SUN Yilin
520030910361

Homework 1

Computer Vision
October 27, 2022

1 Written Assignment

a

The shape of the image will still be a circular disk, with different radius.
Explanations: We use the same setting as our lecture notes, i.e. $\boldsymbol{r_o} = (x_o, y_o, z_o)$ denotes the coordinate of actual object pixel and $\boldsymbol{r_i} = (x_i, y_i, f)$ denotes its image. We set the pinhole $(x, y, z) = (0, 0, 0)$ and let the optical axis be the z-axis. Now we prove our argument by deriving the equation of the image of the disk.
The equation of the circular disk in original space can be described as $(x_o-a)^2+(y_o-b)^2 = c^2$ where (a, b) is its center and c is its radius. Then by applying the rule of similar triangles we get $(x_i, y_i) = (\frac{fx_o}{z_o}, \frac{fy_o}{z_o})$, then by substituting the coordinates we get the equation of image disk as $(\frac{z_o x_i}{f} - a)^2 + (\frac{z_o y_i}{f} - b)^2 = c^2$, or equivalently

$$(x_i - \frac{af}{z_o})^2 + (y_i - \frac{bf}{z_o}) = (\frac{cf}{z_o})^2$$

which is still a circular disk in image plane.

b

Case 1: plane $y = 0$
The direction vectors on this plane can be expressed generally as $(x_i, 0, z_i)$. Then the vanishing points of the family of lines represented by these family of vectors can be expressed as $(f\frac{x_i}{z_i}, 0)$ by our formula from lecture slides. Then we know the vanishing points $(f\frac{x_i}{z_i})$ actual lie on line $y = 0$ in the image plane.
Case 2: plane $x = 0$
Now the direction vectors are $(0, y_i, z_i)$ and the vanishing points are $(0, f\frac{y_i}{z_i})$. This time the vanishing points lie on the line $x = 0$ in the image plane.

c

Generally, the normal vector of plane $Ax + By + Cz + D = 0$ is (A, B, C), thus any direction vector (x_i, y_i, z_i) that lies in this plane must satisfy that $Ax_i + By_i + Cz_i = 0$. Then we know that the vanishing point of line (x_i, y_i, z_i) is $(f\frac{x_i}{z_i}, f\frac{y_i}{z_i})$. (Note that z_i cannot be zero, otherwise there will not be a vanishing point for these lines parallel to the image plane)
These vanishing points $(f\frac{x_i}{z_i}, f\frac{y_i}{z_i})$ actually lie on line

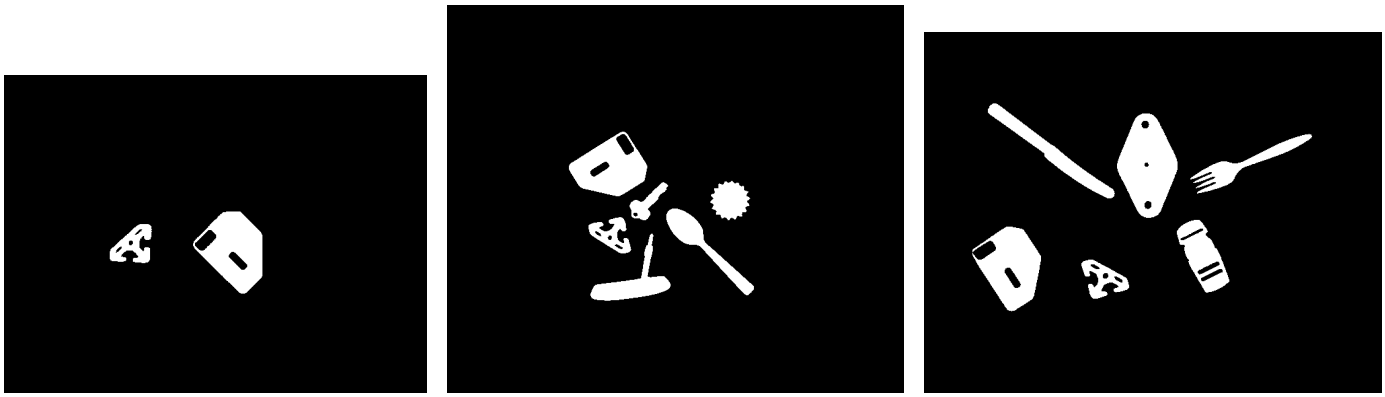
$$Ax + By + Cf = 0$$

2 Programming Assignment

2.1

a

I use the default threshhold value 128.
The function is straightforward. It takes the gray image as input and do an element-wise operation to set all those values larger than threshhold value as 255, meanwhile other values remain zero.



(a) two_objects (b) many_objects_1 (c) many_objects_2

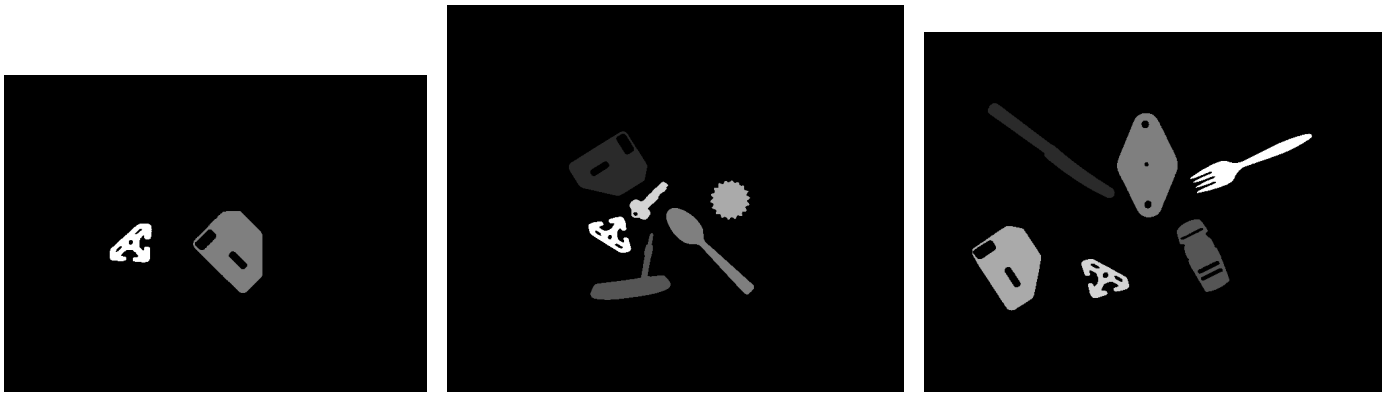
Figure 1: binary images of various input

b

I implemented the sequential labeling algorithm by making two passes of the image. In the first pass I label all those connected components by scanning the array from top to down and from left two right. Meanwhile I used the union-find-set data structure to mark the equivalent classes. In the second pass I merge the equivalent classes, calculate the number of total connected components and mark them with different gray levels by a normalization over interval [0,255].

c

This function is simply the implementation of the formulas given in our lecture notes. The results on various images are listed below.



(a) two_objects (b) many_objects_1 (c) many_objects_2

Figure 2: labeled images of various input

```
➜  CVhw-1023 git:(main) ✗ python p1_object_attributes.py two_objects 128
two_objects
{'position': {'x': 195.3160469667319, 'y': 223.3820939334638}, 'orientation': 0.6875462936637136, 'roundedness': 0.4799636466920353}
{'position': {'x': 349.33298470388286, 'y': 216.45365407242778}, 'orientation': 1.8816361301275926, 'roundedness': 0.533631953475641}
➜  CVhw-1023 git:(main) ✗ python p1_object_attributes.py many_objects_1 128
many_objects_1
{'position': {'x': 268.30828220858893, 'y': 257.85327198364007}, 'orientation': -0.5388371734983285, 'roundedness': 0.4860732206012443}
{'position': {'x': 461.6430812129662, 'y': 313.7504356918787}, 'orientation': 1.2635628997731068, 'roundedness': 0.9902664427338298}
{'position': {'x': 265.97616566814276, 'y': 365.13401927585306}, 'orientation': 0.08042727460236962, 'roundedness': 0.5217196889211292}
{'position': {'x': 326.0154385964912, 'y': 309.29473684210524}, 'orientation': 0.7788385087054034, 'roundedness': 0.13319471993926857}
{'position': {'x': 303.571394686907, 'y': 178.27300759013283}, 'orientation': 0.40520199272654855, 'roundedness': 0.27027118415863505}
{'position': {'x': 417.71620665251237, 'y': 241.29181410710072}, 'orientation': -0.7760238443266956, 'roundedness': 0.02442160982659454}
➜  CVhw-1023 git:(main) ✗ python p1_object_attributes.py many_objects_2 128
many_objects_2
{'position': {'x': 475.3399815894446, 'y': 339.9671678428966}, 'orientation': 0.40324741948779835, 'roundedness': 0.020855451285964482}
{'position': {'x': 130.16157675232074, 'y': 188.1522938248352}, 'orientation': 1.6932113097868653, 'roundedness': 0.5078766943974415}
{'position': {'x': 188.3515625, 'y': 357.90033143939394}, 'orientation': -0.6431420831724863, 'roundedness': 0.007633528961638793}
{'position': {'x': 265.9671412924425, 'y': 169.6462212486309}, 'orientation': -0.4929693290413841, 'roundedness': 0.4809122478567922}
{'position': {'x': 413.6556685685934, 'y': 204.95137682957082}, 'orientation': 2.0236832362775745, 'roundedness': 0.17394416151886075}
{'position': {'x': 331.9617982504706, 'y': 338.21769460746316}, 'orientation': 1.6106730812607657, 'roundedness': 0.3072674402498929}
```

Figure 3: attributes of various images

2.2

a

The Sobel mask is equivalent to what we defined in class, i.e.

$$Sobel_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \qquad Sobel_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

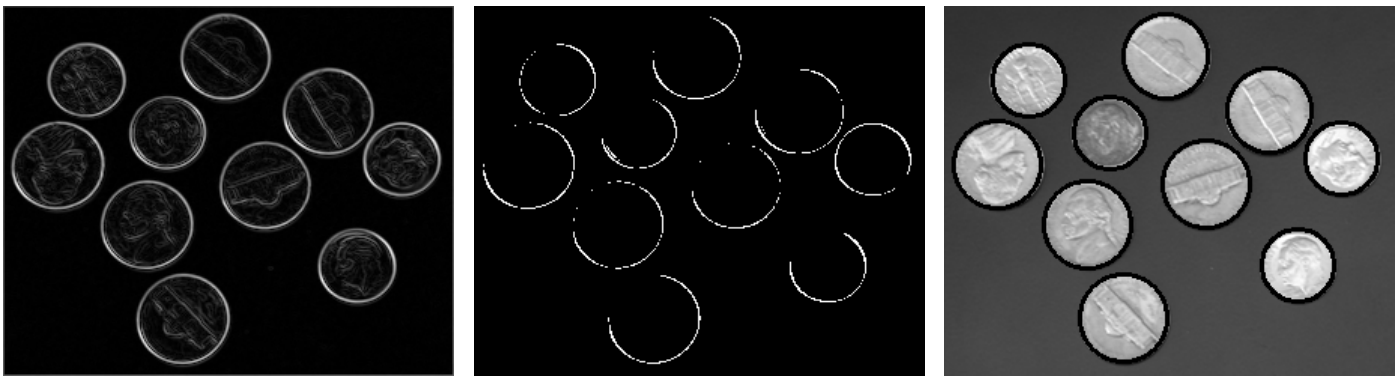
I then implement convolution operation and finish the edge detect operation.

b

I choose edge threshhold value 140 and {24,25,...,30} as possible radius values.

c

The maximum intensity of my accumulation array is about 106, after careful examination, I choose hough threshold value as 75. At this threshold, we are able to detect all the circles at a fast speed. Also I used the NMS operation to filter out multiple circles centerd at the same point.
The circle parameters given as a list of tuples are:
(29, 105, 34)(24, 49, 55)(29, 145, 94)(24, 84, 109)(29, 207, 118)
(28, 33, 146)(29, 118, 173)(28, 69, 215)(24, 172, 234)(24, 101, 263)



(a) edge intensity (b) edge threshholded (c) circles with NMS operation

Figure 4: Results of problem 2