Instituto Tecnológico de Morelia

práctica 4: ejercicios

Graficación

Profesor:
Martinez Guzman Bryan Eduardo

Santiago Gonzalez Lara 22121360

11 de febrero del 2025

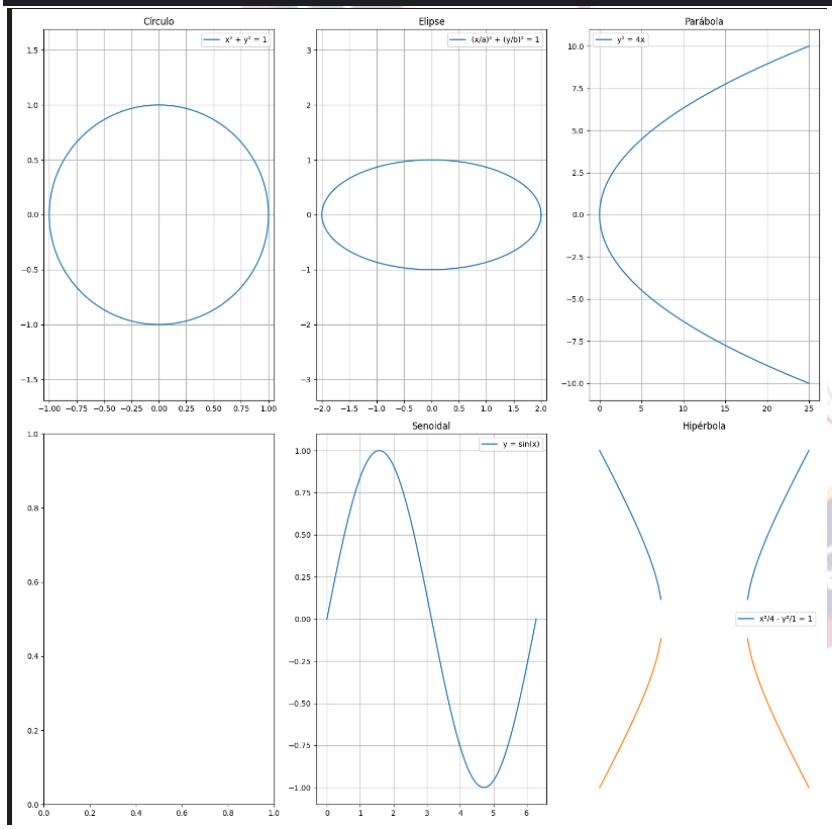
reporte
primitivas graficación

En escencia, son elementos basicos que usamos como base para construir diseños más complejos a partir de ellas, las primitivas son:

- circulo
- elipse
- parabola
- senoidal
- hiperbola

```
import numpy as np
import matplotlib.pyplot as plt
fig, axs = plt.subplots(2, 3, figsize=(15, 15))

# Círculo
theta = np.linspace(0, 2 * np.pi, 100)
x_circle = np.cos(theta)
y_circle = np.sin(theta)
axs[0, 0].plot(x_circle, y_circle, label="x² + y² = 1")
axs[0, 0].set_title("Círculo")
axs[0, 0].axis("equal")
axs[0, 0].legend()
axs[0, 0].grid()

# Elipse
a, b = 2, 1
x_ellipse = a * np.cos(theta)
y_ellipse = b * np.sin(theta)
axs[0, 1].plot(x_ellipse, y_ellipse, label="(x/a)² + (y/b)² = 1")
axs[0, 1].set_title("Elipse")
axs[0, 1].axis("equal")
axs[0, 1].legend()
axs[0, 1].grid()

# Parábola
y = np.linspace(-10, 10, 100)
x_parabola = y**2 / 4
axs[0, 2].plot(x_parabola, y, label="y² = 4x")
axs[0, 2].set_title("Parábola")
axs[0, 2].legend()
axs[0, 2].grid()

# Hipérbola
x = np.linspace(-5, 5, 100)
y_hyperbola_pos = np.sqrt((x**2 / 4) - 1)
y_hyperbola_neg = -y_hyperbola_pos
axs[1, 2].plot(x, y_hyperbola_pos, label="x²/4 - y²/1 = 1")
axs[1, 2].plot(x, y_hyperbola_neg)
axs[1, 2].set_title("Hipérbola")
```

```
axs[1, 2].legend()
axs[1, 2].grid()


# Senoidal
x = np.linspace(0, 2 * np.pi, 100)
y = np.sin(x)
axs[1, 1].plot(x, y, label="y = sin(x)")
axs[1, 1].set_title("Senoidal")
axs[1, 1].legend()
axs[1, 1].grid()

# Desactivar los gráficos restantes
axs[1, 2].axis("off")

plt.tight_layout()
plt.show()
```



Modelos de colores

RGB: Combina tres colores base (rojo, verde y azul) con intensidades entre 0 y 255.

CMY: Se calcula como CMY=1−RGB CMY = 1 - RGB, donde los colores base son cian, magenta y amarillo.
HSV:Representa los colores en términos más intuitivos para los humanos.

- **Hue (Tono):** El color base (rojo, verde, azul, etc.), en un rango de 0° a 360°.
- **Saturation (Saturación):** Intensidad o pureza del color (0 a 1).
- **Value (Valor):** Brillo del color (0 a 1).

HSL: :Similar a HSV, pero en lugar de "Value" usa "Lightness" (luminosidad), que es el promedio del brillo del color.

- **Hue (Tono):** El color base (0° a 360°).
- **Saturation (Saturación):** Intensidad del color (0 a 1).
- **Lightness (Luminosidad):** Qué tan claro u oscuro es el color (0 a 1).

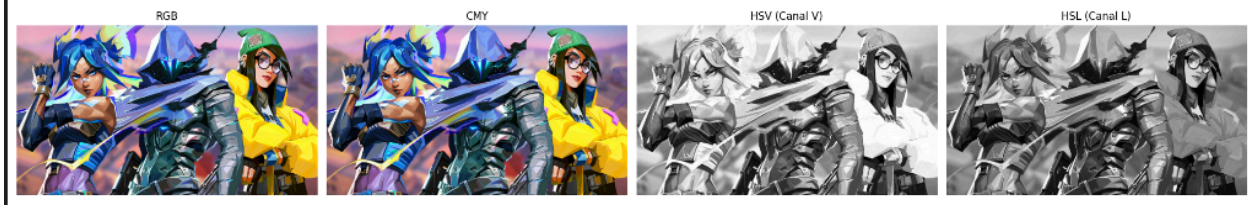
```
from PIL import Image

import numpy as np

import matplotlib.pyplot as plt

import colorsys



# Función para convertir de RGB a CMY

def rgb_to_cmy(image_array):

   return 1 - image_array / 255.0



# Función para convertir de RGB a HSV

def rgb_to_hsv(image_array):

   hsv_image = np.empty_like(image_array, dtype=float)

   for i in range(image_array.shape[0]):

       for j in range(image_array.shape[1]):

           r, g, b = image_array[i, j] / 255.0

           h, s, v = colorsys.rgb_to_hsv(r, g, b)
```

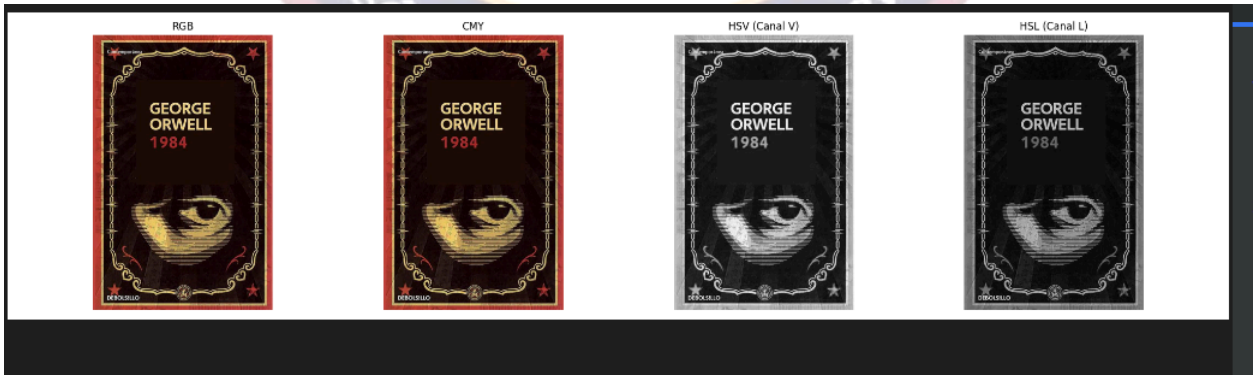
```
            hsv_image[i, j] = [h, s, v]

    return hsv_image


# Función para convertir de RGB a HSL

def rgb_to_hsl(image_array):

    hsl_image = np.empty_like(image_array, dtype=float)

    for i in range(image_array.shape[0]):

        for j in range(image_array.shape[1]):

            r, g, b = image_array[i, j] / 255.0

            h, l, s = colorsys.rgb_to_hls(r, g, b)  # Nota: HLS en colorsys es
equivalente a HSL

            hsl_image[i, j] = [h, s, l]

    return hsl_image


# Función para mostrar una imagen junto con sus modelos de color

def display_color_models(image_path):

    # Cargar la imagen

    image = Image.open(image_path).convert("RGB")

    image_array = np.array(image)


    # Convertir la imagen a los modelos de color

    cmy_image = rgb_to_cmy(image_array)

    hsv_image = rgb_to_hsv(image_array)

    hsl_image = rgb_to_hsl(image_array)


    # Crear un gráfico para mostrar las imágenes
```

```
    fig, axs = plt.subplots(1, 4, figsize=(20, 5))

    # Mostrar la imagen original (RGB)
    axs[0].imshow(image)
    axs[0].set_title("RGB")
    axs[0].axis("off")

    # Mostrar la imagen en CMY
    axs[1].imshow(1 - cmy_image)  # Volver a CMY para mostrar correctamente
    axs[1].set_title("CMY")
    axs[1].axis("off")

    # Mostrar la imagen en HSV (usando el canal V para visualización)
    axs[2].imshow(hsv_image[:, :, 2], cmap="gray")
    axs[2].set_title("HSV (Canal V)")
    axs[2].axis("off")

    # Mostrar la imagen en HSL (usando el canal L para visualización)
    axs[3].imshow(hsl_image[:, :, 2], cmap="gray")
    axs[3].set_title("HSL (Canal L)")
    axs[3].axis("off")

    plt.tight_layout()
    plt.show()

# Lista de rutas de las imágenes
```

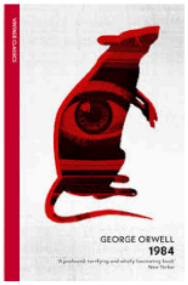
```
image_paths = [
    "3agent.jpg",
    "1984.jpg",
    "godot.jpeg",
    "ILOVEBB.jpg",
    "sova.jpg"
]


# Aplicar la conversión a cada imagen
for path in image_paths:
    display_color_models(path)
```

RESULTADOS





RGB



CMY

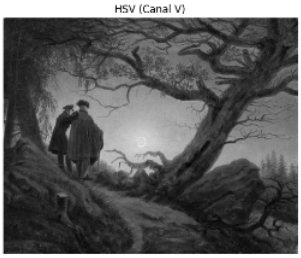
HSV (Canal V)

HSL (Canal L)

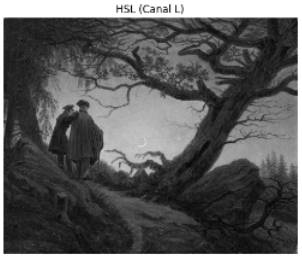
RGB

CMY

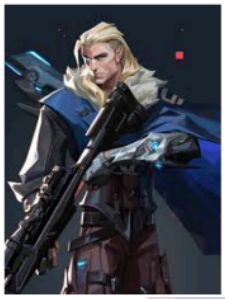
HSV (Canal V)



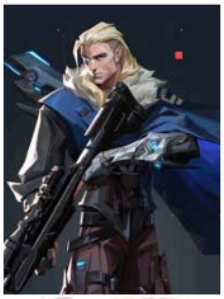
HSL (Canal L)



RGB



CMY



HSV (Canal V)

HSL (Canal L)