Algoritmo 19

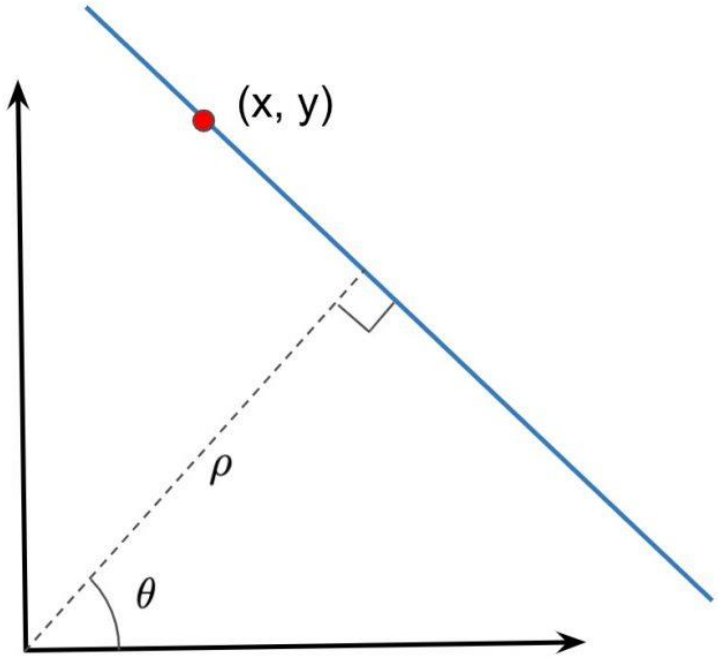
Transformada Hough

Introducción

En este algoritmo aplicaremos la Transformada Hough para detectar líneas y círculos en imágenes.

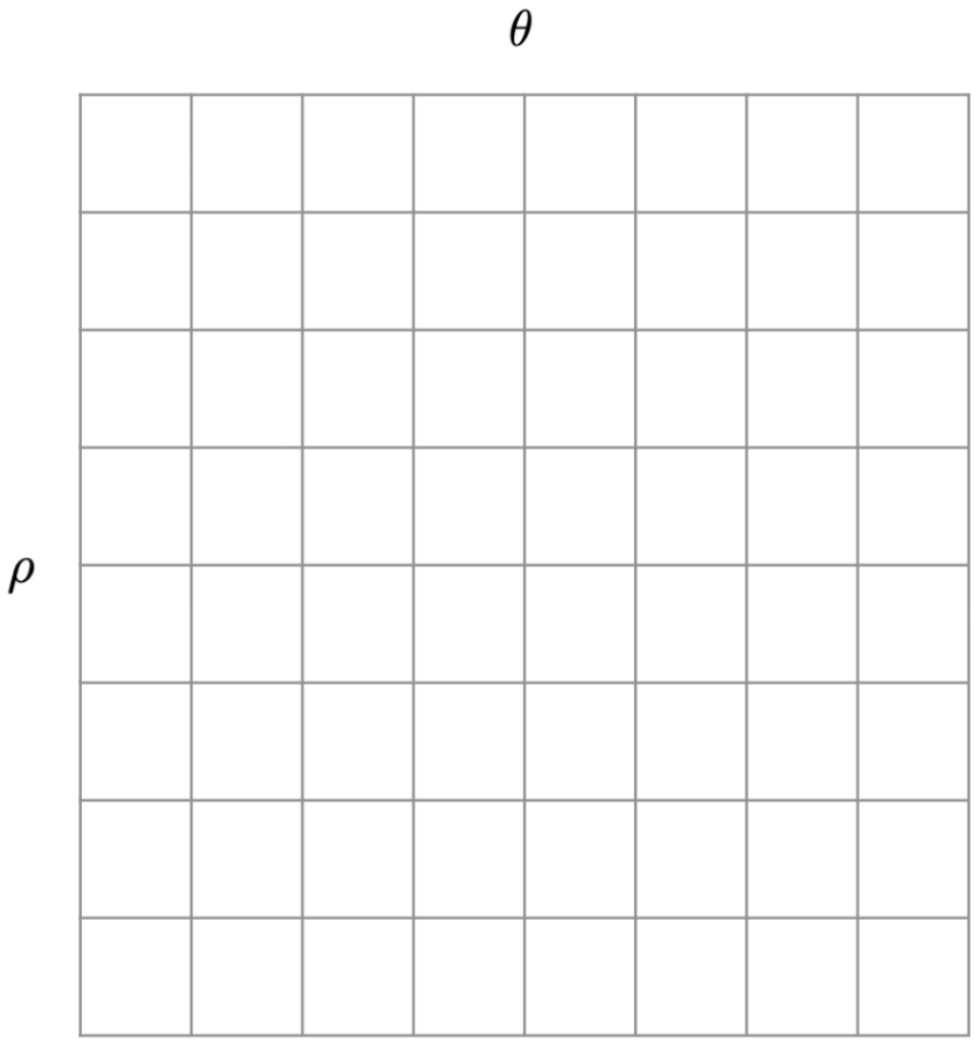
La transformada Hough es un algoritmo que encuentra líneas o círculos en una imagen mediante un sistema de votación.

Las líneas se representan utilizando coordenadas polares



$$\rho = x\cos(\theta) + y\sin(\theta)$$

Para detectar una línea se utiliza una matriz que representan las diferentes combinaciones de ángulos θ y distancia al origen ρ. Los valores con más votos son las líneas detectadas.

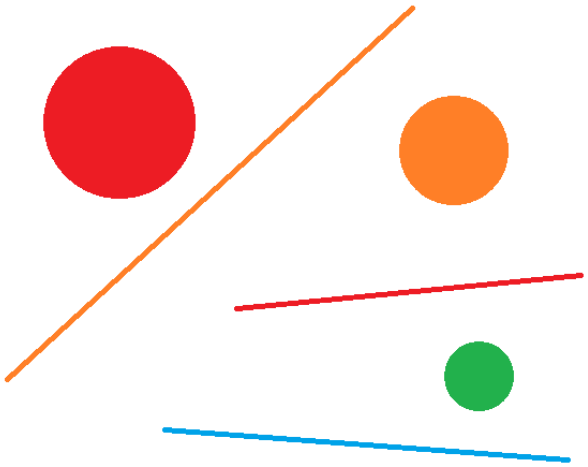


La detección de círculos funciona de manera similar solo que utilizando la ecuación

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

Detección de líneas

Supongamos que tenemos una imagen como la siguiente:



Para detectar las líneas podemos utilizar:

```
import cv2
import numpy as np

img = cv2.imread('lineas.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

edges = cv2.Canny(gray, 50, 200)
lines = cv2.HoughLinesP(edges, 1, np.pi/180, 100,
minLineLength=10, maxLineGap=250)
for line in lines:
    x1, y1, x2, y2 = line[0]
    cv2.line(img, (x1, y1), (x2, y2), (0, 255, 0), 3)

cv2.imshow("Resultado", img)

cv2.waitKey()
```

Este programa dibuja las líneas encontradas en color verde.

Los parámetros de la función HoughLines son los siguientes:

cv2.HoughLinesP(**image, rho, theta, threshold, minLineLength, maxLineGap**)

image: Imagen de un solo canal con la detección de bordes

rho: Resolución de la distancia rho, en el ejemplo es de un pixel

theta: Resolución del ángulo, en el ejemplo es de 1 grado, (el valor que se introduce esta en radianes)

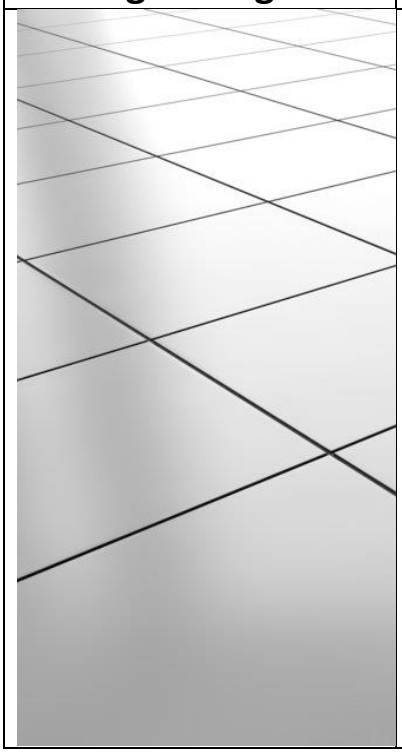
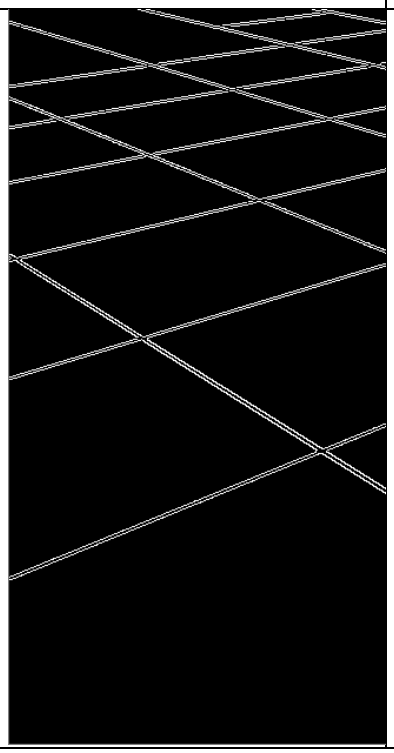
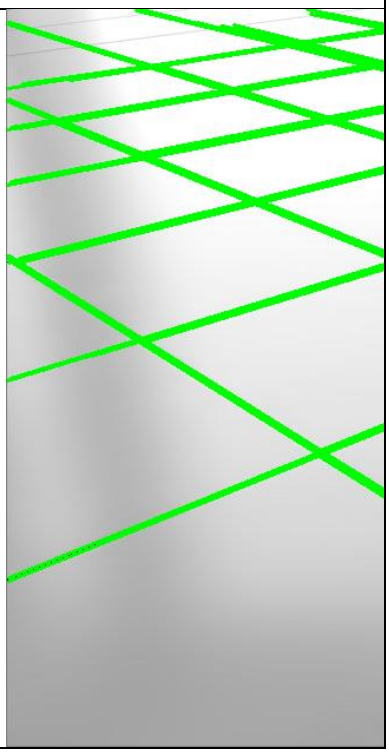
threshold: El valor mínimo con el que se considera que se detectó una línea en la matriz de acumulación. En otras palabras, si este valor tiene un 100 significaría que solo se consideran las líneas que tienen más de 100 votos.

minLineLength: Longitud mínima de una línea en pixels. Si la longitud de una línea es menor que este valor entonces no se regresa en el resultado.

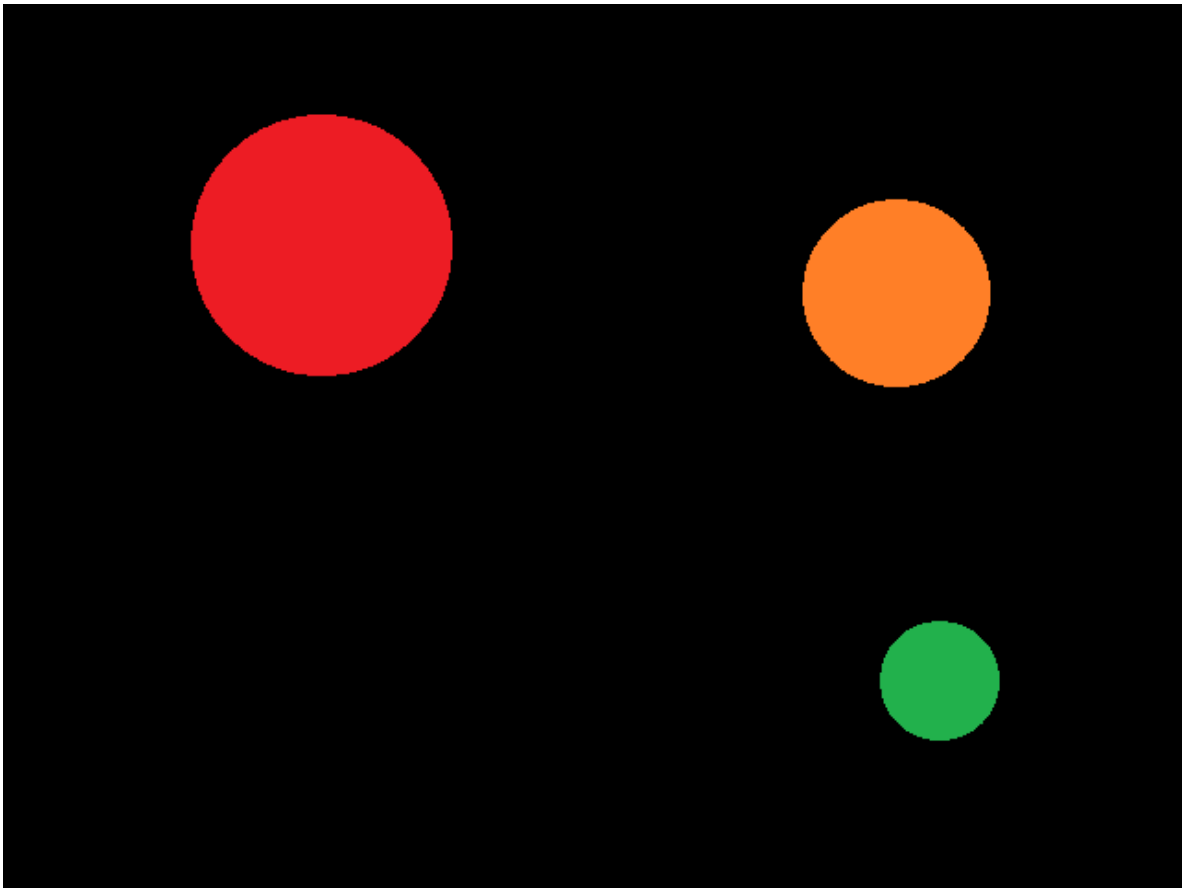
maxLineGap: Distancia máxima entre los puntos para considerar que pertenecen a una misma línea.

Ejemplos de detección de líneas

Para detectar círculos podemos utilizar el siguiente código

Imagen Original	Bordes	Líneas detectadas
		

Detección de círculos



Dr. Carlos López Franco

Para detectar círculos de esta imagen podemos utilizar el siguiente código:

```
import cv2
import numpy as np


img = cv2.imread('circulos.png')


gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)


ret, binary = cv2.threshold(gray, 90, 255, cv2.THRESH_BINARY)


circles = cv2.HoughCircles(binary, cv2.HOUGH_GRADIENT, 1, 20,
param1=10, param2=16, minRadius=1, maxRadius=100)


if circles is not None:
    circles = np.uint16(np.around(circles))
    for i in circles[0, :]:
        cv2.circle(img, (i[0], i[1]), i[2], (0, 255, 0), 2)
        cv2.circle(img, (i[0], i[1]), 2, (0, 0, 255), 3)


cv2.imshow("Resultado", img)
cv2.waitKey()
```

Parámetros de la función HoughCircles

cv2.HoughCircles(imagen, metodo, dp=1, mindist=20, param1=10, param2=16, minRadius=1, maxRadius=100)

imagen: Imagen de un solo canal, puede ser escala de grises o binaria, si es una imagen con mucha textura se recomienda aplicar un filtro suavizador.

método: cv2.HOUGH_GRADIENT

dp: Resolución del acumulador, el valor de 1 significa que es del mismo tamaño que la imagen original

minDist: Distancia mínima del centro del círculo y las coordenadas del círculo. Si el valor de minDist es muy pequeño entonces se pueden detectar incorrectamente múltiples círculos en la misma región de la imagen.

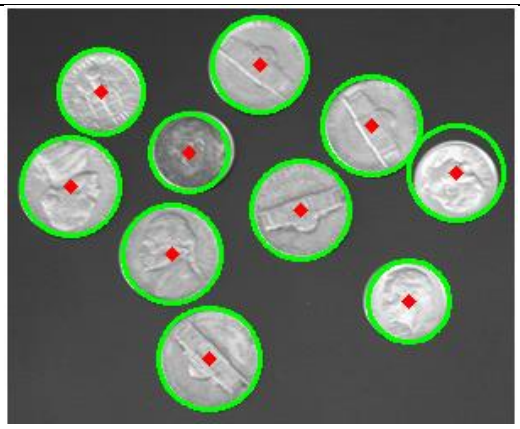
param1: Este valor se pasa directamente al algoritmo de Canny. Es decir la función HoughCircles utiliza internamente el algoritmo de Canny. Por esta razón no necesitamos aplicarlo antes de llamar la función HoughCircles, y por esta razón se le puede pasar una imagen en escala de grises o binaria.

param2: Valor mínimo del acumulador con el que se considera que se detectó un circulo.

minRadius: Radio mínimo de los círculos que se detectarán.

maxRadius: Radio máximo de los círculos que se detectarán.

Ejemplo detección de círculos

Imagen Original	Círculos detectados
	

Dr. Carlos López Franco

Imágenes

Para este algoritmo utilizaremos una imagen para la detección de líneas y una imagen para la detección de círculos.

Las imágenes pueden ser construidas manualmente (usando Paint por ejemplo), o de alguna imagen que contenga líneas o círculos.

Ejercicios

Los ejercicios son los siguientes:

1) Aplica la detección de líneas a una imagen
2) Aplica la detección de círculos a una imagen

Código

Incluye tu código donde se muestre como aplicas los operadores morfológicos.

Reporte

Incluye tu código, así como las imágenes original y resultado de los ejercicios.