*

Tarea 2:
Compensación de iluminación y mejoramiento de contraste

Fecha de entrega: Lunes 15 de noviembre, 18:00 hrs.

Estudiante:	Francisco Molina L.
Profesor:	Claudio A. P.
Auxiliar:	Jorge Zambrano I.
Ayudantes:	Eitan Hasson A.
	Juan Pérez C.
Semestre:	Primavera 2021

Índice

1. Introducción

En el presente informe se desarrollan las actividades correspondientes a la Tarea 2 del curso EL7007-1 del semestre de primavera de 2021. Este informe comprende la implementación computacional de algoritmos de compensación de iluminación y mejoramiento del contraste, basados en métodos clásicos como extensión del contraste y ecualización de histograma y en métodos modernos basados en redes convolucionales como Retinex y MIR.

Los principales objetivos de este informe corresponden a: lograr implementar mediante programación la implementación de los métodos clásicos y el cálculo de ciertas métricas y, principalmente, se desea analiza, cualitativamente y cuantitativamente en base a las mencionadas métricas, el desempeño de los distintos métodos sobre las imágenes entregadas por el cuerpo docente.

En las **Secciones 2, 3 y 4** se detallan las funciones implementadas en Python, adjuntas en los notebook Tarea2.ipynb, Tarea2-RetinexNet.ipynb y Tarea2-MIRNet.ipynb, necesarias para el procesamiento de las imágenes. En la **Sección 5** se presentan los resultados obtenidos de la implementación en Python, presentando individualmente todos los resultados obtenidos, en la **Sección 6** se realiza un análisis de dichos resultados centrando el análisis en torno a cuál método funciona mejor para cada imagen, y en la **Sección 7** se presentan las conclusiones del informe.

2. Implementación de funciones necesarias

En esta sección se explican la programación detrás de las funciones que el enunciado clasifica como necesarias. Estas corresponden a `plot_histogramm_rgb` que muestra el histograma de los tres canales de una imagen BGR y `histogram_saturated` que satura los valores menores a 0 y mayores a 255 de un array.

2.1. Histograma de los canales

La función encargada de esta tarea es `plot_histogramm_rgb(img, title, BGR)`, que recibe como entradas una imagen `img`, el nombre del archivo que contiene la imagen `title` y un bool que determina si la imagen es BGR o RGB `BGR`.

En primer lugar, si la imagen de entrada es BGR, esta se transforma a RGB:

```
if BGR:
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

Luego se separan los canales de la imagen mediante la función `split` de `cv2`:

```
ch_B, ch_G, ch_R = cv2.split(img)
```

Y luego basta con crear una figura y añadir los histogramas, lo que produce una imagen como la que se muestra en la **Figura 1**:

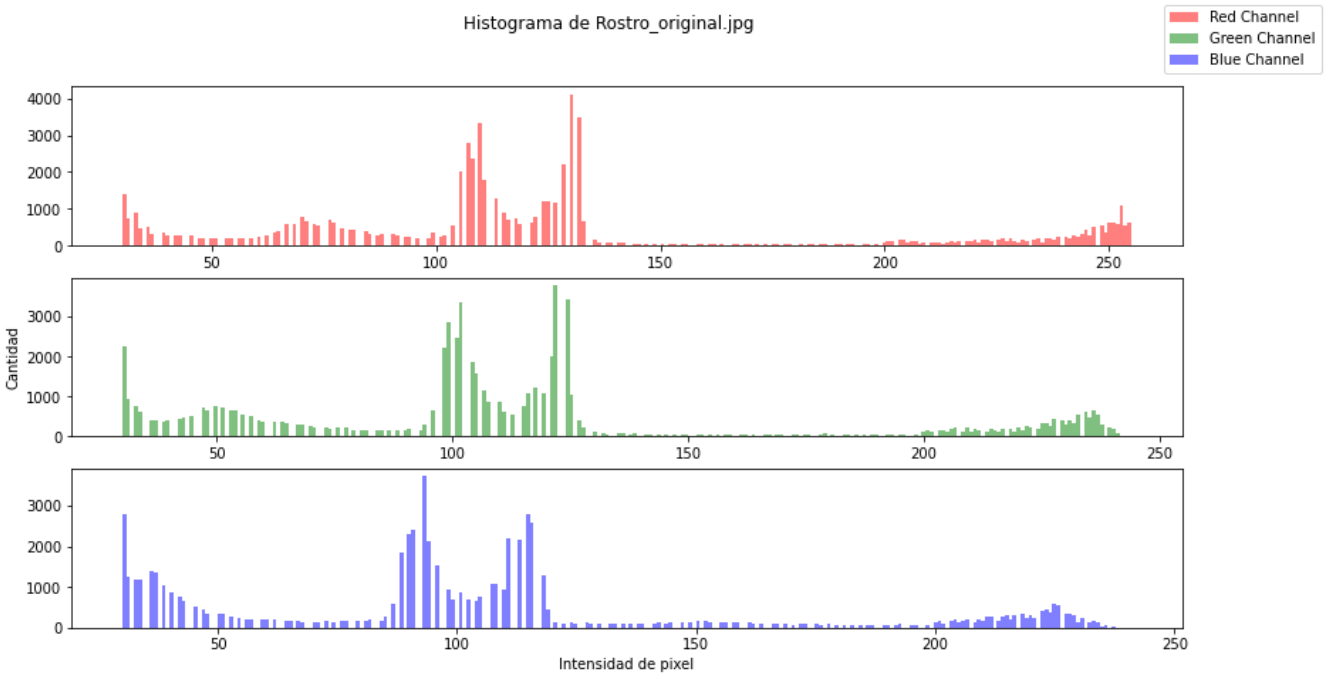

Figura 1: Ejemplo de histograma de los canales R, G y B

2.2. Saturación de histograma

La función encargada de esto es `histogram_saturated(img)` que simplemente recibe una imagen de entrada `img`. Esta función es bastante simple, simplemente utiliza sintaxis de numpy para remplazar todos los valores de la imagen mayores a 255 por 255 y todos los valores menores a 0 por 0.

```
1 def histogram_saturated(img):
2   out = np.copy(img)
3   out[out > 255] = 255
4   out[out < 0] = 0
5   return out
```

3. Corrección de iluminación por métodos clásicos

En esta sección se explica la programación detrás de las funciones correspondientes a los métodos clásicos de corrección de iluminación.

3.1. Extensión de contraste

Este método consiste en básicamente realizar una transformación lineal de la intensidad de los pixeles de una imagen, ponderando distintos tramos por distintos coeficientes como se muestra en la **Figura 2**:

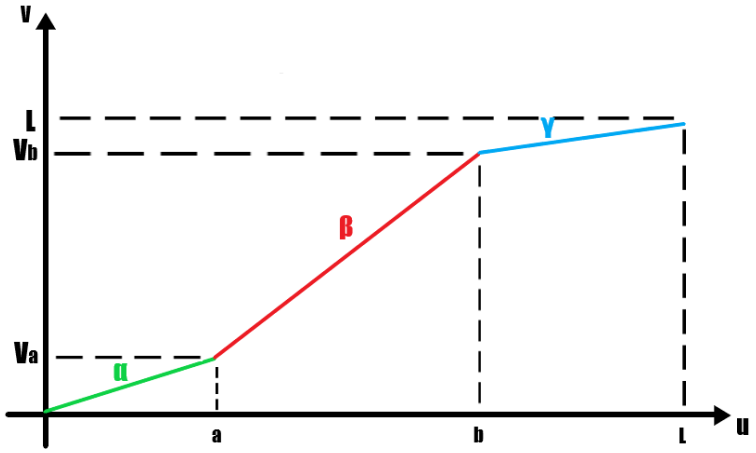

Figura 2: Método de extensión de contraste

Este gráfico quiere decir que para los pixeles con intensidades entre 0 y a se realiza una transformación que depende de α, para los pixeles entre a y b se realiza una transformación que depende de β y para los pixeles entre b y L, que en este caso corresponde a 255, se realiza una transformación que depende de γ, como se muestra en la **Ecuación 1**:

$$v = \begin{cases} \alpha \cdot u & 0 \leq u \leq a \\ \beta \cdot (u-a) + v_a & a < u < b \\ \gamma \cdot (u-b) + v_b & b \leq u \leq L \end{cases} \tag{1}$$

Esta transformación es muy rápida, puesto que se pueden pre-computar sus coeficientes y la transformación entonces consiste en simplemente leer en memoria el valor que le corresponde a cada intensidad de pixel. La función encargada de esta tarea es `contrast_extend(img, a, b)` que recibe como entrada una imagen BGR `img`, y los valores en los que se cortan las secciones `a` y `b`.

De manera arbitraria se selecciona que $v_a = 30$ y $v_b = 200$, por lo que las pendientes α, β y γ se calculan mediante la fórmula de la pendiente entre dos puntos:

$$\begin{aligned} \alpha &= \frac{v_a}{a} \\ \beta &= \frac{v_b - v_a}{b - a} \\ \gamma &= \frac{255 - v_b}{255 - b} \end{aligned} \tag{2}$$

En ciertos casos estas divisiones pueden anularse cuando el denominador resulta ser cero, por lo que en el código se implementa la función auxiliar `div(x,y, opt)` que realiza una división entre `x` e `y`, pero si el denominador es cero entonces retorna un valor grande arbitrario `opt`, de modo que en `contrast_extend` el cálculo de coeficientes es:

```
def contrast_extend(img, a, b):
    va = 30
    vb = 200
    alpha = div(va, a, 9999)
    beta = div(vb - va, b - a, 9999)
    gamma = div(255 - vb, 255 - b, 9999)
```

Para cada canal de la imagen se utiliza la sintaxis de numpy para encontrar y reemplazar los valores de cada tramo por su transformación correspondiente. Para mayor legibilidad se definen las condiciones de cada tramo en variables:

```
    newImg = []
    channels = cv2.split(img)
    for i in range(3): # para cada canal
        out = channels[i]

        cond_alpha = (0 <= out) & (out <= a)
        cond_beta =  (a < out) & (out <  b)
        cond_gamma = (b <= out) & (out <= 255)
```

Una vez identificados los tramos a modificar con las condiciones, sólo basta reemplazar cada sección por la transformación que le corresponde, según la **Ecuación 1** y combinar los canales resultantes usando la función `merge` de `cv2`:

```
        out[cond_alpha] = alpha*out[cond_alpha]
        out[cond_beta] = beta*(out[cond_beta] - a) + va
        out[cond_gamma] = gamma*(out[cond_gamma] - b) + vb

        newImg.append(histogram_saturated(out))
    out = cv2.merge((newImg[2], newImg[1], newImg[0]))
    return out
```

3.2. Ecualización de histograma

Este método consiste en simplemente reemplazar todos los pixeles de la imagen según una *look up table* dada por la **Ecuación 3**:

$$O_{i,j} = \lfloor (L-1) \cdot H(I_{i,j}) \rfloor \tag{3}$$

Donde $O_{i,j}$ corresponde a la imagen ecualizada en la posición [i,j], L corresponde a la cantidad de niveles de intensidad (en este caso 255), H corresponde al histograma acumulado normalizado de la imagen e $I_{i,j}$ corresponde a la imagen de entrada en la posición [i,j].

La función encargada de esta tarea es `equal_hist(img)` que sólo recibe la imagen de entrada `img`. Esta función implementa el método recién descrito, para cada canal primero calcula su histograma acumulado mediante la función `cumsum` de `numpy`:

```
def equal_hist(img):
    out = []
    for img_ch in cv2.split(img):
        H = np.cumsum(cv2.calcHist([img_ch], [0], None, [256], [0, 256]))/
            (img_ch.shape[0]*img_ch.shape[1])
```

Nótese que se divide el resultado de `cumsum` por la cantidad de pixeles de la imagen, con el fin de normalizarlo. Luego se aplica, pixel a pixel, la **Ecuación 3** y se juntan los canales en una sola imagen mediante `merge`:

```
        ch = np.copy(img_ch)
            for y in range(ch.shape[0]):
              for x in range(ch.shape[1]):
                ch[y,x] = np.floor((255 - 1)*H[img_ch[y,x]])
        out.append(ch)
    return cv2.merge((out[2],out[1],out[0])) # R G B
```

3.3. Ecualización adaptativa de histograma CLAHE

Este método se divide en dos categorías, donde en ambas se aplica la función `createCLAHE` de `OpenCV` para realizar la transformación, que en el código se realiza mediante:

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

3.3.1. HSV CLAHE

Consiste en transformar una imagen BGR a HSV, aplicar la transformación CLAHE sólo en el canal V y finalmente transformar la imagen a RGB, lo que en el código se traduce realiza mediante `clahe_hsv(img)` que recibe como entrada sólo la imagen `img`:

```
def clahe_hsv(img):
    out = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    out[:,:,2]= clahe.apply(out[:,:,2])
    return cv2.cvtColor(out, cv2.COLOR_HSV2RGB)
```

3.3.2. RGB CLAHE

Consiste en simplemente aplicar la transformación CLAHE a todos los canales de una imagen RGB, lo que en el código se realiza mediante `clahe_hsv(img)` que también sólo recibe como entrada la imagen `img`:

```
def clahe_hsv(img):
    out = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    out[:,:,2]= clahe.apply(out[:,:,2])
    return cv2.cvtColor(out, cv2.COLOR_HSV2RGB)
```

4. Métricas

Para evaluar los resultados de los métodos cualitativamente se utilizan tres métricas, *Colorfulness*, la saturación de pixeles y el indicador e, las cuales se detallan en la presente sección.

4.1. Coloración

Esta es una métrica que evalúa cómo evoluciona el color de una imagen mediante un cálculo estadístico de los colores oponentes, aquí **valores altos** representan mejor desempeño. El *Colorfulness* $\hat{M}^{(3)}$ está dado por:

$$\hat{M}^{(3)} = \sigma_{rgyb} + 0{,}3 \cdot \mu_{rgyb} \tag{1}$$

Donde:

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

Donde *sigma* y μ corresponden a la desviación estándar y el promedio de los canales rg e yb, donde:

$$rg = |R - G|$$
$$yb = |\frac{R+G}{2} - B|$$

La función encargada de implementar esta métrica es `colorfulness(img)` que recibe como entrada una imagen `img`. Esta función simplemente separa los canales de la imagen mediante `split` y aplica las fórmulas recién descritas para finalmente retornar el valor que arroja la **Ecuación 1**:

```
def colorfulness(img):
    B, G, R = cv2.split(img)
    rg = np.abs(R - G)
    yb = np.abs(0.5*(R + G) - B)

    sigma_rg = np.std(rg)
    mu_rg = np.mean(rg)

    sigma_yb = np.std(yb)
    mu_yb = np.mean(yb)

    sigma_rgyb = np.sqrt(sigma_rg**2 + sigma_yb**2)
    mu_rgyb = np.sqrt(mu_rg**2 + mu_yb**2)

    return sigma_rgyb + 0.3*mu_rgyb
```

4.2. Saturación de pixeles

Esta métrica representa el porcentaje de pixeles que fueron saturados en el procesamiento, donde **valores bajos** reflejan mejores resultados. La saturación Σ se define simplemente como:

$$\Sigma = \frac{n_s}{w \cdot h} \cdot 100\,\% \tag{2}$$

Donde n_s corresponde a la cantidad de pixeles que fueron saturados como blanco (255) o negro (0) después del procesamiento y w y h corresponden a las dimensiones de la imagen. Para obtener el valor correspondiente a n_s basta con restar la cantidad de pixeles saturados en la imagen original con la cantidad de pixeles saturados en la imagen resultante.

La función encargada de este procedimiento es `saturation(original, new)` que recibe como entradas la imagen original `original` y la imagen que resulta del procesamiento `new`. Para calcular n_s se define la función auxiliar `countSat(img)`, la cual calcula la cantidad de pixeles saturados en cada canal de la imagen y retorna el promedio entre esos tres valores:

```
def countSat(img):
    B, G, R = cv2.split(img)
    satB = len(B[(B == 0) | (B == 255)])
    satG = len(G[(G == 0) | (G == 255)])
    satR = len(R[(R == 0) | (R == 255)])

    return (satB + satG + satR)/3
```

Una vez obtenido n_s, basta con aplicar la función de la **Ecuación 2**:

```
def saturation(original, new):
  ns = countSat(original) - countSat(new)
  return (ns/(new.shape[0]*new.shape[1]))*100
```

4.3. Indicador e

Esta métrica evalúa la capacidad de restauración de los bordes de la imagen. No hace falta implementarla puesto que es entregada por el cuerpo docente en el archivo "Metodos.py". Aquí **valores altos** reflejan mejores resultados.

5. Resultados

A continuación se presentan los resultados de aplicar las distintas correcciones de iluminación y contraste a las imágenes entregadas por el cuerpo docente.

5.1. Estiramiento de contraste

El estiramiento de contraste (a = 0, b = 110) en la imagen del edificio transforma los histogramas de la **Figura 3** a los histogramas presentados en la **Figura 4**:

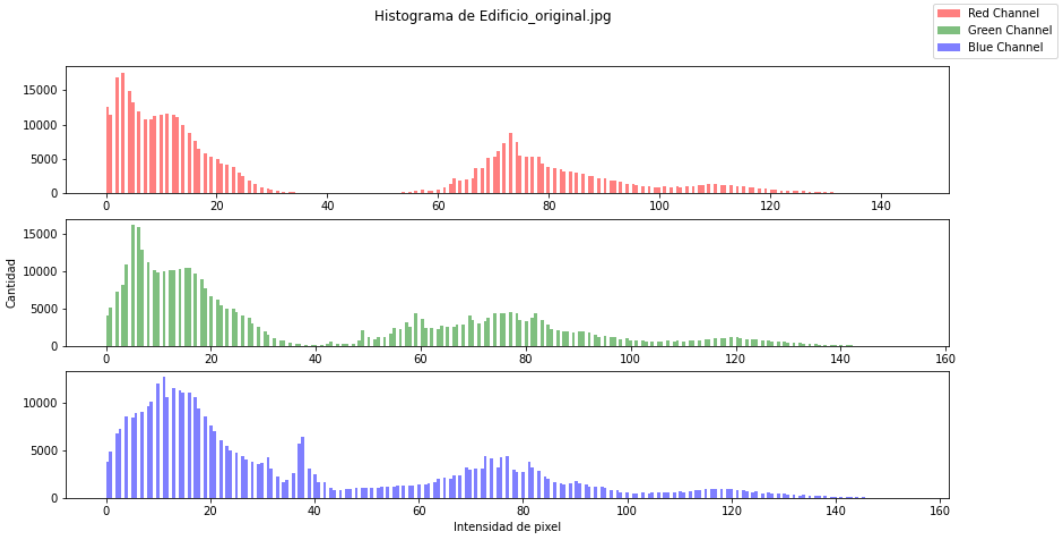

Figura 3: Histograma original edificio

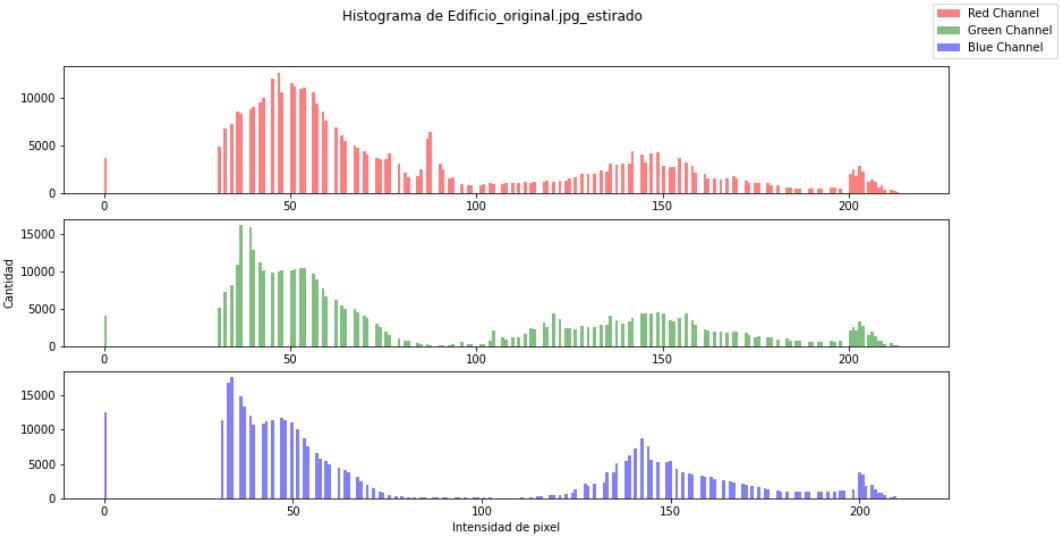

Figura 4: Histograma estirado edificio

El resultado de aplicar el estiramiento de contraste a la imagen del edificio se presenta en la **Figura 5**:

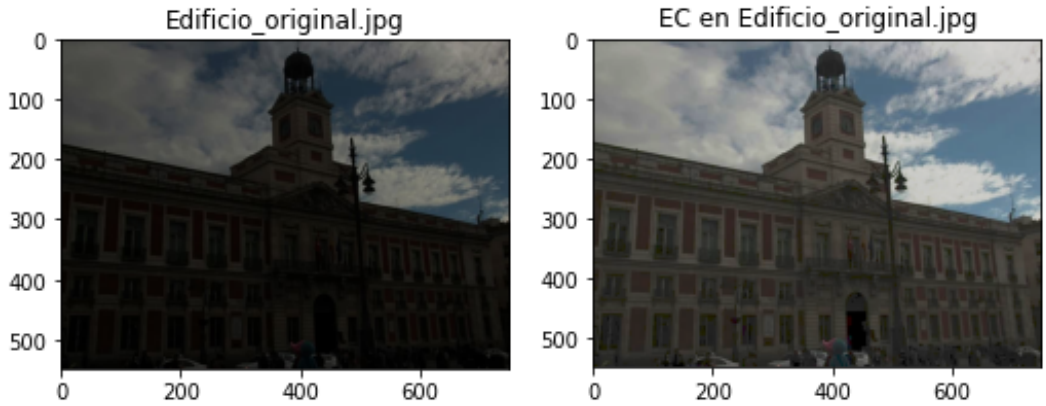

Figura 5: Edificio con estiramiento de contraste

El estiramiento de contraste (a = 0, b = 80) en la imagen del rostro transforma los histogramas de la **Figura 6** a los histogramas presentados en la **Figura 7**:

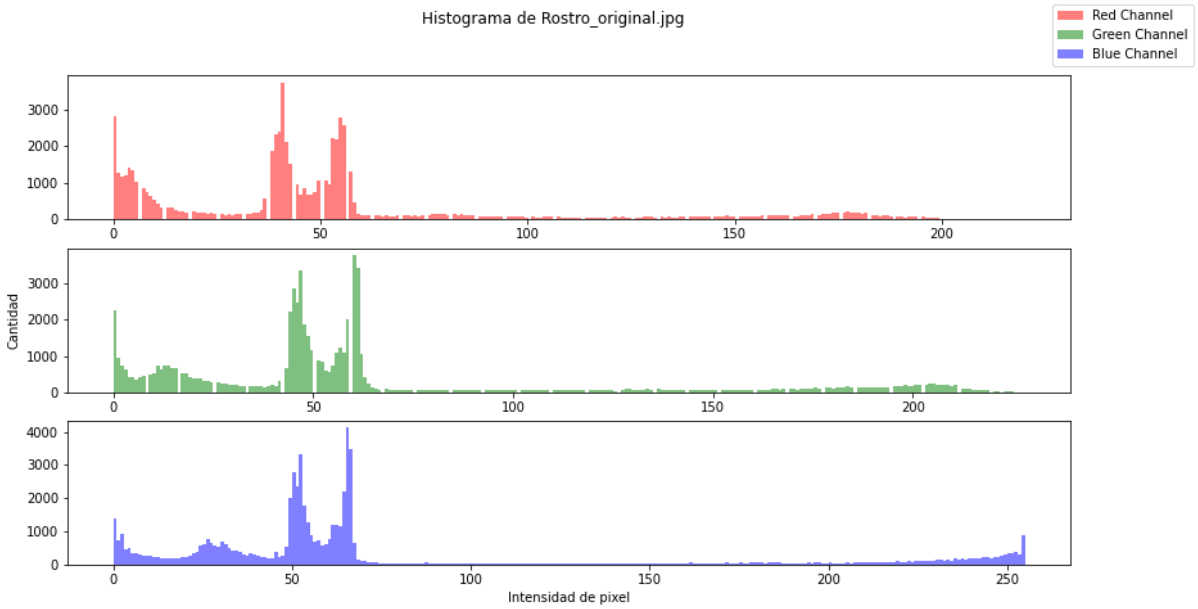

Figura 6: Histograma original rostro

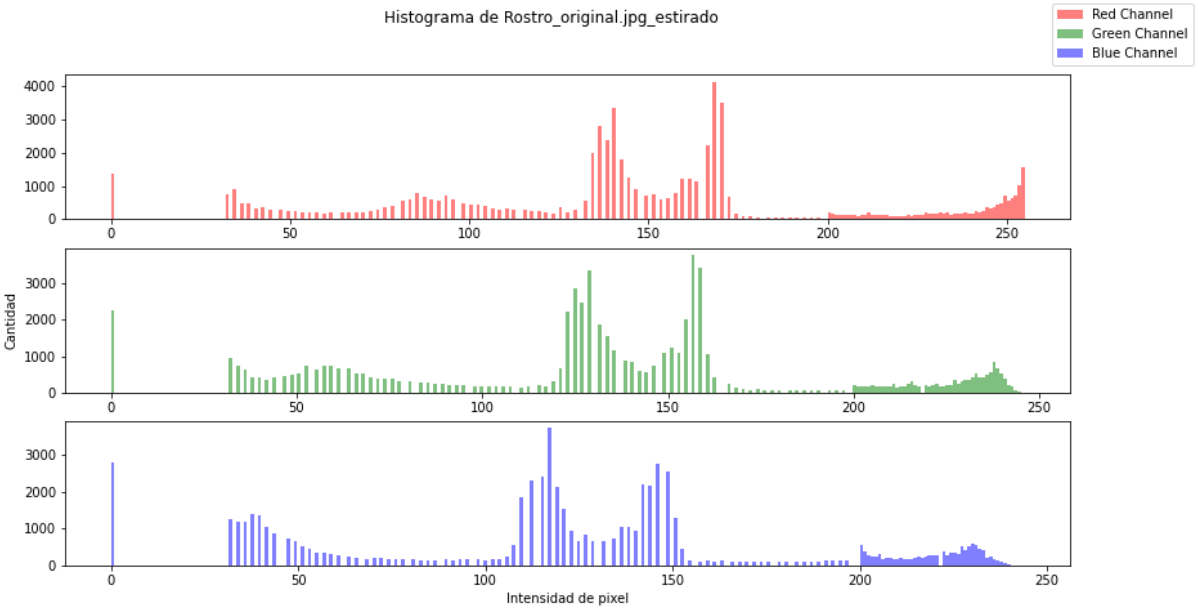

Figura 7: Histograma estirado rostro

El resultado de aplicar el estiramiento de contraste a la imagen del rostro se presenta en la **Figura 8**:

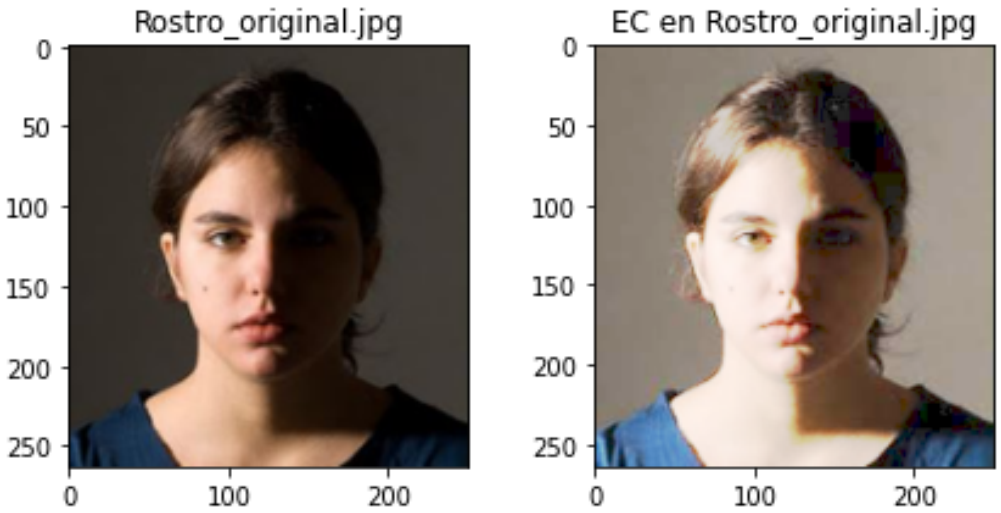

Figura 8: Rostro con estiramiento de contraste

El estiramiento de contraste (a = 0, b = 70) en la imagen de la playa transforma los histogramas de la **Figura 9** a los histogramas presentados en la **Figura 10**:

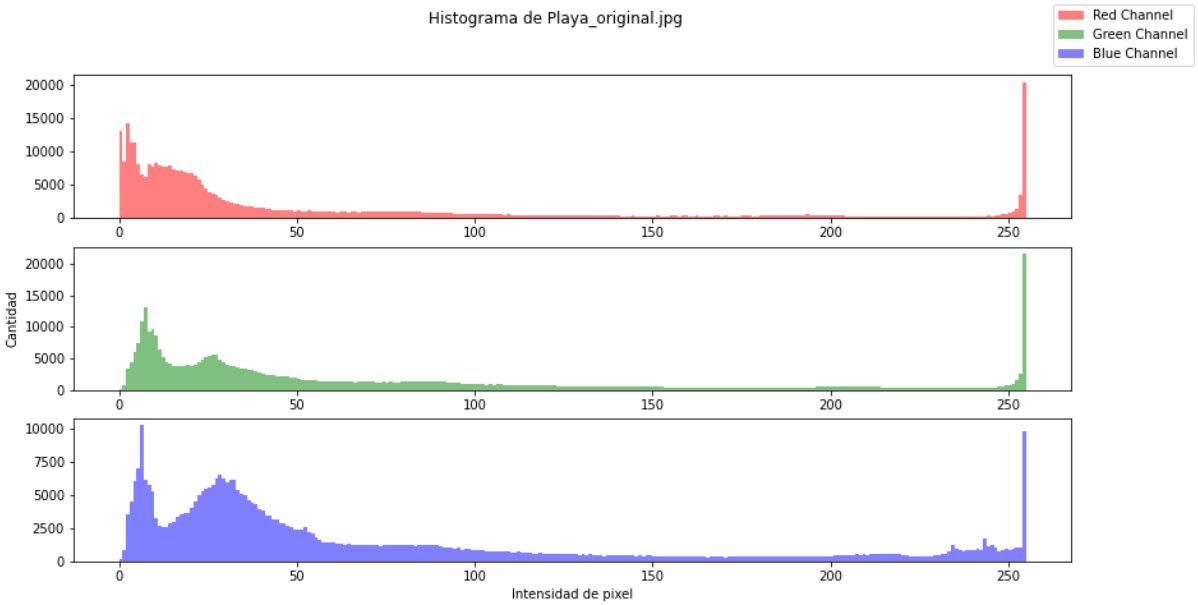

Figura 9: Histograma original playa

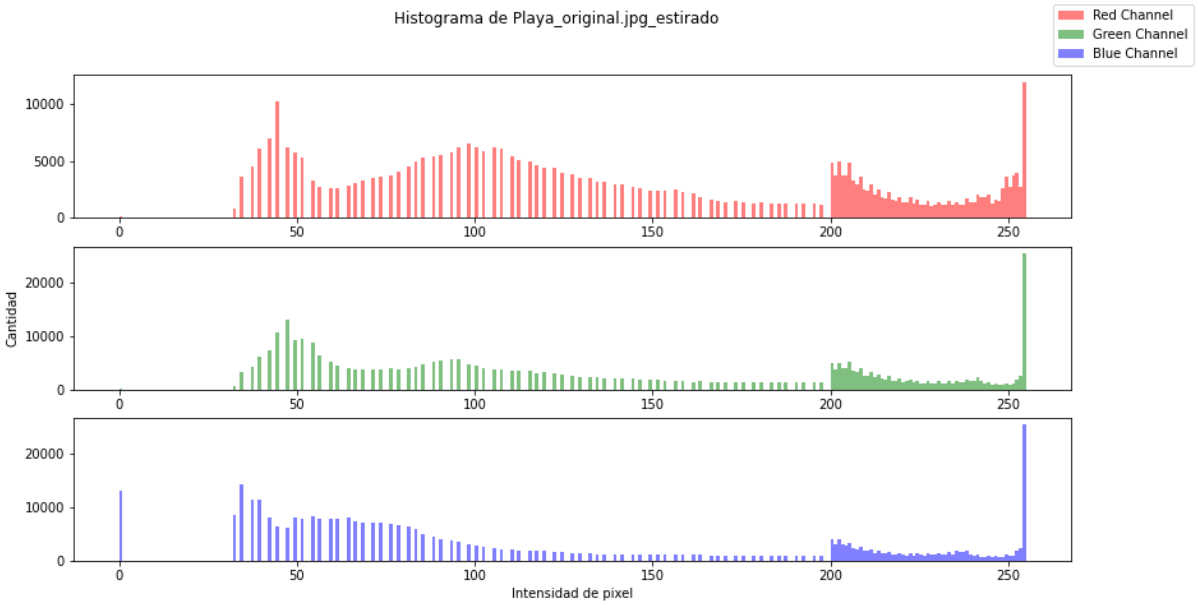

Figura 10: Histograma estirado playa

El resultado de aplicar el estiramiento de contraste a la imagen de la playa se presenta en la **Figura 11**:

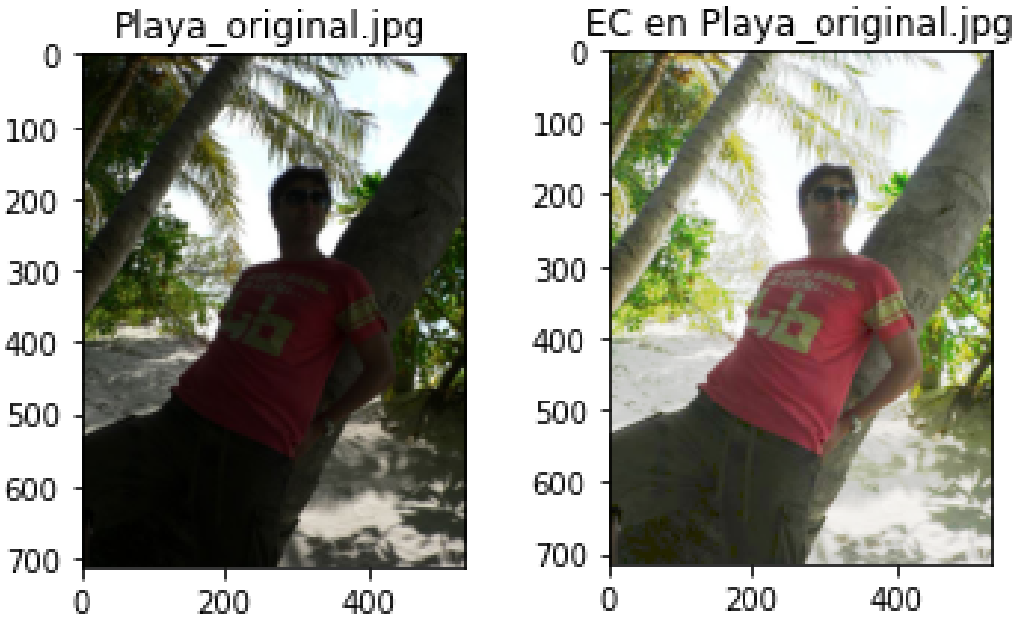

Figura 11: Playa con estiramiento de contraste

El estiramiento de contraste (a = 40, b = 250) en la imagen del vestido transforma los histogramas de la **Figura 12** a los histogramas presentados en la **Figura 13**:

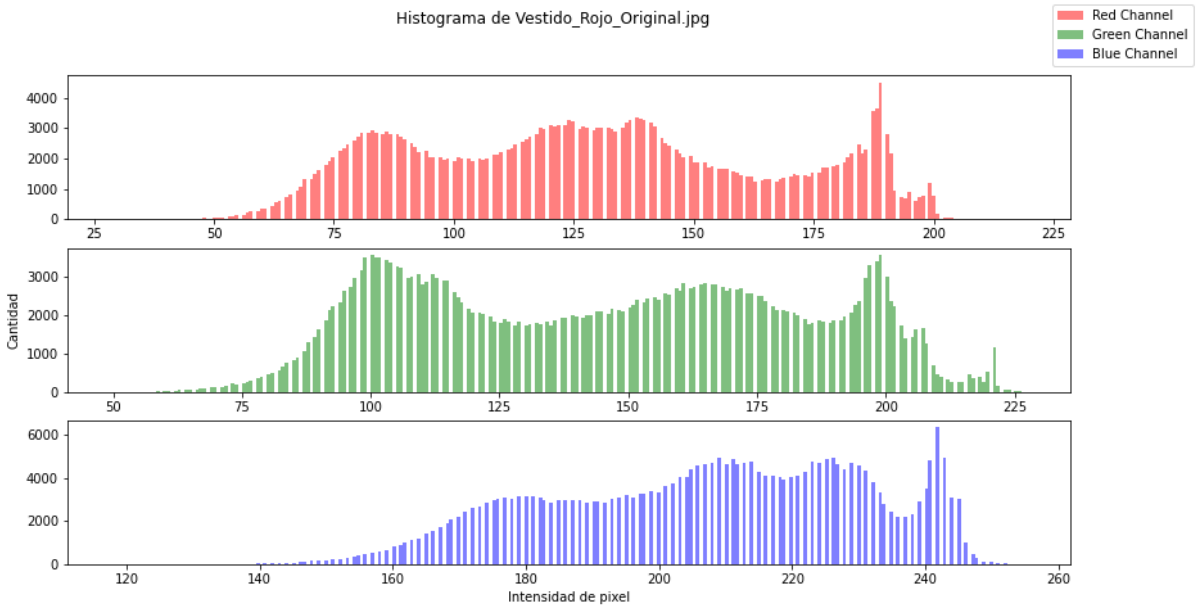

Figura 12: Histograma original vestido

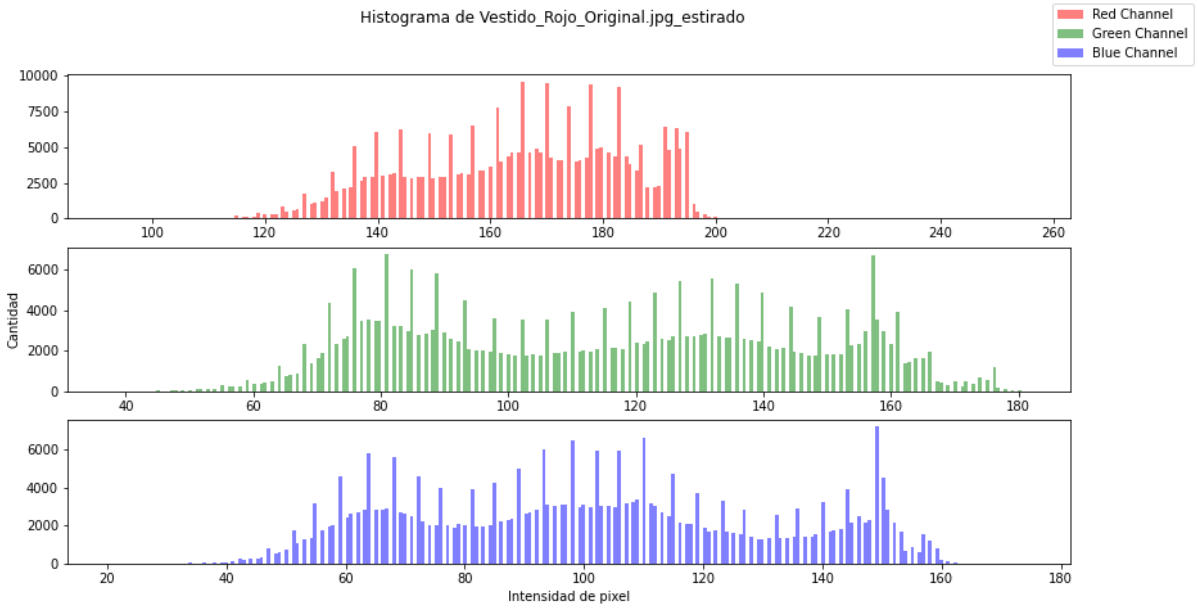

Figura 13: Histograma estirado vestido

El resultado de aplicar el estiramiento de contraste a la imagen del vestido se presenta en la **Figura 14**:

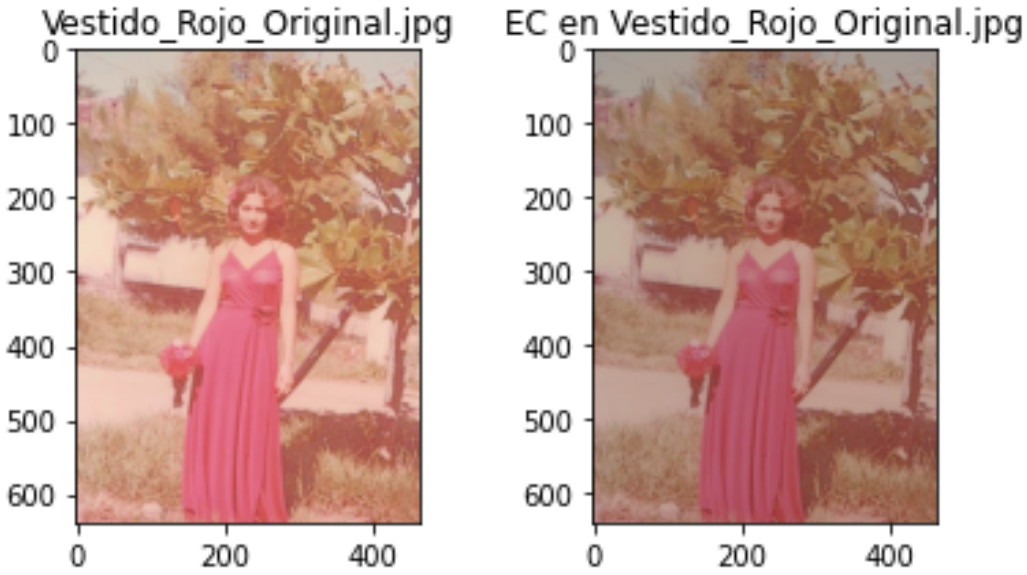

Figura 14: Vestido con estiramiento de contraste

5.2. Ecualización de histograma

Al aplicar la ecualización de histogramas a la imagen del edificio se obtiene el resultado presentado en la **Figura 15**:

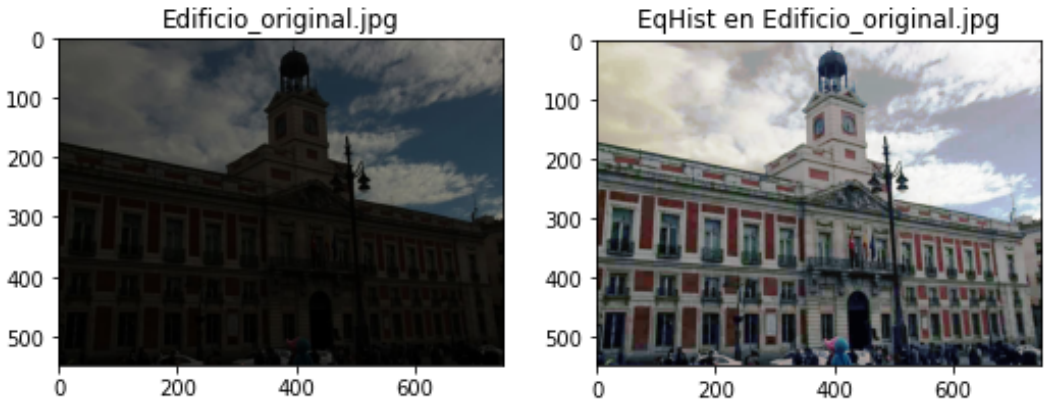

Figura 15: Edificio con ecualización de histograma

Al aplicar la ecualización de histogramas a la imagen del rostro se obtiene el resultado presentado en la **Figura 16**:

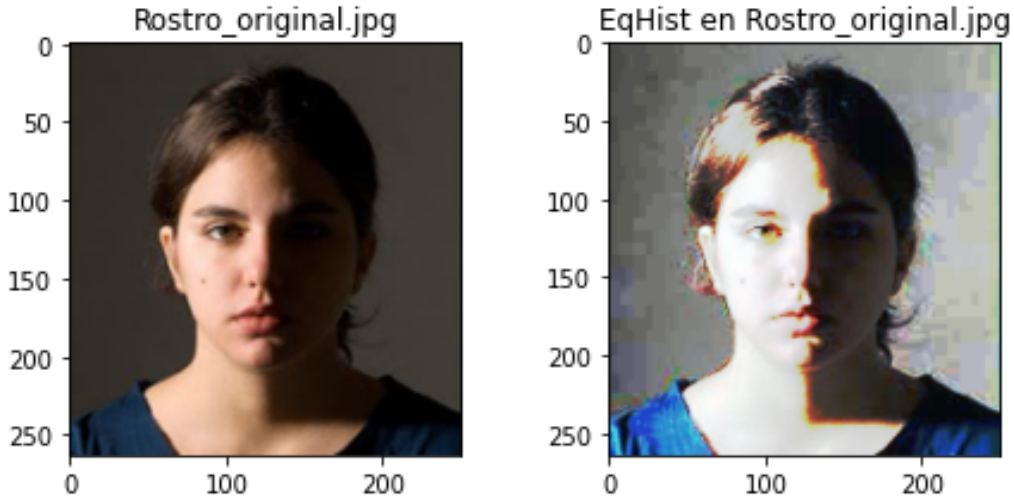

Figura 16: Rostro con ecualización de histograma

Al aplicar la ecualización de histogramas a la imagen de la playa se obtiene el resultado presentado en la **Figura** 17:

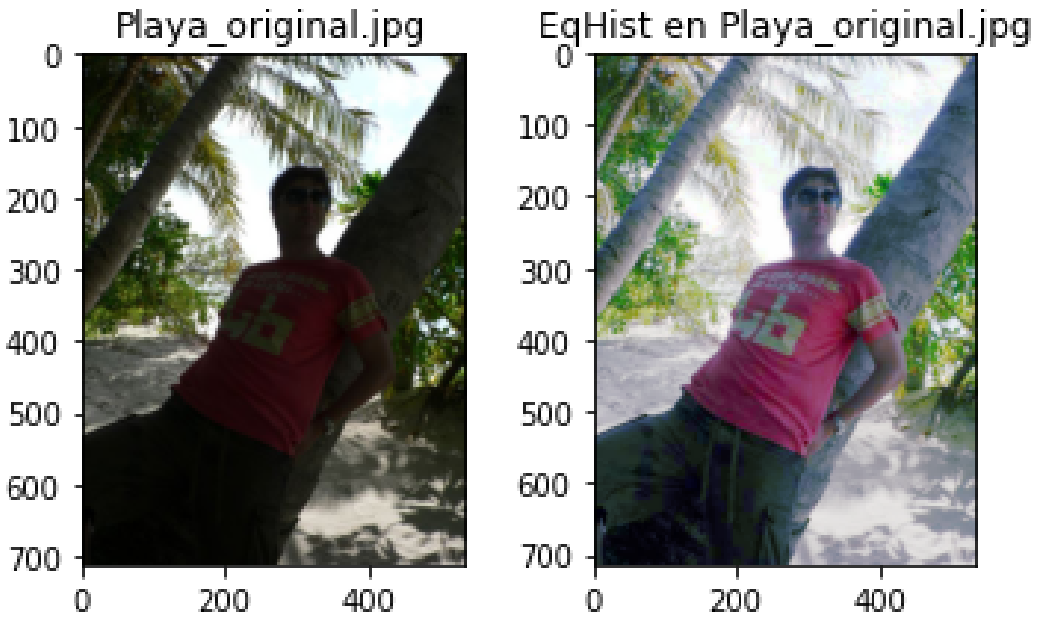

Figura 17: Playa con ecualización de histograma

Al aplicar la ecualización de histogramas a la imagen del vestido se obtiene el resultado presentado en la **Figura** 18:

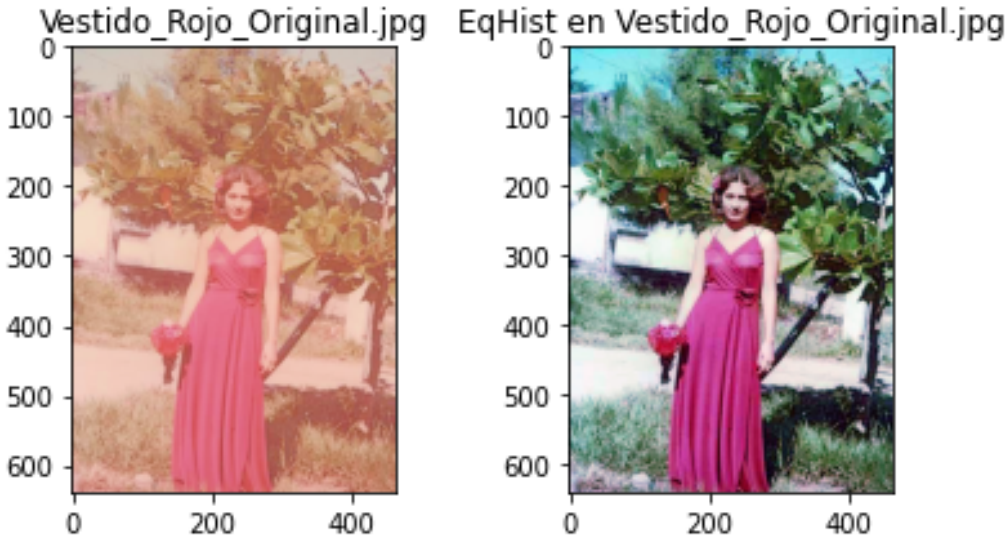

Figura 18: Vestido con ecualización de histograma

5.3. HSV CLAHE

Al aplicar CLAHE HSV a la imagen del edificio se obtiene el resultado presentado en la **Figura 19**:

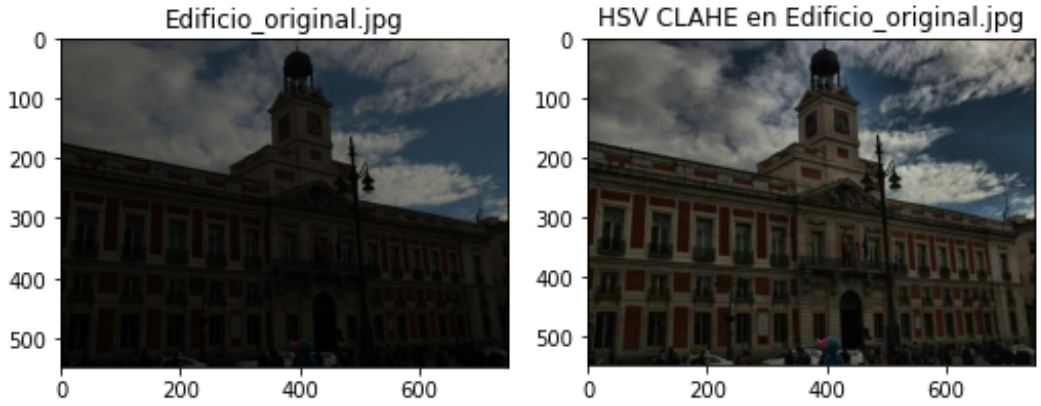

Figura 19: Edificio con CLAHE HSV

Al aplicar CLAHE HSV a la imagen del rostro se obtiene el resultado presentado en la **Figura 20**:

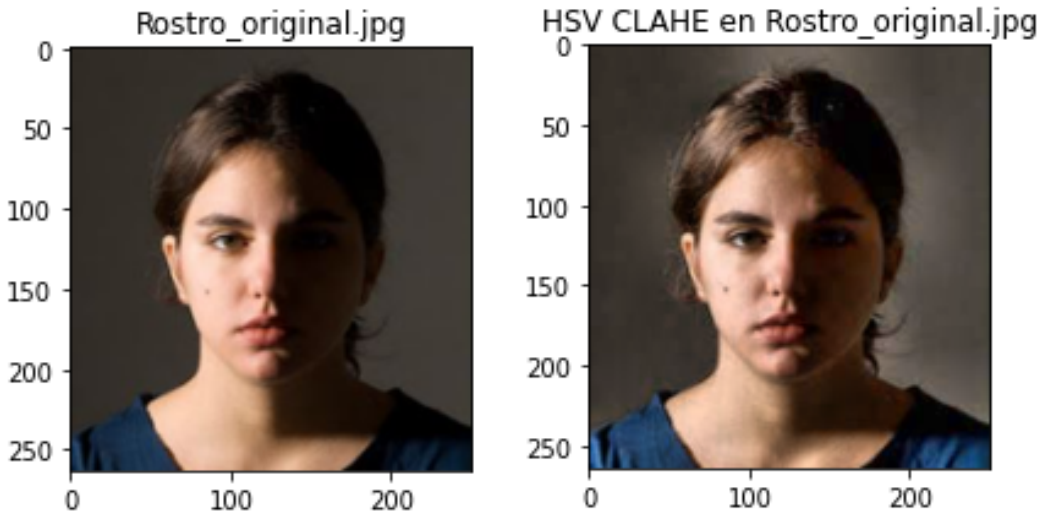

Figura 20: Rostro con CLAHE HSV

Al aplicar CLAHE HSV a la imagen de la playa se obtiene el resultado presentado en la **Figura 21**:

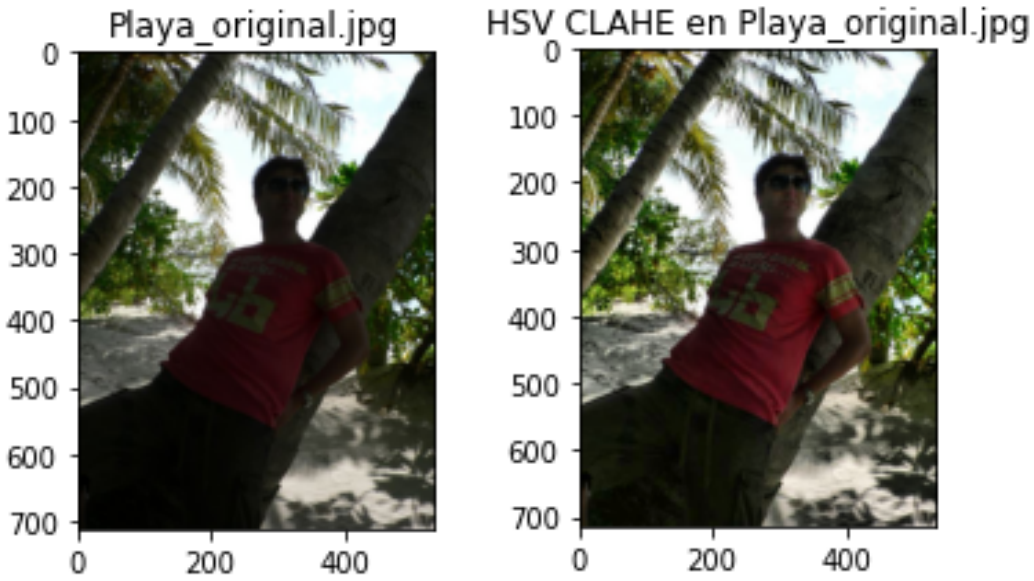

Figura 21: Playa con CLAHE HSV

Al aplicar CLAHE HSV a la imagen del vestido se obtiene el resultado presentado en la **Figura 22**:

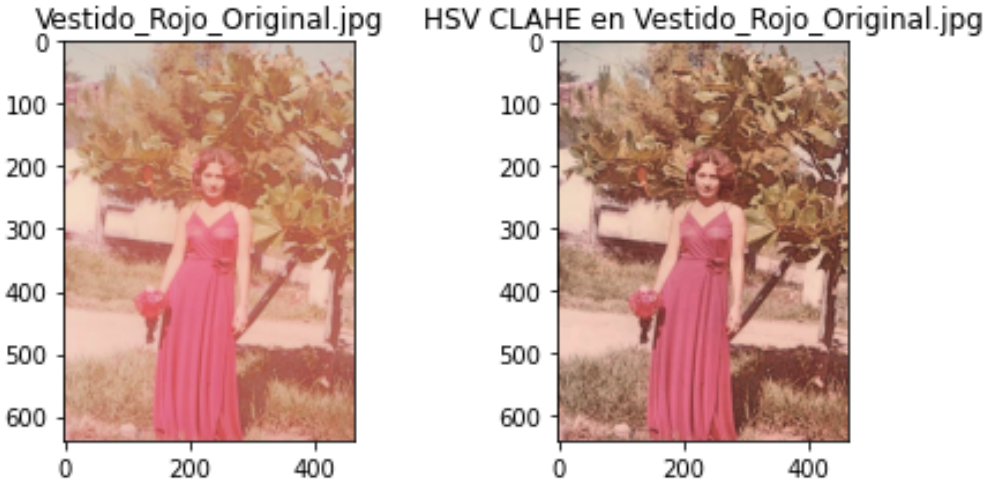

Figura 22: Vestido con CLAHE HSV

5.4. RGB CLAHE

Al aplicar CLAHE RGB a la imagen del edificio se obtiene el resultado presentado en la **Figura 23**:

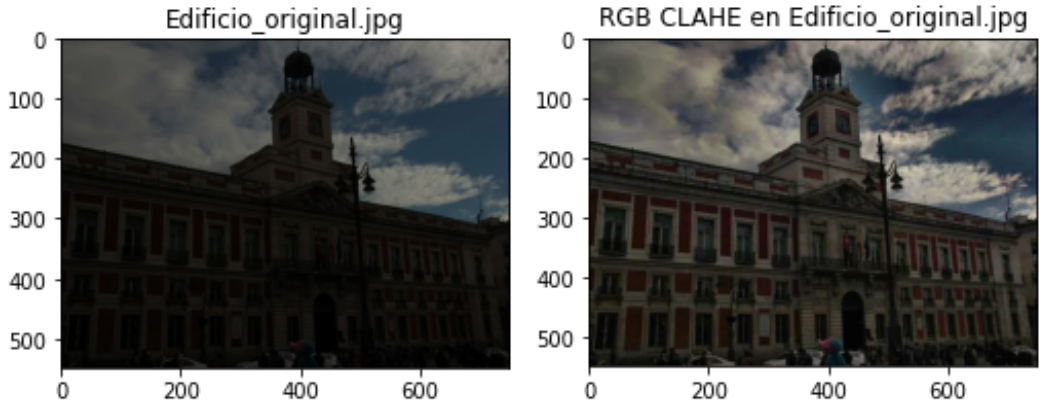

Figura 23: Edificio con CLAHE RGB

Al aplicar CLAHE RGB a la imagen del rostro se obtiene el resultado presentado en la **Figura 24**:

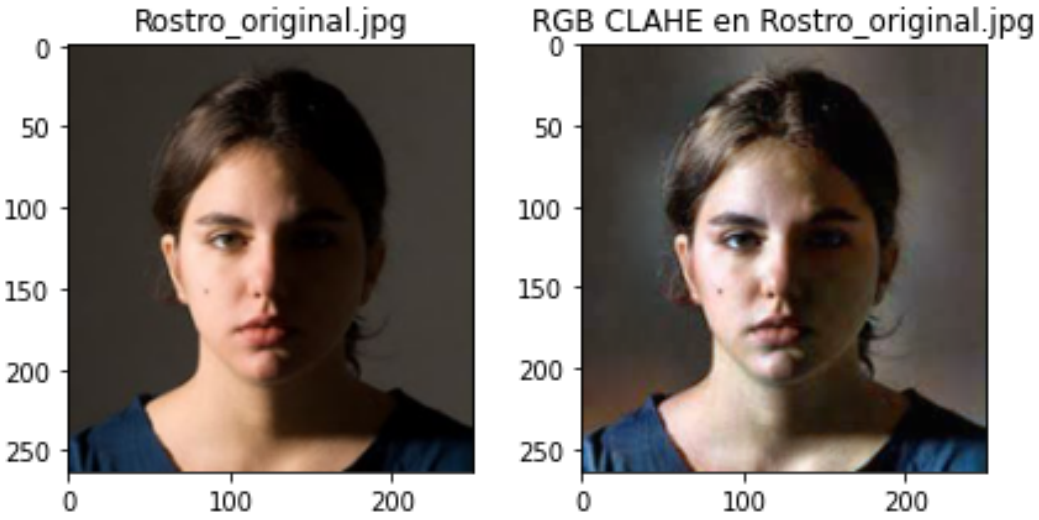

Figura 24: Rostro con CLAHE RGB

Al aplicar CLAHE RGB a la imagen de la playa se obtiene el resultado presentado en la **Figura 25**:

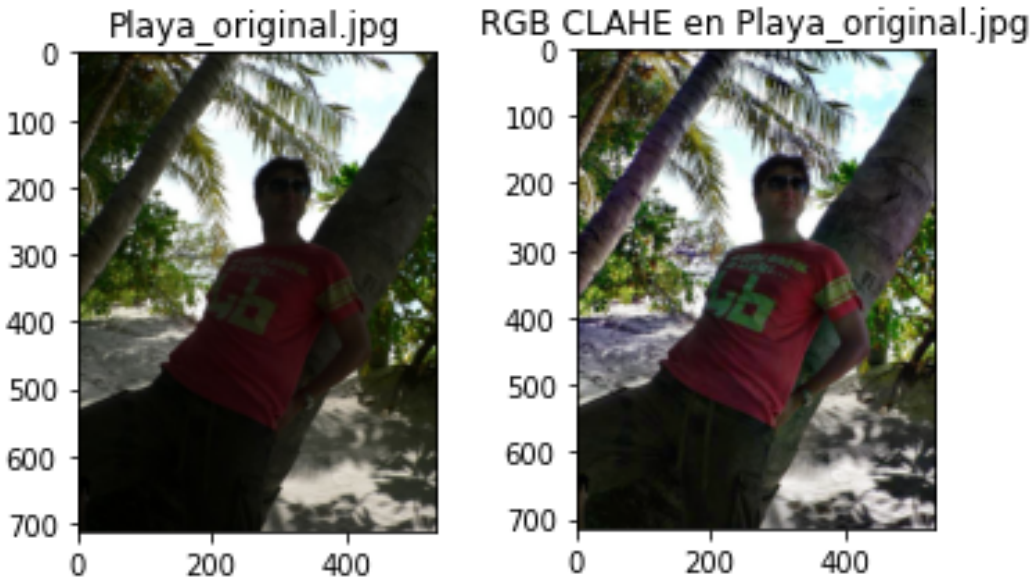

Figura 25: Playa con CLAHE RGB

Al aplicar CLAHE RGB a la imagen del vestido se obtiene el resultado presentado en la **Figura 26**:

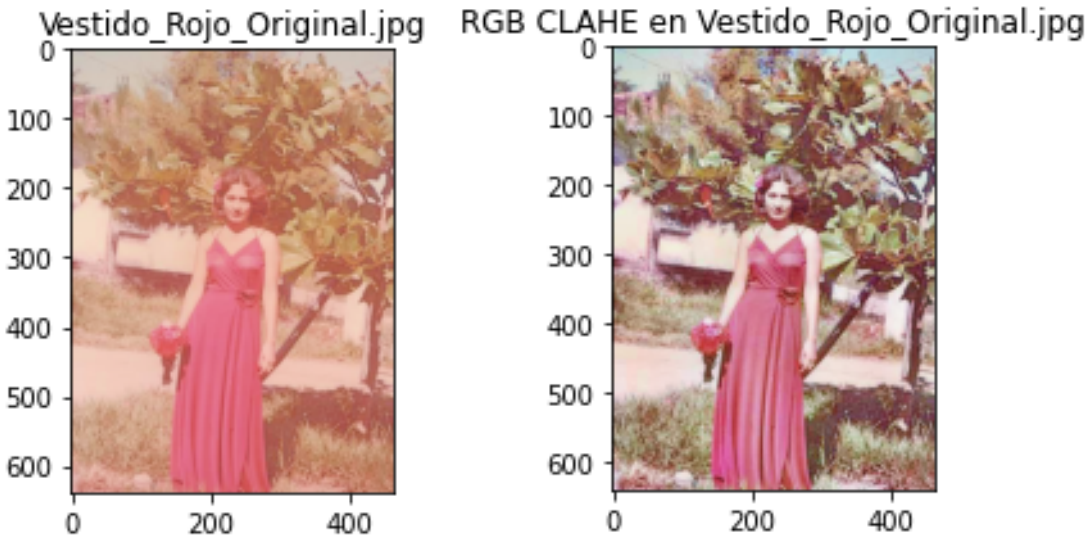

Figura 26: Vestido con CLAHE RGB

5.5. Retinex Net

Al aplicar Retinex Net a la imagen del edificio se obtiene el resultado presentado en la **Figura 27**:

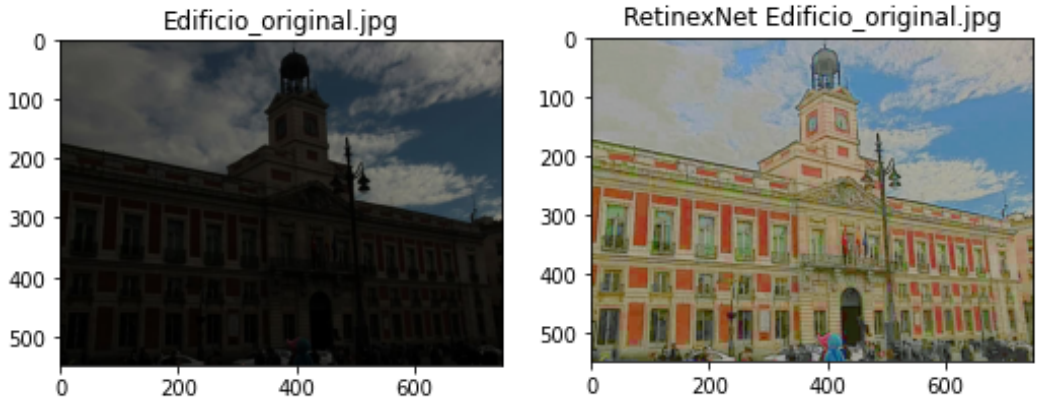

Figura 27: Edificio con Retinex Net

Al aplicar Retinex Net a la imagen del rostro se obtiene el resultado presentado en la **Figura 28**:

Figura 28: Rostro con Retinex Net

Al aplicar Retinex Net a la imagen de la playa se obtiene el resultado presentado en la **Figura 29**:

Figura 29: Playa con Retinex Net

Al aplicar Retinex Net a la imagen del vestido se obtiene el resultado presentado en la **Figura 30**:

Figura 30: Vestido con Retinex Net

5.6. MIRNet

Al aplicar MIR Net a la imagen del edificio se obtiene el resultado presentado en la **Figura 31**:

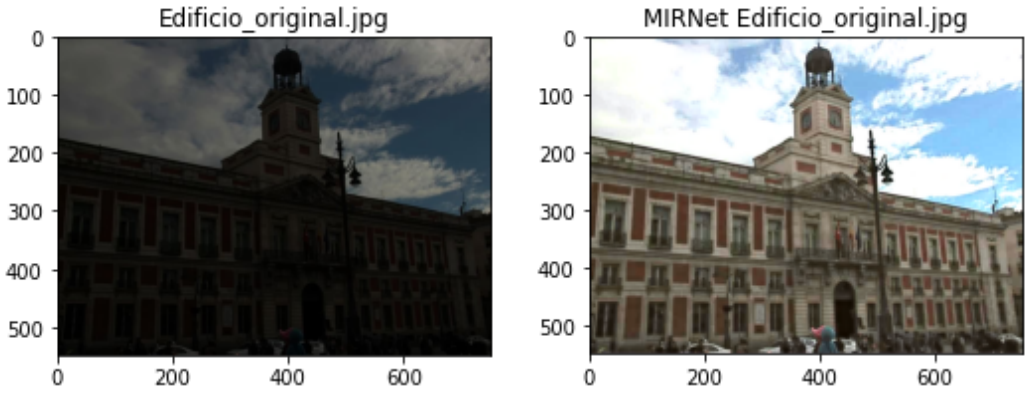

Figura 31: Edificio con MIR Net

Al aplicar MIR Net a la imagen del rostro se obtiene el resultado presentado en la **Figura 32**:

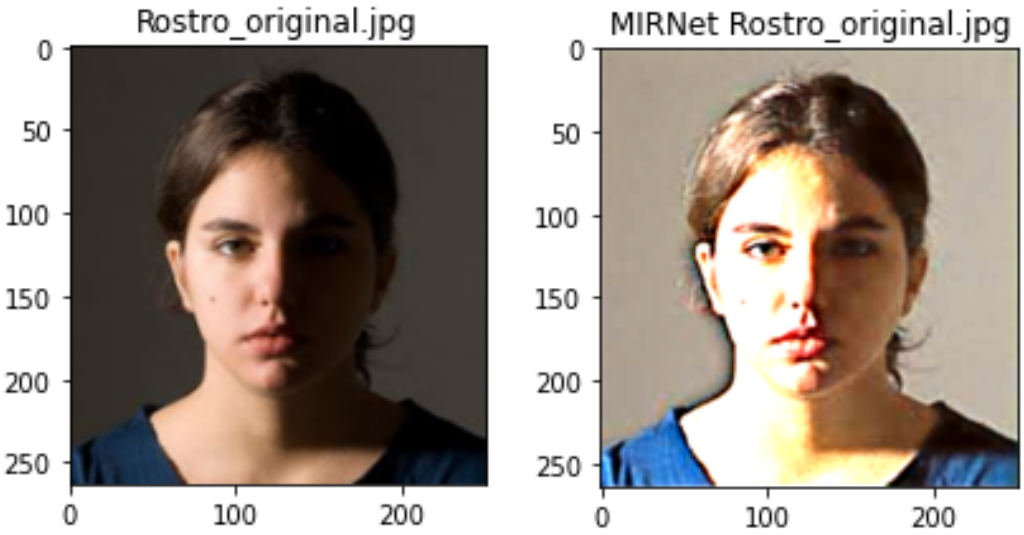

Figura 32: Rostro con MIR Net

Al aplicar MIR Net a la imagen de la playa se obtiene el resultado presentado en la **Figura 33**:

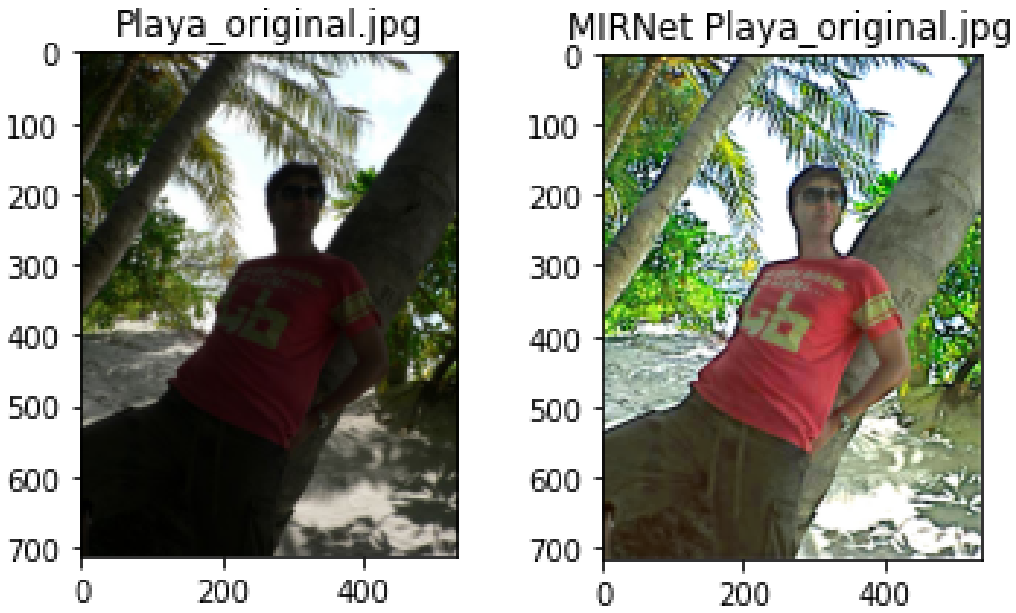

Figura 33: Playa con MIR Net

Al aplicar MIR Net a la imagen del vestido se obtiene el resultado presentado en la **Figura 34**:

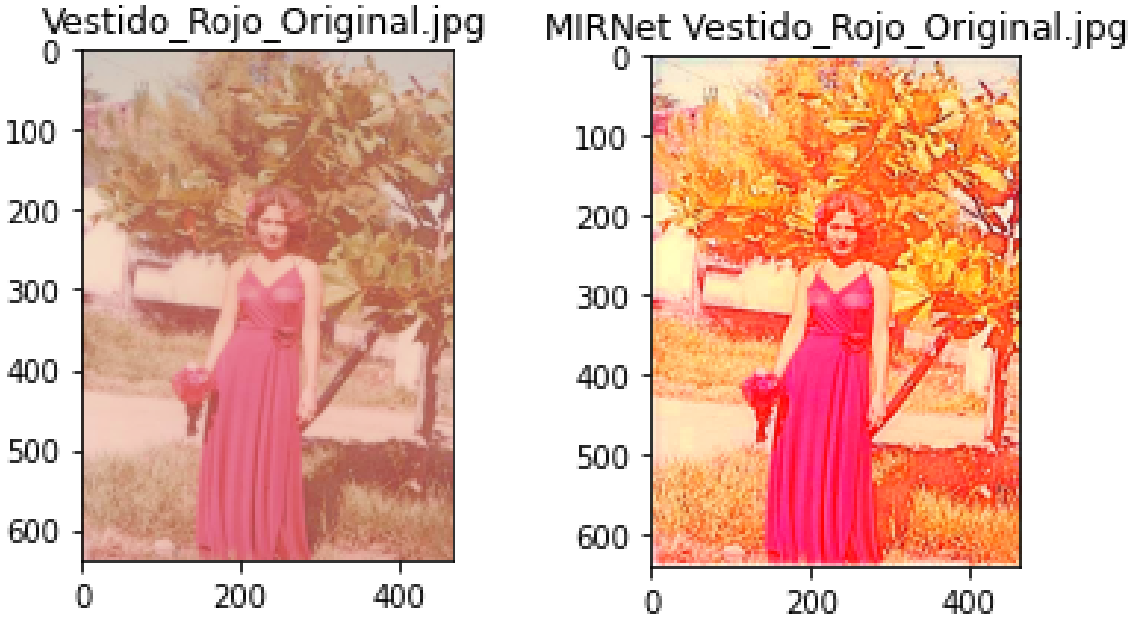

Figura 34: Vestido con MIR Net

5.7. Métricas por imagen

A continuación se presenta un resumen de las métricas obtenidas con cada métrico para cada una de las imágenes.

Tabla 1: Métricas Edificio

Imagen Edificio	Estiramiento de Contraste	Ecualización Histograma	CLAHE HSV	CLAHE BGR	Retinex Net	MIR Net
$\hat{M}^{(3)}$	178,014	176,003	168,161	169,055	163,579	165,640
Σ	0,000	1,649	0,786	1,649	1,260	-12,280
e	0,325	0,214	0,079	0,073	0,057	0,010

Tabla 2: Métricas Rostro

Imagen Rostro	Estiramiento de Contraste	Ecualización Histograma	CLAHE HSV	CLAHE BGR	Retinex Net	MIR Net
$\hat{M}^{(3)}$	161,98	181,475	157,250	169,353	148,958	165,759
Σ	0,000	3,556	1,251	3,248	2,989	-15,741
e	0,024	0,059	0,449	0,488	0,170	0,056

Tabla 3: Métricas Playa

Imagen Playa	Estiramiento de Contraste	Ecualización Histograma	CLAHE HSV	CLAHE BGR	Retinex Net	MIR Net
$\hat{M}^{(3)}$	171,238	151,807	165,372	171,015	158,917	149,805
Σ	0,000	4,608	0,002	1,186	3,868	-33,968
e	0,200	0,160	0,229	0,228	0,198	1,860

Tabla 4: Métricas Vestido

Imagen Vestido	Estiramiento de Contraste	Ecualización Histograma	CLAHE HSV	CLAHE BGR	Retinex Net	MIR Net
$\hat{M}^{(3)}$	159,199	165,081	168,043	178,803	167,054	173,405
Σ	0,000	-0,345	-0,001	-0,001	-0,083	-19,619
e	0,032	1,968	1,601	1,570	0,094	0,353

6. Análisis de resultados

Se puede apreciar en las imágenes presentadas en la **Sección 5.1** que la técnica de estiramiento de contraste efectivamente preserva la forma de la curva seleccionada y al mismo tiempo mapea sus valores al rango deseado de [0,255].

En la **Sección 5** se presentaron todos los resultados obtenidos, sin embargo, para mantener relativamente breve el análisis se opta por discutir qué método tuvo mejores resultados en cada imagen, en vez de analizar uno a uno todos los resultados presentados.

6.1. Edificio

De manera cualitativa definitivamente el mejor resultado lo obtiene la MIR Net (**Figura 31**) pues resalta bien todos los colores sin que la imagen se vea saturada. También preserva bien los colores de la imagen original, a diferencia del resultado obtenido por la ecualización de histograma (**Figura 15**) que si bien ilumina muy bien la imagen no logra preservar bien los colores.

De manera cuantitativa, en las métricas (**Tabla 1**) MIR Net obtiene el mejor resultado en saturación de pixeles, pero el estiramiento de contraste obtiene el mejor resultado en coloración y el indicador e. Simplemente considerando que el mejor método es el que obtiene las mejores métricas se debería escoger el estiramiento de contraste como el mejor, pero cualitativamente (**Figura 5**) la imagen resultante si bien está mejor iluminada, queda muy apagada.

Considerando que el mejor resultado cualitativo lo obtiene MIR Net, y que dicho método es el mejor en una de las métricas, se puede afirmar que la mejor transformación para la imagen del edificio la obtiene **MIR Net**.

6.2. Rostro

Cualitativamente, RetinexNet (**Figura 28**) es el método que mejor ilumina la zona oscura del rostro, pero "ensucia" mucho la imagen. El método que mejor ilumina el lado oscuro pero preservando la integridad de la imagen es el RGB CLAHE (**Figura 24**).

De manera cuantitativa, en las métricas (**Tabla 2**) se puede apreciar que los mejores resultados los obtiene la ecualización de histograma, MIR Net y CLAHE BGR. Volviendo al análisis cualitativo, la ecualización de histograma satura mucho la imagen, y MIR Net también satura la imagen aunque menos que la ecualización.

Considerando que el mejor resultado cualitativo para el rostro lo entrega RGB CLAHE, y que dicho método es el mejor en por lo menos una de las métricas, se puede afirmar que la mejor transformación para la imagen del rostro la obtiene **RGB CLAHE**.

6.3. Playa

Cualitativamente, MIRNet (**Figura 33**) tiene el mejor desempeño, resaltando bien los colores y mejorando la iluminación. Lo único que le resta puntos es que cuando hay transiciones entre zonas muy oscuras y muy claras (como en el contorno de la cabeza de la persona) se forma una linea negra en el contorno de la figura.

Cuantitativamente, según las métricas (**Tabla 3**) MIR Net es el mejor modelo, perdiendo sólo en la coloración contra el estiramiento de contraste que, cualitativamente obtiene muy buenos resultados (**Figura 11**) pero se ve un poco "apagada".

Tomando en cuenta que el mejor resultado cualitativo para la imagen de la playa lo obtiene MIRNet y que dicha red gana en 2 de las 3 métricas, se puede afirmar que la mejor transformación para la imagen de la playa la entrega **MIR Net**.

6.4. Vestido

Sin dudarlo el mejor resultado cualitativo lo obtiene la ecualización de histograma (**Figura 18**) siendo la única versión de la imagen en la que se notan bien los colores sin el tinte rojizo presente en la imagen original. En RGB CLAHE (**Figura 26**) se obtiene un resultado decente, realzando bien los colores y sin saturar la imagen, pero no logra superar el resultado de la ecualización de histograma. En todos los otros métodos el tinte sigue ahí y a penas se aprecia una diferencia con la imagen original.

Cuantitativamente, según las métricas (**Tabla 4**) los mejores modelos para el vestido son CLAHE RGB, MIR Net y la ecualización de histograma. Es curioso que MIR Net (**Figura 34**) tenga buen desempeño en la saturación de pixeles siendo que la imagen que produce se ve más saturada que la original y mucho más rojiza.

En base a lo anterior se puede afirmar que el mejor método para la imagen del vestido es el de la **ecualización de histograma**.

7. Conclusión

Se logra implementar en Python todas las funciones necesarias para la corrección de iluminación y el mejoramiento de contraste. Si bien los resultados en cada imagen variaron considerablemente, se puede decir que el método que entregó consistentemente mejores resultados fue MIR Net. Se evidencia que si bien los métodos clásicos son más simples que los convolucionales, aún así entregan resultados competitivos, superando en varios casos a las redes.

En general, no se podría establecer un método como ganador y desechar al resto puesto que cada uno tiene sus ventajas y desventajas. Por ejemplo, el estiramiento de contraste tiene la obvia desventaja de que requiere intervención humana al momento de escoger los valores de a y b, pero cuando estos se escogen bien producen buenos resultados, como en la imagen de la playa. O también la ecualización de histogramas, que es buena resaltando colores, aunque en ciertos casos se aleja un poco de la paleta original.

Independiente de qué método funcionó mejor que los demás, todos los métodos implementados en esta tarea cumplen con el objetivo de corregir la iluminación y mejorar el contraste de las imágenes de entrada, aunque con variada calidad. Entonces se puede afirmar que se cumplió con el objetivo principal de la tarea y que gracias a esta experiencia se obtienen nuevos mecanismos para enfrentar problemas en los que haya que resolver este problema.