EN2550 Exercise 7 on Camera and Calibration

Ranga Rodrigo

March 22, 2022

1. Display the airplane.ply with the camera matrix

$$P_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -4000 \end{bmatrix}$$

   and a P_2 to affect a rotation of 30° about the z-axis.

2. The following is a cameras matrix in the Hartley and Zisserman

$$\mathrm{P} = \begin{bmatrix} 3.53553 \times 10^{+2} & 3.39645 \times 10^{+2} & 2.77744 \times 10^{+2} & -1.44946 \times 10^{+6} \\ -1.03528 \times 10^{+2} & 2.33212 \times 10^{+1} & 4.59607 \times 10^{+2} & -6.32525 \times 10^{+5} \\ 7.07107 \times 10^{-1} & -3.53553 \times 10^{-1} & 6.12372 \times 10^{-1} & -9.18559 \times 10^{+2} \end{bmatrix}$$

   We know that

$$\mathrm{P} = \mathrm{KR}[\mathrm{I}|-\mathrm{C}].$$

   Find the matrices K, R, and C.

3. Measure the are of the earrings, jewels, and the machine part in the given images. Assume that the above images are a portion of an image acquired using a camera with an the focal length $f = 8$ mm and the distant of the object plane form the camera lens plane is $Z = 720$ mm and the pixel size of the image sensor is 2.2 μm $\times$ 2.2 μm, what is the size of the object? Draw bounding boxes.

4. If the same camera in 3 has been used to make the Allen keys image, measure the width of each allen key. For this, we need to know the direction perpendicular to the Allen keys. The code given below allows you to do that using Hough lines. Note that when the Hough lines appear, you will have to draw a rectangle covering a horizontal strip of the Allen keys.

Upload the pdf generated form the Jupyter notebook as your_index_ex07 .pdf. The report must include important parts of code, image results, and comparison of results.

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
file_name = 'images/allenkeys.jpg'
im = cv.imread(file_name, cv.IMREAD_REDUCED_GRAYSCALE_2)
canny = cv.Canny(im, 50, 150)
```

```python
# Copy edges to the images that will display the results in BGR
canny_color = cv.cvtColor(canny, cv.COLOR_GRAY2BGR)

lines = cv.HoughLines(canny,    1, np.pi / 180, 170, None, 0, 0)

if lines is not None:
    for i in range(0, len(lines)):
        rho = lines[i][0][0]
        theta = lines[i][0][1]
        a = np.cos(theta)
        b = np.sin(theta)
        x0 = a * rho
        y0 = b * rho
        pt1 = (int(x0 + 1000*(-b)), int(y0 + 1000*(a)))
        pt2 = (int(x0 - 1000*(-b)), int(y0 - 1000*(a)))
        cv.line(canny_color, pt1, pt2, (0,0,255), 1, cv.LINE_AA)


cv.namedWindow('Image', cv.WINDOW_AUTOSIZE)
cv.imshow('Image', im)
cv.waitKey(0)
cv.imshow('Image', canny)
cv.waitKey(0)
cv.imshow('Image', canny_color)
r = cv.selectROI('Image', canny_color, showCrosshair = True, fromCenter = False )
cv.waitKey(0)
print(r)

x0, y0 = int(r[0] + r[2]/2), int(r[1] + r[3]/2)
m = b/a # Gradient
m = np.tan(np.median(lines[:, 0,1]))
c = y0 - m*x0   # Intercept

cv.line(canny_color, (0, int(c)), (im.shape[0], int(m*im.shape[0] + c)), (0,255,0), 2, cv.LINE_AA)

cv.imshow('Image', canny_color)
cv.waitKey(0)
cv.destroyAllWindows()

dy = 1
y_sub_pixel = np.arange(0, im.shape[0] - 1, dy)
f_sub_pixel = np.zeros_like(y_sub_pixel)
f_sub_pixel_nn = np.zeros_like(y_sub_pixel)
# https://youtu.be/v9CFu4r6tPY

for i, y in enumerate(y_sub_pixel):
    # Your code hear to generate the pixel values along the line

fig, ax = plt.subplots(figsize=(30,5))
ax.plot(f_sub_pixel_nn)

# Your code hear to compute the widths. Keep in mind of the angle.
```