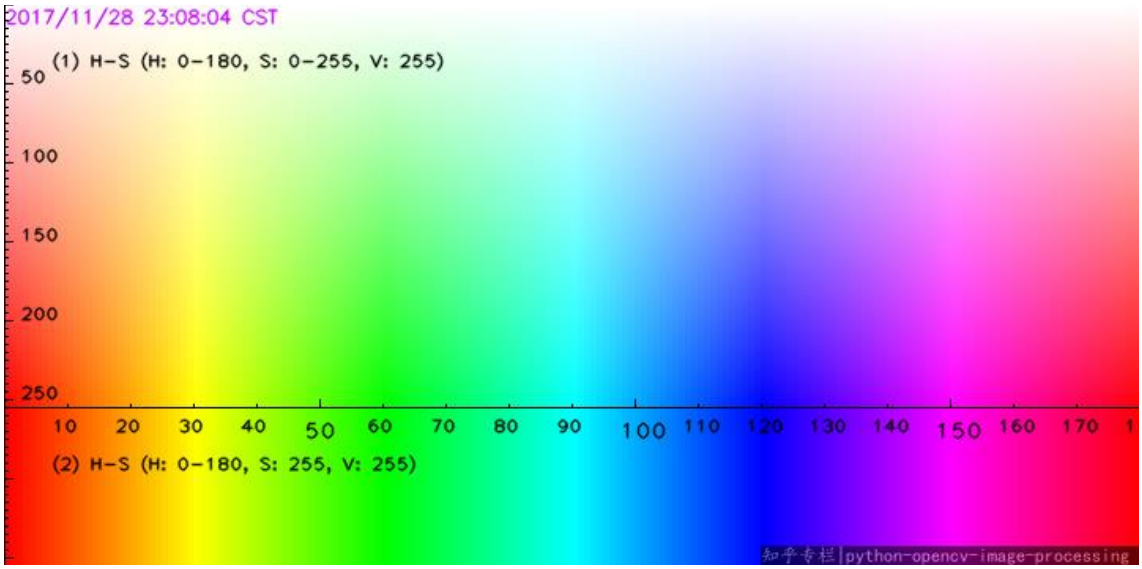
Reconocimiento de colores con openCV y Python

La visión artificial es una disciplina científica que incluye métodos para adquirir, procesar, analizar y comprender las imágenes del mundo real, en base a este concepto se ha puesto en práctica la utilización de la librería openCV con el lenguaje Python como base para programar, la librería nos ofrece múltiples usos pero en esta ocasión se ha tomado el módulo de reconocimiento de colores.

La composición de colores se toma a partir de sus 3 primarios que son azul, rojo y amarillo estos tienen diferentes tonalidades según sus capas y ambientes ya sea luz, opacidad o incluso por medio de la cama que se les este visualizando, esto quiere decir que rara vez se podrá obtener una tonalidad pura del color deseado y es por ello que se a tomado rangos que van desde lo mas oscuro del color hasta lo mas claro esto con el fin de encontrar la tonalidad mas cercana al color que se desea detectar, para ello tenemos que saber desde donde a donde se tomaría los valores de los colores como se muestra en la imagen.



Con la referencia de la imagen podemos obtener los valores de los colores que deseamos detectar, los mismos se componen de tres capas que anteriormente se mencionó (azul, rojo y amarillo) cada uno define su tonalidad creando el deseado.

Lo primero que se necesita para crear un reconocimiento de colores con la librería openCV es una imagen o fotograma para su detección, la librería por defecto lee las imágenes o fotogramas en BGR, por ello es necesario hacer una trasformación al espacio de color HVS, para ello nos ofrece una función que es “cv2.ctvColor” que como primer argumento pasaremos “cv2.Color_BGR2HVS” esto indica que trasformaremos BGR a HVS.

A partir de la imagen antes mencionada nos ayudaremos a determinar los colores que necesitamos como por ejemplo H va de 0 a 179 y por este rango pasa colores rojos, naranja, amarillo, verde, azul, violeta entre otros, se determinada dos rangos para el color rojo como ejemplo para este tomamos “0, 100, 20”-“8, 255, 255” después de definir los rangos se procederá a crear caras o también llamadas mascaras " cara1 = cv2.inRange(hvs, azul_osc, azul_cla)" seguido a esto procedemos a guardar en una constaten definiendo de la siguiente manera

```
cnst1 = cv2.findContours(cara1, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

cnst1 = imutils.grab_contours(cnst1), con esto lo que hacemos es una comparación entre los rangos declarados anteriormente para encontrar el más aproximado, como últimos pasos declaramos un bucle for y le damos el siguiente comportamiento

```
  for c in cnst1:
     area1 = cv2.contourArea(c)
     if area1>5000:
        cv2.drawContours(frame, [c], 0, (255, 0, 0), 3)
        M = cv2.moments(c)
        cx = int(M["m10"]/M["m00"])
        cy = int(M["m01"] / M["m00"])

        cv2.circle(frame, (cx, cy), 7, (0, 255, 0), -1)
        cv2.putText(frame, "azul oscuro", (cx-20, cy-20), cv2.FONT_ITALIC, 2, (255, 255, 255), 2)

        cursorInsert = conexionc.con.conexion.cursor()
        consulta = "INSERT INTO dbo.coloresDatos(clr_name) VALUES ('azul oscuro');"
        cursorInsert.execute(consulta)
        cursorInsert.commit()
        cursorInsert.close()
```

En el for colocamos una variable “c” y la contante “cnst1” después el área le damos igualdad a “cv2.contoursArea(c)” para contornear el objeto, declaramos un if y colocamos el área mayo a 500 indicando que localice el objeto a cierta distancia, con “cv2.drawContours(frame, [c], 0, (255, 0, 0), 3)” se dibujara un contorno, aplicamos una fórmula “M = cv2.moments(c) cx = int(M["m10"]/M["m00"]) cy = int(M["m01"] / M["m00"])” esta nos ayuda a encontrar en punto central del objeto consiguiendo una mejor captura de la misma, ubicamos un texto para saber que color es como un identificador con la siguiente función “cv2.circle(frame, (cx, cy), 7, (0, 255, 0), -1) cv2.putText(frame, "azul oscuro", (cx-20, cy-20), cv2.FONT_ITALIC, 2, (255, 255,255), 2)”, por penúltimo paso procedemos a guardar el nombre de los colores en una base de datos de la siguiente manera:

```
cursorInsert = conexionc.con.conexion.cursor()
 consulta = "INSERT INTO dbo.coloresDatos(clr_name) VALUES ('azul oscuro');"
 cursorInsert.execute(consulta)
cursorInsert.commit()
cursorInsert.close()
```

y como último paso procedemos a cerrar la vista y la ventana creado para visualizar la cámara.

```
cap.release()
cv2.destroyAllWindows()
```