6.819 Advances in Computer Vision_Spring2022

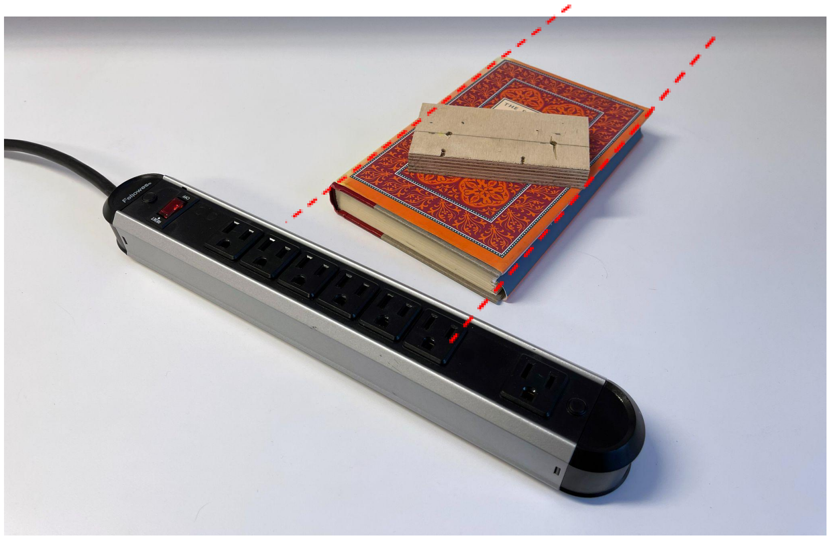
Ibrahim Ibrahim

914997060

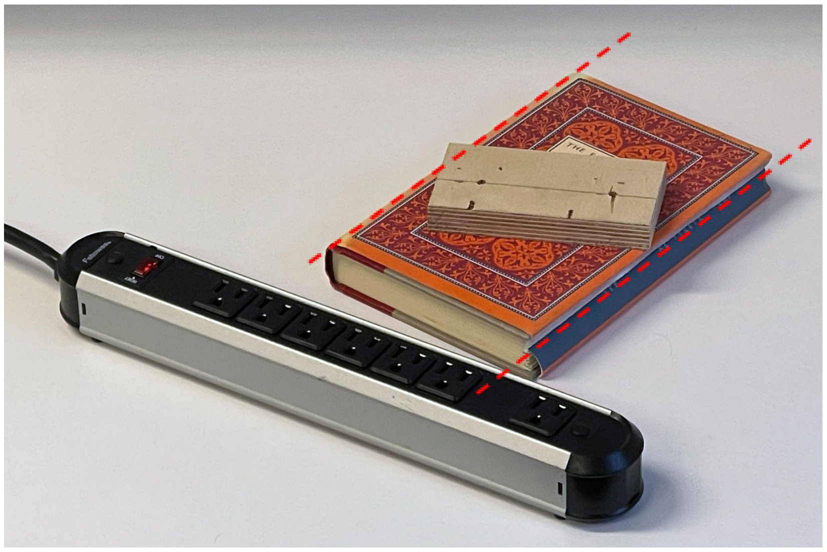
iibrahim@mit.edu / ibrahimibrahim@gsd.harvard.edu

Pset1

Problem 1



Perspective View (edge lines (red) converging towards a focal point / horizon line)



Orthographic (Parallel) View (edge lines are parallel to one another & are of approximately the same length)

Problem 2

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

$$= a \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

$$= a \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

$$= a \cdot \begin{bmatrix} & X & \\ \cos(\theta) & Y & \\ -\sin(\theta) & Z & \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

$$x = a \cdot X + x_0$$
$$y = a \cdot (\cos(\theta)\, Y - \sin(\theta) Z) + y_0$$

P is a 2x3 orthographic projection matrix

R is a 3x3 rotation matrix along the X-axis

Alpha scalar is used for sizing of the camera lens factor

x0 and y0 are translation offsets that are added since the origin of the 3D world and the origin of the camera plane do not align.

(0,0,0) to (0,0), x0=0, y0=0, a=1

(1,0,0) to (3,0), x0=0, y0=0, a=3

Problem 3

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

$$Z = \frac{a \cdot \cos(\theta) Y - y + y_0}{a \cdot \sin(\theta)}$$

$$Z = \left(\frac{\cos(\theta) Y}{\sin(\theta)}\right) - \left(\frac{y - y_0}{a \cdot \sin(\theta)}\right)$$

$$\frac{dZ}{dy} = -\frac{1}{\sin(\theta)}$$

$$\frac{dZ}{dt} = \nabla Z \cdot t \quad \text{where the vector t denotes direction tangent to the horizontal, } t = (-n_y; n_x)$$

$$= \frac{dZ}{dx} \cdot -n_y + \frac{dZ}{dy} \cdot n_x$$

$$= \frac{dZ}{dx} \cdot -n_y + -\frac{1}{\sin(\theta)} \cdot n_x$$

We can find ny and nz using angle theta, whereny=sin(theta) and nx=cos(theta)

$$\frac{d^2Z}{dx^2} = 0$$

$$\frac{d^2Z}{dy^2} = \frac{dZ}{dy}\left(-\frac{1}{\sin(\theta)}\right)$$

$$= 0$$

$$\frac{d^2Z}{dy^2dx^2} = \frac{dZ}{dy} \cdot \frac{dZ}{dx}$$

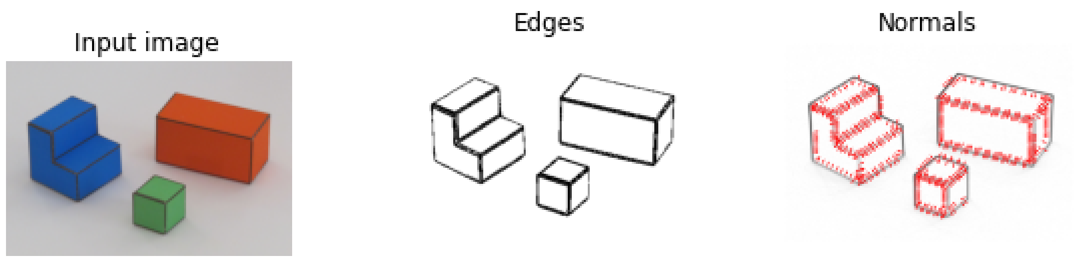
$$= \left(-\frac{1}{\sin(\theta)}\right) \cdot \frac{dZ}{dx}$$

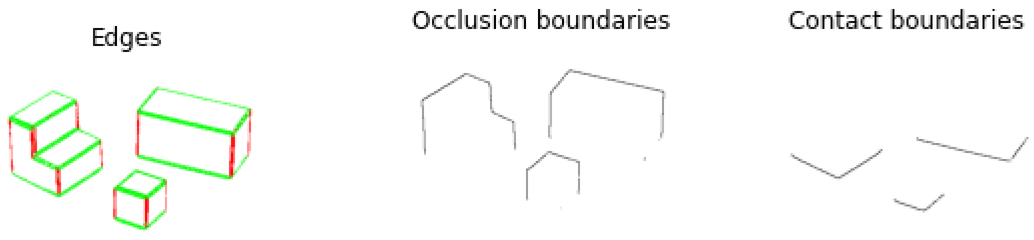
$$= 0$$

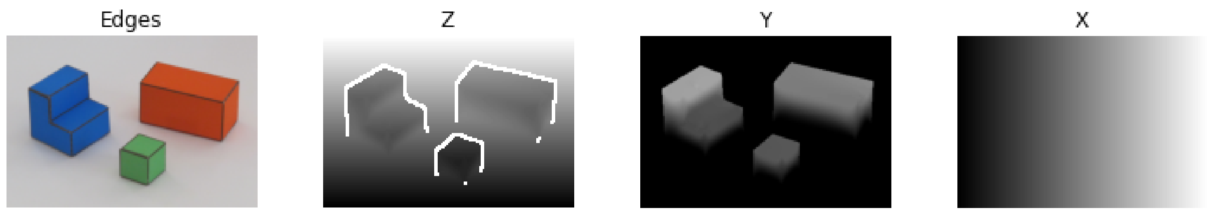
Problem 4:

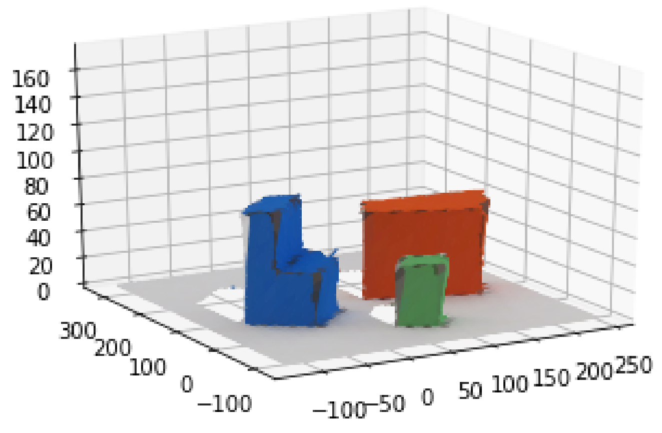
```
# Contact edge: dY/dy
# Requires: a transform matrix
if contact_edges[i, j]:
  Aij[:,:,c] = np.array([[0, 0, 0], [0, 1, 0], [0, 0, 0]])
  b[c]       = 0
  update_indices()

# Vertical edge: dY/Dy = (1/math.cos(alpha))
# 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]])
  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:
  # using trignometry to find n_x, n_y
  n_x = np.cos(theta[i, j])
  n_y = np.sin(theta[i, j])

  dYdy = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]])
  dYdx = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])
  dYdt = -n_y * dYdx + n_x * dYdy

  Aij[:,:,c] = dYdt
  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([[0, 0, 0], [-1, 2, -1], [0, 0, 0]])
  b[c]       = 0
  update_indices()

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

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

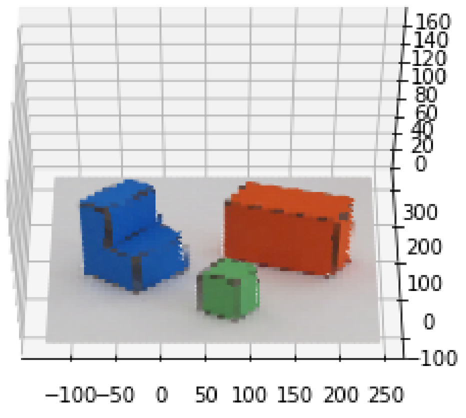
Results from the input image:









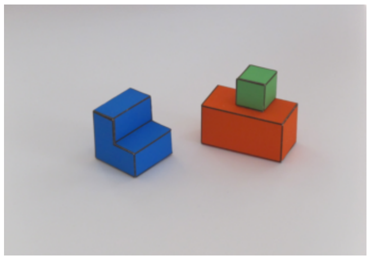


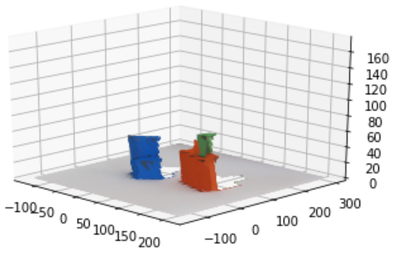
Angles: 20 degrees, -120 degrees

30deg, -80deg

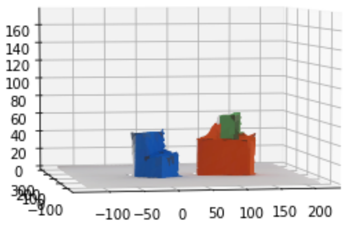
Problem 5:

Img2:



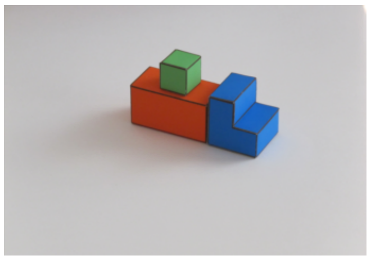


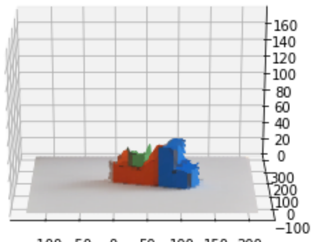
Angles: 20deg, -150deg



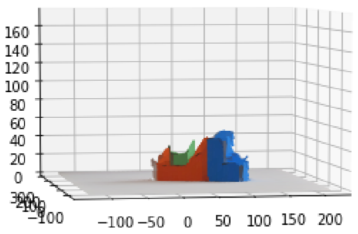
5deg, -90deg

Img3:



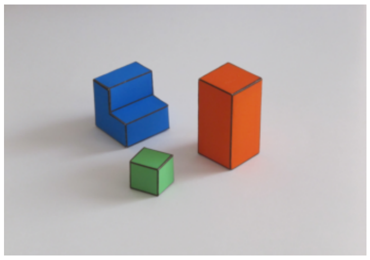


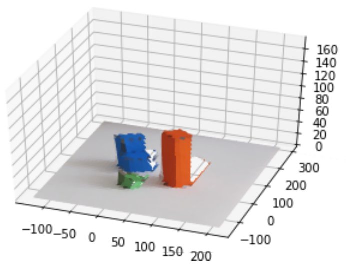
Angles: 20deg, -120deg



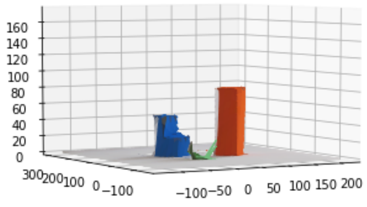
5deg, -90deg

Img4:





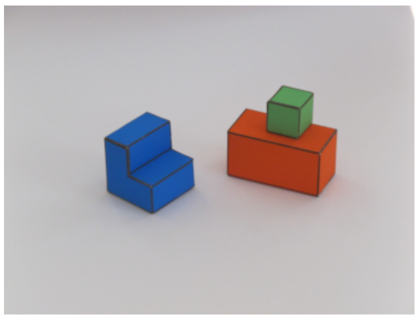
Angles: 20deg, -120deg

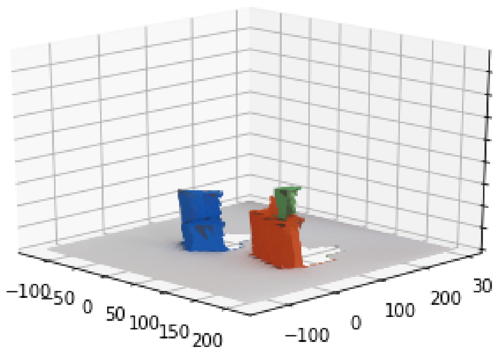


5deg, -120deg

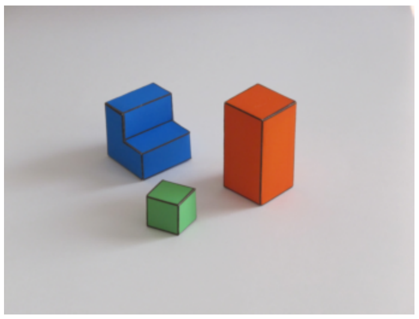
Problem 6:

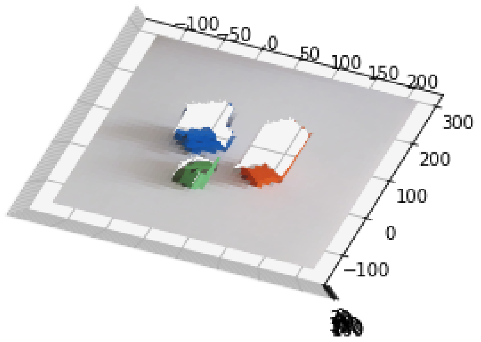
Img2:





Img4:



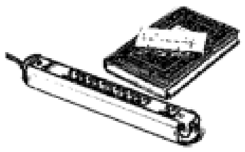


The image fails as it turns around the objects in the scene, trying to understand the backsides of the objects. Unable to reconstruct the scene from the camera input, the results display glitches as you change the angle. Besides, certain parts of the model begin to fail as the assembly becomes more complex. Example, the green box fails to be reconstructed as one of its Y edges is not detected when touching the ground due to the increased brightness on that face. Same as to when it is stacked.

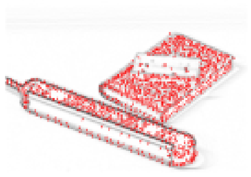
Problem 7 [optional]:

Input image

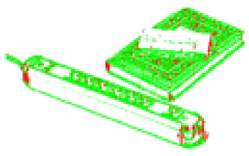
Edges



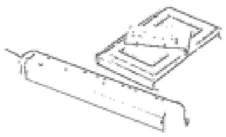
Normals



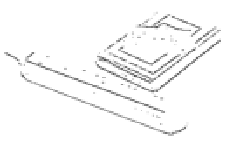
Edges

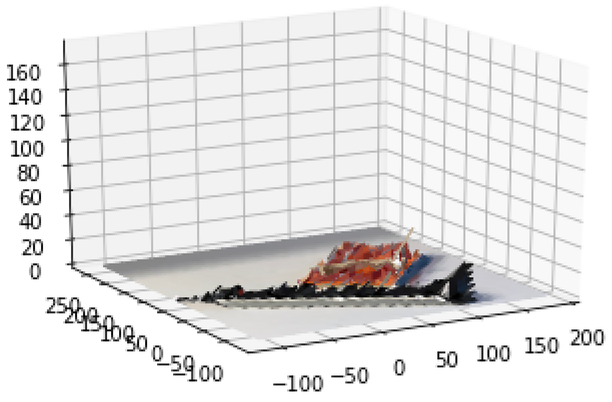


Occlusion boundaries



Contact boundaries




160
140
120
100
80
60
40
20
0
250
200
150
100
50
0
−50
−100
−100
−50
0
50
100
150
200