Tarea 4

Compensación de iluminación y mejora de contraste

Profesor: Claudio Pérez F.
Auxiliar: Diego Maureira
Ayudantes: Juan Pablo Pérez, Jhon Pilataxi, Jorge Zambrano

1. Objetivo

El objetivo de esta tarea es explorar distintas alternativas para la corrección de iluminación y mejora de contraste en imágenes digitales, desde métodos clásicos hasta un modelo moderno de *Deep Learning*. Para ello, se trabajará con un conjunto de imágenes que se encuentran afectadas singularmente, en donde se reconocerán las técnicas que funcionan de mejor manera en cada una de ellas y por qué. En este contexto, la tarea finaliza con una actividad en donde se muestra la utilidad de estas técnicas en aplicaciones de la vida real.

2. Descripción

Tanto la compensación de la iluminación como la mejora de contraste son temas clásicos abordados por el procesamiento digital de imágenes y en los que se busca mejorar la calidad visual de las imágenes. Para lograr este objetivo se han desarrollado varias técnicas manuales efectivas y, en los últimos años, se han hecho populares las técnicas basadas en *Deep Learning* que también resuelven este problema.

Esta tarea tiene como finalidad implementar, aplicar, analizar y comparar distintas técnicas de corrección de iluminación y mejora de contraste clásicas, tales como el estiramiento de contraste, la ecualización de histograma, ecualización de histograma adaptativa limitada por contraste (CLAHE) y, por otro lado, el modelo convolucional *Bread*.

3. Implementación de funciones generales útiles

Se recomienda implementar los siguientes métodos que serán de ayuda para las siguientes secciones.

- Implemente un método llamado *plot_histogram_rgb*, el cual debe recibir como parámetro de entrada solamente una imagen. El resultado será el gráfico del histograma de cada uno de los canales de la entrada.

- Desarrolle un método llamado *saturated_histogram* que reciba como entrada una imagen en forma de array. Este método deberá saturar los valores mayores a 255 y menores a 0 en 255 y 0, respectivamente (este método se usará solo una vez, en el estiramiento de contraste).

4. Mejora usando Modelos Clásicos

En esta sección se pide implemenar los métodos clásicos de estiramiento de contraste, ecualización de histograma y CLAHE.

1. Implemente una función llamada *contrast_extend* para realizar el estiramiento de contraste en una imagen. Los parámetros de entrada deberán ser: la imagen a procesar, el canal objetivo y dos valores a y b, que corresponden a los límites del rango seleccionado del histograma. Estos valores deben ser seleccionados de acuerdo con el histograma de la imagen, y deberán ser transformados linealmente al intervalo [0 - 255]. Un ejemplo de selección de valores a y b se muestra en la Figura 1.

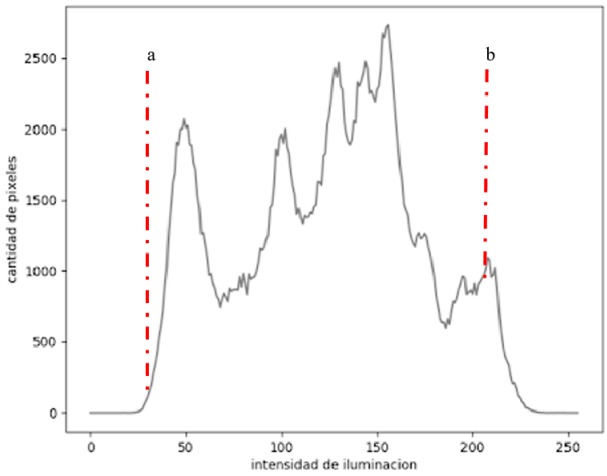

Figura 1: Cantidad de pixeles por valor de intensidad.

2. Implemente una función llamada *equal_hist*, que se encargará de hacer la ecualización del histograma de una imagen de entrada. Se recomienda hacer la ecualización de cada canal por separado y al final reconstruir la imagen en el orden original de canales.

   En este caso, para ecualizar el histograma, se ocupa una función llamada $look-up\ \ table$ cuyo resultado de obtiene al normalizar el histograma acumulado, como se indica en la siguiente ecuación:

$$O_{i,j} = floor\{(L-1) \times H'(I_{i,j})\} \tag{1}$$

   Donde I es la imagen de entrada, O es la imagen ecualizada, H' es el histograma acumulado y L es la cantidad de niveles de intensidad usados (256 en este caso). Los subíndices i, j son la posición de cada pixel a evaluar.

Hint: Puede calcular el histograma acumulado de la imagen con:

```
Hpr = np.cumsum(cv2.calcHist([img_one_ch], [0], None, [256], [0, 256]))

```

Donde *img_one_ch* es el canal de la imagen que está siendo procesado.

3. Defina una función llamada *clahe_enhancement*, que debe recibir una imagen y entregar una versión de la misma pero corregida por CLAHE. En este caso, se ecualizará el tercer canal del espacio de imágenes HSV. Para lograr el objetivo de esta pregunta se sugiere seguir los siguientes pasos:

   - Transforme la imagen RGB al dominio HSV.
   - Haga la ecualización adaptativa del tercer canal de la imagen HSV (el canal Value).
   - Reasigne el canal ecualizado a la imagen original del espacio HSV.
   - Regrese la imagen al dominio RGB.

   Genere una versión alternativa de la función *clahe_enhancement* (podría generarse con un parámetro adicional de entrada), en la cual no se transforme el espacio RGB y se ejecute la ecualización adaptativa en cada uno de los canales, para posteriormente reconstruir la imagen.

   Hint: Para hacer la ecualización adaptativa del histograma utilice la función cv2.createCLAHE de OpenCV (ver documentación) con los siguientes parámetros:

```
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))

```

5. Mejora usando *Bread*

En esta sección se busca implementar un flujo de trabajo que permita ejecutar la corrección de iluminación usando un modelo de *Deep Learning*, para lo cual se solicita solamente obtener la predicción del modelo y descargar los resultados.

- Investigue y comente acerca del funcionamiento de *Bread*.
- Genere un flujo de trabajo (más comúnmente llamado *pipeline*) que le permita generar la corrección de iluminación de las imágenes utilizando *Bread*. En el repositorio oficial podrá encontrar el link a un Jupyter Notebook en Colab que le permitirá cumplir este punto de forma simple. No se exige cambiar los valores de los parámetros *gamma* y *strength*, pero podría hacerlo si lo considera conveniente.

6. Evaluación y comparación de métodos implementados

Para poder evaluar y comparar los métodos implementados, se utilizarán tanto métricas cuantitativas como la inspección visual de los resultados. En específico, el primer tipo de evaluación mencionado se llevará a cabo mediante 2 métricas que deberá implementar y utilizar: MSE y SSIM.

Para completar el objetivo de esta sección cumpla los siguientes puntos:

1. Comente sobre las diferencias entre las dos métricas que implementará y utilizará ¿Qué mide cada una de ellas?, ¿En qué rango están los valores de salida?
2. Aplique los métodos de mejora digital que implementó, en cada una de las imágenes de la subcarpeta *low* de la carpeta *sub_eval15*, la cual corresponde a un subconjunto de las imágenes de evaluación de la base de imágenes LOL. En este contexto, las imágenes contenidas en la subcarpeta *high* corresponden al *ground truth* de este problema.
3. Genere una tabla para cada imagen procesada. En cada tabla deberá disponer el valor de las 2 métricas para todos los métodos de mejora digital utilizado, comparando el resultado de la mejora con el *ground truth* del problema.
4. Muestre en 4 casos el mejor y peor resultado obtenido en base a la métrica de su preferencia.
5. Comente acerca de los resultados cuantitativos obtenidos ¿Tiene relación lo que observa visualmente con el valor obtenido de las métricas?
6. Concluya sobre ventajas y desventajas de cada método de mejora de imágenes utilizado ¿Existe algún método superior en todos los escenarios probados?

7. Aplicación real

Las técnicas recién implementadas son muy útiles en aplicaciones de procesamiento de imágenes en el mundo real, por ejemplo, en la mejora de la detección o segmentación de imágenes. En la carpeta *aplicacion* existe una imagen de muy baja luminosidad, donde - con cierto esfuerzo - se pueden ver algunas personas. Supongamos que la tarea a resolver es detectar automáticamente el rostro de la persona que está en el plano más cercano. Para esto, utilizará una librería ampliamente conocida de python, llamada DeepFace y deberá seguir los siguientes pasos:

1. Ejecute la detección de rostros de *DeepFace* sobre la imagen *1.png*. El respositorio compartido está muy bien documentado, pero si necesita ayuda, cree un Jupyter Notebook para asistir el proceso.
2. ¿Pudo el algoritmo utilizado generar la detección del rostro de la persona en esta imagen?

3. Aplique *Bread* sobre la imagen de baja luminosidad, generando una nueva imagen que deberá guardar con el nombre *1_bread.png*.

4. Ejecute la detección de rostros, esta vez sobre la imagen mejorada ¿Qué sucede ahora?

5. Concluya sobre la utilidad de los métodos aprendidos.

8. Entregables

- Presente un reporte del trabajo. Incluya en el informe las respuestas a todas las interrogantes introducidas en el enunciado.
- El código también debe ser entregado, asegúrese de que pueda ser ejecutado por los revisores para comprobar los algoritmos.

9. Recordatorio

- El reporte es individual.

La fecha de entrega de trabajo será el día 12/11/2024 a las 23:59 hrs por medio de U-cursos.