Computer Vision

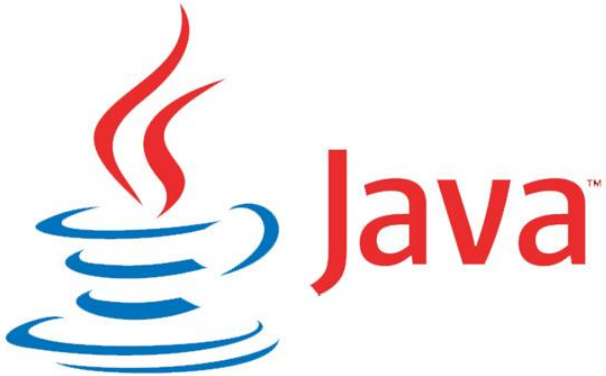
# *Circle detection using Hough Transform*

A.Y. 2021/2022


**Professor:** Luca Lombardi
**Student:** Domenico Ragusa

# Objective

Creating a program for finding circles in an image by implementing the Hough Transform



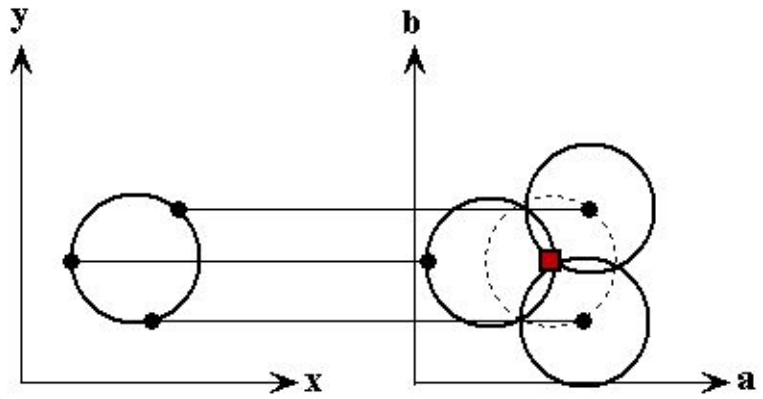
# Introduction

The **Circle Hough Transform** (**CHT**), a particular implementation of the Hough Transform, is a feature extraction technique used in digital image processing for detecting circles in an image.

In a 2D space a circle can be described by $\mathbf{(x-a)^2+(y-b)^2=r^2}$, where (a,b) is its center and r is the radius.

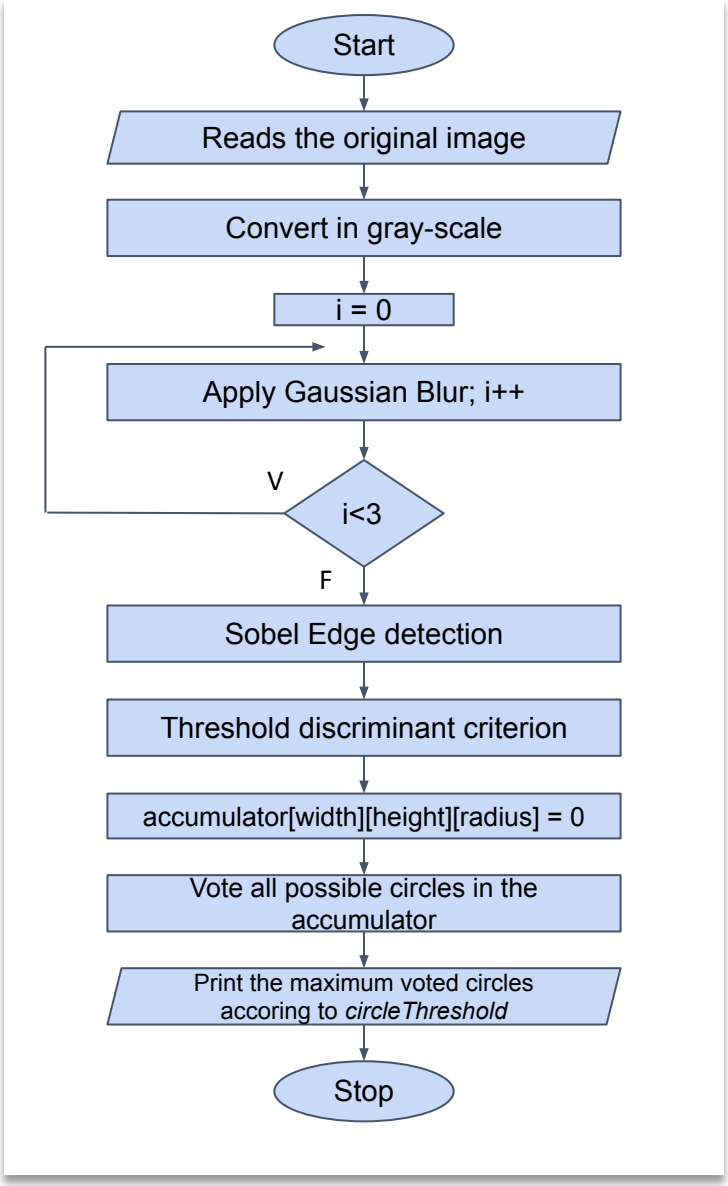
If the radius is fixed the parameter space is 2D and one have to look at the intersections of the circles obtained using the pixels of the image as a center, the intersection is the point that result in a local maxima.

In case of a variable radius, the parameter space would be 3D (a,b,r), the working principle is the same but here one have to look at the intersections of cones.




**2D example**. In a 3D space it works the same but one have to look at the intersection of the cones.

# Algorithm




***Incrementing mapping rule - pseudocode***

```
set each accumulator element to 0

for each cell(x,y) above threshold
        for r=minRad to r=maxRad
                for t = 0 to 360
                        b = y - r*sin(t*pi/180)
                        a = x - r*cos(t*pi/180)
                        accumulator[a,b,r] += 1
                end
        end
end
```

# How to use

***Application run:***

1. Open a terminal in the project directory
2. compile all the java source codes running the command ***make***
3. Run the program by typing
   ***java circleHough.Main imagePath [sobelThreshold][minRadius][maxRadius][circleThreshold]***

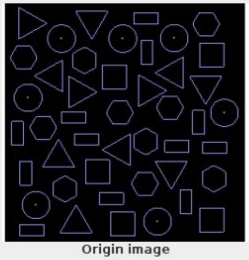
***Parameters***:

- *imagePath* (**required**): is the path of the image to process
- *sobelThreshold*: is the threshold for the sobel filter, used to discard useless details (default 150)
- *minRadius*: minimum radius of the circles to be detected in px (default 10)
- *maxRadius*: maximum radius of the circles to be detected in px (default 100)
- *circleThreshold*: number of detected circles to print (default 1)

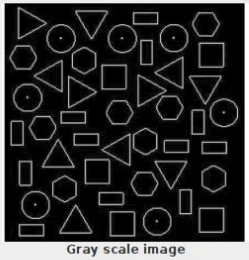
# Results

sobelThreshold = 128 | minRadius = 10 | minRadius = 30 | circleThreshold = 1

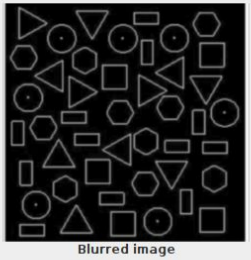
Hough Transform result



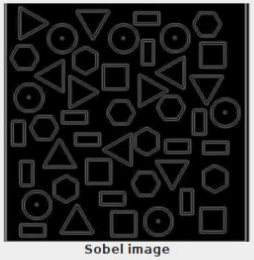
Origin image



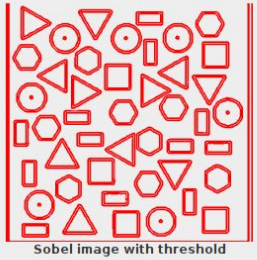
Gray scale image



Blurred image



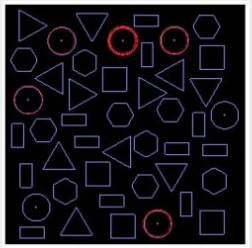
Sobel image



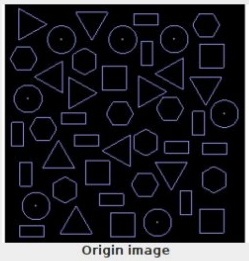
Sobel image with threshold

# Results

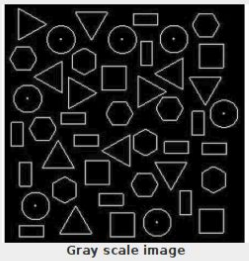
sobelThreshold = 128 | minRadius = 10 | minRadius = 30 | circleThreshold = 10



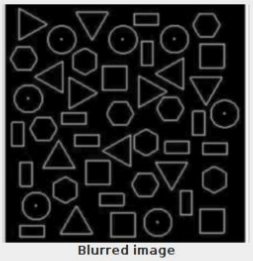
Hough Transform result



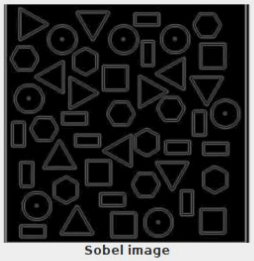
Origin image



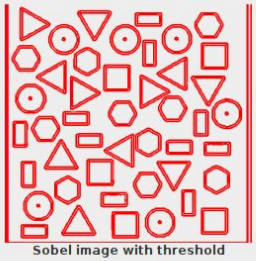
Gray scale image



Blurred image



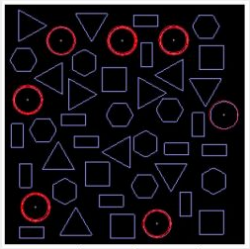
Sobel image



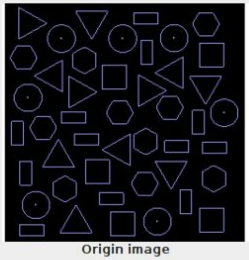
Sobel image with threshold

# Results

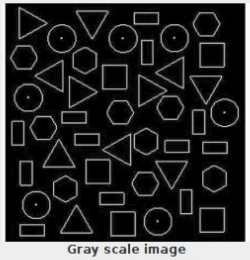
sobelThreshold = 128 | minRadius = 10 | minRadius = 30 | circleThreshold = 25



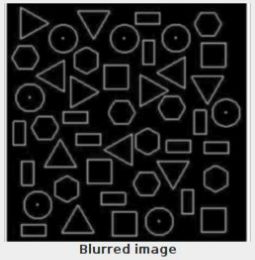
Hough Transform result



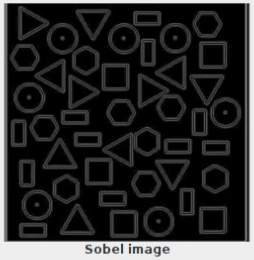
Origin image



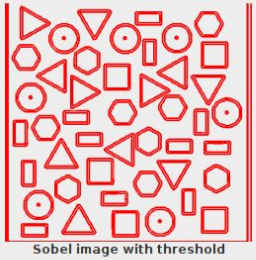
Gray scale image



Blurred image



Sobel image



Sobel image with threshold

# Conclusions

- Sometimes the same circle is highlighted more than one due to noise and possible occlusion.
  Possible solutions:
  - use the Canny Edge Detector instead of the Sobel filter
  - check if the detected circles have close coordinates and eventually discard copies
- For big images or big circle radius the algorithm is slow due to the 3D parameter space, the solution is parallelization.
- Using OpenCV library could improve performance significantly

# Thank you
# for your attention