Universidad del Valle de Guatemala
Departamento de Ingeniería Mecatrónica
MT3006 - Robótica 2
MSc. Miguel Zea

Laboratorio 2

# Visión de computadora con OpenCV y MATLAB

## Objetivos

- Emplear la auto-generación de código de las herramientas de Color Thresholder e Image Region Analyzer de MATLAB.
- Combinar thresholding y segmentación para desarrollar una aplicación de detección sencilla.
- Desarrollar destrezas para el desarrollo de aplicaciones de visión de computadora en OpenCV.

## Procedimiento

En la práctica de esta semana usted empleará las herramientas de thresholding y segmentación que presenta MATLAB para desarrollar un detector simple de resistencias de 4.7 kΩ. Luego, repetirá el proceso pero realizando la implementación en Python mediante OpenCV. Para ello, siga los siguientes pasos:

1. Descargue de Canvas el archivo `mt3006lab2.zip` y extraiga sus contenidos dentro de una carpeta en una ubicación de su preferencia. Cambie el folder de trabajo de MATLAB para que coincida con esta carpeta.

2. Las imágenes `res1, res2, res3` contienen una resistencia de 4.7 kΩ en diversas orientaciones y condiciones de luz, mientras que la imagen `res4` contiene una resistencia de 47 kΩ. Emplee estas imágenes junto con la herramienta **Color Thresholder** para generar (el código de) las máscaras que sean necesarias para detectar cada una de las bandas de color de una resistencia de 4.7 kΩ, sin importar la orientación o iluminación. Tome nota de verificar la validez de sus máscaras utilizando la imagen de la resistencia de 47 kΩ, ya que sólo deberían detectarse las bandas con los colores que comparten ambos valores de resistencia. **NOTA: el set de máscaras generadas debe funcionar para *todas* las imágenes, es decir, no pueden generarse sets de máscaras para cada imagen en específico. Por lo mismo trate de emplear un espacio de color que sea robusto ante los cambios de iluminación.**

3. De la aplicación de las máscaras del inciso anterior, usted debería obtener una serie de imágenes binarias (blanco y negro) las cuales ya pueden emplearse para realizar segmentación. Antes de esto, sin embargo, verifique que las regiones detectadas para las bandas de color sean regiones de color blanco sin agujeros. Puede emplear la función `imfill` para rellenar los agujeros.

4. Para cada una de las imágenes binarias, emplee segmentación ya sea mediante la herramienta **Image Region Analyzer** o bien directamente con la función `regionprops`, la cual obtiene las propiedades de cada uno de los *blobs* correspondientes a las bandas de color. Pida a la función (o a la herramienta) que extraiga los centroides de las regiones (dentro de la herramienta puede filtrar por tamaño en caso que todavía se detecten múltiples blobs) y emplee esta información, junto con la información provista por las máscaras (es decir, el color de cada banda de la resistencia) para determinar si la imagen procesada corresponde a una resistencia de 4.7 kΩ o no (puede ser simplemente mediante un mensaje en consola usando `disp`).

5. Verifique que el detector funciona para las tres imágenes de la resistencia de 4.7 kΩ y que falla para la de 47 kΩ. Tome nota que este será el último inciso en donde empleará MATLAB.

6. Replique la aplicación pero ahora implementándola dentro de Python mediante OpenCV. Las siguientes funciones pueden serle de utilidad durante el *port* de la aplicación:

   - `img_hsv = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)`: cambia la imagen del espacio de color RGB al espacio HSV (por ejemplo). La función `cvtColor` cambia la imagen al espacio de color especificado mediante los `cv2.ColorConversionCodes`.
   - `mask = cv2.inRange(img, lower, upper)`: crea una máscara de color para la imagen, empleando los thresholds suministrados. El resultado de esta función es similar a la imagen binaria `BW` que retorna MATLAB mediante el código generado por la herramienta Color Thresholder. **NOTA:** los thresholds `lower` y `upper` deben ser arrays de `numpy` de tamaño igual a los componentes de espacio de color empleado.
   - `res = cv2.bitwise_and(frame,frame, mask= mask)`: retorna la imagen en el espacio de color original luego de haber sido pasada por la máscara. El resultado de esta función es similar a la imagen `maskedRGBImage` que retorna MATLAB mediante el código generado por la herramienta Color Thresholder.
   - OpenCV no posee un reemplazo directo a la función `imfill` de MATLAB pero puede obtenerse un comportamiento idéntico mediante la función `cv2.floodFill`, tal como se describe en el siguiente enlace.
   - OpenCV no posee un reemplazo directo a la función `regionprops` de MATLAB pero puede obtenerse un comportamiento idéntico mediante la función `cv2.findContours`, tal como se describe en los siguientes tutoriales.

Cuando esté satisfecho con sus resultados, presente los mismos al catedrático del laboratorio y responda a las preguntas que se le hagan. Recuerde que entregas tardías representan una penalización del 25 % por semana.