UNIVERSILDAD DEL VALLE

INVESTIGACIÓN EL MÉTODO DE OTSU

ESTUDIANTE: IBLING GABRIEL VALLE NUÑEZ

MATERIA: PROCESAMIENTO DIGITAL DE IMAGENES

CURSO: QUINTO SEMESTRE

CARRERA: INGENIERIA DE SISTEMAS INFORMATICOS

FECHA: 05/02/2025

SUCRE – 2025

CONCEPTO

Según (Ning, 2019):

> El método de Otsu se puede utilizar para la reducción de una imagen en escala de grises a una imagen binaria. Supone que la imagen contiene dos clases de píxeles que siguen un histograma bimodal (píxeles de primer plano y píxeles de fondo). La clave es que este método puede generar un umbral basado en el histograma de una imagen.
>
> Como muestra la figura del histograma, encuentra el umbral t que minimiza la suma ponderada de las varianzas dentro del grupo (lo analizaremos más adelante; tenga en cuenta que se refiere a las variaciones causadas por las diferencias dentro de los grupos individuales) para los dos grupos que resultan de separar los tonos de gris en el valor t.

Código implementado por el autor (Ning, 2019)

```
img_hist = np.histogram(img_flattened,bins=255)
lista_de_brillo = img_hist[1][1:].astype(int) #ignorar el
primer elemento
hist_counts = img_hist[0]
normed_hist_counts = hist_counts/hist_counts.sum()
probabilidad_ponderada =
lista_de_brillo*normed_hist_countsomiga_arr =
recuentos_hist_normados.cumsum()
Ex_arr = probabilidad_ponderada.cumsum()def
calcular_varianza_entre_grupos(umbral):
    eps = 1e-10 #evitar dividir por cero
    omiga0 = omiga_arr[umbral-1]
    Ex0 = Ex_arr[umbral-1]
    varianza_entre_grupos = omiga0*(Ex0-EX)**2/(1-omiga0+eps)
    return varianza_entre_grupos

varianza_entre_clases_arr =
np.array(list(map(calcular_varianza_entre_grupos,lista_brillo
)))
```

Código del autor (Sakshi, 2020)

```
    # Set total number of bins in the histogram
    bins_num = 256

    # Get the image histogram
    hist, bin_edges = np.histogram(image, bins=bins_num)

    # Get normalized histogram if it is required
    if is_normalized:
        hist = np.divide(hist.ravel(), hist.max())

    # Calculate centers of bins
    bin_mids = (bin_edges[:-1] + bin_edges[1:]) / 2.

    # Iterate over all thresholds (indices) and get the probabilities w1(t), w2(t)
    weight1 = np.cumsum(hist)
    weight2 = np.cumsum(hist[::-1])[::-1]

    # Get the class means mu0(t)
    mean1 = np.cumsum(hist * bin_mids) / weight1
    # Get the class means mu1(t)
    mean2 = (np.cumsum((hist * bin_mids)[::-1]) / weight2[::-1])[::-1]

    inter_class_variance = weight1[:-1] * weight2[1:] * (mean1[:-1] - mean2[1:]) ** 2

    # Maximize the inter_class_variance function val
    index_of_max_val = np.argmax(inter_class_variance)

    threshold = bin_mids[:-1][index_of_max_val]
    print("Otsu's algorithm implementation thresholding result: ", threshold)
```

REFERENCIA BIBLIOGRÁFICA

Ning, M. (29 de Marzo de 2019). *El método de Otsu*. Obtenido de Medium: https://medium.com/@MinghaoNing/otsus-method-db49e2f85093

Sakshi, A. (5 de Agosto de 2020). *Umbralización de Otsu con OpenCV*. Obtenido de LearnOpenCV: https://learnopencv.com/otsu-thresholding-with-opencv/

https://github.com/muthuspark/ml_research/blob/master/Otsu%20Thresholding%20implementation.ipynb