EL7007-1 - Primavera 2021
Introducción al Procesamiento Digital de Imágenes

TAREA 2

Profesor: Claudio Pérez F.
Auxiliar: Jorge Zambrano I.
Ayudante: Juan Pablo Pérez C.

COMPENSACIÓN DE ILUMINACION Y MEJORAMIENTO DE CONTRASTE

1. Objetivo:

El objetivo de esta tarea es explorar distintas alternativas para la corrección de iluminación y mejora de contraste, desde métodos clásicos hasta dos de los modelos más modernos de redes convolucionales. Para ello, se trabajará con un conjunto de imágenes que se encuentran afectadas singularmente, en donde se reconocerá la técnica que funciona de mejor manera en cada una de ellas, y por qué.

2. Descripción:

La compensación de la iluminación y el mejoramiento de contrate, son temas clásicos abordados por el procesamiento de digital imágenes, en el que se busca de manera digital mejorar la calidad visual de la imagen. Se han desarrollado varias técnicas efectivas para lograr este objetivo. En los últimos años, han aparecido técnicas basadas en Deep Learning que también resuelven este problema.

Esta tarea tiene como finalidad implementar distintas técnicas de corrección de iluminación clásicas tales como el estiramiento de contraste, la ecualización de histograma, ecualizaciones de histograma adaptativo (CLAHE) y también utilizar dos modelos convolucionales, RetinexNet y MIRNet.

Se recomienda hacer esta tarea en Python utilizando el entorno de Google Colab por facilidad, con la ayuda de OpenCV.

3. Implementación:

Corrección de Iluminación por Modelos Convolucionales:

En esta sección se hará la corrección de iluminación usando modelos Deep Learning de las imágenes entregadas, para lo cual se solicita solamente hacer la predicción del modelo y descargar los resultados.

3.1. Ejecute el mejoramiento de contraste de las imágenes utilizando la red llamada RetinexNet, publicada en el siguiente repositorio: weichen582/RetinexNet: A Tensorflow implementation of RetinexNet (github.com) [1].

HINT: Para una rápida ejecución puede utilizar las siguientes líneas en el entorno de Google Colab:

```
!git clone https://github.com/weichen582/RetinexNet.git
!pip install tensorflow==1.15
%cd "/content/RetinexNet/"
# Antes de seguir, almacene las imágenes en el directorio:
/content/RetinexNet/data/test/low. Ejecute la línea siguiente en una
nueva celda.
!python main.py --phase=test
```

En este punto, los resultados se almacenarán en el directorio mostrado, usted deberá recuperar las imágenes y cargarlas posteriormente para hacer el análisis.
`'/content/RetinexNet/test_results'`

3.2. Utilizando el modelo MIRNet, ejecute la corrección de las imágenes, y al igual que en el punto anterior, almacénelas para analizarlas con el resto de las imágenes. Las instrucciones están en el repositorio [2].

https://github.com/soumik12345/MIRNet

HINT: Utilice este cuaderno de Jupyter Notebook como referencia:
https://github.com/bhattbhavesh91/low-light-image-enhancement-tutorial

Implementación de funciones necesarias:

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

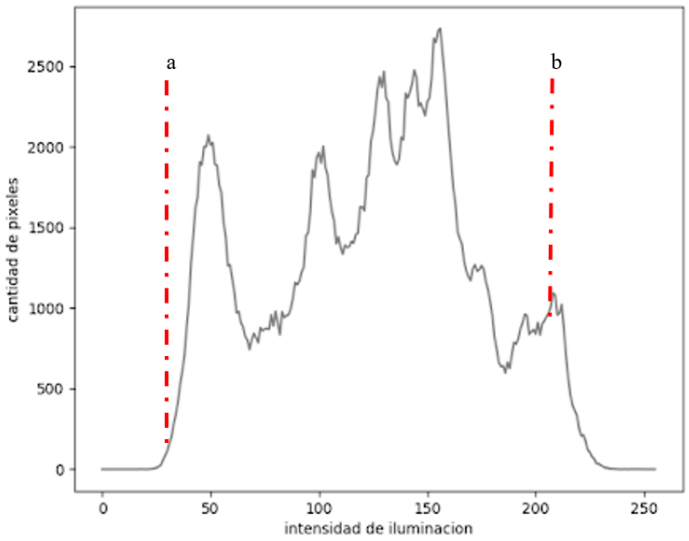
3.3. Implemente un método llamado plot_histogram_rgb, el cual recibirá como parámetro de entrada solamente una imagen. El resultado será el gráfico del histograma de cada uno de los canales.

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

Corrección de Iluminación por Modelos Clásicos:

Se pide desarrollar los métodos clásicos de estiramiento de contraste, ecualización de histograma, y ecualizaciones adaptativas CLAHE.

3.5. Implemente un método llamado contrast_extend para realizar el estiramiento de contraste. Los parámetros de entrada deberán ser: la imagen, el canal, a y b, siendo a y b los límites del rango seleccionado del histograma. Estos valores son seleccionados de acuerdo con el histograma de la imagen, y deberán ser transformados linealmente al intervalo [0 - 255] de acuerdo con la siguiente figura:



3.6. Implemente un método llamado equal_hist que se encargará de hacer la ecualización del histograma. Se recomienda hacer la ecualización de cada canal por separado y al final recuperar la imagen en BGR (Hint al final del documento).

Para ecualizar el histograma, se ocupa una función llamada *look-up table* la cual se consigue al normalizar el histograma acumulado como se indica en la ecuación:

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

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, para este caso 255. Los subíndices *i, j* son la posición de cada pixel a evaluar.

HINT: Calcule el histograma acumulado de la imagen con la línea de código entregada a continuación. (img_1ch es el nombre del canal respectivo de la imagen)

```
Hpr = np.cumsum(cv2.calcHist([img_1ch], [0], None, [256], [0, 256]))
```

3.7. Desarrolle un método llamado clahe_enhanced, el cual recibirá una imagen de entrada, a la cual se le aplicará una corrección por CLAHE. Para este caso, se ecualizará el tercer canal del espacio de imágenes HSV, para lo cual siga los siguientes pasos:

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

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

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

Métricas:

Para evaluar los resultados cuantitativamente, se considerarán 3 métricas: Colorfulness, Saturación de Pixeles y el Indicador e.

- Colorfulness $\hat{M}^{(3)}$. Esta métrica evalúa como evoluciona el color de una imagen por medio de un cálculo estadístico de los colores oponentes [3]. Valores más altos representan mejor desempeño.
  Esta métrica se define como:

$$\hat{M}^{(3)} = \sigma_{rgyb} + 0.3\,\mu_{rgyb}$$

  Para lo cual:

$$\sigma_{rgyb} = \sqrt{\sigma_{rg}^2 + \sigma_{yb}^2}$$

$$\mu_{rgyb} = \sqrt{\mu_{rg}^2 + \mu_{yb}^2}$$

  Donde σ y μ son la desviación estándar y el promedio de los canales rg y yb, siendo estos:

$$rg = |R - G|$$

$$yb = \left|\frac{1}{2}(R+G) - B\right|$$

  Donde B, G y R son los canales Azul, Verde y Rojo respectivamente.

- Saturación de Pixeles $\sum$. Esta métrica representa el porcentaje de pixeles que fueron saturados en el procesamiento. Se define como:

$$\sum = \frac{n_s}{m \times n} \times 100\%$$

  Siendo n_s la cantidad de pixeles que fueron saturados como blanco o negro después del procesamiento, y m y n las dimensiones de la imagen. Para obtener n_s, reste la cantidad de pixeles negros y blancos absolutos de la imagen procesada menos el de la original (haga esto por cada canal y promedie los resultados). Un valor bajo indica una mejor habilidad de preservación del rango de valores de los pixeles [4].

- Indicador e. Está métrica evalúa la capacidad de restauración de los bordes de la imagen. No hace falta implementarla, utilice la función entregada por equipo docente llamada `e_indicator`. Valores altos reflejan mejores resultados.

Resultados:

Evalúe las métricas correspondientes sobre las imágenes entregadas utilizando las siguientes técnicas:

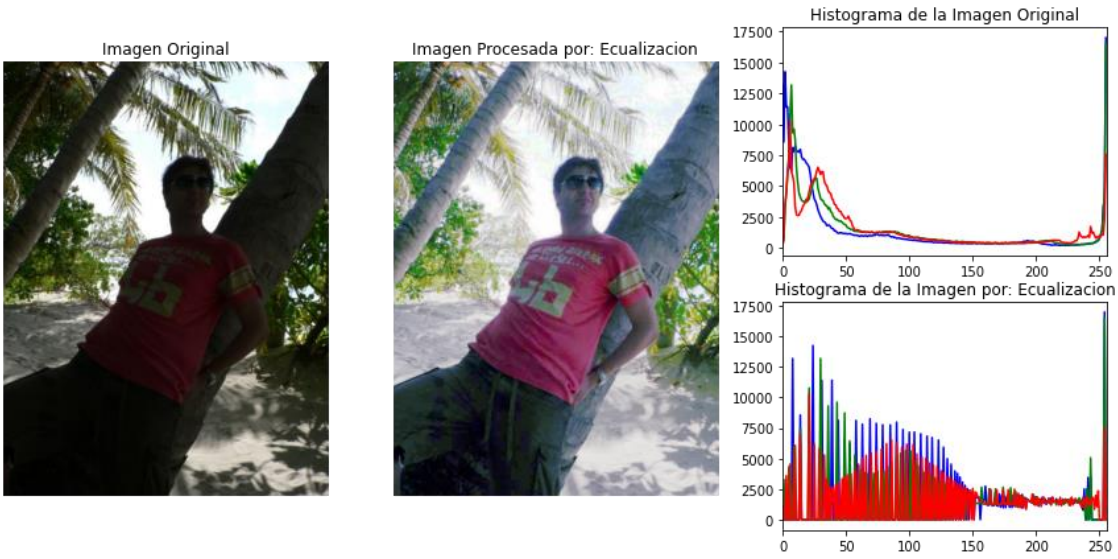
1. Estiramiento de contraste. Para ello deberá desplegar el histograma de los tres canales usando el método plot_histogram_rgb y seleccionar en cada imagen los valores de a y b por cada canal (pueden ser los mismos si así lo considera). Ecualice los tres canales y forme nuevamente una imagen BGR (sección 3.5).
2. Ecualización de histograma (sección 3.6).
3. Ecualización adaptativa HSV (sección 3.7).
4. Ecualización adaptativa BGR. Para ello, debe hacer la ecualización adaptativa en cada uno de los canales por separado, y finalmente formar la imagen BGR.
5. Aplicación de RetinexNet (sección 3.1).
6. Aplicación de MIRNet (sección 3.2).

Analice los resultados **cuantitativa y cualitativamente**, y concluya que métodos tienen mejores resultados para cada imagen. Para ello complete la siguiente tabla por imagen.

Imagen x	Estiramiento de Contraste	Ecualización Histograma	CLAHE HSV	CLAHE BGR	Retinex Net	MIR Net
$\widehat{M}^{(3)}$						
$\sum$						
e						

Muestre solamente el mejor y peor resultado haciendo uso del método entregado llamado show_img_hist. Además, recuerde analizar el histograma original vs el procesado.

```
show_img_hist(img, salida, 'Ecualizacion', Flag_vertical=True)
```



4. **Entregables.**
   a) Presente un reporte individual del trabajo realizado (solo considere lo que se solicita).
   b) El código debe ser entregado en formato **.ipynb** junto con las imágenes resultantes.

Nota:
Trabaje directamente con imágenes en el dominio BGR (Blue, Green and Red) por facilidad, dado a que muchas de las funciones de OpenCV toman esta norma.
Para leer una imagen a color utilice:

```
Img_bgr = cv2.imread('path de la imagen')
```

Utilice las siguientes funciones para separar por canales y unir canales en OpenCV:

```
ch_B, ch_G, ch_R = cv2.split(img_bgr)
img_bgr = cv2.merge((ch_b, ch_g, ch_r))
```

Fecha de Entrega: 5 de Noviembre 2021 (18:00 hrs).

Bibliografía:

[1] C. Wei, W. Wang, W. Yang, and J. Liu, "Deep retinex decomposition for low-light enhancement," *Br. Mach. Vis. Conf. 2018, BMVC 2018*, no. 61772043, 2019.

[2] S. W. Zamir *et al.*, "Learning Enriched Features for Real Image Restoration and Enhancement," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12370 LNCS, pp. 492–511, 2020, doi: 10.1007/978-3-030-58595-2_30.

[3] D. Hasler and S. E. Suesstrunk, "Measuring colorfulness in natural images," *Hum. Vis. Electron. Imaging VIII*, vol. 5007, no. June 2003, p. 87, 2003, doi: 10.1117/12.477378.

[4] Q. C. Tian and L. D. Cohen, "A variational-based fusion model for non-uniform illumination image enhancement via contrast optimization and color correction," *Signal Processing*, vol. 153, pp. 210–220, 2018, doi: 10.1016/j.sigpro.2018.07.022.