Proyecto 1 - INFO1128
Eliaser Concha

4. Programe un script en Python + PIL + PyGame que calcule los Momentos Invariantes de Hu para las siguientes imágenes:

Código:

Una vez abierto la imagen, se procede a convertir en escala de grises, pasando como argumento un solo canal (L), y posteriormente extraer 6 regiones para cada vocal mediante el método crop, el cual recibe una tupla de 4 elementos dónde el primero es la cantidad de pixeles a desplazar desde la izquierda, el segundo elemento corresponde a la cantidad de pixeles a desplazar desde arriba, el tercer elemento corresponde a la cantidad de pixeles a desplazar desde la derecha, y el último elemento de la tupla corresponde a la cantidad de pixeles a desplazar hacia abajo. Posteriormente, se redimensiona a una imagen de 120 píxeles de ancho x 120 píxeles de alto.

```
6   figuras = figuras.convert("L")
7
8   vocales = { "a": [], "e": [], "i": [], "o": [], "u": [] }
9
10  # Obtener una región para cada letra presente en la imagen
11  vocales["a"].append(figuras.crop((0,0,90,90)).resize((120,120)))
12  vocales["a"].append(figuras.crop((0,90,90,180)).resize((120,120)))
13  vocales["a"].append(figuras.crop((0,180,90,250)).resize((120,120)))
14  vocales["a"].append(figuras.crop((0,250,90,340)).resize((120,120)))
15  vocales["a"].append(figuras.crop((0,330,90,420)).resize((120,120)))
16  vocales["a"].append(figuras.crop((0,410,90,510)).resize((120,120)))
```

A partir de aquí, podemos calcular los momentos invariantes de HU, recorriendo cada vocal y llamando al módulo con “momentos” con la respectiva implementación del momento.

```
52      for a in vocales["a"]:
53          data = []
54          data.append(momentos.momento_hu1(a))
55          data.append(momentos.momento_hu2(a))
56          data.append(momentos.momento_hu3(a))
57          data.append(momentos.momento_hu4(a))
58          data.append(momentos.momento_hu5(a))
59          data.append(momentos.momento_hu6(a))
60          data.append(momentos.momento_hu7(a))
61          m_a.append(data)
```

A continuación, generamos las filas y los respectivos índices de un dataframe. Un dataframe es una estructura de filas y columnas muy utilizada para almacenar y representar datos. En

cada fila se representa la información de cada vocal, mientras que en cada columna se representa un momento de HU en específico.

```
111     data = [m_a[0], m_a[1], m_a[2], m_a[3], m_a[4], m_a[5],
112         m_e[0], m_e[1], m_e[2], m_e[3], m_e[4], m_e[5],
113         m_i[0], m_i[1], m_i[2], m_i[3], m_i[4], m_i[5],
114         m_o[0], m_o[1], m_o[2], m_o[3], m_o[4], m_o[5],
115         m_u[0], m_u[1], m_u[2], m_u[3], m_u[4], m_u[5]]
116
117     # Ajustar los índices de las filas para indicar a qué imagen corresponde
118     index = ["a0","a1","a2","a3","a4","a5",
119         "e0","e1","e2","e3","e4","e5",
120         "i0","i1","i2","i3","i4","i5",
121         "o0","o1","o2","o3","o4","o5",
122         "u0","u1","u2","u3","u4","u5"]
```

Generamos el dataframe, ajustamos los índices de las filas para indicar a que vocal corresponde la información, y exportamos la información a un archivo cuyos se separan a través de una coma (.csv).

```
124     # Crear un dataframe (estructura de filas y columnas) para almacenar la información obtenida
125     moments = pd.DataFrame(data, columns=["HU1","HU2","HU3","HU4","HU5","HU6","HU7"])
126     # Ajustar los índices
127     moments.index = index
128     print(moments)
129     # Almacenar el dataframe en un archivo separado por comas.
130     moments.to_csv("./result/ejercicio4/moments.csv",sep=",")
```

Resultados:

Los resultados de los cálculos, se almacenan en un archivo separados por comas (csv) generado a través de la librería de pandas. Este archivo se almacena en el directorio “/result/ejercicio4/moments.csv”

Conclusiones:

- La principal conclusión que se puede obtener es que los momentos invariantes de HU se aplican en la rotación de las imágenes, y bajo este aspecto se puede analizar a través de los información extraída que los momentos invariante de HU, varían según el ángulo de inclinación de los píxeles en negro de las vocales. Otro factor muy importante que pudo realizar ciertas variaciones en sus resultados, es la precisión a la hora de extraer las regiones respectivas para cada vocal.
- Con respecto a las aplicaciones que pueden involucrar, de manera breve puedo comentar que un uso específico sería en problemas relacionados con transformaciones geométricas como problemas de escala, transformación y rotación. Ejemplo de ello, involucra a software especializado en el diseño gráfico, a escala de empresas como Adobe tipo After Effects, Photoshop y/o Illustrator, dónde usualmente incorpora funcionalidades para trabajar con transformaciones geométricas dentro de imágenes.

5. Programe un script en Python + PIL + PyGame que calcule y grafique los Histogramas Normal y Acumulativo para cada figura (F1-F6) en modo GREY.

Código:

Definir la función que permite calcular el histograma acumulado como la sumatoria de los valores del histograma original, es utilizado principalmente para realizar ciertas operaciones que involucren histogramas.

```python
# Generar el histograma acumulado
def cumulative_histogram(img:Image):
    img = img.copy()
    w = img.width
    h = img.height
    H = img.histogram()
    for j in range(len(H)):
        H[j] = H[j-1] + H[j]
    return H
```

Implementar función que permite generar botones interactivos con pygame. Recibe como argumentos, la superficie principal (que puede ser la ventana o display principal), recibe un rectángulo que dará forma al botón, una tupla en formato rgb para el color de fondo o background, otra tupla en rgb para el color de fondo cuando se le pase el mouse por encima, la posición en la que esté ubicado y txt representa al texto que contiene. Dentro de la función se genera una fuente para el texto, se valida que la posición del mouse se encuentre dentro del botón para generar un efecto de hover o cambio de fondo dibujando botones con características que lo diferencian entre sí (en este caso el color de fondo). Luego se establece el texto el color verde, y se centra de forma vertical como horizontal dentro del botón.

```python
# Función para dibujar un botón en pygame
def draw_button(surface:pygame.Surface, button:pygame.Rect, background:tuple, background_hover:tuple, posicion:tuple, txt:str):
    # Generar el tipo de letra con su tamaño
    myFont = pygame.font.SysFont("Calibri",30)
    # Validar la posición del mouse dentro del botón
    if button.collidepoint(pygame.mouse.get_pos()):
        # Dibujar un botón redondeando de color definido por el background
        pygame.draw.rect(surface, background, button, border_radius=50)
    else:
        # Dibujar un botón redondeando de color definido por el background_hover
        pygame.draw.rect(surface, background_hover, button, border_radius=50)

    # Establecer el texto en color verde
    texto = myFont.render(txt,True, (255,255,255))
    # Establecer el texto dentro de la superficie centrado tanto horizontal como verticalmente con respecto al botón
    surface.blit(texto,(posicion[0]+(button.width-texto.get_width())/2,posicion[1]+(button.height-texto.get_height())/2))
```

También se incorpora una función para generar la interfaz del carrusel, sin embargo voy a documentar lo importante, que es la detección de eventos. Para cada evento que se detecte en pygame, se valida cuando se cierra la ventana con el ícono de cerrar (nativo de cada

sistema operativo). En caso contrario, se detecta cuando se hace click izquierdo sobre los botones. La ventana principal cambiará de imagen según la posición de la figura en el directorio. Si esta posición al hacer click en next supera a la cantidad de elementos, entonces regresa a la figura inicial. Cuando se detecta el botón prev, la lógica es contraria, es decir, retrocede de posición y de figura.

```
62      while True:
63          # Para cada evento detectado
64          for event in pygame.event.get():
65              # Cerrar la aplicación
66              if event.type == pygame.QUIT:
67                  sys.exit()
68              # Detectar cuando se hace click izquierdo sobre el botón next
69              if event.type == pygame.MOUSEBUTTONDOWN and event.button==1 and next.collidepoint(pygame.mouse.get_pos()):
70                  pos_figure += 1
71                  if pos_figure > len(figure)-1: # Regresar a la imagen inicial
72                      pos_figure = 0
73                  ventana.blit(figure[pos_figure],(0,0)) # Cambiar de imagen
74              # Detectar cuando se hace click izquierdo sobre el botón prev
75              elif event.type == pygame.MOUSEBUTTONDOWN and event.button==1 and prev.collidepoint(pygame.mouse.get_pos()):
76                  pos_figure -= 1
77                  if pos_figure < 0: # Regresar a la imagen final
78                      pos_figure = len(figure)-1
79                  ventana.blit(figure[pos_figure],(0,0))
```

Lo siguiente, es generar un script que recorra todas las figuras en el directorio, para cada una de ellas se debe transformar a escala de gris y posteriormente obtener su histograma normal y acumulado. Y con matplotlib, generar sus respectivas figuras.

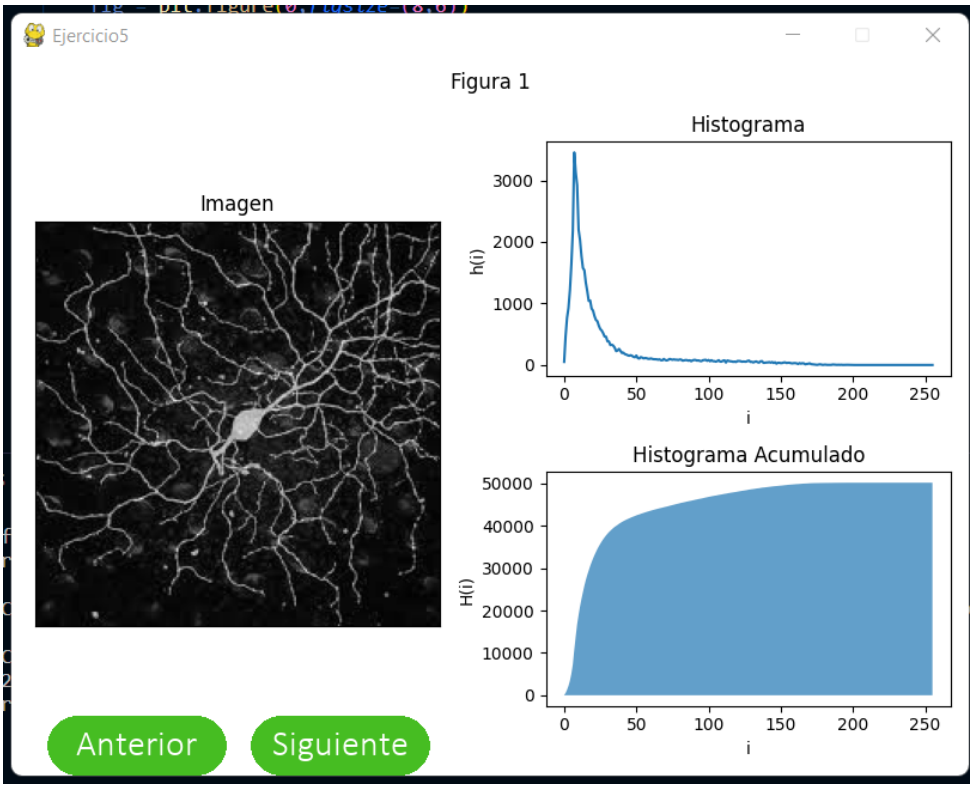
```
94      path = pathlib.Path("./img/figures")
95
96      for i in path.iterdir():
97          img.append(ImageOps.grayscale(Image.open(i)))
98
99      for i in img:
100         h.append(i.histogram())
101         c.append(cumulative_histogram(i))
102
103     for i in range(len(img)):
104         fig = plt.figure(0,figsize=(8,6))
105         fig.suptitle("Figura {}".format(i+1))
106         ax_1 = fig.add_subplot(121)
107         ax_1.imshow(img[i],"gray")
108         ax_1.set_title("Imagen")
109         plt.xticks([]), plt.yticks([])
110         ax_2 = fig.add_subplot(222)
111         ax_2.set_title("Histograma")
112         ax_2.set_ylabel("h(i)")
113         ax_2.set_xlabel("i")
114         ax_2.plot(h[i])
115         ax_3 = fig.add_subplot(224)
116         ax_3.set_title("Histograma Acumulado")
117         ax_3.set_ylabel("H(i)")
118         ax_3.set_xlabel("i")
119         ax_3.fill_between(np.arange(len(c[i])),np.array(c[i]),alpha=0.7)
120         fig.tight_layout()
121         fig.savefig("./result/ejercicio5/F{}.png".format(i+1))
122         plt.close(0)
```

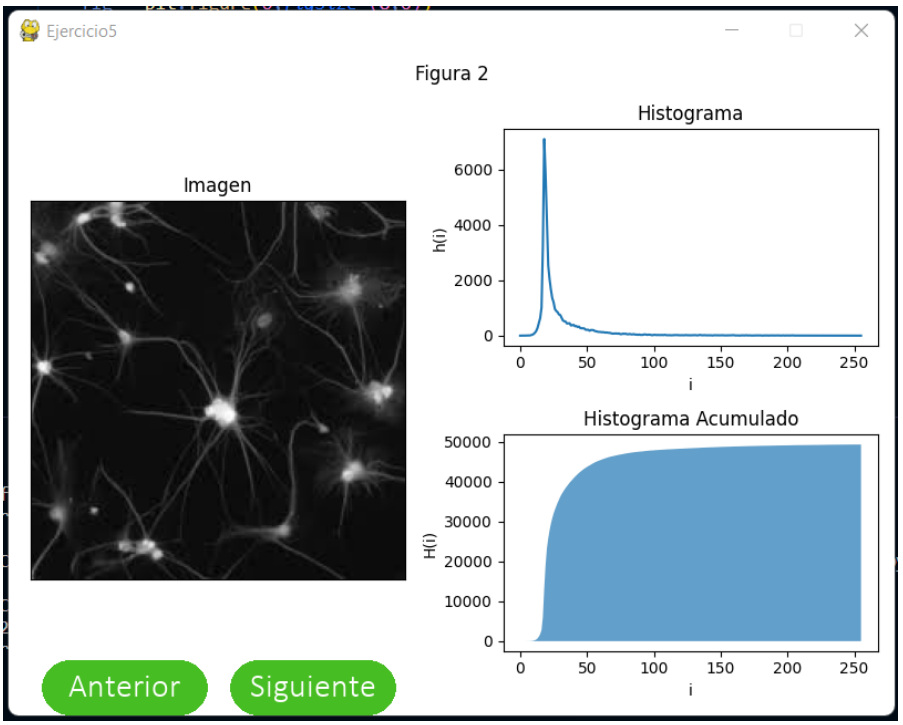
Seguido de ello, se recorre cada figura de matplotlib grabada y estas se transmiten a la función slider, para generar un carrusel de imágenes un poco más interactivo y atractivo, según mi creatividad.

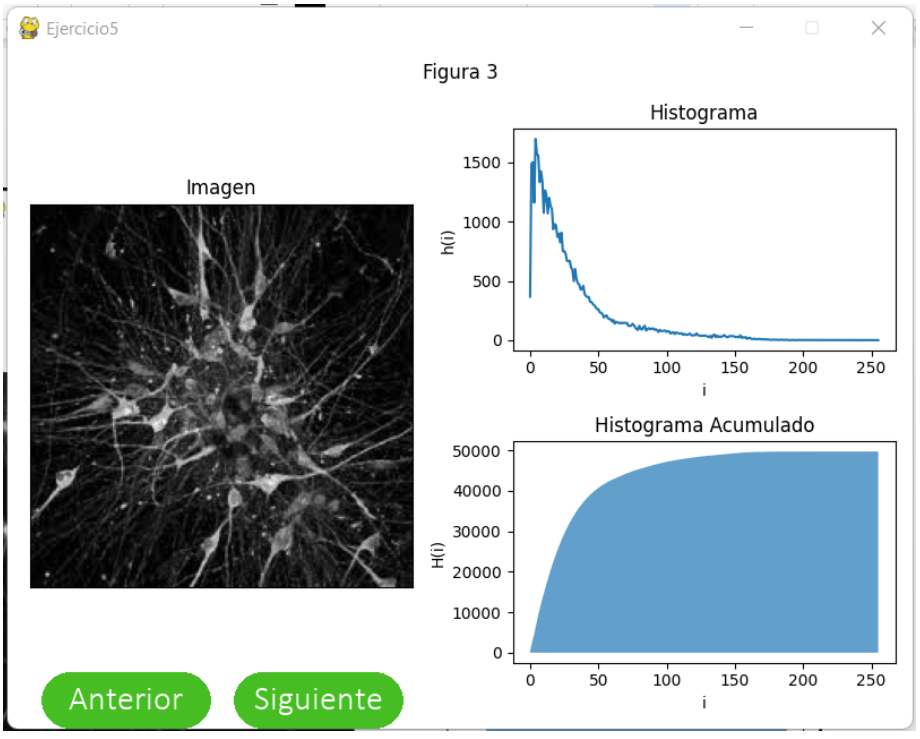
```
124    figure = []
125
126    result = pathlib.Path("./result/ejercicio5")
127
128    for e in result.iterdir():
129        file = pygame.image.load(e)
130        figure.append(file)
131
132    width = figure[0].get_width()
133    height = figure[0].get_height()
134
135    slider("Ejercicio5",size=(width,height),btn_next=[200,550,150,50],btn_prev=[30,550,150,50],
136        figure=figure,color_btn=[(237,128,19),(70,189,34)])
```

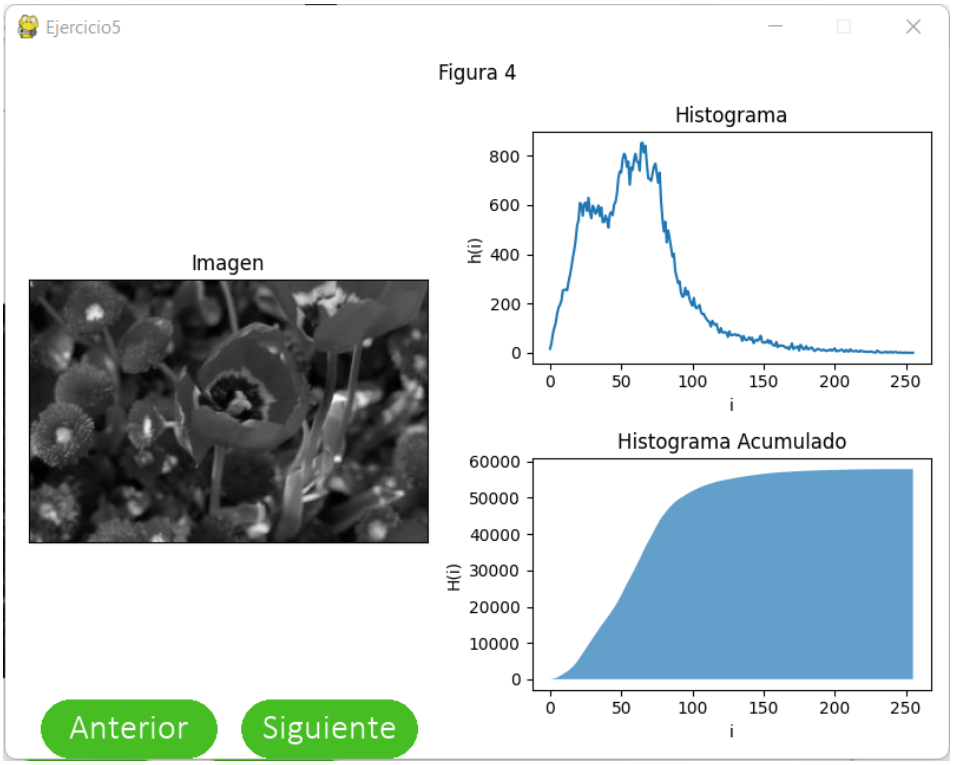
Resultados:

En las figuras 1, 2, 3 y 4, se puede concluir que el histograma normal indica una mayor presencia de tonalidades oscuras, mientras que su histograma acumulado me puede indicar que falta una mayor presencia en tonalidades medias y claras.

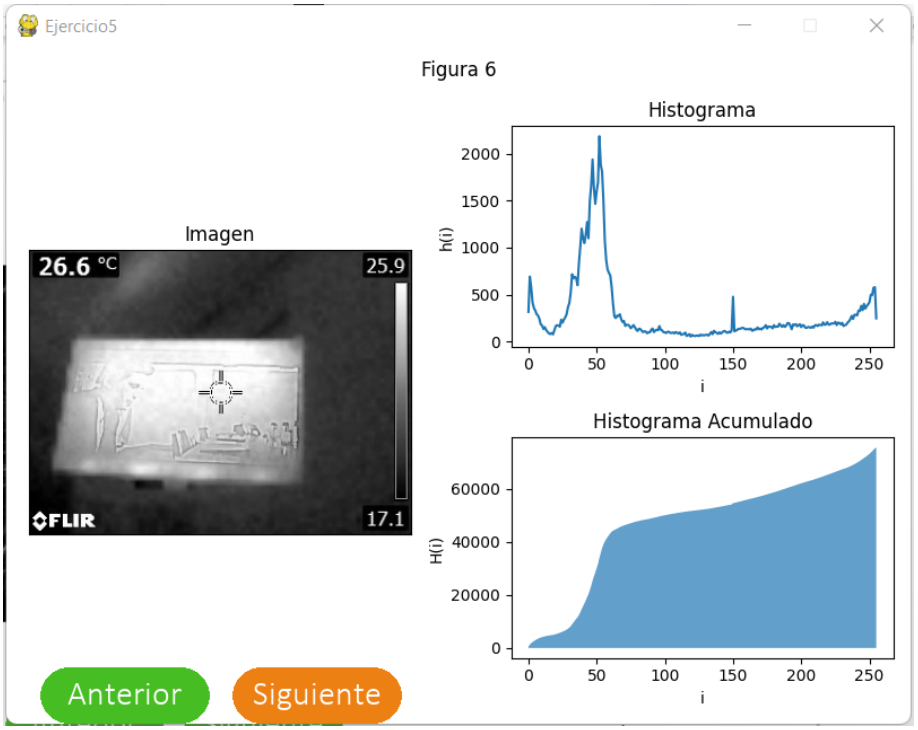



Ejercicio5
Figura 2
Imagen
Histograma
h(i)
i
Histograma Acumulado
H(i)
i
Anterior
Siguiente

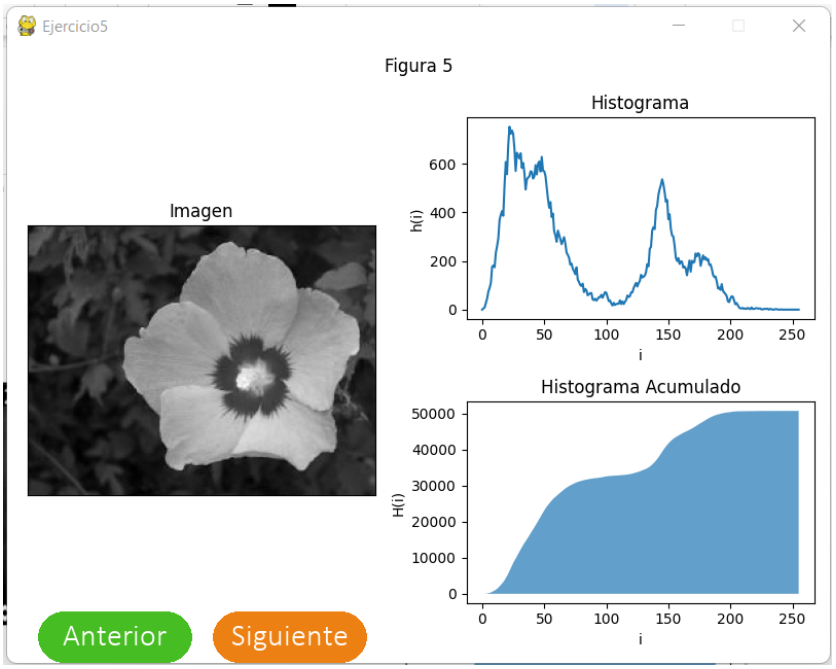

Ejercicio5
Figura 3
Imagen
Histograma
h(i)
i
Histograma Acumulado
H(i)
i
Anterior
Siguiente



En la figura 6, se puede visualizar un aumento en las tonalidades medias y claras, sin embargo, su resultado es similar a figuras 1, 2, 3 y 4.



La figura 5, se puede observar un cambio más notable en sus histogramas, incorporando una mayor presencia de tonalidades medias entre negro y blanco, pero que al final, su histograma acumulado, indica que falta mayor presencia de tonalidades claras.



7. Programe un script en Python + PIL + PyGame que calcule y grafique el Histograma Normal y el Acumulativo para las figuras F1 y F4 en modo CMYK.

Código:

Al abrir las imágenes de interés, se convierte en modo CMYK (cian, magenta, yellow, black) y para cada uno de los canales de la imagen se genera su respectiva gráfica.

```
9     # Abrir las figuras de interés
10    f1 = Image.open("./img/figures/F1.png")
11    f2 = Image.open("./img/figures/F4.png")
12
13    # Convertir cada figura en modo CMYK
14    img = [f1.convert("CMYK"), f2.convert("CMYK")]
15
16    # Para cada figura
17    for i in range(2):
```

Utilizamos subplots para generar una figura de 3 filas y 4 columnas, dónde la primera fila, contiene a las imágenes en su respectivo canal, la segunda fila corresponde a su histograma normal, y la tercera fila corresponde a su histograma acumulado.

```
25        # Generar un subplots de 3 filas y 4 columnas
26        fig, ax = plt.subplots(3,4)
27        # Establecemos una resolución de 15 pulgadas de ancho y 7 pulgadas de alto
28        fig.set_size_inches(15,7)
29        # Mostrar la figura con un mapa de colores en escala de grises
30        ax[0,0].imshow(c1,"gray")
31        # Establecer un título para la figura
32        ax[0,0].set_title("Canal C")
33        # Ocultar los valores en los ejes horizontal y vertical
34        ax[0,0].set_xticks([]), ax[0,0].set_yticks([])
35        ax[0,1].imshow(m1,"gray")
```

Guardamos las figuras creadas con matplotlib, para posteriormente recorrer su directorio y generar una interfaz gráfica similar al ejercicio 5.

```python
        # Guardar cada figura
        plt.savefig("./result/ejercicio7/fig_{}.png".format(i+1))

    result = Path("./result/ejercicio7")

    file = []

    # Iterar cada figura obtenida con matplotlib
    for f in result.iterdir():
        file.append(pygame.image.load(f))

    # Se asume que cada figura de matplotlib contiene el mismo ancho y alto
    width = file[0].get_width()
    height = file[0].get_height()

    # Generar un carrusel de imágenes con la función implementada en el ejercicio 5.
    slider("Ejercicio7",size=(width,height+80),btn_next=[200,720,150,50],btn_prev=[30,720,150,50],
        figure=file, color_btn=[(237,128,19),(70,189,34)])
```

Resultados:
Figura F1:

Los canales C, M y Y me llaman bastante la atención, cada uno muestra detalles totalmente diferentes, siendo el canal C los glóbulos rojos de la imagen original (esto es así porque el canal Cyan es opuesto al canal rojo, por lo tanto, absorbe ese color), el canal magenta representa a la célula verde con sus ramificaciones (esto es así porque el canal Magenta es opuesto al canal verde, por lo tanto, absorbe ese color.), y el canal amarillo o Yellow representa a los puntos de color azul en la figura 1 (absorbiendo el color azul, siendo el amarillo, el opuesto a este mismo). Con respecto a su histograma normal, los 3 primeros canales presentan una fuerte presencia de tonalidades claras, con respecto a sus histogramas acumulados, estas no sufren una gran variación con respecto a la forma gráfica de la recta del histograma normal, confirmando que las 3 tendencias de los 3 primeros canales se acercan bastante a tonalidades más cercanas al blanco.

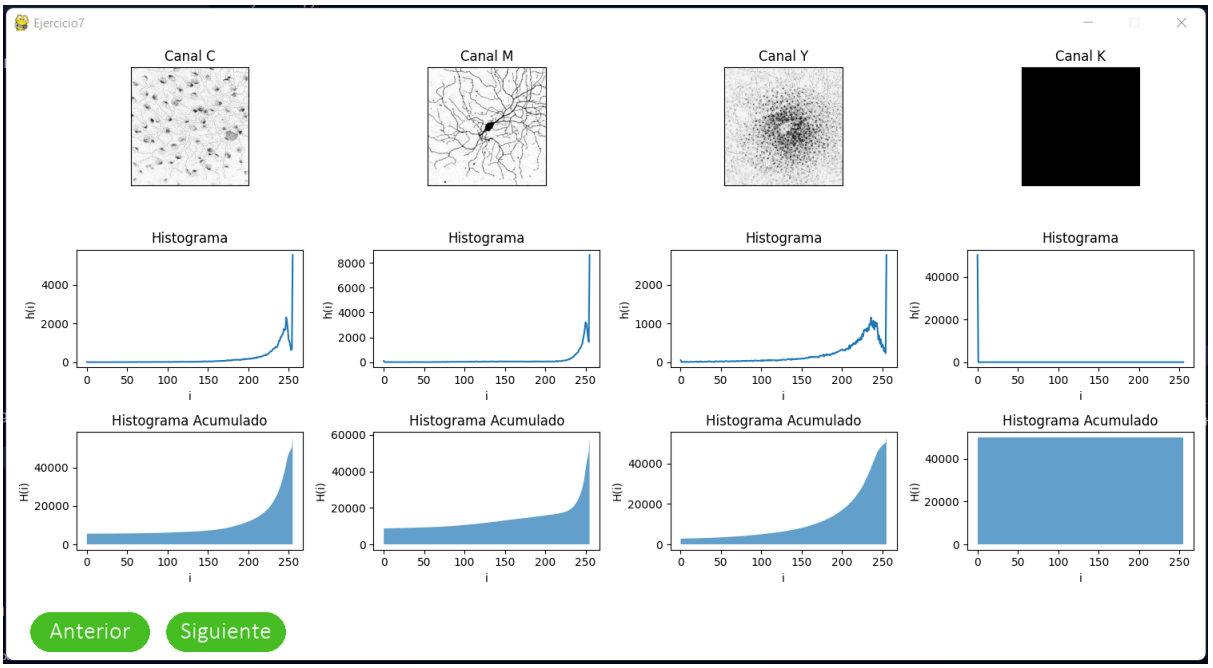


Figura F4:

En el caso de las imágenes, el canal Cyan presenta mayor presencia de tonalidad oscura para los colores rojos de la imagen. En cambio, el canal Magenta presenta una débil presencia de tonalidades oscuras para los colores verdes de la imagen y una mayor presencia de tonalidades claras para otros canales de la imagen, mientras que el canal amarillo viene a representar todos los planos conformados por el color azul.