HSV

Tomas Lopez Pérez

June 2024

1 Introduction

El espacio de color HSV (Hue, Saturation, Value) es un modelo alternativo al espacio de color RGB (Red, Green, Blue) ampliamente utilizado en el procesamiento de imágenes y gráficos por computadora. A diferencia de RGB, que describe los colores en términos de la intensidad de las componentes primarias de luz, HSV se centra en cómo percibimos y categorizamos los colores en la vida cotidiana. Importancia del Espacio de Color HSV El modelo HSV es especialmente útil en aplicaciones donde la interpretación intuitiva del color es crucial. Por ejemplo, en el análisis de imágenes médicas, la detección de objetos basada en el color y la segmentación de regiones de interés pueden beneficiarse de las propiedades descriptivas de HSV.

2 Fundamento Teorico

El modelo de color HSV es una transformación no lineal del modelo RGB en coordenadas cilíndricas, donde cada color se define por las siguientes dimensiones:

- **Tinte o Matiz (Hue):** Es un ángulo que representa el matiz del color, típicamente definido entre 0° y 360°.
- **Saturación (Saturation):** Es el nivel de saturación del color, expresado entre 0 y 1. Un valor de 0 indica ausencia de saturación (blanco puro), mientras que 1 representa el matiz en su máxima saturación. También se puede expresar en porcentaje de 0
- **Brillo (Value o Brightness):** Indica el nivel de brillo del color, variando de 0 a 1. Un valor de 0 representa negro puro, y 1 representa blanco puro. Al igual que la saturación, se puede expresar en porcentaje de 0

3 Metodologia

La metodologia propuesta en este trabajo constituye en la implementacion de un programa que permita realizar la segmentacion por medio del espacio de

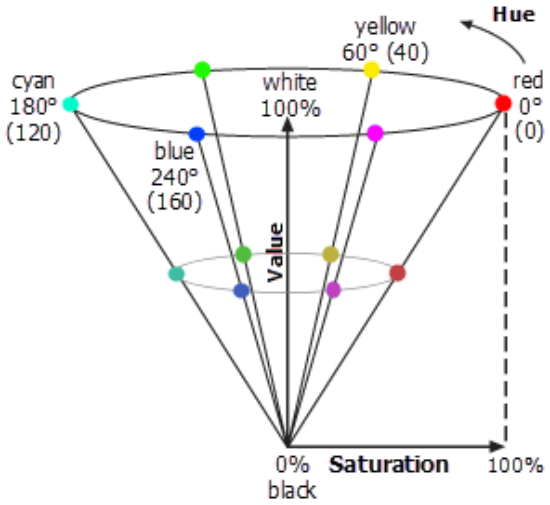

Figure 1: Modelo HSV

color HSv. Para realizar esto, se utilizo la herramienta OpenCV para el manejo de la camara, junto con la libreria PyQt5 para poder acomodar las capturas de el video. 3

Se implementa el uso de los events, para que al dar click al la imagen de la camara 1 obtenga la posicion del pixel y extraer el color RGB para posteriormente convertirlo a HSV y mostrar en la imagen resultante, cabe mencionar que se definen valores estaticos de lower blue y upper blue. 4

4 Resultados

En la pruebas del la segmentacion se realizo los click en diferentes partes de la imagen con en la pared del fondo, la cara del sujeto, una chammara. 6 8 7

5 Conclusiones

La implementación del espacio de color HSV (Hue, Saturation, Value) en aplicaciones de procesamiento de imágenes y visión por computadora ofrece varias ventajas significativas sobre el uso del espacio de color RGB. Este modelo de color es especialmente útil para tareas que requieren la separación del matiz de un color de su brillo e intensidad, lo cual es una necesidad común en diversas aplicaciones prácticas como la detección y seguimiento de objetos, la mejora de imágenes y el reconocimiento de patrones.

```
class MainWindow(QWidget):
    lower_blue = np.array([50, 50, 50])
    upper_blue = np.array([130, 255, 255])

    tabnine: test | explain | document | ask
    def __init__(self):
        super().__init__()       You, last week • update

        self.video_label = QLabel()
        self.mask_label = QLabel()
        self.res_label = QLabel()
        self.initUI()
        self.cap = cv.VideoCapture(1)
        self.timer = QTimer(self)
        self.timer.timeout.connect(self.update_frame)
        self.timer.start(10)
        self.video_label.mousePressEvent = self.mouse_click_event
        self.lower_blue = np.array([0, 0, 0])
        self.upper_blue = np.array([179, 255, 255])
```

Figure 2: Inicicializacion de la clase MainWindow

```
tabnine: test | explain | document | ask
def initUI(self):
    layout = QHBoxLayout()
    layout.addWidget(self.video_label)
    layout.addWidget(self.mask_label)
    layout.addWidget(self.res_label)
    self.setLayout(layout)
    self.setWindowTitle('Dale click sobre un color en el video')
```

Figure 3: camaras acomodadas

```
tabnine: test | explain | document | ask
def mouse_click_event(self, event):
    x = event.pos().x()
    y = event.pos().y()
    ret, frame = self.cap.read()
    if ret:
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        color_rgb = frame_rgb[y, x]
        color_rgb_reshaped = np.uint8([[color_rgb]])
        color_hsv = cv2.cvtColor(color_rgb_reshaped, cv2.COLOR_RGB2HSV)[0][0]
        margin = np.array([10, 50, 50])
        self.lower_blue = np.maximum(color_hsv - margin, [0, 0, 0])
        self.upper_blue = np.minimum(color_hsv + margin, [179, 255, 255])
        self.update_frame()
        print(f"Color RGB en coordenadas ({x}, {y}): {color_rgb} : Color HSV {color_hsv}")
```

Figure 4: Event Onclick

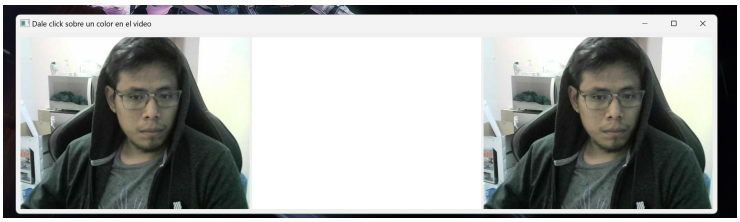

Figure 5: Imagen original

Figure 6: Clikc en la pared del fondo

Figure 7: Click en la chamarra

Figure 8: Click en la cara