Card19_HSV_com_Opencv

Davi Bezerra Barros

O Vídeo fala sobre como detectar cores em vídeos usando a biblioteca OpenCV(Open Source Computer Vision Library), utilizada para desenvolvimento de softwares de visão computacional. As etapas realizadas são as seguintes:

1. Configuração inicial:

Inicializa o stream de dados da câmera e loop para exibir seus frames:

```
cap = cv2.VideoCapture(0) #Abre a câmera para captura de vídeo;

while True:
    _, frame = cap.read() # Lê e retorna um frame da captura de vídeo;
    cv2.imshow("frame", frame ) # Exibe a imagem em uma janela;
    key = cv2.waitKey(1) #Delay de 1ms para um evento de tecla
    if key == 27:
        break
```

2. Convertendo o sistema de cores:

O sistema de cores nativo do OpenCV é o RGB. Para este projeto foi utilizado o sistema **HSV**:

- **H(Hue):** Define a tonalidade da cor, diferentes valores representam diferentes cores .
- **S(Saturation):** Define a densidade da cor.
- **V(Value):** Define o brilho da cor.

```
hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
# converte uma imagem de um espaço de cores para o outro
# frame: Imagem de entrada
# v2.COLOR_BGR2HSV) código de conversão da cor
```

3. Definindo intervalo para as cores

Para cada cor específica há um intervalo de valores que define seu *hue*, *Saturation* e *value*:

```
#Intervalo para o vermelho:
low_red = np.array([161, 155, 84])
high_red = np.array([179, 255, 255])

#Intervalo para o azul
low_blue = np.array([94, 80, 2])
high_blue = np.array([126, 255, 255])

#Intervalo para o verde:
low_green = np.array([25, 52, 7])
high_green = np.array([102, 255, 255])
```

O OpenCv utiliza numpy para converter a lista em um array.

4. Criação de máscaras

As máscaras isolam os pixels correspondentes ao intervalo de uma cor espcífica, definido anteriormente.

```
red_mask = cv2.inRange(hsv_frame, low_red, high_red)
blue_mask = cv2.inRange(hsv_frame, low_blue, high_blue)
green_mask = cv2.inRange(hsv_frame, low_green, high_green)
# hsv_frame: Imaggem de entrada em HSV
# low_red/gblue/green: Limite inferior da cor, em array
# high_red/gblue/green: Limite superior da cor, em array
```

Os pixels dentro do intervalo de cores recebem valor 1(branco), e os pixles fora recebem 0(preto).

5. Filtrando objetos

Com a criação das máscaras, agora é possível aplicá-las em objetos nos frames capturados pela câmera:

```
red = cv2.bitwise_and(frame, frame, mask=red_mask)
blue = cv2.bitwise_and(frame, frame, mask=blue_mask)
green = cv2.bitwise_and(frame, frame, mask=green_mask)
# frame(esquerda): Primeira imagem
# frame(direita): A mesma imagem
# mask=red/greeen/blue_mask: a mascara binária definida anteriormente
```

Isso é feito utilizando a função **Bitwise_and** que executa uma operação *AND* bit a bit entre os pixels da imagem e a máscara. Onde a máscara é 1, o valor da imagem original é mantido e onde é 0, valor é removido, criando uma imagem preto e branco.

Detectar todas as cores exceto o branco:
Ajustando os valores de saturação da máscara, é possivel exibir todas as cores exceto o branco, pois os valores mais baixos de saturação são evitados, excluindo as cores que mais se aproximam do branco.

```
low = np.array([0, 42, 0])
high = np.array([179, 255, 255])
mask = cv2.inRange(hsv_frame, low, high)
```

Encerramento:
Para encerrar o loop, basta pressionar a tecla ESC:

```
cap.release() # libera a câmera para outros processos
cv2.destroyAllWindows() # fecha todas as janelas abertas pelo OpenCV
```