

Visión Computacional

Nombre: Oscar Alexis Gonzalez Rivera

Matricula: 1517049

Actividad 1: Detección de bordes

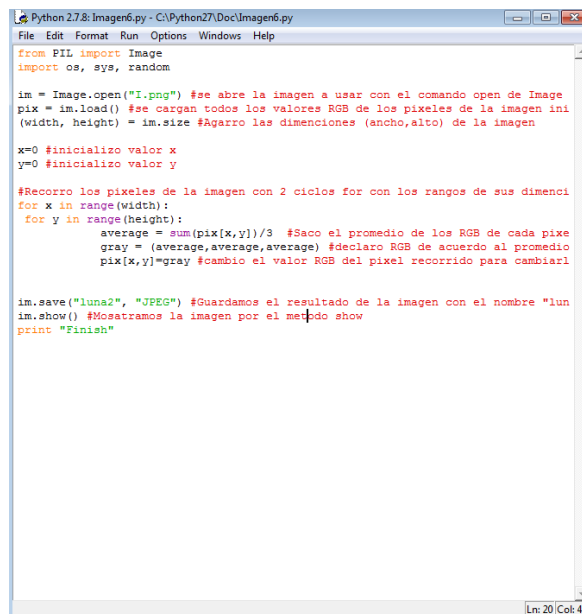
Descripción:

- Acceder a los valores de los pixeles individuales de la imagen de entrada.
- Calcular el gradiente — magnitud y dirección — (en RGB o monocromático) de cada pixel.
- Calcular la distribución de magnitudes de gradientes.
- A partir de la distribución, determinar un umbral de corte para identificar las magnitudes más fuertes.
- Clasificar como borde aquellos pixeles que superan el umbral establecido.
- Visualizar en una imagen de salida los bordes detectados, cambiando el color de aquellos pixeles a un tono llamativo no presente en la imagen de entrada.

Paso 1:

Para poder acceder a los valores de los pixeles empleé el uso de un doble for para poder recorrer todo el arreglo, en el paso anterior se muestra el recorrido del arreglo sumado el cambio de la imagen a escala de gris para facilitar la detección de los bordes.

En seguida el código empleado para el cambio a escala de grises.



```
Python 2.7.8: Imagen6.py - C:\Python27\Doc\Imagen6.py
File Edit Format Run Options Windows Help

from PIL import Image
import os, sys, random

im = Image.open("I.png") #se abre la imagen a usar con el comando open de Image
pix = im.load() #se cargan todos los valores RGB de los pixeles de la imagen ini
(width, height) = im.size #Agarro las dimensiones (ancho,alto) de la imagen

x=0 #inicializo valor x
y=0 #inicializo valor y

#Recorro los pixeles de la imagen con 2 ciclos for con los rangos de sus dimenci
for x in range(width):
    for y in range(height):
        average = sum(pix[x,y])/3 #Saco el promedio de los RGB de cada pixe
        gray = (average,average,average) #declaro RGB de acuerdo al promedio
        pix[x,y]=gray #cambio el valor RGB del pixel recorrido para cambiari

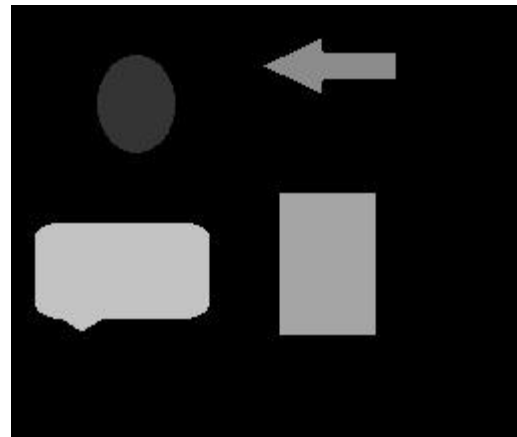
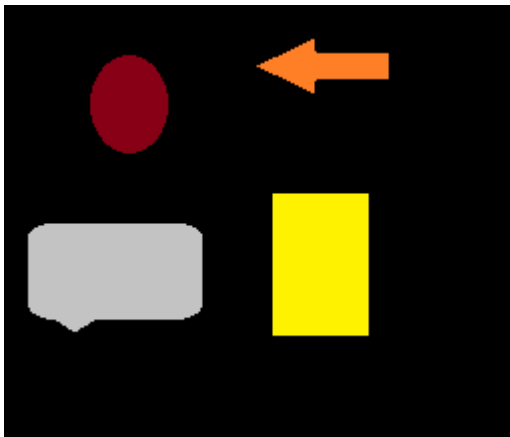
im.save("luna2", "JPEG") #Guardamos el resultado de la imagen con el nombre "lun
im.show() #Mostramos la imagen por el metodo show
print "Finish"
```

Visión Computacional

Nombre: Oscar Alexis Gonzalez Rivera

Matricula: 1517049

Resultado del programa:



Paso 2:

Para calcular el gradiente se empleó la siguiente formula:

$$G = \sqrt{gx + gy}$$

Mostrado en el código de la siguiente manera:

```
grad = int(math.sqrt(value_x + value_y)) #calculo el gradiente
```

Paso 3:

Para calcular los bordes de la imagen emplea las máscaras de Sobel para las coordenadas "x" y "y",

Empleando una máscara para cada una de estas coordenadas una vez empleado esto se empezó a calcular el gradiente para cada uno de los pixeles que componían la imagen, partiendo de la coordenada (0,0), con las máscaras podemos visitar los vecinos del pixel y así poder determinar la magnitud del mismo y con este poder clasificarlos dependiendo del umbral de corte.

Visión Computacional

Nombre: Oscar Alexis Gonzalez Rivera

Matricula: 1517049

El código donde se muestran la máscara es el siguiente:

```
from PIL import Image
import os, sys, random
import math
import numpy as np

im = Image.open("I.png") #se abre la imagen a usar con el comando open de Image
pix = im.load() #se cargan todos los valores RGB de los pixeles de la imagen ini
(width, height) = im.size #Agarro las dimensiones (ancho,alto) de la imagen

x=0 #inicializo valor x
y=0 #inicializo valor y
mask_x = ([-1,0,1],[-2,0,2],[-1,0,1])
mask_y = ([1,2,1],[0,0,0],[-1,-2,-1]) #mascara para las coordenadas en y
```

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$
$$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \quad P_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Paso 4:

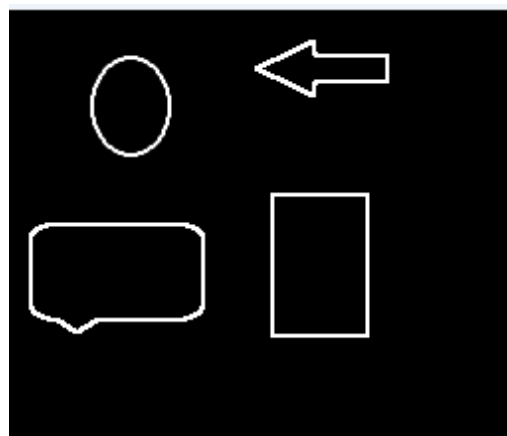
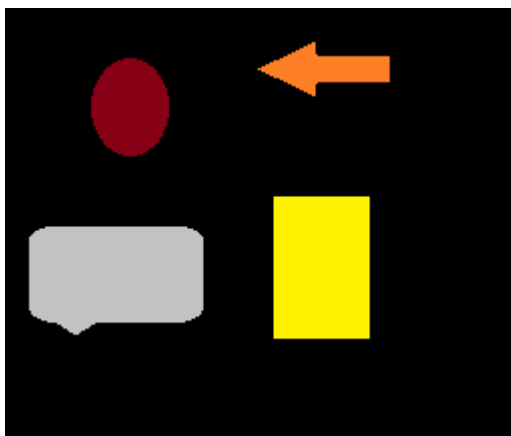
El umbral de corte se determinó en 135 como valor máximo y 0 para como valor mínimo, estos siendo blanco y negro respectivamente, se determinaron estos umbrales con el fin de eliminar el ruido al mostrar las áreas de las figuras delimitadas por bordes, para poder mostrar una imagen más limpia.

```
if grad <= 0: # declaro el umbral de corte para limpiar posible ruido, lo decl
    grad = 0
elif grad >= 135: # asi mismo declaro el umbral de corte para el blanco
    grad = 255
```

Paso 5:

Entonces los que superaban el umbral eran marcados como bordes y se pintaban de color blanco, mientras que los que no superaban el umbral eran marcados de color negro, dando como resultado las siguientes imágenes:

Imagen de Prueba 1:



Visión Computacional

Nombre: Oscar Alexis Gonzalez Rivera

Matricula: 1517049

Imagen de Prueba 2:

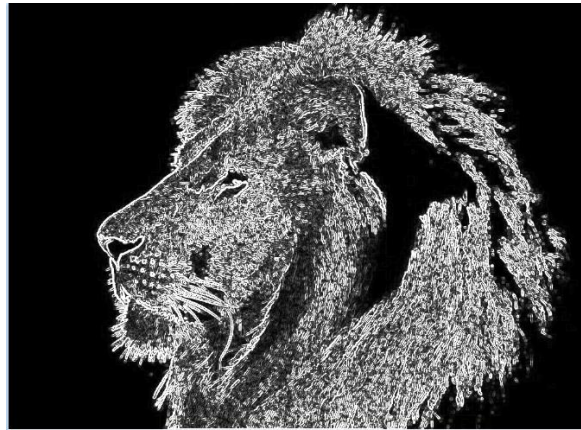


Imagen de Prueba 3:



Imagen de Prueba 4:



Visión Computacional

Nombre: Oscar Alexis Gonzalez Rivera
Código Fuente:

Matricula: 1517049

```
Python 2.7.8: Imagen6.py - C:\Python27\Doc\Imagen6.py
File Edit Format Run Options Windows Help

from PIL import Image
import os, sys, random
import math
import numpy as np

im = Image.open("C:\jpg.jpg") #se abre la imagen a usar con el comando open de Image almacenandolo en im
pix = im.load() #se cargan todos los valores RGB de los pixeles de la imagen inicializandolo en pix
(width, height) = im.size #Aqarro las dimensiones (ancho,alto) de la imagen

x=0 #inicializo valor x
y=0 #inicializo valor y
mask_x = [[-1,0,1],[-2,0,2],[-1,0,1]]
mask_y = [[1,2,1],[0,0,0],[-1,-2,-1]]#mascara para las coordenadas en y

#Recorro los pixeles de la imagen con 2 ciclos for con los rangos de sus dimensiones
for x in range(width):
    for y in range(height):
        average = sum(pix[x,y])/3 #Saco el promedio de los RGB de cada pixel recorrido
        gray = (average,average,average) #declaro RGB de acuerdo al promedio de cada pixel recorrido
        pix[x,y]=gray #cambio el valor RGB del pixel recorrido para cambiarlo a escala de grises

#declaro un nuevo recorrido de la imagen inicializando las variables
x=0
y=0
for x in range(width):
    for y in range(height):
        sumx=0.0
        sumy = 0.0
        #recorro la mascara para cada pixel recorrido
        for m in range(len(mask_x[0])):
            for h in range(len(mask_y[0])):
                try: #declaro el try para reparar el posible error de salida de la imagen al aplicar la mascara
                    x2=x+m #Visito los vecinos del pixel seleccionado
                    y2=y+h
                    sum_x= mask_x[m][h] * pix[x2, y2][0] #multiplico el valor de la mascara a la coordenada en x del pixel
                    sum_y= mask_y[m][h] * pix[x2, y2][0] #multiplico el valor de la mascara a la coordenada en y del pixel
                except: #delaro el except si se salio del area de la imagen y declaro que la suma de x y y es igual a 0
                    sum_x=0
                    sum_y=0

            sumx=sum_x+sumx #sumo el resultado de la suma en x y asi mismo en y
            sumy=sum_y+sumy
        value_x = pow(sumx,2) #elevo a la segunda potencia la resultante de la suma
        value_y = pow(sumy,2)
        grad = int(math.sqrt(value_x + value_y)) #calculo el gradiente
        if grad <= 0: # declaro el umbral de corte para limpiar posible ruido, lo declaro en 0 para declarar que es negro
            grad = 0
        elif grad >= 135: # asi mismo declaro el umbral de corte para el blanco
            grad = 255
        pix[x,y] = (grad, grad, grad)

im.save("luna2", "png") #Guardamos el resultado de la imagen con el nombre "luna2" y el tipo "JPEG"
im.show() #Mostramos la imagen por el metodo show
print "Finish"
```

Conclusión: En conclusión esta práctica es el empleo de matemáticas simples aplicadas a los pixeles con el fin de poder analizar una imagen para así poder separar los patrones comunes con un borde esto empleado con otras ciencias puede ser muy útil al momento de querer analizar más afondo una imagen, los conocimientos usados para esta actividad fueron de mucha ayuda para orientarme a poder comprender y analizar el funcionamiento de diversos sistemas orientadas a la visión computacional tales como las cámaras inteligentes.

Bibliografía:

Machine Vision by E.R. Davies Third Edition, Chapter 5.

<http://www.mathsisfun.com/algebra/matrix-introduction.html>

<https://docs.python.org/2/library/math.html>