UNIVERSIDAD SANTO TOMÁS

PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

SECCIONAL TUNJA

VIGILADA MINEDUCACIÓN - SNIES 1732

Acreditación Institucional Internacional

OTORGADA POR EL IAC CINDA ACUERDO 55 DEL 8 DE MAYO–VIGENCIA 5 AÑOS

ACREDITACIÓN INSTITUCIONAL DE ALTA CALIDAD MULTICAMPUS

Vigencia por seis años

QS STARS RATED FOR EXCELLENCE

ISO 9001 icontec SC4289-1

CERTIFIED IQNet MANAGEMENT SYSTEM

UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
SECCIONAL TUNJA
VIGILADA MINEDUCACIÓN - SNIES 1732

Faculty: Systems engineer
Course: Deep Learning
Topic: Computer visión - filters

Professor: Luis Fernando Castellanos Guarin
Email: Luis.castellanosg@usantoto.edu.co
Phone: 3214582098

CONTENIDO

- Representación de imágenes
- Tipos de datos e imágenes en OpenCV
- Canales
- Filtros lineales
- Suavizado del ruido
- Resaltado de aristas con filtros lineales
- Operadores morfológicos
- Filtros y operaciones en OpenCV

¡Siempre hacia lo alto!

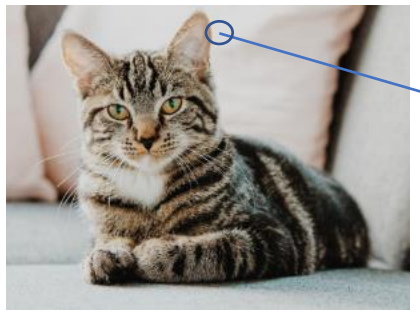
Describir Imágenes como un función

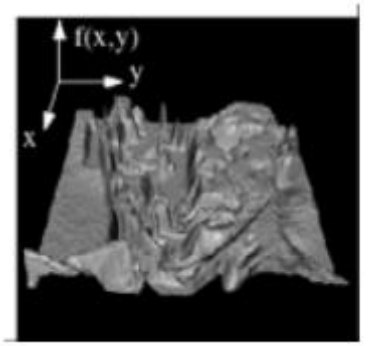
- Podemos pensar en una **imagen** como una función *f*, de R^2 to R:
- *f* (*x, y*) devuelve la **intensidad** en la posición (*x, y*)
- En la realidad, la imagen sólo está definida sobre un rectángulo [*a*,*b*]x[*c*,*d*]

Una imagen de color son tres funciones juntas.

Podemos formalizar este concepto con una función color donde la imagen es solo tres funciones pegadas juntas. Podemos escribir esto como una función de "valor vectorial".

$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$



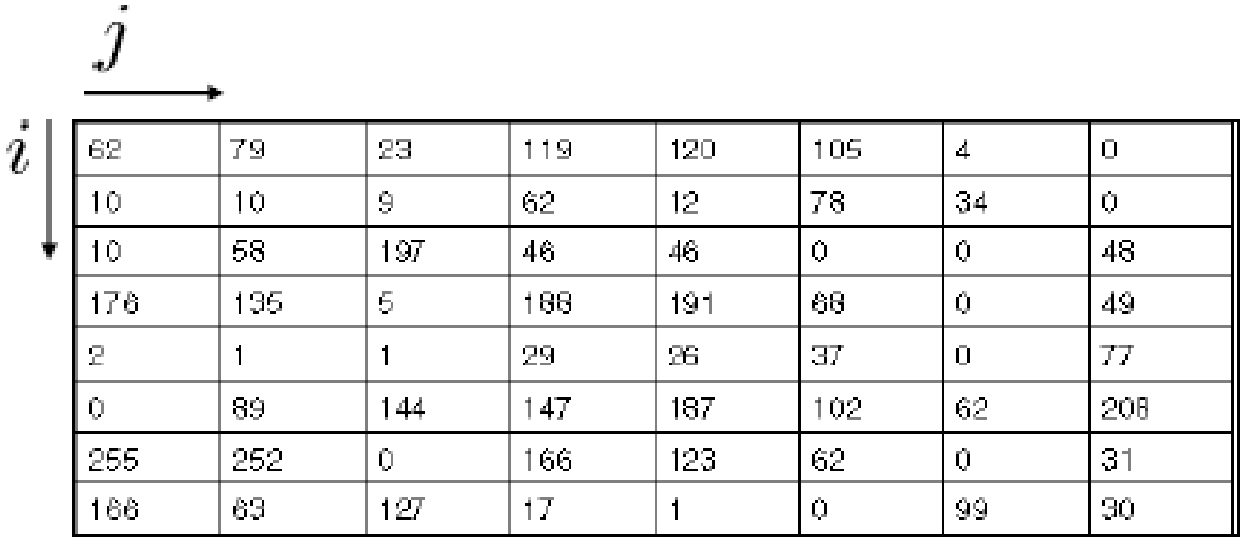


¡Siempre hacia lo alto!

¿Cómo describir una imagen digital?

Fotografías o imágenes digitales (discretas):

- **Muestreamos** al espacio 2D sobre una matriz
- **Discretizamos** cada muestra (donde redondeamos hasta el entero más cercano)
- La imagen puede representarse entonces como una matriz de valores enteros

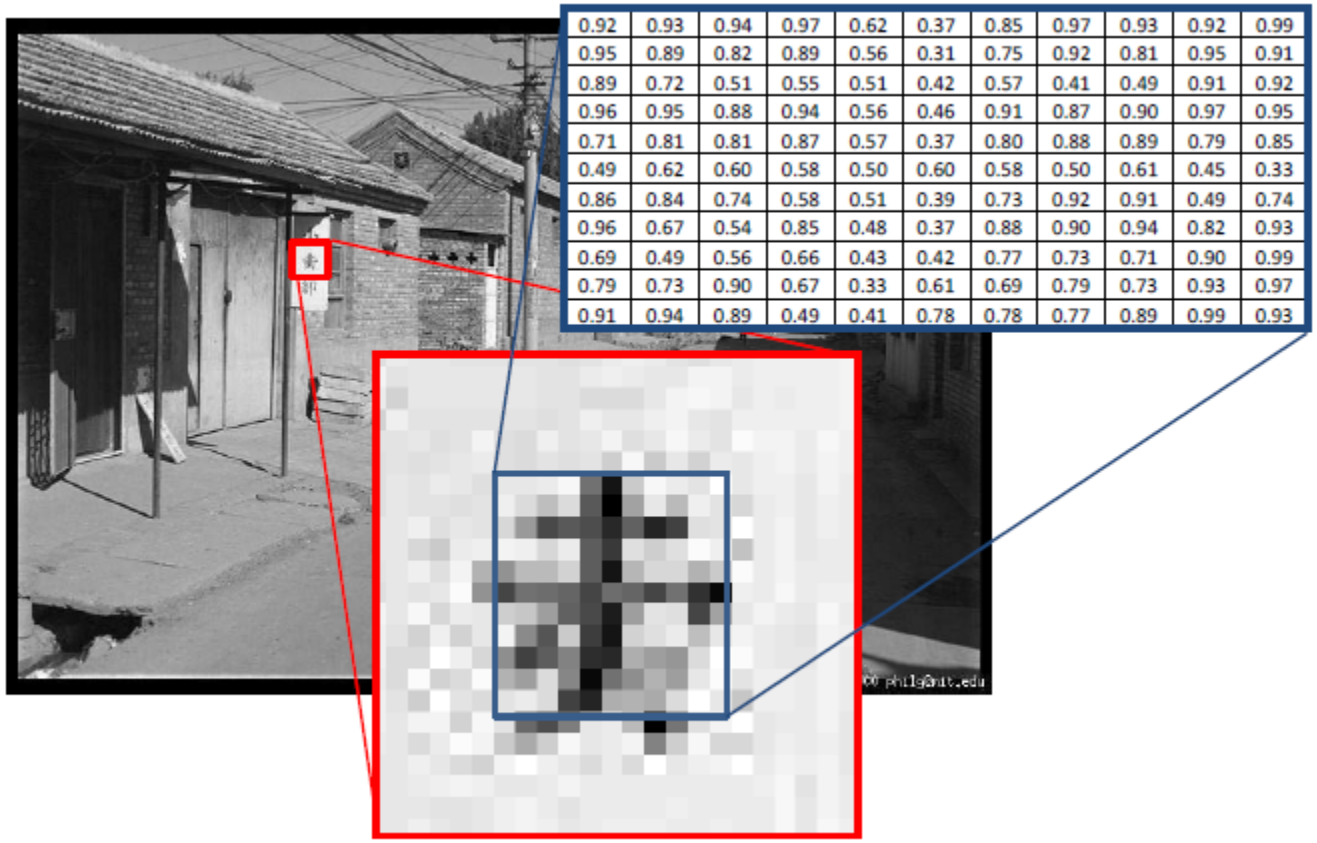


62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

¡Siempre hacia lo alto!

¿Cómo describir una imagen digital?

Niveles de gris



¡Siempre hacia lo alto!

¿Cómo describir una imagen digital?

Color, 3 canales (R,G,B)

Red

Green

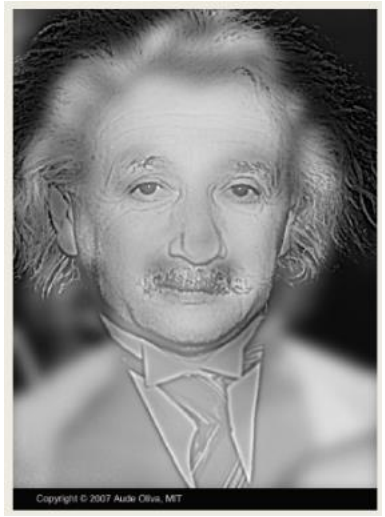
Blue

¡Siempre hacia lo alto!

Filtros que podemos aplicar en imágenes

Transforman imágenes in nuevas imágenes
Aplicaciones

- Suavizado del ruido
- Resaltado de aristas y otras características de la imagen
- Búsqueda de plantillas



- Filtros lineales y convolución
- Muestreo y aliasing
- Filtros como plantillas
- Pirámides gaussianas

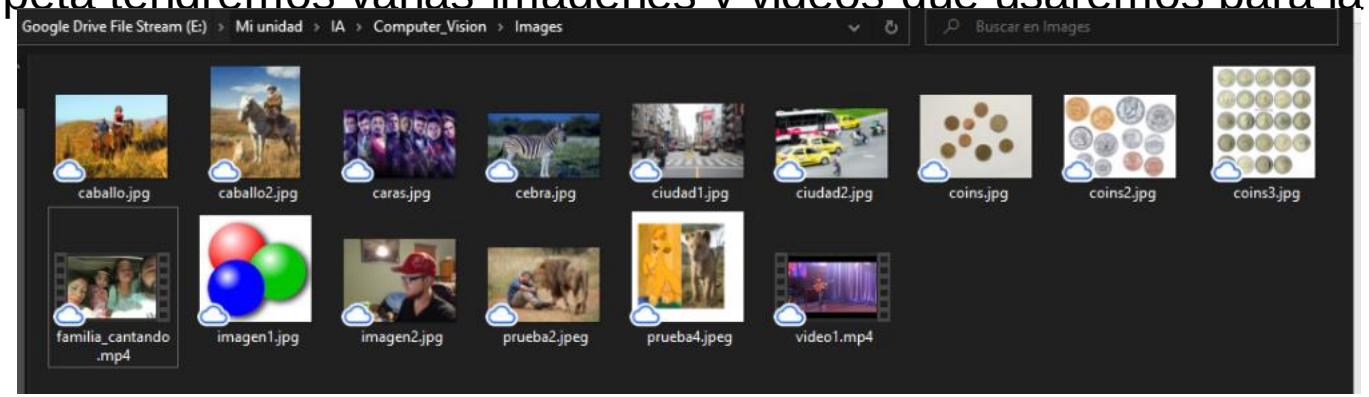
¡Siempre hacia lo alto!

¡Menos charla
y
más código!

Opencv en Google Colaboratory

Paso 0: En Google Drive crear una carpeta denominada "**Computer_vision**" y dentro de ella una subcarpeta denominada "**images**" similar a como esta creada en la carpeta compartida de la asignatura: **"https://github.com/luisFernandoCastellanosG/Machine_learning/tree/master/Computer_vision/OpenCv**

Dentro de esa carpeta tendremos varias imágenes y videos que usaremos para la clase.



Crear un nuevo libro de jupyter denominado **"computer_vision_opencv_basic.ipynb"**

hacia lo alto!

Opencv en Google Colaboratory

Paso 1: Montar Google Drive en Google colaboratory

```
from google.colab import drive
drive.mount('/content/drive')
```

Paso 2: Importando librerías necesarias.

```
import cv2                                  #libreria de opencv
import numpy as np                          #libreria para manejo de arrays
from matplotlib import pyplot as plt        #libreria para visualizar graficos y otros
from google.colab.patches import cv2_imshow #modulo para visualizar imagenes en google colaboratory
```

Paso : Cargando y visualizando imágenes.

```
img = cv2.imread("/content/drive/My Drive/IA/Computer_Vision/Images/imagen1.jpg")
if img.size == 0:
  print("Error: la imagen no fue cargada con exito.")
else:
  print('Image Dimensiones  : ',img.shape)
  print("Alto: ",str(img.shape[0]),",Ancho: ",img.shape[1],"Número de canales:",img.shape[2])
  cv2_imshow(img)
```

Opencv en Google Colaboratory

Información de la imagen.

1. Alto(Height): representa el número de filas de píxeles en la imagen o el número de píxeles en cada columna de la matriz de imágenes.

2.Ancho(Width): representa el número de columnas de píxeles en la imagen o el número de píxeles en cada fila de la matriz de imágenes.

3. Numero de canales(Number of Channels): representa el número de componentes utilizados para representar cada pixel.

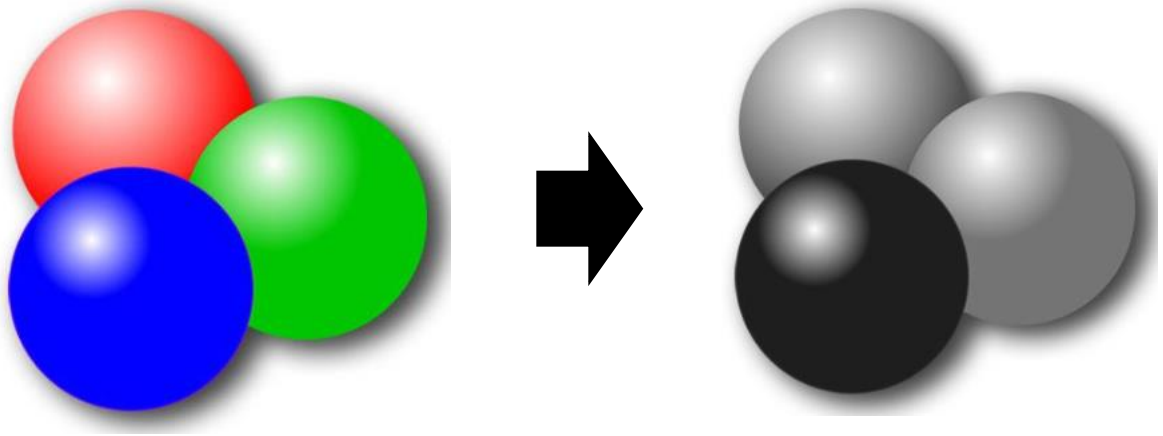
Por ejemplo = 4 representa ***Alpha***, ***Red***, ***Green*** y ***Blue*** canales.

¡Siempre hacia lo alto!

Paso 4. Aplicando filtros

Gris (Gray)

```
#convirtiendo la imagen a escala de grises
processed_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
#guardando la imagen en drive
cv2.imwrite('/content/drive/My Drive/IA/Computer_Vision/Images/image1_processed_gray.png', processed_image)
cv2_imshow(processed_image)
```



¡Siempre hacia lo alto!

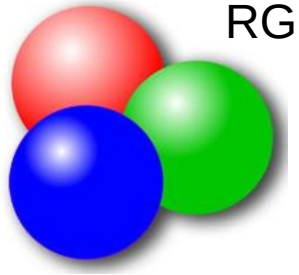
Opencv en Google Colaboratory

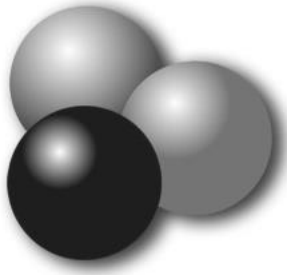
Filtros

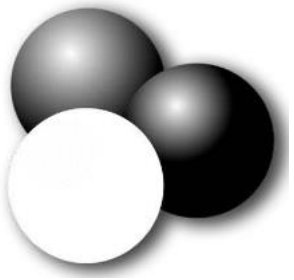
Escala de colores (Channels colors)

```
#extrayendo el canal rojo (RED channel)
red_channel = img[:,:,2]
#extrayendo el canal verde (Green channel)
green_channel = img[:,:,1]
#extrayendo el canal azul (blue channel)
blue_channel = img[:,:,0]
cv2.imwrite('/content/drive/My Drive/IA/Computer_Vision/Images/image1_processed_red.png', red_channel)
cv2.imwrite('/content/drive/My Drive/IA/Computer_Vision/Images/image1_processed_green.png', green_channel)
cv2.imwrite('/content/drive/My Drive/IA/Computer_Vision/Images/image1_processed_blue.png', blue_channel)
```

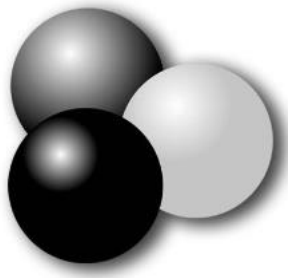
RGB



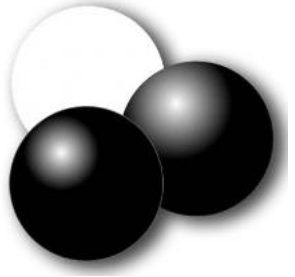




Blue



Green



Red

¡Siempre hacia lo alto!

Opencv en Google Colaboratory

Adquisición y representación de imágenes - Resolución en amplitud

Dependiendo del número de niveles de gris, generalmente se usan 256 niveles

Pero para trabajar en Deep learning necesitaremos usar una de estas tres:

16 Niveles de intensidad
160 x 160 píxeles

8 Niveles de intensidad
160 x 160 píxeles

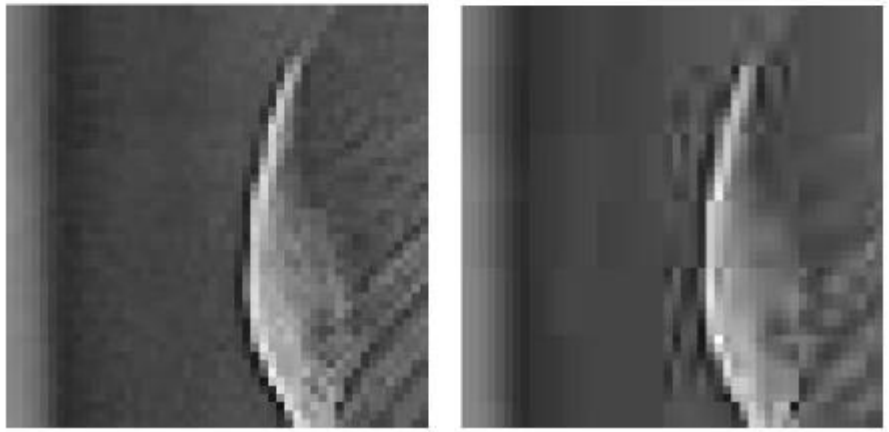
2 Niveles de intensidad
160 x 160 píxeles

¡Siempre hacia lo alto!

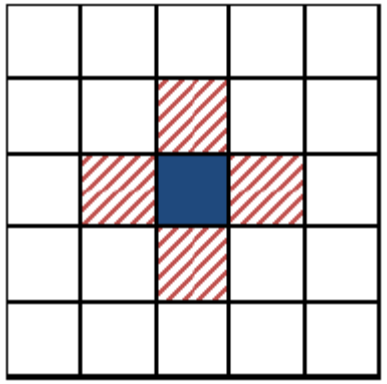
Opencv en Google Colaboratory

Adquisición y representación de imágenes –Comprensión de imágenes

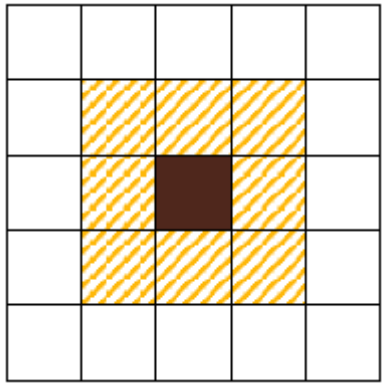
(a) (b)



Vecindad x 4



Vecindad x 8



¡Siempre hacia lo alto!

Opencv en Google Colaboratory

Filtros

Resaltar saturación de Color es un practica muy común en la clasificación de imágenes:

```
#Cargamos la imagen de la que queremos reslartar el rojo
img = cv2.imread("/content/drive/My Drive/IA/Computer_Vision/Images/imagen2.jpg")
#convertimos la imagen a RGB HSV
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
lower_red = np.array([30,150,50]) # codigo minimo del rojo en del B,G,R
upper_red = np.array([255,255,180]) # codigo maximo del rojo en B,G,R
#Creamos una mascara de la imagen donde resaltaremos el color rojo
mask_red = cv2.inRange(hsv, lower_red, upper_red)
#aplicamos la mascara en la imagen
img_result = cv2.bitwise_and(img,img, mask = mask_red)
#visualizamos la imagen
cv2_imshow(img)
cv2_imshow(mask_red)
cv2_imshow(img_result)
```

Opencv en Google Colaboratory

Filtros

Gradiente de Imágenes

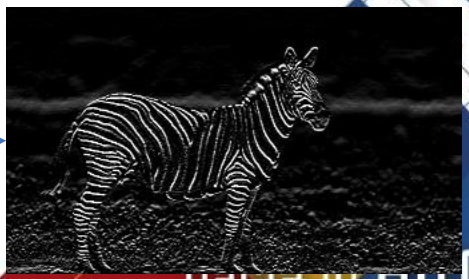
El gradiente de una imagen mide cómo esta cambia en términos de color o intensidad.

La magnitud del gradiente nos indica la rapidez con la que la imagen está cambiando, mientras que la dirección del gradiente nos indica la dirección en la que la imagen está cambiando más rápidamente.

Matemáticamente, el gradiente se define por la derivadas parciales de una función dada (intensidad en el caso imágenes) a lo largo de las direcciones X e Y.

- **Derivadas Laplacianas**
- **Derivadas Sobel y Scharr**

```
# cargamos la imagen y la pasamos de una a gris
img = cv2.imread("/content/drive/My Drive/IA/Computer_Vision/Images/cebra.jpg",0)
#cambiar CV_64F  por CV_8U, si quiere verlo en negro/blanco
laplacian = cv2.Laplacian(img,cv2.CV_64F)
sobelx = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=3)
sobely = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=3)
cv2_imshow(img)
cv2_imshow(sobelx)
cv2_imshow(sobely)
```

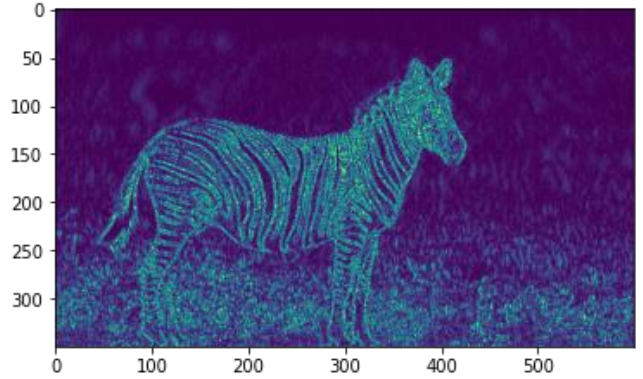


Opencv en Google Colaboratory

Filtros

Gradiente de Imágenes mejorado

al utilizar **cv2.CV_8U** todas las transiciones negativas se hacen cero y por lo tanto no son detectadas por los filtros y se pierden los bordes.
Para evitar este problema y detectar ambos bordes, manteniendo la salida final en blanco y negro, la mejor opción es mantener el tipo de datos de salida en algunas formas superiores, como **cv2.CV_16S, cv2.CV_64F** etc; tomar su valor absoluto y luego convertirlo de nuevo **a cv2.CV_8U**.




```
sobelx8u=cv2.Sobel(img,cv2.CV_8U,1,0,ksize=3)
#Utilizando cv2.CV_64F. Luego toma el valor absoluto y hace la conversión a cv2.CV_8U
sobelx64f=cv2.Sobel(img,cv2.CV_64F,1,0,ksize=3)
abs_sobel64f=np.absolute(sobelx64f)
sobel_8u=np.uint8(abs_sobel64f)
plt.imshow(sobel_8u)
plt.show()
cv2_imshow(sobel_8u)
```

¡Siempre hacia lo alto!

Opencv en Google Colaboratory

Filtros

Detector de bordes Canny, cv2.Canny ()

Canny Edge Detection es un algoritmo de detección de bordes popular. Fue desarrollado por John F. Canny en 1986.

Es un algoritmo de etapas múltiples y revisaremos cada etapa:

1. Convertir la imagen en escala de grises con OpenCV
2. Filtro Guassiano para la eliminación de ruido en imágenes: Dado que la detección de bordes es susceptible al ruido en la imagen, el primer paso es eliminar el ruido en la imagen con un filtro gaussiano de 5x5
3. Detector de bordes Canny
    - Detección de bordes con Sobel
    - Dibujar los contornos en una imagen con OpenCV

.

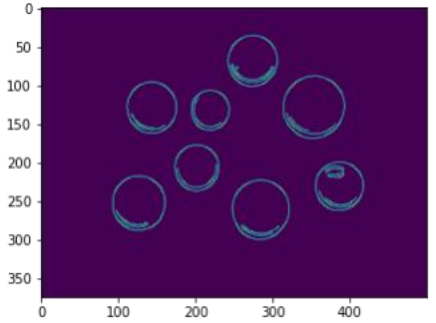
¡Siempre hacia lo alto!

Opencv en Google Colaboratory

Filtros

Detector de bordes Canny, contando monedas

```
img_gris = cv2.imread("/content/drive/My Drive/IA/Computer_Vision/Images/coins.jpg",0)
cv2_imshow(img)
#Aplicamos un filtro gausiano para suavizar la imagen
img_gauss = cv2.GaussianBlur(img_gris, (5,5), 0)
cv2_imshow(img_gauss)
img_canny = cv2.Canny(img_gauss, 50, 150)
plt.imshow(img_canny)
plt.show()
cv2_imshow(img_canny)
```



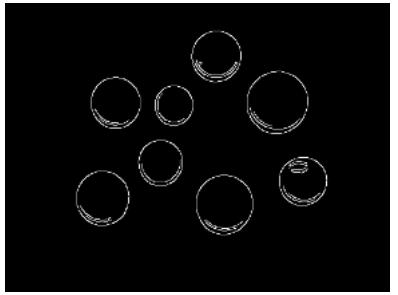
3 parámetros:

- El primero es la imagen donde queremos detectar los bordes. Esta imagen es la que ya hemos suavizado en el paso anterior.
- El segundo parámetro será el umbral mínimo
- El tercer parámetro será el umbral máximo.

Debemos buscar los más adecuados para que en la imagen se detecten más bordes siempre y cuando no sean contornos es decir, curvas cerradas.

Un valor de 50 para el mínimo y 150 para el máximo puede ser un buen comienzo.

Intenta subir el máximo para eliminar esos bordes innecesarios.



Opencv en Google Colaboratory

Filtros

Detector de bordes Canny, contando monedas

Buscamos los contornos

```
contornos,_=cv2.findContours(img_canny.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
print("He encontrado {} objetos".format(len(contornos)))
```

```
He encontrado 8 objetos
```

- cv2.RETR_EXTERNAL, devuelve todos los contornos externos (si tiene contornos internos no los contabiliza).
- cv.CHAIN_APPROX_SIMPLE . Elimina todos los puntos redundantes y comprime el contorno, ahorrando así memoria.

¡Siempre hacia lo alto!

Detector de bordes Canny, contando monedas

Taller:

Para las imágenes **coins2.jpg** y **coins3.jpg**, diseñar programa en Python para que reconozca la cantidad de monedas.

Varia en algo el programa para cada imagen…y como lo podría optimizar?

UNIVERSIDAD SANTO TOMÁS

PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

SECCIONAL TUNJA

VIGILADA MINEDUCACIÓN - SNIES 1732

¡Siempre hacia lo alto!

USTATUNJA.EDU.CO

@santotomastunja