Universidad De Guadalajara

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERIAS

ALGORITMO 3

VISION ROBOTICA

INGENIERIA EN ROBOTICA

Alumno: Carlos Omar Rodriguez Vazquez

Introducción

El presente informe aborda el tema del cálculo y particionamiento del histograma de una imagen en el contexto de la visión robótica. El histograma de una imagen en escala de grises constituye una representación visual de la distribución de intensidades de píxeles en la misma, lo que resulta fundamental en diversas aplicaciones de procesamiento de imágenes. A lo largo de este documento, se examinarán los fundamentos teóricos del cálculo del histograma y se explorarán diversas técnicas de particionamiento que permiten segmentar el rango completo de intensidades de píxeles en intervalos específicos. Se presentarán también ejemplos prácticos de la implementación de algoritmos para el cálculo y visualización del histograma en Python utilizando la biblioteca OpenCV.

Marco Teorico

Histograma de una imagen

Calcular el histograma de una imagen en escala de grises implica la representación gráfica de la distribución de intensidades de píxeles en la imagen. En el contexto de OpenCV, se utiliza la función cv.calcHist(). Los parámetros de esta función son:

```
hist = cv.calcHist([ig], [0], None, [256], [0,256]).flatten()/(M*N)
```

- **'images':** Una lista de imágenes de entrada. En este caso, '[ig]', donde 'ig' representa la imagen en escala de grises.
- **'channels':** Índice de canales para los cuales se calculará el histograma. Aquí '[0]' indica que se calcual el histograma para el primer (y único) canal de la imagen en escala de grises.
- **'mask':** Una máscara de imagen. Si se proporciona, se calculará el histograma solo para los píxeles donde la máscara sea diferente de cero.
- **'histSize':** El número de contenedores del histograma. '[256]' especifica que habrá 256 contenedores, representares, representando las intensidades de píxeles desde 0 hasta 255.
- **'ranges':** El rango de intensidades de píxeles que se tendrán en cuenta en el histograma. '[0,256]' indica que se considerarán píxeles con intensidades desde 0 hasta 255.
- El métdo '.flatten()' transforma la estructura multidimensional a un array unidimencional, donde todos los valores del histograma se almacenan secuencialmente. Esto simplifica el acceso y manipulación de los datos del histograma.

Después de calcular el histograma, es común normalizarlo para facilitar su interpretación y comparación entre diferentes imágenes. Para ello, primero se necesita obtener las dimensiones de la imagen utilizando [M, N] = ig.shape[0:2], donde ig es la imagen en escala de grises. Esta operación proporciona las dimensiones de altura (M) y ancho (N) de la imagen. Luego, se procede a normalizar el histograma dividiendo cada bin por el número total de píxeles en la imagen, es decir, hist_normalizado = hist / (M * N). Esto asegura que el histograma normalizado represente la distribución de intensidades relativa en lugar de absoluta.

Particionamiento del Histograma

El cálculo del histograma de una imagen proporciona una representación visual de la distribución de intensidades de píxeles en la misma, lo que resulta fundamental en diversas aplicaciones de procesamiento de imágenes. Sin embargo, la elección de los intervalos o contenedores para la construcción del histograma puede influir significativamente en la interpretación de los datos. En este sentido, se pueden emplear diferentes técnicas de particionamiento, tales como terciles, cuartiles, quintiles, deciles y percentiles, para dividir el rango completo de intensidades de píxeles en intervalos específicos. Los terciles dividen el conjunto de datos en tres partes iguales, mientras que los cuartiles lo hacen en cuatro partes iguales, y así sucesivamente hasta los percentiles, que representan el porcentaje de datos que se encuentran por debajo de un valor dado. La elección de uno u otro método de particionamiento dependerá del grado de detalle requerido en el análisis del histograma y de la naturaleza específica de la imagen en cuestión. Por ejemplo, el uso de cuartiles puede ser útil para identificar umbrales de intensidad en imágenes médicas, mientras que los percentiles pueden ser más adecuados para resaltar áreas de interés en imágenes de alto contraste. En última instancia, la selección del método de particionamiento del histograma dependerá de los objetivos específicos de análisis y del contexto de la aplicación.

Ejercicios

Imagen Utilizada

Figura 1: Imagen utilizada para los ejercicios.

- Ejercicio 1

```
hist = cv.calcHist([ig], [0], None, [256], [0,256]).flatten()/(M*N)

fig = plt.figure()
plt.bar(range(len(hist)), hist)
plt.show()
```

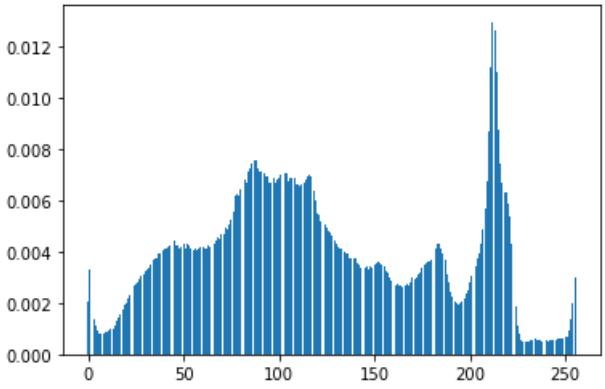

Figura 2: Histograma del Ejercicio 1.

- Ejercicio 2

```
hist2 = cv.calcHist([ig], [0], None, [3], [0,256]).flatten()/(M*N)

plt.bar(range(len(hist2)), hist2)
plt.show()
```

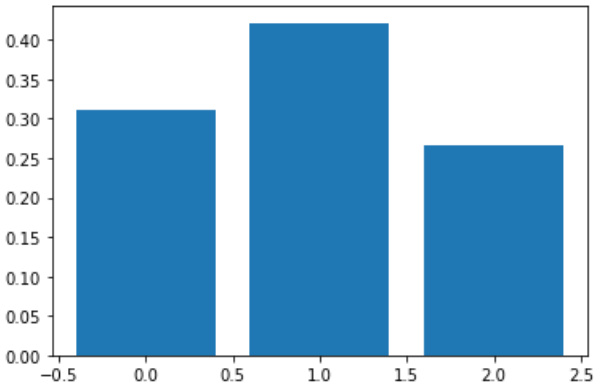

Figura 3: Histograma del Ejercicio 2.

- Ejercicio 3

```
hist3 = cv.calcHist([ig], [0], None, [4], [0,256]).flatten()/(M*N)

plt.bar(range(len(hist3)), hist3)
plt.show()
```

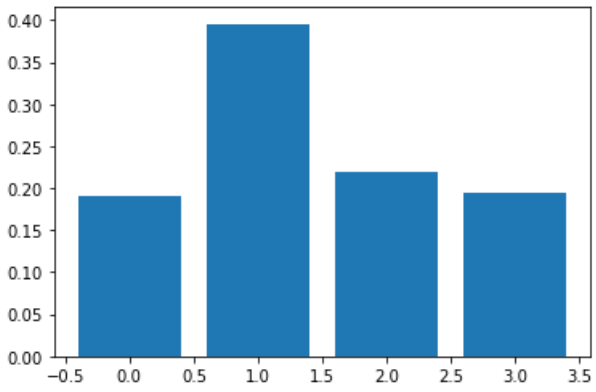

Figura 4: Histograma del Ejercicio 3.

- Ejercicio 4

```
hist4 = cv.calcHist([ig], [0], None, [5], [0,256]).flatten()/(M*N)

plt.bar(range(len(hist4)), hist4)
plt.show()
```

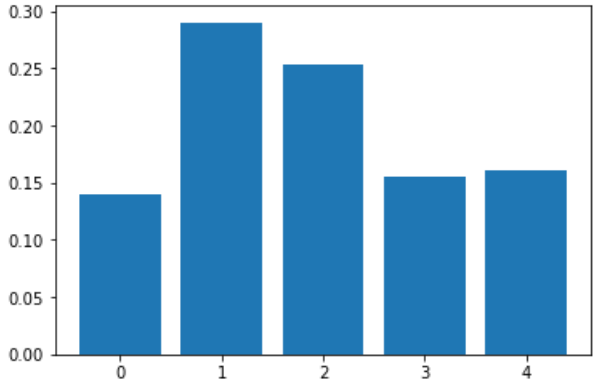

Figura 5: Histograma del Ejercicio 4.

- Ejercicio 5

```
hist5 = cv.calcHist([ig], [0], None, [10], [0,256]).flatten()/(M*N)

plt.bar(range(len(hist5)), hist5)
plt.show()
```

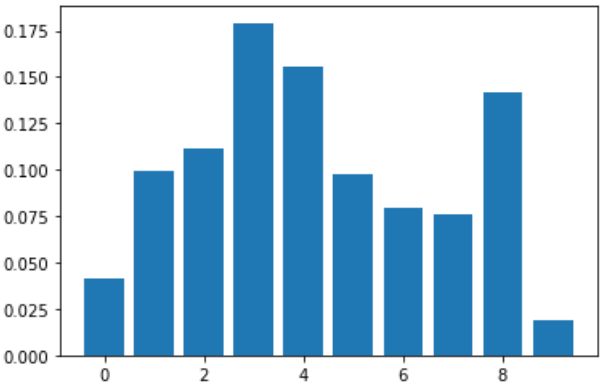

Figura 6: Histograma del Ejercicio 5.

- Ejercicio 6

```
hist6 = cv.calcHist([ig], [0], None, [100], [0,256]).flatten()/(M*N)

plt.bar(range(len(hist6)), hist6)
plt.show()
```

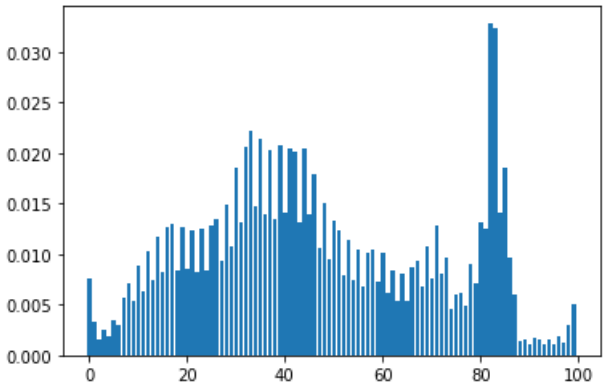

Figura 7: Histograma del Ejercicio 6.

Conclusión

En conclusión, el cálculo y análisis del histograma de una imagen son procesos esenciales en el campo de la visión robótica y el procesamiento de imágenes. A través del estudio detallado de estos conceptos, se ha demostrado su relevancia en la extracción de características y la comprensión de la distribución de intensidades en las imágenes. Asimismo, se ha destacado la importancia de técnicas de particionamiento del histograma, las cuales permiten una mayor comprensión y análisis de los datos. La implementación de estos conceptos en Python utilizando la biblioteca OpenCV proporciona herramientas poderosas para el desarrollo de aplicaciones en visión por computadora y robótica.