Computer Vision

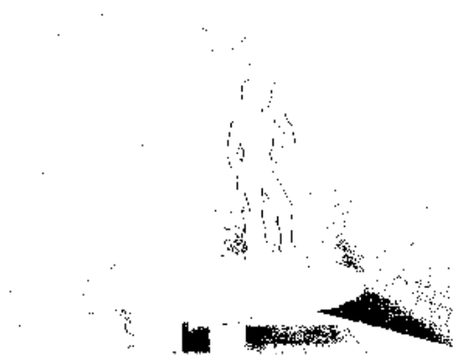
Exercise 9: Shape from Silhouettes

Viviane Yang 16-944-530

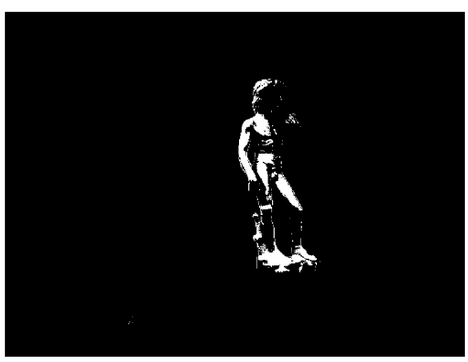
4th December 2019

1 Silhouette extraction

In this subtask, we want to find a good threshold value such that the important parts, which in our case is the whole figure will be differentiated from it's background. All points with a gray value $\in [0, 255]$ higher than the threshold, will be classified as part of the silhouette.

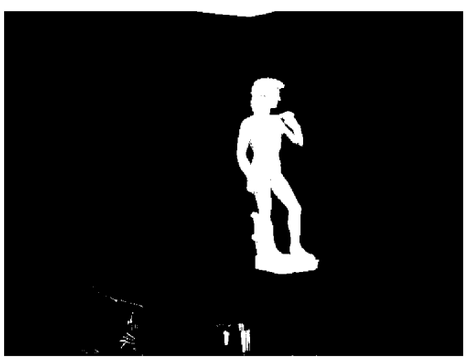


(a) $thres = 20$, which is too low, part of the background is also classified as the silhouette



(b) $thres = 220$, which is too high, parts of the silhouette are not correctly classified

Figure 1: Choosing a incorrect threshold value



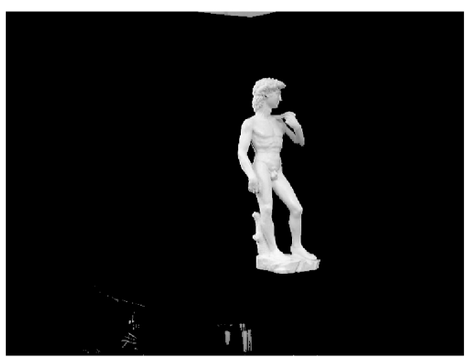


Figure 2: $thres = 120$ works well for detecting the whole silhouette, parts on the roof and on the table that are misclassified are handled by the volume of interest

2 Volume of interest

The volume of interest is defined using 6 parameters of the `bbox` and 3 for defining the number of interval across each axis `volumeX`, `volumeY` and `volumeZ`. The `bbox` values define the ranges of a cuboid which covers the whole volume of interest. We need this since some points which are not part of the silhouette might get misclassified. So restricting silhouette points to only be in a restricted volume part in the image can filter out those misclassified points. This is why it is important to find a bounding box that fits the silhouette as close as possible.

I chose the following parameters for `bbox`:

	x	y	z
min	0.4	-0.2	-1.8
max	2.2	1.2	2.6

The bounding box using the following parameters can be seen in Figure [3]:

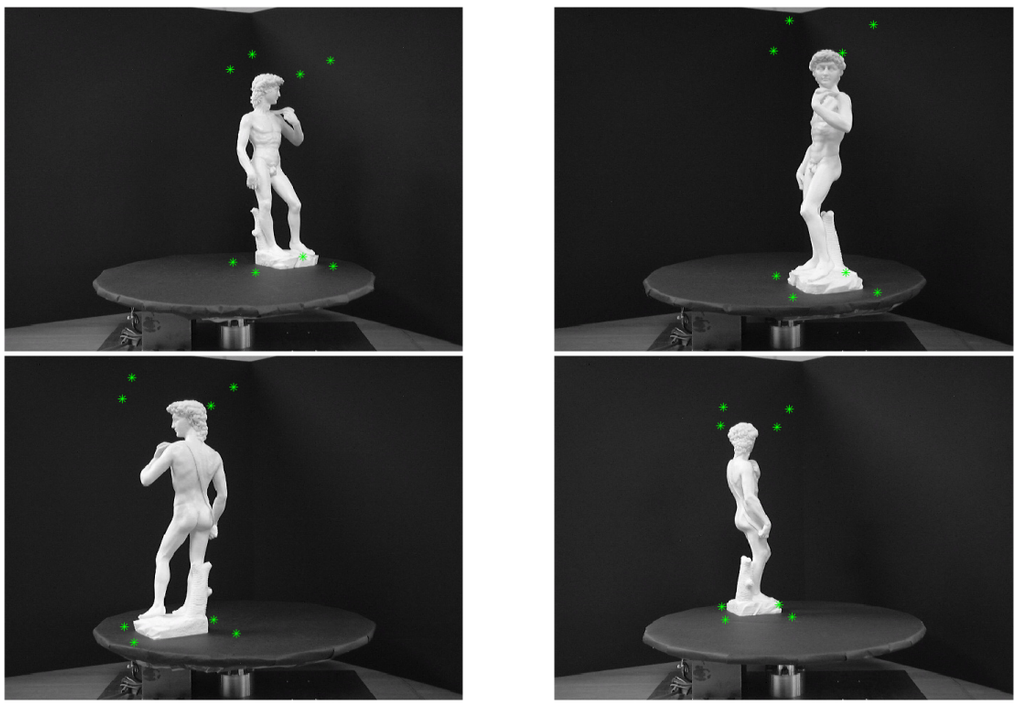


Figure 3: Bounding box

3 Visual hull

The parameters `volumeX`, `volumeY` and `volumeZ` define in how many voxels we want to divide the edge of the bounding box. The number of voxels correspond to the quality or resolution of the 3D model. For each voxel we need to map it to the silhouette image of each camera and count the number of voxels that lie in the region of the silhouette.

```
for i=1:volumeX
      for j=1:volumeY
         for k=1:volumeZ
             voxel = [i j k 1]'; %center of the voxel
             for n=1:numCameras
                 %project to image
                 pvoxel = Ps{n}*T*voxel;
                 pvoxel = pvoxel./repmat(pvoxel(3,:),3,1);
                 xp = round(pvoxel(2)); %change x and y axis
                 yp = round(pvoxel(1));
                 %check out of bound
                 min_x = 1;
                 min_y = 1;
                 [max_x, max_y] = size(sils{n});
                 in_bound = xp ≥ min_x && xp ≤ max_x && yp ≥ min_y && yp ≤max_y;
                 if in_bound
                  if sils{n}(xp, yp)
                      volume(i,j,k) = volume(i,j,k)+1;
                  end
                 end

             end
         end
      end
   end
```

The isosurface that in the end is computed using the volume slices is pretty rough for a resolution of $volumeX = 10, volumeY = 10, volumeZ = 20$. This is shown in Figure [4].

However, using a higher resolution of $volumeX = 64, volumeY = 64, volumeZ = 128$, we are able to find a surface that is much more precise than the one with lower resolution (Figure [5]).

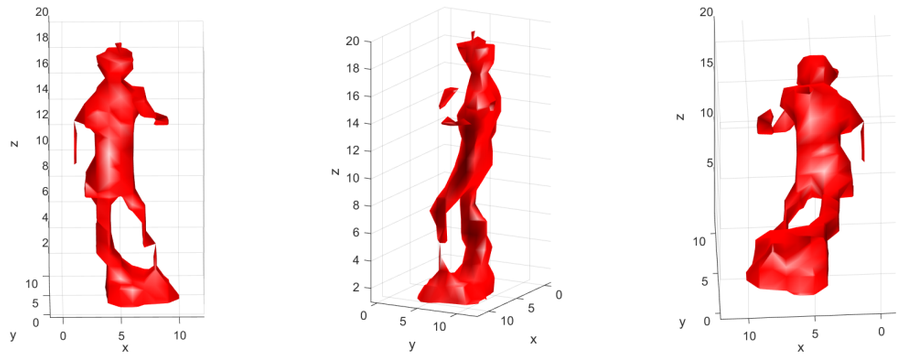


Figure 4: Using a resolution of $10 : 10 : 20$

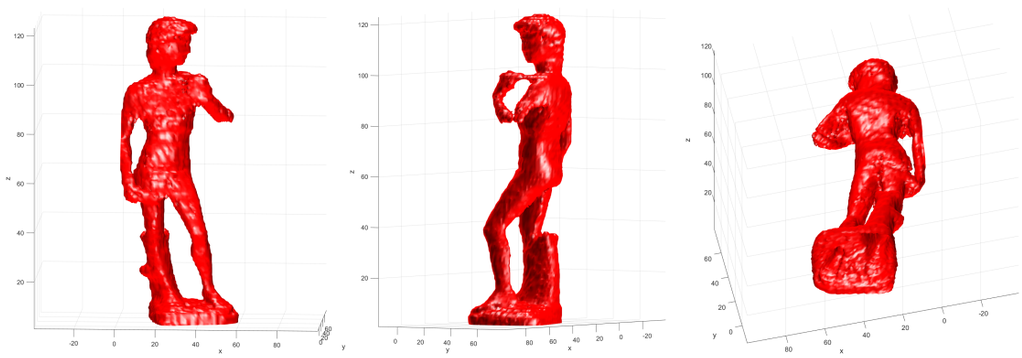


Figure 5: Using a resolution of $64 : 64 : 128$

4 Improvements

The silhouette extraction part only worked, because we used a white statue in a black room, so that it's contrast to the background was easily detectable. Therefore one could convert the image to gray scale and easily use a threshold to detect pixels that are part of the stature. This property however cannot be simply assumed in real world scenarios. We have seen different approaches during the image segmentation exercise which can be used instead like mean-shift or EM segmentation. Using image segmentation techniques, we also save the manual work of drawing the bounding box.
Another idea would be to use graph cut with a disparity map. However, as seen in the stereo matching exercise, graph cut can be quite computationally expensive. However, it returned results which are quite impressive. Using only 2 images (stereo matching), we were able to make an good 3D reconstruction. Here we have 18 pictures. We would need to do the graph cut pairwise and then use a voting scheme to decide on the edge to be cut. With $n = 18$ pictures, if we do an pairwise one-vs-one scheme, that would result in $\frac{n(n-1)}{2} = 153$ graph cut runs, which would take a long time.