Herramientas computacionales: el arte de la programación

Color Match
Herramienta de Relaciones de Color

Rodrigo Quiñonez
Mariel López

Complementary

Analogous

Triadic

Problemáticas a solucionar

Aplicaciones

- Problemática de Accesibilidad:

Muchas interfaces usan combinaciones de color con bajo contraste o difíciles de distinguir para personas con daltonismo o baja visión, limitando su acceso a la información.

Generación de colores complementarios y contrastantes de forma automática puede ayudar a diseñadores a crear interfaces más inclusivas y fáciles de leer.

- Errores en el Diseño de Señalizaciones:

Colores mal elegidos en señalizaciones pueden ser difíciles de distinguir en emergencias.

Ayuda a elegir combinaciones claras y legibles para mejorar la seguridad y la orientación en espacios públicos.

Implementación de la respuesta

1. **Imagen es ingresada**
2. **Lectura y conversión de la imagen** BGR → RGB, RGB → HSV
3. **Cálculo de color promedio**
4. **Definición de rango de color**
5. **Creación de máscara**
6. **Extracción de píxeles del color detectado**
7. **Conversión del color a HEX**
8. **Generación de colores complementario, análogos y triádicos**
9. **Visualización de resultados**

Filtros/ librerías

- OpenCV (cv2)
- NumPy (np)
- Matplotlib (plt)
- colorsys
- Google Colab files.upload()

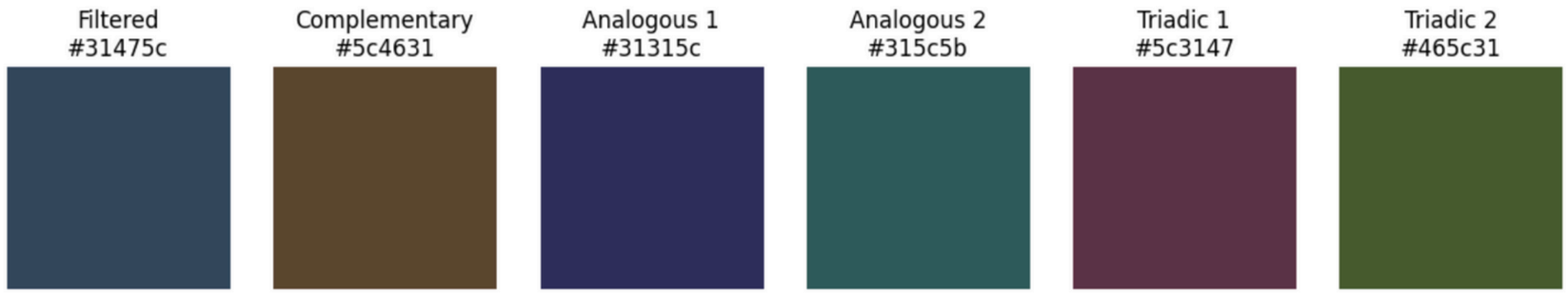
- Conversión de la imagen a HSV para separar tono, saturación y valor.
- Creación de una máscara (cv2.inRange) para filtrar píxeles cercanos al color dominante de la imagen.
- Aplicación de bitwise_and para aislar los píxeles filtrados y calcular el color promedio.

Archivo de entrada/ salida

```
[ ]
```

```
    ana1_h = (h + 0.0833) % 1.0
    ana2_h = (h - 0.0833) % 1.0
    matches["Analogous 1"] = tuple(int(c*255) for c in colorsys.hsv_to_rgb(ana1_h, s, v))
    matches["Analogous 2"] = tuple(int(c*255) for c in colorsys.hsv_to_rgb(ana2_h, s, v))

    tri1_h = (h + 0.3333) % 1.0
    tri2_h = (h - 0.3333) % 1.0
    matches["Triadic 1"] = tuple(int(c*255) for c in colorsys.hsv_to_rgb(tri1_h, s, v))
    matches["Triadic 2"] = tuple(int(c*255) for c in colorsys.hsv_to_rgb(tri2_h, s, v))

    return matches

matching_colors = generate_matching_colors(dominant_color)

fig, ax = plt.subplots(1, len(matching_colors)+1, figsize=(15, 4))

ax[0].imshow(np.ones((100,100,3), dtype=np.uint8)*dominant_color)
ax[0].set_title(f"Filtered\n{rgb_to_hex(dominant_color)}")
ax[0].axis("off")

for i, (name, color) in enumerate(matching_colors.items(), start=1):
    ax[i].imshow(np.ones((100,100,3), dtype=np.uint8)*color)
    ax[i].set_title(f"{name}\n{rgb_to_hex(color)}")
    ax[i].axis("off")

plt.show()
```

```
Elegir archivos  Sin archivos seleccionados Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Blue_Ex.png to Blue_Ex.png
```



Conclusión

Este proyecto demuestra cómo el procesamiento de imágenes y la generación automática de paletas de colores pueden ser herramientas prácticas para resolver problemas reales de diseño y accesibilidad. Al identificar el color ingresado y calcular y sugerir combinaciones complementarias, análogas y triádicas, se facilita la creación de interfaces inclusivas y señalizaciones claras. Esto no solo mejora la legibilidad y la experiencia del usuario, sino que también contribuye a entornos más seguros y accesibles para todas las personas, incluyendo aquellas con daltonismo o baja visión.

#F0D3BA

#CEB07D

#C49F65

#B68E52

#CEBCC6

#DECBCD

Gracias

#6B8298

#849DB6

#8F8163

#8F8163

#3C220F

#4A2F18