# Computer Vision

Lab 09: Shape from Silhouettes

Sumeet Gyanchandani

## 1 Silhouette extraction

In this section, we extract silhouettes from the provided images, using a simple thresholding technique. I just re-adjusted the silhoutteThreshold so that the silhouettes of the statue are clearly extracted. I observed that silhouetteThreshold = 100 is the optimal value. Fig. 1 and 2 show the extracted silhouette of two images.

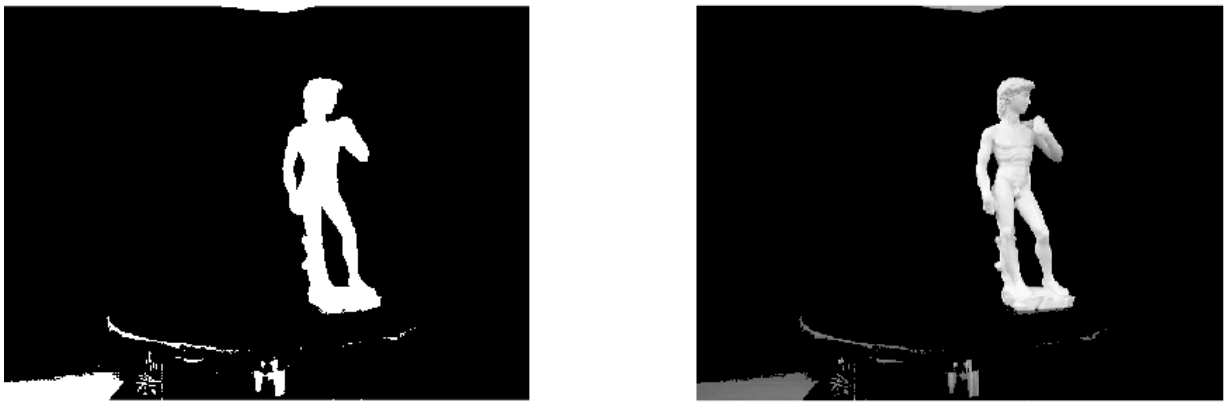

**Figure 1.** Extracted exemplar silhouette from image 0

**Figure 2.** Extracted exemplar silhouette from image 9

## 2 Volume of interest

In this section, I adjusted the min and max values of x, y and z of the bounding box. It took several trials to get a tight bounding box: bbox = [.3 -.15 -2; 2.2 1.2 2.5]. Fig. 3 and 4 show the volume corners around the statue.

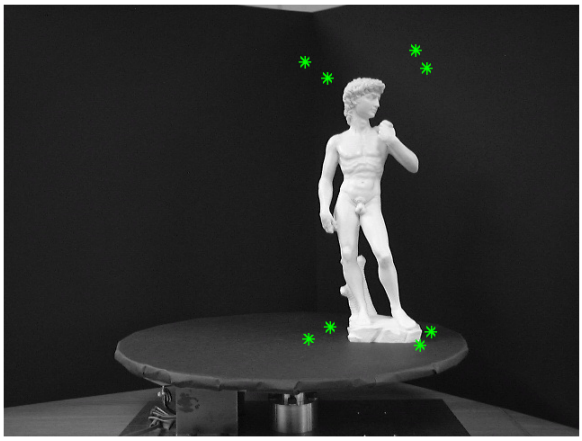

**Figure 3.** Bounding Box on image 1

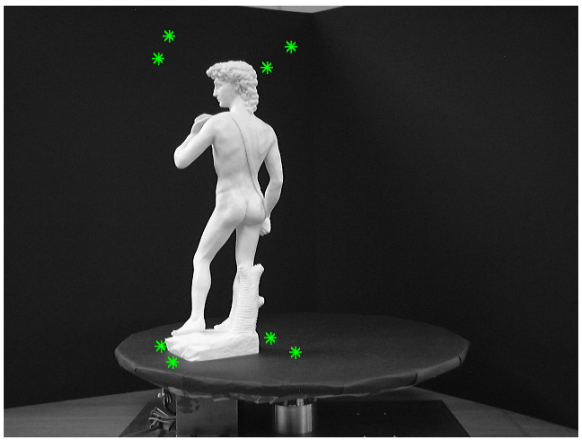

**Figure 4.** Bounding Box on image 8

## 3 Visual hull

I chose $64 \times 64 \times 128$ as the volume resolution because it is much faster to reconstruct a 3D model at this resolution.

First, I compute the occupancy score at each voxel. For each voxel, I iterate over all the cameras and project the voxel center from volume to world coordinates using the transformation provided. I divide the homogeneous coordinate system with the last coordinate (normalize it). We need to round the values of the resulting coordinate system, as the silhouettes have integer indices. Finally, I check if the point lies within the image, if it does, I add the silhouette value to the volume value of that voxel. This way, I add 1 to the score if a projected point falls within the silhouette and 0 if not.

Next, I extract an iso-surface from the volume based on *volumeThreshold*. In our case, it is set to **17**, which means it will consider voxels that have been seen in at least 17 of the 18 images.

Fig. 5 and 6 show the 3D reconstruction from 1 image and 4 images respectively. Fig. 7 through 10 show the 3D reconstruction from 18 images from various angles. This 3D model is also saved as *David3D.fig* in the code folder.

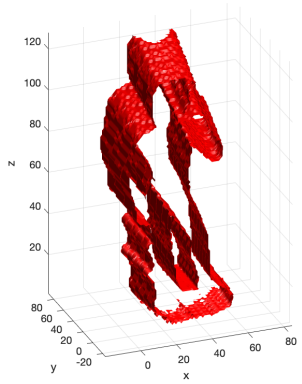


**Figure 5.** 3D model from 1 image

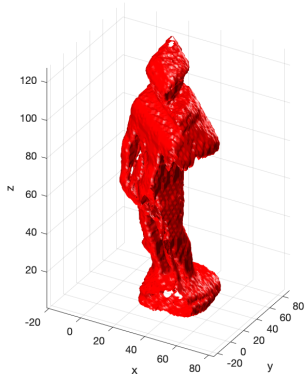


**Figure 6.** 3D model from 4 images

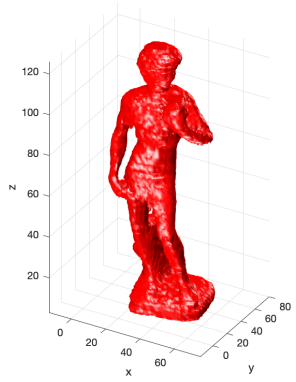

**Figure 7.** 3D model from 18 images-1

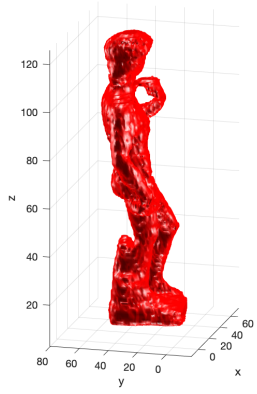

**Figure 8.** 3D model from 18 images-2

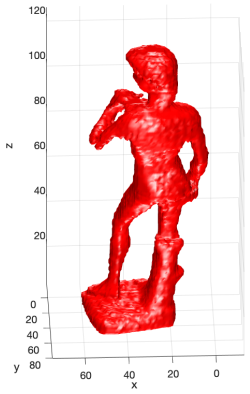

**Figure 9.** 3D model from 18 images-3

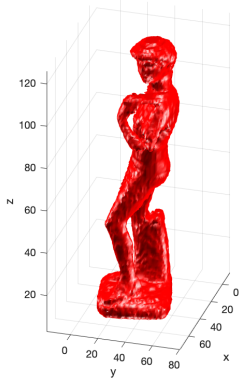

**Figure 10.** 3D model from 18 images-4

Table 1 summarizes the parameter values I chose for our case.

| Parameter | Value |
|---|---|
| Silhouette Threshold | 100 |
| Bounding Box | $[.3\ -.15\ -2;\ 2.2\ 1.2\ 2.5]$ |
| Volume Resolution | $64 \times 64 \times 128$ |

**Table 1.** Parameter values used in the algorithm

# 4 Improvements

I divide this section into 2 parts, problems and probable solutions:

## 4.1 Problems:

- The most prominent problem with this method is that it fails to reconstruct the concavities of the object, if the object is non-convex
- Another limitation is that silhouettes extraction requires a contrast between the object and the background. This heavily affects the accuracy of the 3D reconstruction. Hence, the production studios rely on green/blue screen for such a task

- Reconstruction is not photo-consistent
- Silhouettes are also depth ambiguous, i.e. different depth ordering of objects can result in same silhouette. This results in incorrect visual hulls and consequently inaccurate 3D reconstruction

## 4.2 Solutions:

- Extracting more information from the images is definitely a great idea. More information includes Shape from X, where X can be Shading, Occlusions, Multiple Light Sources (photometric stereo), Texture, Focus/Defocus, Specularities, Shadows. The idea is to perceive the depth of the pixels of the image. If we a have a depth map we can easily set the weight of all the voxels as 0 and add a weight corresponding to pixel depth to the associated voxel. This way the background would be isolated from the weighted object, as it's value would remain 0. This will result in non-ambiguous visual hulls
- Another solution is using pairwise stereo matching between neighboring images to constrain the visual hull
- If the computation isn't a constraint, there are many advance solutions that can get the job done, for instance, we can use scene traversal depth ordering, panoramic depth ordering, multi-pass plane sweep, level-set methods, volumetric graph-cut, adaptive mesh refinement or a mixture of these.