Practica 5
Github

https://github.com/Leogaltre/Practica-5-Segmentaci-n-de-Umbralizaci-n-de-Imagenes.git

Objetivo: Utilizar las funciones de umbrales para la recuperación de información.
Threshold1 binary, b_inv, Trunc, To Zero, Tz_inv, Mean, Gaus, Otsu.

Tipo de letra de escritura en Word para mostrar la programación -> 3ds tamaño 11
Tipo de Letra de Python -> Courier New tamaño 12

Objetivo dibujar sobre una imagen creada en fondo negro, posteriormente a la hora de entrega se agregará la segmentación y ROI

Información

Introducción – Segmentación de Umbral

La segmentación de umbral se usa comúnmente en imágenes en escala de grises. El valor de la escala de grises se divide en 0 o 255 por un cierto umbral, de modo que el valor de píxel de la imagen sea solo 0 o 255 (no blanco ni negro).

Dado que los valores de píxeles de los diferentes objetos son diferentes, los objetos de la imagen se segmentan a nivel de píxeles de acuerdo con el umbral establecido, lo que favorece el procesamiento posterior de la imagen, simplifica la imagen y reduce la cantidad de datos, lo que puede resaltar el objetivo de interés. Contorno. Para procesar y analizar la imagen binaria, la imagen gris debe binarizarse primero para obtener una imagen binaria.

La imagen en color original se usa en el ejemplo, y cada canal de la imagen se binariza durante la binarización. Al utilizar imágenes en escala de grises, puede mostrar imágenes en blanco o negro.

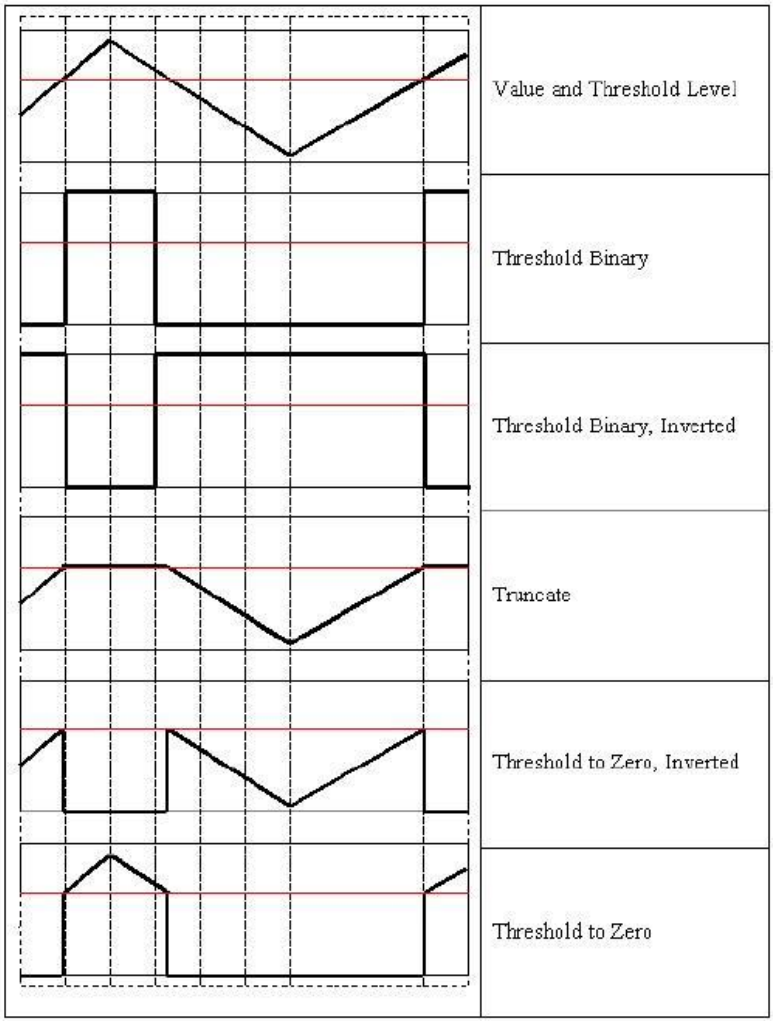
THRESH_BINARY : $\texttt{dst}(x,y) = \begin{cases} \texttt{maxVal} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$

THRESH_BINARY_INV : $\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxVal} & \text{otherwise} \end{cases}$

THRESH_TRUNC : $\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$

THRESH_TOZERO : $\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$

THRESH_TOZERO_INV : $\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$



Umbral Adaptativo

Cuando diferentes partes de la misma imagen requieren diferente brillo, el umbral global no es adecuado. En este caso, se necesita un umbral adaptativo. El umbral adaptativo calculará diferentes umbrales en diferentes áreas de la imagen para obtener una jerarquía clara. imagen.

Aprendiendo código

Cuando se usa el umbral automático, solo se pueden usar imágenes de un solo canal

adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize,C,dst=None)

#maxValue: tipo doble, el valor máximo del umbral

#adaptiveMethod: tipo Int, hay dos opciones

ADAPTIVE_THRESH_MEAN_C (para obtener el valor promedio promediando)

ADAPTIVE_THRESH_GAUSSIAN_C (valor gaussiano obtenido a través de gaussiano)

thresholdType: tipo Int, el método es el siguiente:

THRESH_BINARY Umbral binario -> mayor que el umbral es 1 y menor que el umbral es 0

THRESH_BINARY_INV Umbral binario inverso -> mayor que el umbral es 0, menor que el umbral es 1

THRESH_TRUNC Truncar umbral -> mayor que el umbral es el umbral, menor que el umbral permanece sin cambios

THRESH_TOZERO El umbral se cambia a 0 -> el valor mayor que el umbral no cambia, y el valor menor que el umbral es todo 0

THRESH_TOZERO_INV De threshold se reduce a 0 -> mayor que el umbral es 0, menos que el umbral permanece sin cambios

#blockSize: tipo Int, este valor determina qué tan grande es el bloque de vecindad del píxel.

C: Cantidad de ajuste del valor de compensación, que calcula los parámetros utilizados por adaptiveMethod.

¡¡Nota si se usa -> algún adaptiveMethod como ADAPTIVE_THRESH_MEAN_C o ADAPTIVE_THRESH_GAUSSIAN_C es necesario que se use un solo canal en gris!! Y también por alguna razón necesitan ingresarse valores pares menos 1, ejemplo 1, 3, 5, 7, 9, 11, ... 255 (Aun no se investiga su razón)

Código

```
"Tratamiento de Imagenes Mediante Umbrales"

import cv2
from matplotlib import pyplot as plt

# imag = cv2.imread('AMERIKEos.jpeg', 0)
# img = cv2.resize(imag, (300, 600))

imag = cv2.imread('amosc.jpeg', 0)
img = cv2.resize(imag, (600, 450))

#img - imagen en color
#El valor de umbral aquí 80 se establece como el
umbral, todos los valores superiores a 80 se
convierten en 255 dados más tarde, y los menores de 80
se convierten en 0
# Máximo 255
# Formatcv2.THRESH_BINARY procesamiento binario,
# ret, umbral; bin, imagen después del procesamiento
binario

Val = int(input('Valor de Umbral regulador: ')) #10/
80
if Val%2 == 0:
    Val1 = Val+1
```

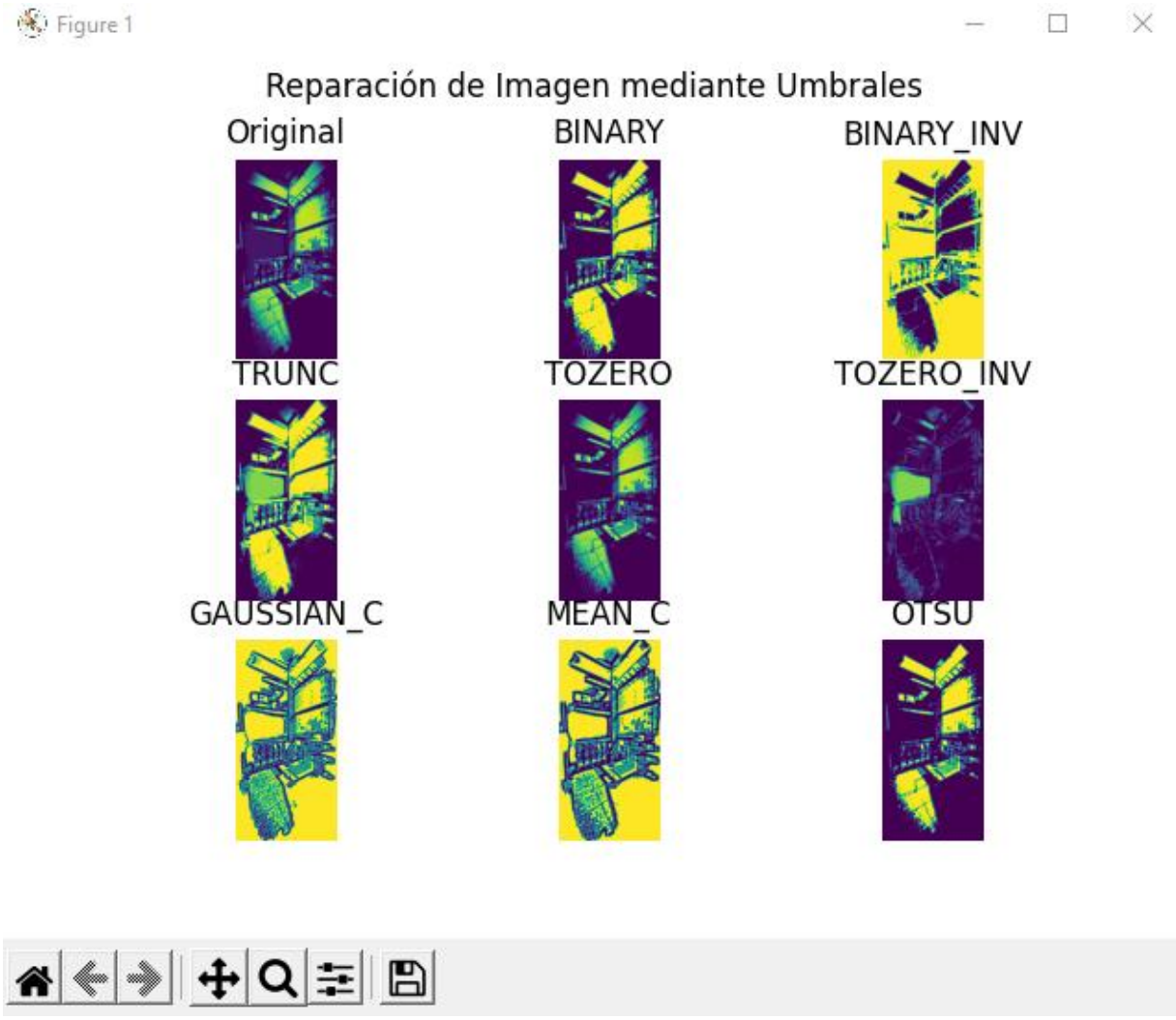
```
else:
    Val1 = Val

# THRESH_BINARY
ret, thresh1 = cv2.threshold(img, Val, 255,
cv2.THRESH_BINARY)
# THRESH_BINARY_INV
ret, thresh2 = cv2.threshold(img, Val, 255,
cv2.THRESH_BINARY_INV)
# THRESH_TRUNC
ret, thresh3 = cv2.threshold(img, Val, 255,
cv2.THRESH_TRUNC)
# THRESH_TOZERO
ret, thresh4 = cv2.threshold(img, Val, 255,
cv2.THRESH_TOZERO)
# THRESH_TOZERO_INV
ret, thresh5 = cv2.threshold(img, Val, 255,
cv2.THRESH_TOZERO_INV)

"Necesita de una Imagen en un solo canal es decir en
Gris"

# ADAPTIVE_THRESH_GAUSSIAN_C
thresh6 = cv2.adaptiveThreshold(img, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,
Val1, 2)
# ADAPTIVE_THRESH_MEAN_C
thresh7 = cv2.adaptiveThreshold(img, 255,
cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, Val1,
2)
# THRESH_BINARY + THRESH_OTSU
ret, thresh8 = cv2.threshold(img, Val, 255,
cv2.THRESH_BINARY+cv2.THRESH_OTSU)

titles = ['Original', 'BINARY', 'BINARY_INV','TRUNC',
'TOZERO',
```

```
          'TOZERO_INV','GAUSSIAN_C', 'MEAN_C', 'OTSU']

'''
# Como deberían llamarse
titles = ['Original', 'THRESH_BINARY',
'THRESH_BINARY_INV',
          'THRESH_TRUNC', 'THRESH_TOZERO',
'THRESH_TOZERO_INV',
          'ADAPTIVE_THRESH_GAUSSIAN_C',
'ADAPTIVE_THRESH_MEAN_C'...]
'''

images = [img, thresh1, thresh2, thresh3, thresh4,
thresh5, thresh6, thresh7, thresh8]

for i in range(9):
    #print(i)
    # Tamaño 2 en Y, 4 en X
    plt.subplot(3, 3, i+1)
    plt.imshow(images[i])
    plt.title(titles[i])
    plt.axis('off')
plt.suptitle('Reparación de Imagen mediante Umbrales')
plt.show()
```

Como luce la práctica

```
Valor de Umbral regulador: 15
```


Figure 1
Reparación de Imagen mediante Umbrales
Original
BINARY
BINARY_INV
TRUNC
TOZERO
TOZERO_INV
GAUSSIAN_C
MEAN_C
OTSU
AMERIKE
LICENCIATURA EN



Referencias:

Segmentación de umbral de imagen

https://programmerclick.com/article/11811566694/

RGB a HSV (Fallo en H tiene un rango de 0 - 360 y no de 0 - 179)

https://programmerclick.com/article/1668408560/

Organización para impresión con Matploit