Midterm

Perception for Autonomous Robots

March 16, 2022

Instructors:
S. Charifa

Student:
Jerry Pittman, Jr. (117707120)

Semester:
Spring 2022

Course code:
ENPM673

Contents

1 Generated Images **3**

2 Problem 1: Separate and count coins **3**

3 Problem 2: Stitch Images **3**

4 Problem 3: Calibrate the camera **4**

5 Problem 4: Separate image based on color **4**

6 References **4**

Whole bibliography **4**

List of Figures

1 Generated Images

Images generated for the midterm are located here and are also hyperlinked in their appropriate sections.

2 Problem 1: Separate and count coins

Algorithm 1 Count Coins Pipeline

function count coins(image)
    Use Canny to find edges. Pick threshold.
    Use morphology to separate coins (dilation, erosion, closing, or opening). Pick kernel size (odd number).
    use FindContours to count number of coins
end function

See "question1.py" script. I had best visual results using dilation to separate the coins although I expected/wanted to solve using Opening. I found 3-4 iterations of dilation with kernal size of 5x5 to have worked best. I used canny thresholds of 50 and 80 for findCountours to count correct number of coins. I then decided to use the houghCircles function. This method takes a lot of troubleshooting in order to determine the appropriate minDist, minRadius, and maxRadius parameters. The difficulty was filtering out black circles (non-coins), not having duplicate circles for a singular coin, and counting/seeing all white (coin) circles. The best values (still not accurate) are: minDist=15, minRadius=20, and maxRadius=60. These values have only coins found but several coins are not found. Images generated from the various methods are here.

Algorithm 2 Count Coins using HoughCircles Pipeline

function count coins with Hough(image)
    Use Canny to find edges. Pick threshold.
    Use Hough circles to find circles based on given minDist, minRadius, and maxRadius parameters.
    Use shape[1] to count number of coins based on number of circles.
end function

3 Problem 2: Stitch Images

Algorithm 3 Stitch Images Pipeline

function stitch images(
)Convert to Grayscale
    Find sift or orb points in each image and measure distances in each image.
    Match points using Brute-Force Matcher and RANSAC
    Find best matches using Lowe's ratio test
    Solve for the homography matrix between the image1 and image2 points.
    Warp perspective of 2nd image.
    Stitch the images together using the homography
end function

See "question2.py" script. Stitched image and comparison here.

4 Problem 3: Calibrate the camera

Q1. What is the minimum number matching points to solve this mathematically?

A: Six (6) points since 3D mapping using homogeneous coordinates (12x1 vector) so that will have enough knowns to solve the 11 unkowns

Q2. What is the pipeline or the block diagram that needs to be done in order to calibrate this camera given the image above.

Algorithm 4 solve Intrinsic Matrix Pipeline

1: **function** solveIntrinsicMatrix(image_points, world_points)
2: Type in values of image and world points as numpy arrays.
3: Solve for the homography matrix between the world and image points.
4: Solve for homogeneous coordinates to then estimate matrix P.
5: Get 3x3 submatrix M from P then do RQ (or QR) factorization to solve for K since R is orthogonal. Gram-schmidt process. This is multiplying each row by it's respective unit vector and dividing by their normal.
6: **end function**

Q3. First write down the mathematical formation for your answer including steps that needs to be done to find the intrinsic matrix K

A: where:

$$p' = p \cdot R + t$$

$$t = P^{-1} \cdot p'$$

$$proj_u(v) = \frac{\langle u,v \rangle}{\langle u,u \rangle} u$$

$$u = v - \sum_{j=1}^{k-1} proj_{u_j}(v_k)$$

Q4: See "question3.py" script. K matrix solved to be (doesn't look right, last row should be 0,0,1).

$$K = \begin{bmatrix} -2.637e-13 & -2.975e-13 & 1 \\ 7.465e-01 & -6.653e-01 & -9.992e-16 \\ 6.653e-01 & 7.465e-01 & 3.974e-13 \end{bmatrix}$$

5 Problem 4: Separate image based on color

See "question4.py" script. Post-cluster image here. Doesn't look like 4 colors but possibly has different shades of colors.

6 References

Whole bibliography

[1] S. Charifa, "Enpm673 spring 2022 notes," University of Maryland - College Park, MAGE, College Park, MD, Tech. Rep. Lectures, Jan. 2022.

[2] openCV. "Opencv documentation." (), [Online]. Available: https://docs.opencv.org. (accessed: 04.16.2022).