

Kyle Verdeyen

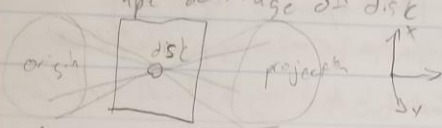
Computer Vision EN.601.461

Assignment 1 README

Written section 1:

Computer Vision HW1 written section

1a) Shape and image of a disk



Makes a cone

Cone: $z = \lambda \sqrt{x^2 + y^2}$ where $\lambda = \cot(\theta/c)$

Image plane: $z = p - mx - ny$

$\lambda \sqrt{x^2 + y^2} = p - mx - ny$ (combine)

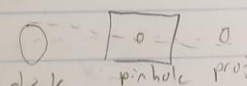
$(\lambda^2 - m^2)x^2 + (\lambda^2 - n^2)y^2 - 2m\lambda xy + 2mpx + 2npy - p^2 = 0$

$(\lambda^2 - m^2)$ and $(\lambda^2 - n^2)$ must be of the same sign

Equation is continuous so the projection must be a closed shape - since it is of 2nd order it must be an ellipse.

If $(2m=0) \text{ and } ((\lambda^2 - m^2) = (\lambda^2 - n^2))$ then it forms a circle

1b)



$A_0 = 1 \text{ m}$ $A_0 = 1 \text{ mm}^2$

$A_1 = 2 \text{ m}$ $A_1 = ?$

Area of circle = $\pi r^2 \rightarrow r_0 = \frac{1}{2\pi} \rightarrow (0, \frac{1}{2\pi})$

Pinhole projections: $(x_i/\lambda = x_0/z_0, y_i/\lambda = y_0/z_0)$

Magnification: $A/z_0, \frac{\text{area}_i}{\text{area}_0} = m^2$

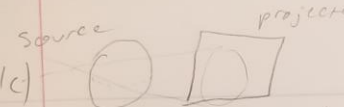
Solve for z_0 using $r_0, v_i = \frac{1}{2}\pi$

$z_0 = z/\pi$. Double λ to 2 m to get $m = 3.14$

$A_1/A_0 = m^2 \rightarrow x_1/\lambda = \pi^2$

$A_1 = \pi^2 \text{ mm}^2$

1c)



Incident surface intersects the cone

The ray projection of a sphere makes the shape of a cone. When an incident surface receives the image projection it takes a slice out of this cone.

No matter the angle, the projection will take the form of a circle, ellipse, parabola, or hyperbola on the surface.

Programming section 1:

Pyplot needs to be closed manually. Cv2 image windows will all close when any key is pressed but pyplot needs to be closed first.

- a) P1: Used 127 as the threshold value. Reason for use: half of 255. Binary image of two_objects has window titled 'p1: two_objects_binary'.
- b) P2: output shown in both initial pyplot window and 'p2: labels_out (cv2 image)'
- c) P3: Attributes chosen: title, x position, y position, moment, orientation, roundness. Output is shown in 'p3: two_objects_output_image'. Reason for these attributes is simply because that's how it was taught in lecture...
 - a. To calculate the above, needed: "area" (number of counts of label), xpos, ypos, a, b, c, minimum and maximum theta, minimum and maximum moment)
 - b. first needed to get shifted coordinates from raw positions in the image, then calculate a, b, c (ref: Lect 3, slide 15). Using these values we can then calculate theta and moment. Saving to the output db, theta is 'encoded' as orientation and the ratio between moments is roundness.
- d) I used a brute force method that took the minimum ratio of both moment and roundness between labeled analysis objects and the comparison database object. If the ratio (should always be ≤ 1 , with 1 being perfect match) of both the moment and roundness is greater than 80% then the objects are considered a match. I used these criteria together because one alone was producing false positives and misses, and the simple ratio test does not require extensive computation to compare the two objects.

Programming Section 2:

- a) P5: I used a Sobel 3x3 convolution mask because it is less sensitive to noise and has better detection than a Roberts or Prewitt, while also saving much on computational load compared to a Sobel 5x5.
- b) P6:
 - 1. Theta can range from $-\pi/2$ to $\pi/2$ (-90 to 90) and the maximum rho is the longest distance possible in the image, e.g. the diagonal. (Edge detection, slide 66) Theta used is 180, and rho used is 800.
 - 2. The accumulator is of size 180 x 800. This encompasses all possible values of theta. Setting rho to 800 will ensure it always has enough space to span the entire image, since the given images are 640x480 (diagonal of 800). This balance allows excellent detection while avoiding the computational load of higher resolution accumulator arrays.
 - 3. The voting scheme is a classical straight line Hough transform and has nothing really special about it. The accumulator starts at 0 and is allowed to freely increment, but is scaled after calculation to give a range 0-255.
 - 4. The edge threshold chosen is 20. This allowed the best detection while minimizing noise according to testing with the given image data.
- b) P7: The threshold used for all images was 127, to consider the upper 50 percentile edges a "strong" edge. Couldn't get this to work. While the hough image looks OK, I am having difficulty recovering the rho and theta (and therefore X and Y coordinates) to subsequently draw lines in the base image.