

**Proyecto Final: Seguimiento de un Objeto con Python**

Asignatura en Robótica Industrial

*Facultad de Ingeniería*

*Michel López Franco*

Paola Guadalupe Paniagua Camacho

Brunella Pamela Merino Luque

Yolanda Stephania Lara Cervantes

Lunes 25 de mayo del 2020

**DESCRIPCIÓN DEL PROYECTO**

Este proyecto consta de la programación de un sistema de seguimiento de un objeto usando el lenguaje de python. Este programa funciona usando la cámara de la computadora y se programa para que detecte los rangos de las tonalidades de un color RGB.

Al detectar el tono deseado que previamente se le indicó por medio de una imagen de muestra del color, hace una máscara para extraer todo el color del video y pasarlo a una escala de grises, en donde el color deseado o en este caso, el objeto del color deseado, toma el color blanco y todo lo demás se torna negro.

A partir de esa máscara se programa una silueta o recuadro que seguirá en todo momento al objeto, de esta manera cuando movemos el objeto de un lado a otro dentro del rango visual de la cámara, un recuadro enmarcará el objeto en todo momento para indicar que ahí se encuentra.

**Código**

#Librerías

pip install opencv-python #OpenCV

import cv2 #OpenCV

import numpy as np

#Imagen

img = cv2.imread("Prueba.jpg") #Cargar imagen.

**#Syntax: cv2.imread(path, flag)**

#flag: Forma en que se debe leer la imagen. (opcional)

#cv2.IMREAD\_COLOR (Predeterminado): Escala a color. (1)

#cv2.IMREAD\_GRAYSCALE: Escala de grises. (0)

#cv2.IMREAD\_UNCHANGED: Composición alfa (opacidad de un pixel) (-1)

roi = img[252:395, 354: 455] #Área de interés en la imagen (píxeles).

#Variables y Constantes

x = 354

y = 252

width = 455-x

height = 395-y

# Conversión de imagen

hsv\_roi = cv2.cvtColor(roi, cv2.COLOR\_BGR2HSV) #Convertir imagen de un espacio de color a otro

**#Syntax: cv2.cvtColor(src, code[, dst[, dstCn]])**

#src: Imagen a cambiar.

#code: Código de conversión del espacio de color.

#Hay más de 150 métodos de conversión de espacio de color disponibles en OpenCV.

#dst: Imagen de destino. (mismo tamaño y profundidad que src) (opcional)

#dstCn: Número de canales en la imagen de destino. (opcional)

#Histograma: Gráfico o diagrama relacionado con la frecuencia de píxeles en una imagen en escala de grises con valores de píxeles (que van de 0 a 255).

roi\_hist = cv2.calcHist([hsv\_roi], [0], None, [180], [0, 180]) #Encontrar histograma de la imagen.

**#Syntax: cv2.calcHist (imágenes, canales, máscara, histSize, rangos)**

#imágenes: Imagen. (tipo int o float)

#canales: Índice del canal para el que calculamos el histograma.

#[0]: Escala de grises.

#[0], [1] o [2]: Canal azul, verde o rojo, respectivamente.

#máscara: Máscara de imagen. (Para encontrar el histograma de imagen completa, se proporciona como "None")

#histSize: Representa recuento de BIN. (escala completa [256])

#rangos: Rango. (normalmente es [0, 256])

#Encender camara web.

cap = cv2.VideoCapture(0) #Reproducir vídeo.

#cv2.VideoCapture (0): Primera cámara o cámara web.

#cv2.VideoCapture (1): Segunda cámara o cámara web.

#cv2.VideoCapture ("nombre de archivo.mp4"): Archivo de video.

term\_criteria = (cv2.TERM\_CRITERIA\_EPS | cv2.TERM\_CRITERIA\_COUNT, 10, 1)

while True:

\_,frame= cap.read() #vídeo

print(frame.shape)

hsv= cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

mask= cv2.calcBackProject([hsv], [0], roi\_hist, [0, 180], 1)

ret, track\_window = cv2.CamShift(mask, (x, y, width, height), term\_criteria)

pts = cv2.boxPoints(ret)

pts = np.int0(pts)

cv2.polylines(frame, [pts], True, (0, 0, 0), 2) #GENERA RECUADRO

cv2.imshow("mask", mask) #cv2.imshow(window\_name, image)

cv2.imshow("Frame", frame) #cv2.imshow(window\_name, image)

key = cv2.waitKey(1)

if key == 27:

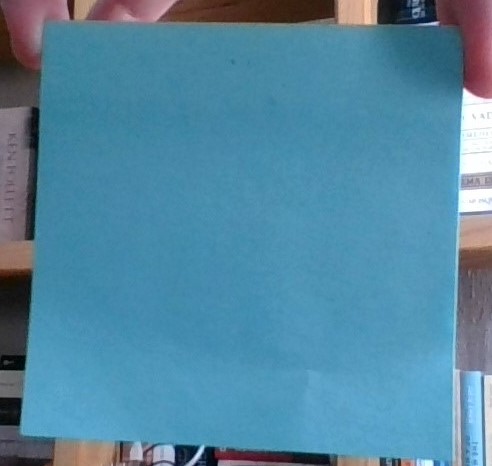
break

cap.release()

cv2.destroyAllWindows()

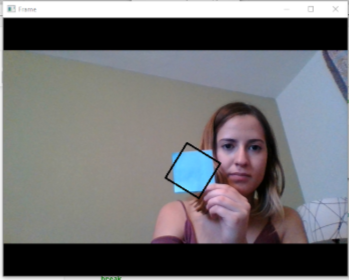
**Parámetros y Resultados**

El objeto que vamos a seguir es un post- it de tono azul claro. La imagen de muestra que se le cargó al programa es la siguiente:



El resultado obtenido es el esperado, al mover el objeto por el rango visual de la cámara, hace el seguimiento adecuado a éste como se muestra en la siguiente imagen:





**CONCLUSIÓN**

Este proyecto nos permite conocer una de las tantas aplicaciones útiles de python que es la de rastrear, en este caso es un color ya que la imagen extrae la tonalidad que se le asigna, sin embargo, esto tiene un alcance sumamente destacable puesto que es solo la introducción a temas de seguimiento e identificación, ya sea de objetos simples o incluso de rostros más detallados.

A pesar de que no tuvimos que hacer el código desde cero y que nos ayudamos mucho de comandos pre-definidos y una librería pre-establecida (OpenCV) la cual ayuda con este tipo de prácticas ya que se especializa en funciones y comandos de visión de la computadora en tiempo real, es decir, nos permite usar la WebCam e identificar objetos que se encuentren delante de ella en ese momento.

Todo esto es vital entender si queremos darle un mayor alcance a este proyecto lo cual es el objetivo de todo, seguir indagando y ampliar nuestro repertorio de herramientas para en un futuro, estar mejor preparadas para cualquier obstáculo que se nos presente.