EL7007-1 - Introducción al Procesamiento Digital de Imágenes

TAREA 2

Profesor de Cátedra: Claudio Pérez F.
Profesor Auxiliar: Jorge Zambrano I.
Semestre: Otoño 2022

COMPENSACIÓN DE ILUMINACIÓN Y DETECCIÓN DE PERSONAS

1. **Objetivo:**

   El objetivo de esta tarea es explorar distintas alternativas para la corrección de iluminación, desde los métodos clásicos hasta el uso de redes convolucionales. Para ello se trabajará con algunas imágenes del **conjunto de prueba** de la base de datos DARK FACE [1], sobre las cuales, una vez preprocesadas, se procederá a utilizar un método de detección de personas. Se podrá así evaluar la importancia de la etapa de preprocesamiento.

2. **Descripción:**

   La compensación de iluminación y el mejoramiento de contraste son temas clásicos abordados por el procesamiento de digital imágenes, que permiten mejorar el resultado de otros algoritmos de detección o clasificación. Se han desarrollado varias técnicas para lograr este objetivo, incluyendo técnicas basadas en Deep Learning.

   Esta tarea tiene como finalidad implementar diferentes técnicas de mejora de contraste clásicas tales como, Estiramiento Lineal de Contraste, Ecualización de Histograma, Contrast Limited Adaptive Histogram Equalization (CLAHE), y un modelo convolucional llamado RetinexNet [2]. Una vez procesadas las imágenes, se pide aplicar un algoritmo de detección de personas basado en YOLO V5 [3], para evaluar la calidad de las imágenes obtenidas. Recomendamos realizar esta tarea en Python con librerías como OpenCV, Skimage, y PIL, entre otras.

3. **Implementación de Técnicas de Mejora de Contraste:**

   3.A Mejora de Contraste por modelos clásicos:
   Se pide desarrollar los métodos clásicos: Estiramiento Lineal de Contraste, Ecualización de histograma, y Ecualizaciones adaptativa CLAHE HSV.

   3.1. Desarrolle un método para la mejora de contraste por **Estiramiento Lineal**. Descomponga la imagen RGB en tres canales, y aplique el estiramiento por separado. Para decidir el rango de estiramiento, construya los histogramas de las imágenes a color. Nota: Cada imagen puede tener un rango diferente si así lo considera necesario, o puede establecerse un solo rango para todas.

3.2. Implemente un método 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 en la sección Notas).
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 256, ya que las imágenes están codificadas en 8 bits.

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)

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

3.3. Desarrolle un método, el cual recibirá una imagen de entrada, a la cual se le aplicará una corrección por **CLAHE HSV**. 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 de solamente el 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 directa de OpenCV *cv2.createCLAHE* (ver documentación). Puede usar los parámetros mostrados o si prefiere, elegir los propios con el fin de mejorar los resultados.

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

3.B Corrección de contraste utilizando una CNN:

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

3.4. Ejecute el mejoramiento de contraste de las imágenes originales utilizando la red llamada *RetinexNet*, publicada en el siguiente repositorio: weichen582/RetinexNet: A Tensorflow implementation of RetinexNet (github.com) [2].

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

```
!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 a continuación, usted deberá recuperar las imágenes y cargarlas posteriormente para hacer la detección con YOLO.

```
'/content/RetinexNet/test_results'
```

4. Detección de Personas usando YOLO

Para la detección de personas se utilizará YOLO V5. Puede guiarse en la implementación de Google Colaboratory de este repositorio: YOLOv5 | PyTorch.

HINT: Los objetos detectados se almacenan en variables tipo Pandas. Para iterarlos puede basarse en el siguiente código:

```
import torch
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)
results = model(img)
P = results.pandas().xyxy[0]
for i in P.index:
  print(P["name"][i])
```

Debe detectar solamente la clase persona, cuya confidencia sea mayor a 0.5. Además, debe **dibujar bounding boxes** sobre la imagen e **imprimir la confidencia** de cada detección. A continuación, se muestran unos ejemplos.

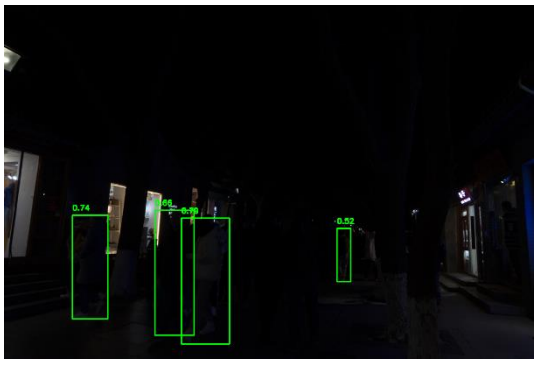

Imagen Original: 4 Personas

Estiramiento de Contraste: 10 Personas

Ecualización: 9 Personas

CLAHE HSV: 9 Personas

Figura 1: Ejemplos de las detecciones requeridas.

5. **Notas:**
   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)
new_img_bgr = cv2.merge((ch_B, ch_G, ch_R))
```

6. **Entregables:**
   - **a)** Presente un reporte individual del trabajo en PDF.
   - **b)** Dentro del reporte, para una imagen de ejemplo (a su elección), muestre el histograma original y los histogramas obtenidos por cada método. Comente los resultados.
   - **c)** También, presente una tabla que muestre el número de personas detectadas para cada imagen y para cada método. Así mismo, analice y muestre algunos resultados obtenido.

Tabla 1: Número de detecciones de personas por cada método

	Img1: 0.png	**Img2: 1.png**	**Img3: 2.png**	**Img4: 10.png**	**Img5: 17.png**	**Img6: 18.png**	**Img7: 31.png**	**Img8: 94.png**	**Img9: 96.png**
Imagen Original									
Estiramiento Lineal									
Ecualización de Histograma									
CLAHE HSV									
CNN RetinexNet									

   - **d)** El código debe entregarse junto con las detecciones sobre las imágenes originales y procesadas por cada uno de los métodos. Entregue todas las imágenes, pero en el reporte muestre solamente algunos ejemplos.

7. Referencias:

[1] W. Yang, Y. Yuan, W. Ren, J. Liu, W. J. Scheirer, Z. Wang, Zhang, and et al., "Advancing image understanding in poor visibility environments: A collective benchmark study," IEEE Transactions on Image Processing, vol. 29, pp. 5737–5752, 2020.

[2] 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.

[3] P. Jiang, D. Ergu, F. Liu, Y. Cai, en B. Ma, "A Review of Yolo Algorithm Developments", Procedia Computer Science, vol 199, bll 1066–1073, 2022.

La fecha de entrega de la tarea será el día 28/04/2022 a las 18:00 hrs por medio de U-cursos.