Department of Computer Science
University of Bristol

COMS30030 - Image Processing and Computer Vision

Lydian electrum trite

Lab Sheet Week 03

The Coin Counter Challenge

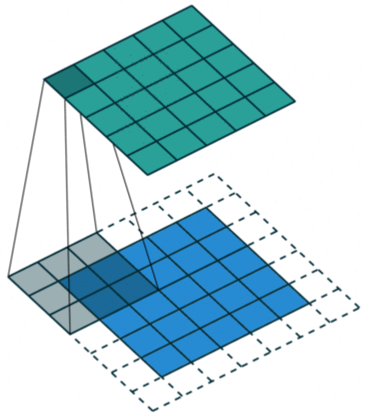
Pui Anantrasirichai | n.anantrasirichai@bris.ac.uk
Original Slides: Tilo Burghardt | Majid Mirmehdi

Catch up – Lab2

Catch up – Lab 2: The Mandrill Challenge

Task 2 : Convolution basics

$$* \ 1/9 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



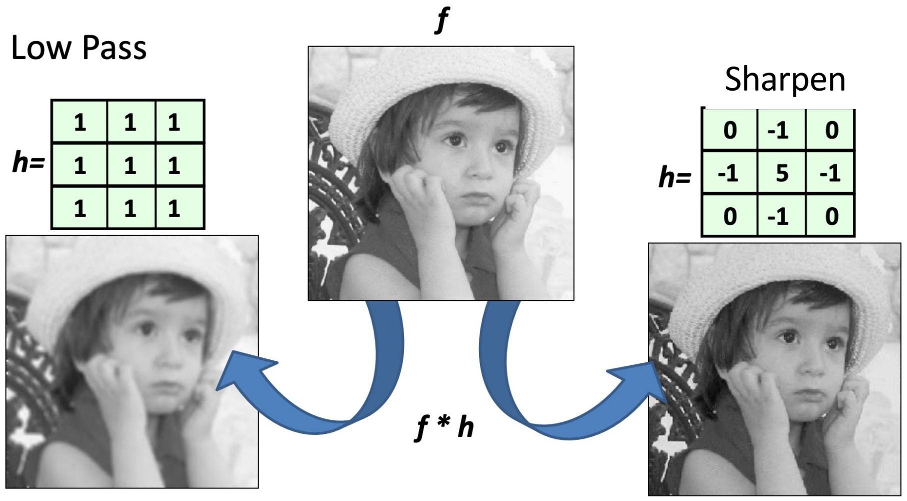
- Self-Study Video Lecture: [04 - Filtering in Spatial Domain]()

Low Pass

$$h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpen

$$h = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



- filter2d.py
- Sharpening

Catch up – Lab2

Catch up – Lab 2: The Number Plate Challenge

Image restoration – deblurring, denoising

Task 2.1 : Sharpening

- unsharp masking

- convolution.py

```
carBlurred =
GaussianBlur(gray_image,23);

detail = gray_image - carBlurred;
sharpened = gray_image + 1.0*detail;
```

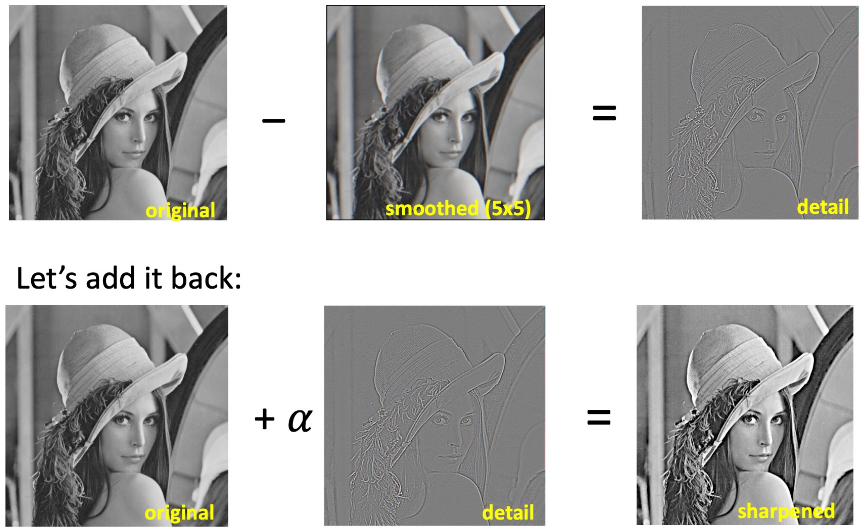
- Self-Study Video Lecture: [04 - Filtering in Spatial Domain]

Sharpening Revisited

Source: S. Lazebnik

What does blurring take away?

Let's add it back:



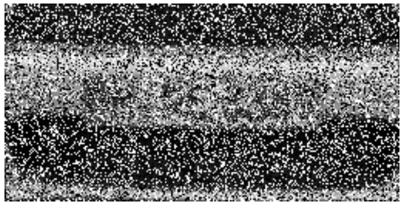
Catch up – Lab2

Catch up – Lab 2: The Number Plate Challenge

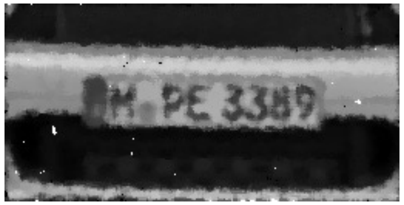
Image restoration – deblurring, denoising

Task 2.2 : Noise removal

- median filter



cv2.medianBlur (gray_image, **ksize**)

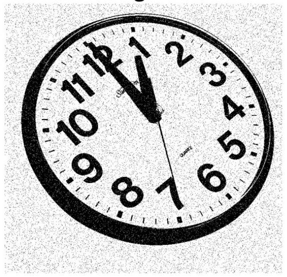


- Self-Study Video Lecture: [04 - Filtering in Spatial Domain]()

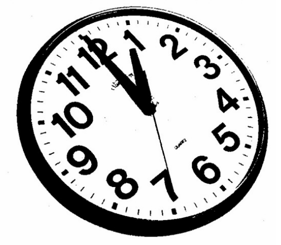
Median Filter: the algorithm

1. Let I be a monochrome image.
2. Let Z define a neighborhood of arbitrary shape.
3. At each pixel location, $\mathbf{p}=(x,y)$, in I …
4. … select the n pixels in the Z-neighborhood of $\mathbf{p}$,
5. … sort the n pixels in the neighborhood of $\mathbf{p}$ by value into a list $L(j)$ for $j = 1,\ldots,n$.
6. The output value at $\mathbf{p}$ is $L(m)$, where $m = \lfloor n/2 \rfloor + 1$.

Original



Median filtered



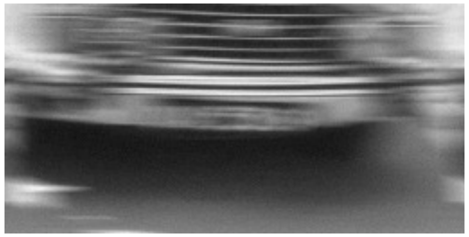
Catch up – Lab2

Catch up – Lab 2: The Number Plate Challenge

Image restoration – deblurring, denoising

Task 2.3 : Removing motion blur

- Wiener Deconvolution



$$y(t) = h(t)*x(t) + n(t)$$

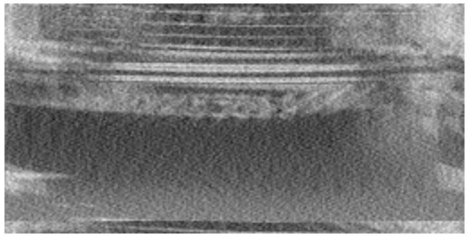
inverse of original kernel

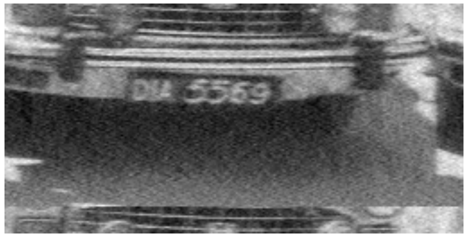
$$\boxed{\boxed{\frac{1}{H(f)}}\left[\frac{|H(f)|^2}{|H(f)|^2+\boxed{\frac{1}{\mathrm{SNR}(f)}}}\right]}$$

estimated loss at frequency f

Output = WienerDeconvoluition(input,
motionLength, motionAngle,
noiseSignalRatio,
displayPowerSpectrums)

recover = WienerDeconvoluition(gray_image,**15**,**3**,**0.001**,0)





Try using for loops to get the best parameters

Department of Computer Science
University of Bristol

COMS30030 - Image Processing and Computer Vision

Lab Sheet **Weeks 03-04**

The Coin Counter Challenge

Pui Anantrasirichai | n.anantrasirichai@bris.ac.uk
Original Slides: Tilo Burghardt | Majid Mirmehdi

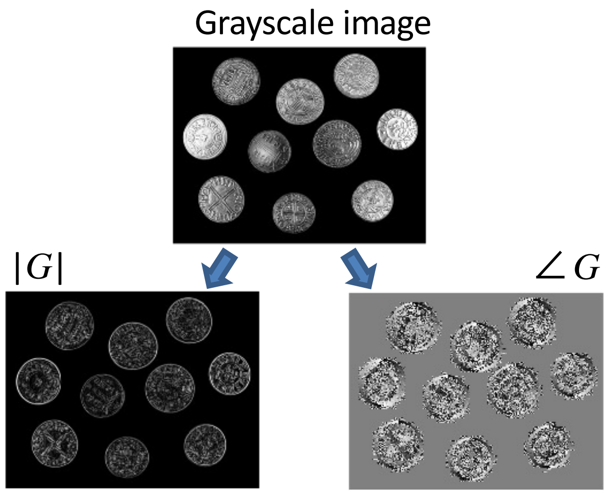
Introduction: The Coin Counter Challenge

Task Overview: Counting objects is an essential task in many vision applications. Your task is to implement an OpenCV program that can count round coins from basic imagery by utilising the Hough transform. (However, try to implement the Hough transform explicitly, that is without using the pre-implemented function calls for the Hough Transform. You can utilise your code from former labs.)

This task will be formative and we will, as always, be there during labs to provide you with help and feedback. The assignment is to be worked on individually and will prepare you for the upcoming summative assessment.

Task Part 1: Sobel Edge Detection

As detailed in this week's lecture, the detection of edges is pivotal in the task of distinguishing shapes. Your first task is to implement the Sobel edge detector by convolving an image with the Sobel kernels and calculate the gradient information. (*without using the convolutional and sobel functions offered by OpenCV.)*



Task Breakdown

1. Implement a function `sobel`.
2. This function should take an image as an input and compute the following
    - Image containing the derivative in the x direction $\frac{\partial f}{\partial x}$
    - Image containing the derivative in the y direction $\frac{\partial f}{\partial y}$
    - Image containing magnitude of the gradient $\nabla f\,(x, y)$
    - Image containing the direction of the gradient ρ
3. Be able to display the four images above and discuss their appearance in relation to the original image.

Task Part 1: Sobel Edge Detection

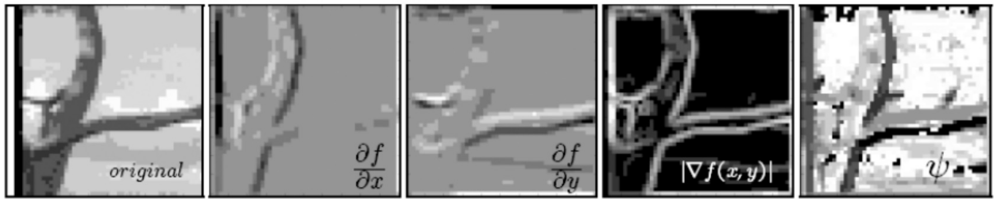
Implementation Details

- Implementation of a Sobel filter requires convolution. You have already used convolution in OpenCV for last week's task – so feel free to look back at those labsheets and review how convolution can be implemented.
- Similar to the sharpening filter from last week, convolution with the Sobel filter can result in a negative response value and generally in a wider range of values. Think carefully about the quantisation of these results, the handling of the image boundary and the data type of the matrix in which you will hold them.

- Self-Study Video Lecture: [07 - Edge Decetion]

$$\frac{\partial}{\partial x} \approx \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \frac{\partial}{\partial y} \approx \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$grad(f) = |\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \quad \psi = \arctan\left(\frac{\partial f / \partial y}{\partial f / \partial x}\right)$$


original
$\frac{\partial f}{\partial x}$
$\frac{\partial f}{\partial y}$
$|\nabla f(x,y)|$
ψ

Task Part 2: Hough Circle Transform

In this task, you will be using the gradient images from task one and perform a Hough Circle transform to detect the location of the coins in the provided imaged (… and one coin image of your own choice).

Task Breakdown

1. Take the gradient magnitude image you generated for task one and apply a thresholding operation (e.g. set all pixels larger than a threshold value T to 255 and all others to 0) to determine the set of pixels with the strongest gradient magnitudes to be considered for circle detection.
2. Implement a function **hough** that calculates the (3D) Hough Space (x_0, y_0, r) from the thresholded gradient magnitude image and the gradient orientation image. You need to make a decision about the size (i.e. number of cells) in your Hough Space. The function should at least have the following input parameters:
   - The thresholded gradient magnitude image and the gradient orientation image
   - Threshold by which a peak in the Hough Space is actually considered a circle
   - You may also wish to consider minimum / maximum radius and distance between circle centres.
3. Display the Hough Space for each image. As the Hough Space for circles is 3D, you can create a 2D image by summing the values of the radius dimension. This will highlight circle centres.
4. Threshold the Hough Space and display your set of found circles on the original images.

Implementation Details

- Consider representing your Hough Space as a 3D array.
- When you are displaying the Hough Space, you might want to take the logarithm of the image to make the image values more descriptive.

Task Part 2: Hough Circle Transform

- Self-Study Video Lectures: 08 - Hough Transform for Shapes

Circle Detection Algorithm

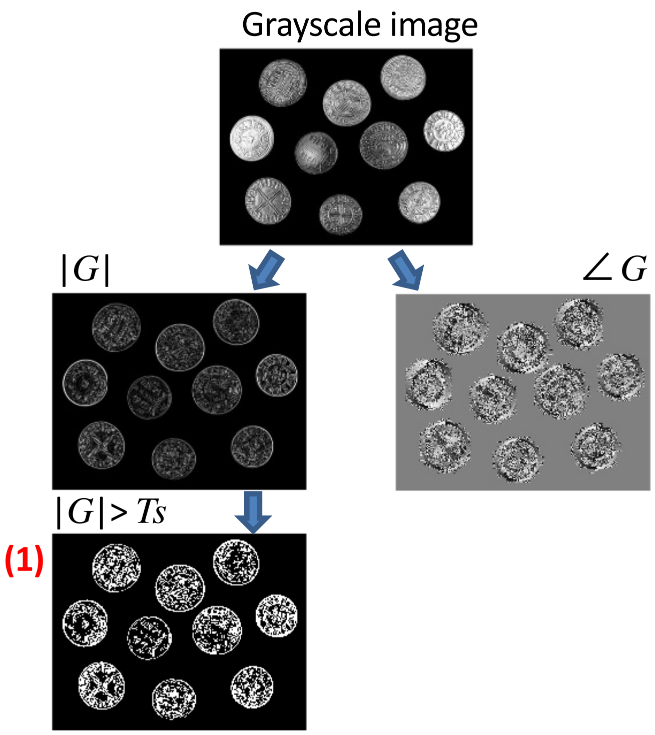
(1)

1. For any pixel satisfying $\boxed{|G(x,y)| > T_s}$ increment all elements satisfying the two simultaneous equations

$$\forall r, \quad \begin{cases} x_0 = x \pm r\ cos\ \angle G \\ y_0 = y \pm r\ sin\ \angle G \end{cases}$$

$$H(x_0, y_0, r) = H(x_0, y_0, r) + 1;$$

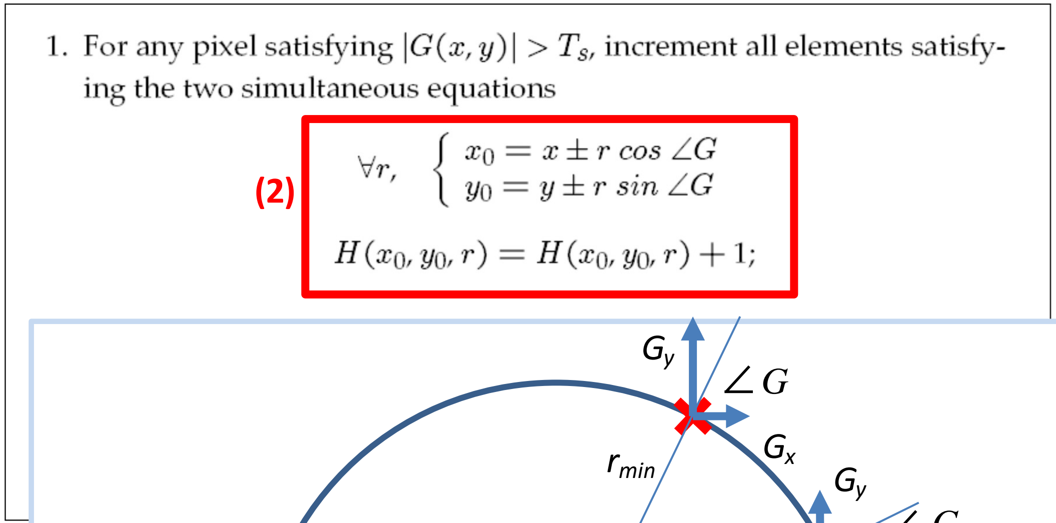
2. In the parameter space, any element $H(x_0, y_0, r) > T_h$ represents a circle with radius r located at (x_0, y_0) in the image.



Task Part 2: Hough Circle Transform

- Self-Study Video Lectures: 08 - Hough Transform for Shapes

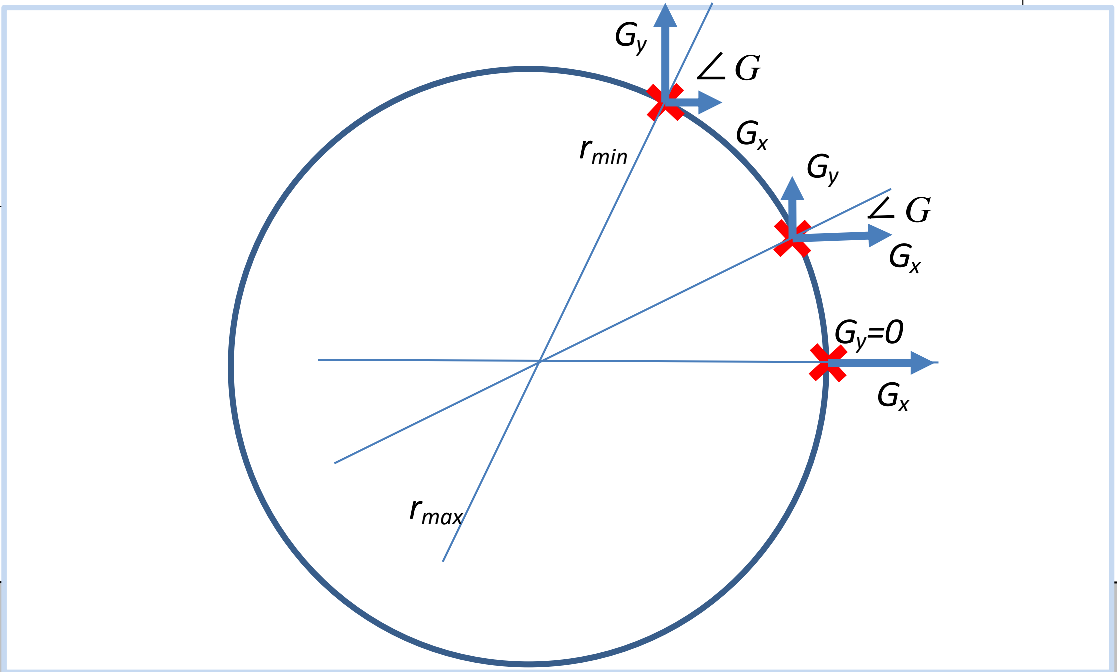
Circle Detection Algorithm

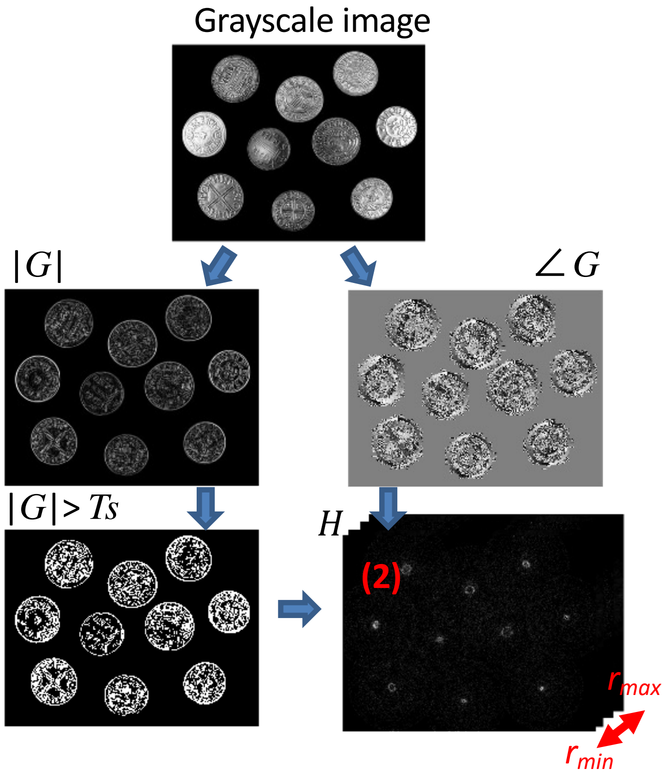


1. For any pixel satisfying $|G(x,y)| > T_s$, increment all elements satisfying the two simultaneous equations

(2)

$$\boxed{\forall r, \quad \begin{cases} x_0 = x \pm r\cos\angle G \\ y_0 = y \pm r\sin\angle G \end{cases} \qquad H(x_0, y_0, r) = H(x_0, y_0, r) + 1;}$$





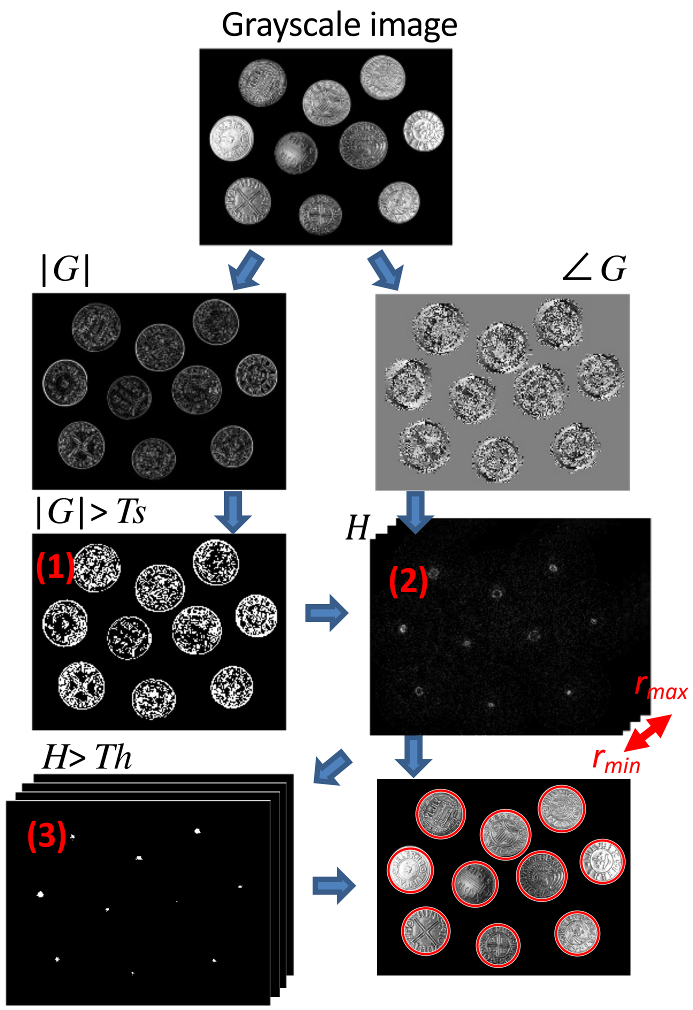
- Self-Study Video Lectures: 08 - Hough Transform for Shapes

Circle Detection Algorithm

1. For any pixel satisfying (1) $\boxed{|G(x,y)| > T_s}$ increment all elements satisfying the two simultaneous equations

(2)
$$\boxed{\begin{gathered}\forall r, \quad \begin{cases} x_0 = x \pm r\,cos\,\angle G \\ y_0 = y \pm r\,sin\,\angle G \end{cases} \\ H(x_0, y_0, r) = H(x_0, y_0, r) + 1; \end{gathered}}$$

2. In the parameter space, any element (3) $\boxed{H(x_0, y_0, r) > T_h}$ represents a circle with radius r located at (x_0, y_0) in the image.



Grayscale image
|G|
∠G
|G|> Ts
(1)
H
(2)
r max
r min
H> Th
(3)