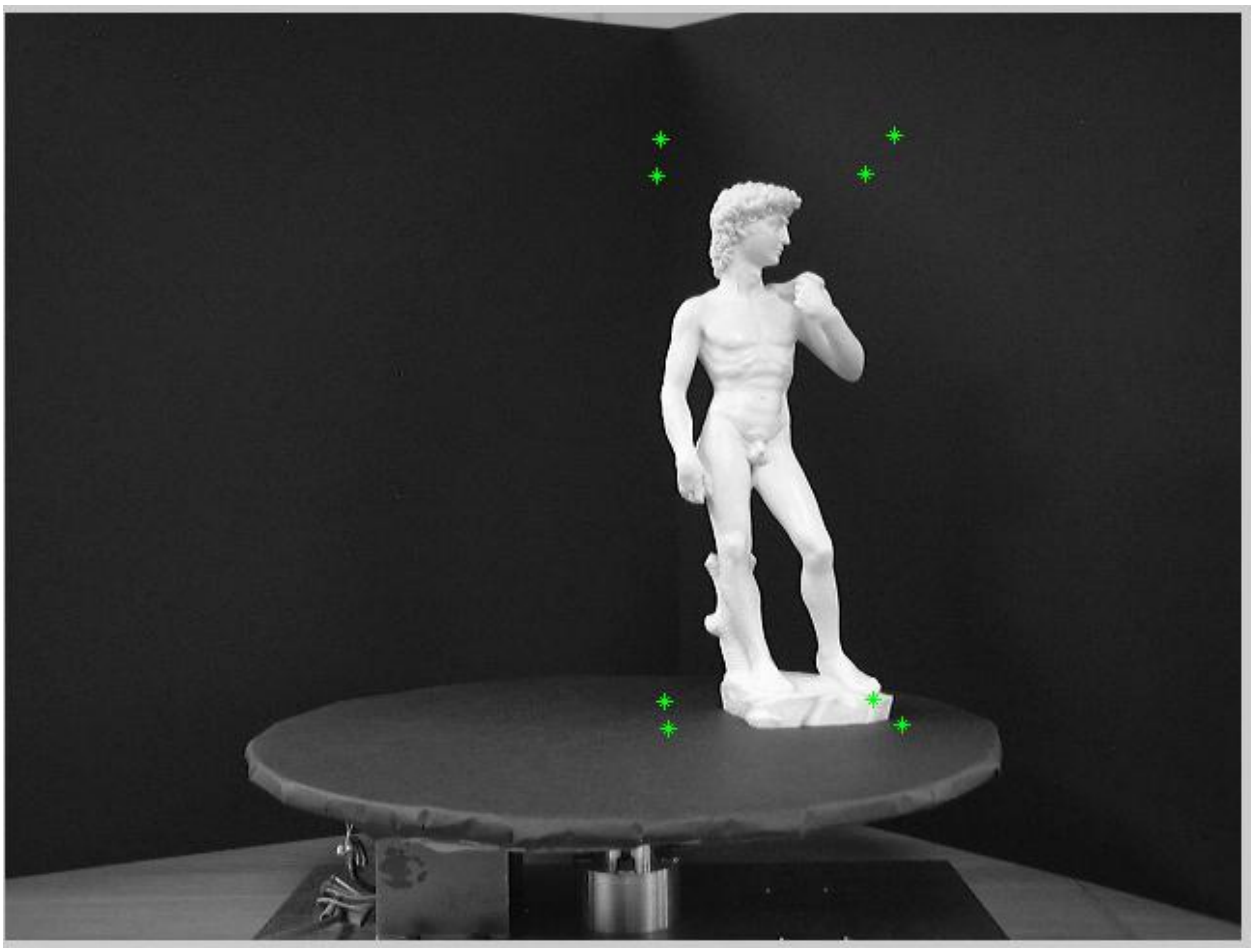


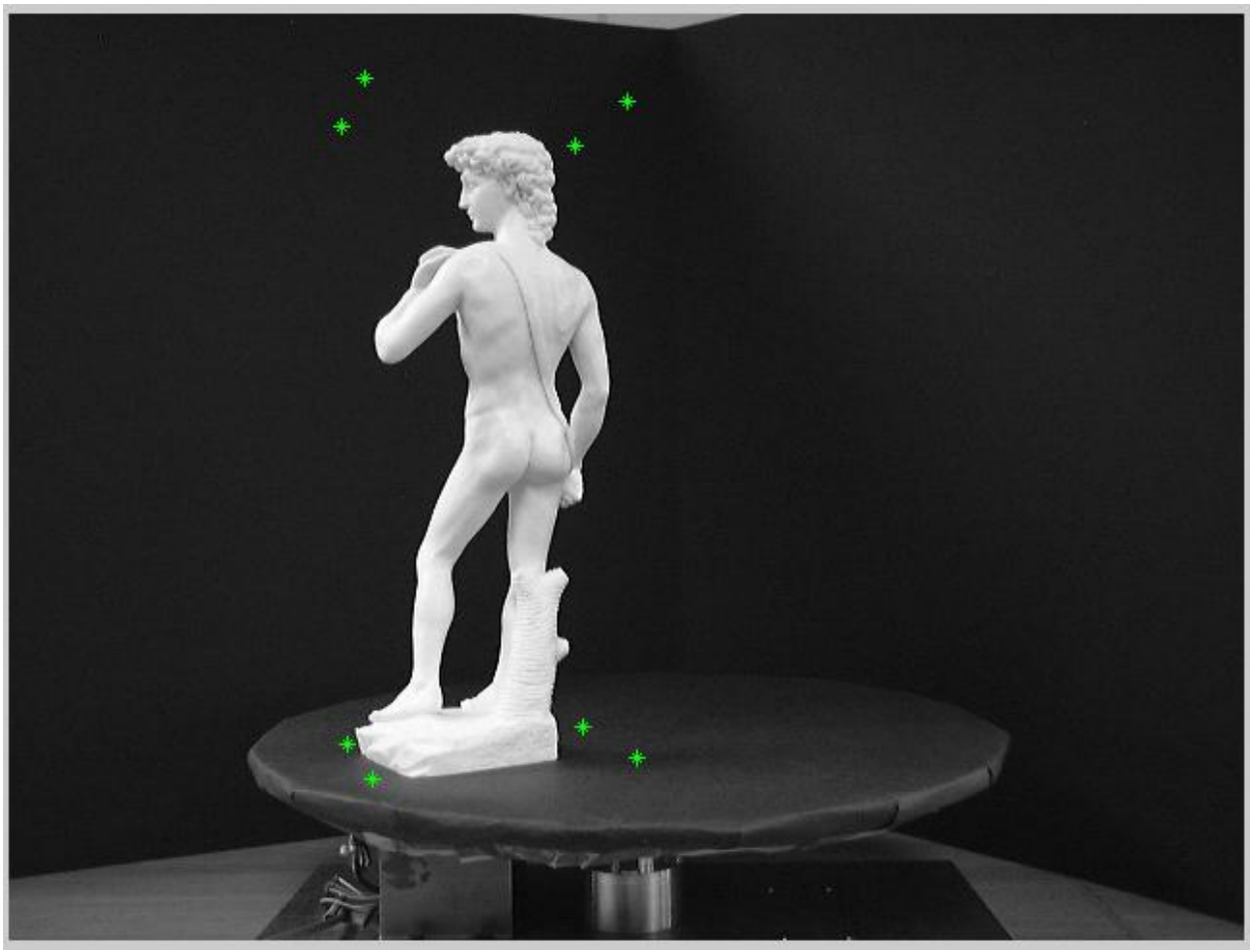
Task 2)

For this part, I also increased/decreased Min and Max for X,Y,Z as the bounding box around the statue. After each change, I checked if the statue fits in the bounding box or not. With considering the parts that fall outside of box or fall far from box edges, I decided to adjust which number out of those 6 parameters. At the end, my optimum bounding box which is tight and also covers the statue completely is this:

```
bbox = [.35 -.1 -1.8; 2.1 1.1 2.6];
```

Here are two sample of images with bounding box corners shown around statue:





Task 3)

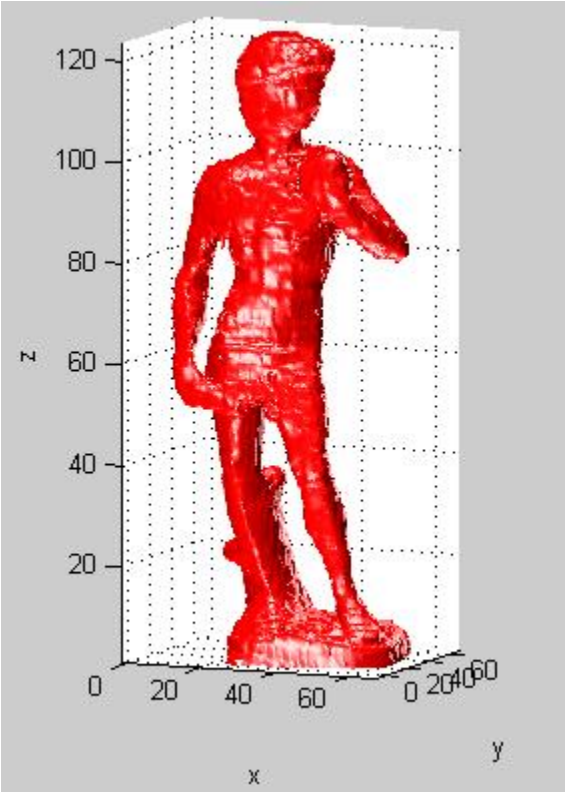
I set volume resolution for this part as follows (according to exercise slide guids):

```
volumeX = 64;
volumeY = 64;
volumeZ = 128;
```

Then I iterate over the whole volume by discrete voxels. For each voxel, I iterate over all the cameras and project that voxel center to the image coordinates using the transformation provided in code framework and camera matrix for to current picture. Then, I divide all point coordinates to the last one, because it is homogenous coordinate. Then I round the result x,y to the nearest integer, because we need to check for corresponding pixels in silhouettes and they have integer indices. Then, I check if this point coordinates fall inside of image borders at all, if it is so, I will add the value of corresponding pixel in silhouette of current image to the voxel which I am iterating on. This means that if point falls inside of silhouette, I will add one and otherwise I will add 0.

After iterating over all voxels of volume, we know that each voxel has been seen in which number of silhouettes. Then, we threshold the result to just consider those voxels that has been seen in at least 17

mages out of 18 images we had. This threshold is because some parts are not visible on all cameras. The rest of code is for plotting volume slices and 3D model. Here it is the 3D model result of my code. It also has been attached as a .fig file to this report.



Task 4)

The space carving method which I explained in task 3 is a method for extracting 3D model from image silhouettes. In this method, we first take pictures in different viewpoint from an object, then we extract the object silhouette from the images. The computed silhouettes for every image along with the center of the corresponding camera is then used to define a volume which if back projected to 3D space can be assumed to bound the object. The intersection of these volumes associated with the set of acquired images yields a reasonable approximation of the real object. This approximated volume is called the visual hull. One of the problems associate with this method is limitation for reconstructing non-convex objects like a cup. In these cases the concavities will not be in reconstruction. Second problem with this method is depth ambiguity resulting from silhouettes. For example, if a person keeps his hand in front of his chest or keep it on his back, the resulting silhouettes are the same. So, this make the algorithm go wrong about visual hull. The third problem is about silhouette extraction which needs a fairly homogenous background and ideally different than object intensity. Otherwise, the resulting silhouette and consequently the 3d model would be not accurate.

Solutions:

For overcoming to these problems I think we should extract more information from images. For example we can use the shadings[1], occlusions[2], colors[3], edges and texture as the clues for perceiving depth of pixels in image. We might also use stereo algorithms to extract disparity map and depth from neighbor images. This is with the assumption that we don't want to (or cannot) use depth camera in the first place. So, after having the estimated depth images (using iterative combination of clues above), we use the same approach as SFS[4] but this time we add a weight corresponding to pixel depth to the associated voxel. Of course we set the background depth as zero. Then if we threshold the value on voxel, a nice 3d depth map would have been reconstructed using a modified version of ISO surface. It is helpful if we re-project the resulted shape to each camera viewpoint to get new 2d depth images. Then we can repeat the shape reconstruction with new images to refine the final shape more. We can do this until the amount of changes between two respective shapes goes below some threshold. I hope with this approach we would be able to perceive some concavities (that are visible to cameras) in objects like a cup. I know this method needs lots of optimizations and is not complete with the steps I described here, but this is what I can think of, now.

As the other simpler (and faster) approach we can extract shape using SFS, then refine it by using pairwise stereo matching between neighbor images to constrain the visual hull more accurately. If we only care about accuracy and not time, there are bunch of cool algorithms for background subtraction which help us get a more precise silhouettes even in a non-homogenous environment[5].

Mesh deformation for topology, may also be helpful in this problem but I am not familiar with that to propose a method.

[1] Depth from Shading Based on 2D Maximum Entropy (by Shuzhen Wang)
[2] Revisiting Depth Layers from Occlusions (by Adarsh Kowdle et. Al)
[3] Only applicable after removing background.
[4] Shape from silhouette
[5] To diminish impact of third problem of SFS