Problem Set #1

Bennett Hellman
6.869 - Advances in Computer Vision
MIT

February 15, 2022

Problem 1: *Perspective and orthographic projections*

Clearly, the lines in Figure 2 are more parallel as compared to Figure 1. This is a result of the second image being an orthographic projection, caused by zooming in on a camera. This could also be demonstrated by the larger difference in widths between the top and bottom edges of the laptop in Figure 1.

Figure 1: Perspective Projection

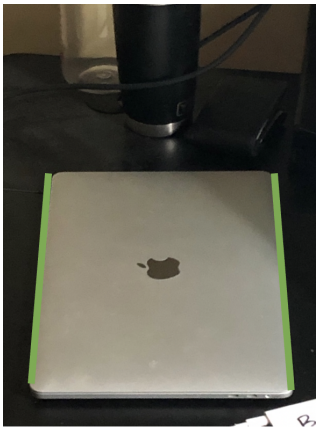

Figure 2: Orthographic Projection

Problem 2 *Orthographic projection equations*

$$\begin{bmatrix} x \\ y \end{bmatrix} = \alpha \cdot P \cdot R_x(\theta) \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

Where

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix}, P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$x = \alpha X + x_0$$

$$y = \alpha Y cos(\theta) - \alpha Z sin(\theta) + y_0$$

Solving the system of equation to get $(0, 0, 0)$ to project onto $(0, 0)$ and $(1, 0, 0)$ to project onto $(3, 0)$, you get $\boldsymbol{\alpha = 3, x_0 = 0, y_0 = 0}$

Problem 3 *Edge and surface constraints*

Using both intuition and the equations from *Problem* 2 we can derive the constraints for the Z dimension:

- Along a vertical edge:

$$\frac{\partial Z}{\partial x} = 0$$

- Along a horizontal edge:

$$\frac{\partial Z}{\partial y} = \frac{1}{-sin(\theta)}$$

- Along a flat surface:

$$\frac{\partial Z}{\partial y^2} = 0, \frac{\partial Z}{\partial x^2} = 0, \frac{\partial Z}{\partial xy} = 0$$

Problem 4 *Complete the code*

```
# Contact edge: dY/dy = ?
# Requires: a transform matrix
if contact_edges[i, j]:
```

```
  Aij[:,:,c] = np.array([[0, 0, 0], [0, 1, 0], [0, 0, 0]], dtype=np.float32)
  b[c]       = 0
  update_indices()

# Vertical edge: dY/dy = 1/cos(theta)
# Requires: a transform matrix, alpha

if verticalsum > 0 and groundsum == 0:
  Aij[:,:,c] = 0.125*np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]],
  dtype=np.float32)
  b[c]       = 1/np.cos(alpha)
  update_indices()


# dY/dt = 0 (you'll have to express t using other variables)
# Requires: a transform matrix, i, j, theta
if horizontalsum > 0 and groundsum == 0 and verticalsum == 0:
  g_norm = np.sqrt(dmdx[i,j]**2 + dmdy[i,j]**2)
  dmdx_norm = dmdx[i,j]/g_norm
  dmdy_norm = dmdy[i,j]/g_norm
  dx_kern = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]], dtype=np.float32)
  dy_kern = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]], dtype=np.float32)
  Aij[:,:,c] = -dx_kern*dmdy_norm +  dy_kern*dmdx_norm
  b[c]       = 0
  update_indices()

# laplacian = 0 (weighted by 0.1 to reduce constraint strength)
# Requires: multiple transform matrices
if groundsum == 0:
  Aij[:,:,c] = 0.1*np.array([[1, -2, 1], [2, -4, 2], [1, -2, 1]],
  dtype=np.float32)
  b[c]       = 0
  update_indices()

  Aij[:,:,c] = 0.1*np.array([[1, -2, 1], [2, -4, 2], [1, -2, 1]],
  dtype=np.float32).T
  b[c]       = 0
  update_indices()

  Aij[:,:,c] = 0.1*np.array([[1, 0, -1], [0, 0, 0], [-1, 0, 1]],
  dtype=np.float32)
  b[c]       = 0
  update_indices()
```

Problem 5 *Run the code*

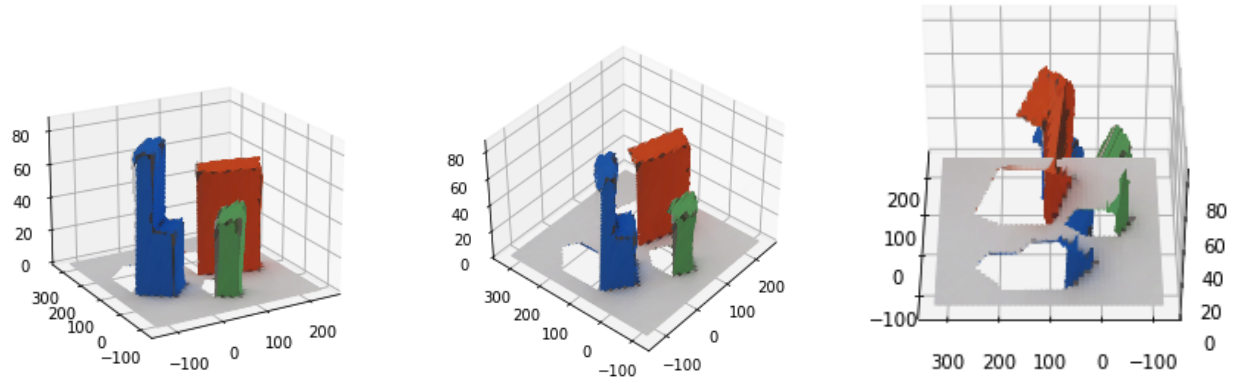

Figure 3: Image 1 from (20,-120), (40,-140) and (40,-180)

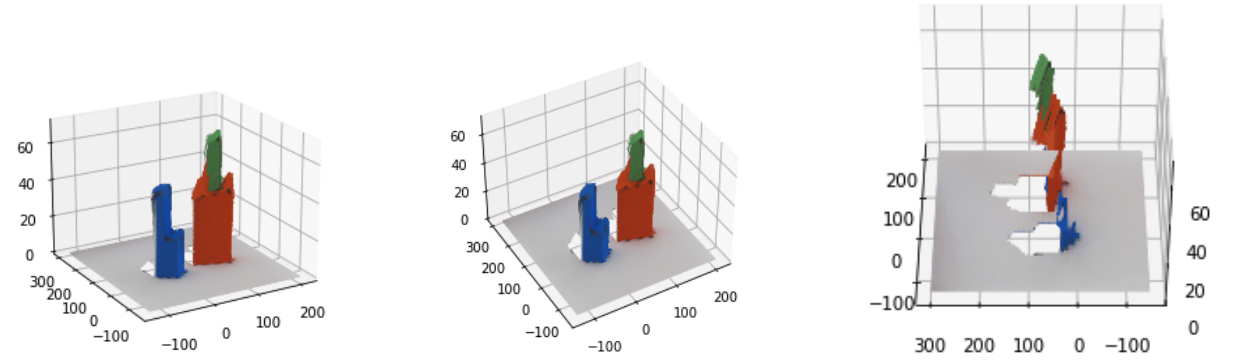

Figure 4: Image 2 from (20,-120), (40,-140) and (40,-180)

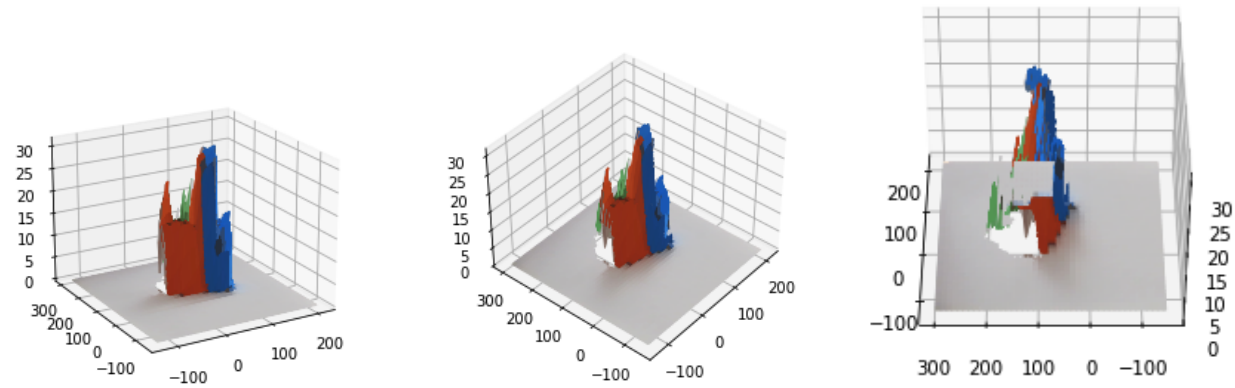

Figure 5: Image 3 from (20,-120), (40,-140) and (40,-180)

Problem 6 *Violating simple world assumptions*

Image 3 violates a couple of our simple world assumptions.
First, "In this simple world, we will assume that objects do not occlude each other (this can be relaxed) and that the only occlusion boundaries are the boundaries between the objects and the ground." (p. 6). Parts of the red shape are occluded by the green shape and the blue shape.
Second, "In this simple world, if we assume that all the objects rest on the ground plane, then we can set Y (x, y) = 0 on the contact edges." (p.7) Obviously, the green shape does not rest on the ground. The image on the far right demonstrates how this causes problems. The green shape is partially rendered on the ground because the model recognizes it as a shape based on its occlusion boundaries and thus attempts to impose the assumptions on the shape.

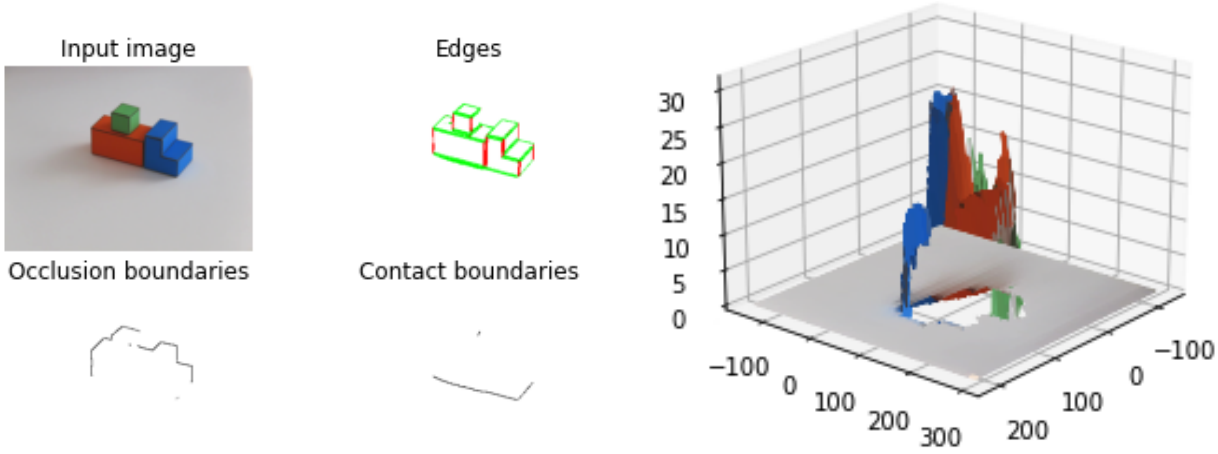

Figure 6: Image 3: Demonstrating the Problems with Simple World Assumptions

References