El objetivo de esta práctica es aprender a segmentar de manera correcta una imagen en función de un cierto color. Para la realización de esta práctica se utilizaron un conjunto de librerías como OpenCV o Numpy y paquetes como cv_bridge, esto para poder obtener una imagen y hacerla compatible con los nodos de ROS. OpenCV se encarga de obtener la imagen del sistema, Numpy se encarga de representar imágenes como matrices y cv_bridges se encarga de "traducir" las imágenes de los diversos formatos a ROS.

Todas las librerías previamente mencionadas tienen un conjunto de funciones que nos permitirán, para el propósito de la siguiente página, realizar las siguientes tareas:

- Transformar la imagen del espacio BGR al HSV
- Obtención del rango de color en ciertos pixeles
- Obtención de la posición de la región
- Observar la imagen mostrada en blanco y negro.

Inicialmente, se lanzo el archivo de simulación del robot "robotino" y se ejecuto el nodo de *color_segmentation.py* el cual es el que se encarga de la implementación de todas las librerías y la correspondiente segmentación por color de la imagen del lanzador *robotino simul.launch.*

Principalmente, se modificó la función InRange la cual se encarga de determinar un rango de colores en el espacio HSV para el cual se segmentará una imagen.

La imagen por segmentar en este caso es la siguiente:

*Imagen en el espacio BGR*

En este caso lo que se desea segmentar es la taza color verde del resto de la imagen. El nodo lo que se encarga es de recibir el mensaje de una imagen, la transforma al espacio BGR, esta imagen la convierte al espacio HSV, se realiza la segmentación y se muestran las siguientes imágenes:

*Imagen en el espacio HSV*

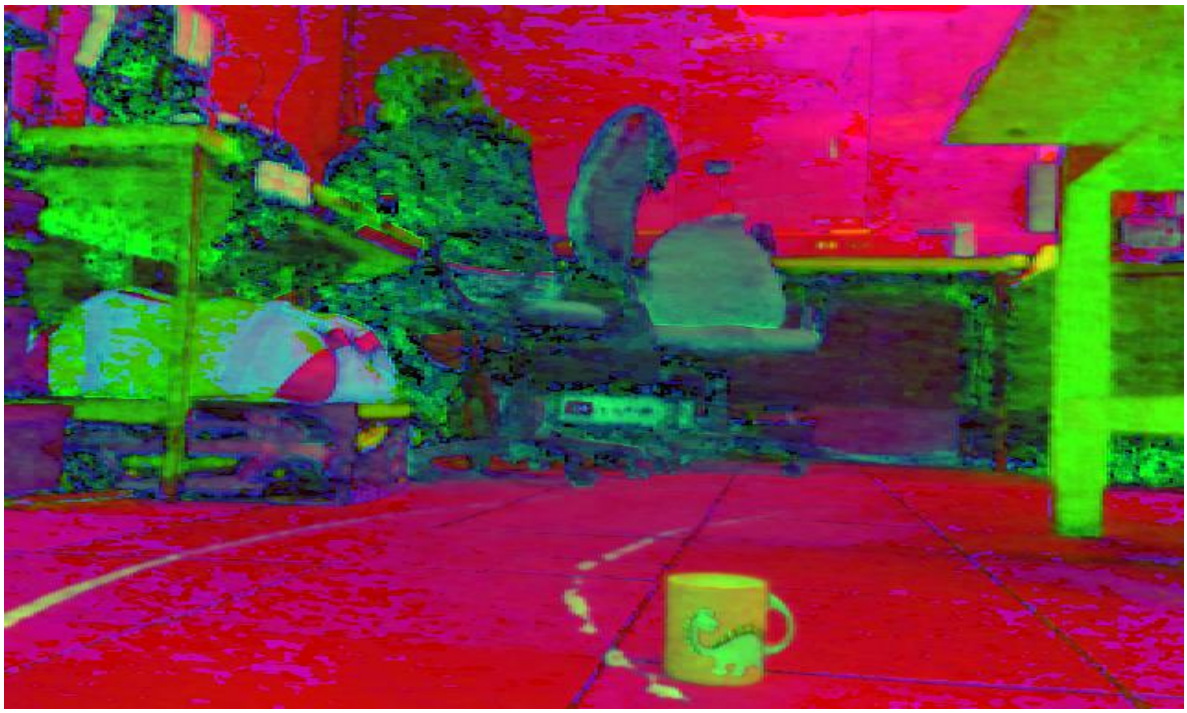

*Imagen segmentada correctamente en blanco y negro*

***Comentarios sobre los resultados obtenidos***

En un comienzo, la segmentación únicamente mostraba la forma del dinosaurio estampando en la taza, esto por el rango que definía el color a segmentar, pero al modificar los valores dentro del arreglo que definía este rango, se logró observar la forma de la taza básicamente es su totalidad.

Los valores que se ubican en el arreglo son los HSV (Hue, Saturation and Value). Al modificar dichos valores en la línea 13 de (60,200,60) a (20,85,70) se logró corregir la segmentación de la imagen. La segmentación por colores utilizando el espacio HSV en lugar de la BGR es mucho más robusta ante los cambios de iluminación de una cierta imagen.

```
1   #!/usr/bin/env python
2
3   import rospy
4   import cv2
5   import cv_bridge
6   import numpy
7   from sensor_msgs.msg import Image
8
9   def callback_rgb(msg):
10      bridge = cv_bridge.CvBridge()
11      img_bgr = bridge.imgmsg_to_cv2(msg, desired_encoding="bgr8")
12      img_hsv = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2HSV)
13      img_bin = cv2.inRange(img_hsv, numpy.array([44,85,70]), numpy.array([80, 255, 255]))
14      idx = cv2.findNonZero(img_bin)
15      [centroid_x, centroid_y, a, b] = cv2.mean(idx)
16      cv2.circle(img_bgr, (int(centroid_x), int(centroid_y)), 20, [0, 255, 0], thickness=3
17      cv2.imshow("Image BGR", img_bgr)
18      cv2.imshow("Image HSV", img_hsv)
19      cv2.imshow("Image Binary", img_bin)
20      cv2.waitKey(1)
21
22  def main():
23      print "INITIALIZING COLOR SEGMENTATION NODE..."
24      rospy.init_node("color_segmentation")
25      rospy.Subscriber("/camera/color/image_raw"    , Image,       callback_rgb )
26      loop = rospy.Rate(30)
27      while not rospy.is_shutdown():
28          loop.sleep()
29
30  if __name__ == "__main__":
31      main()
32
```

***Comentarios sobre los problemas encontrados***

Cabe destacar que la segmentación por colores de esta manera es mucho más complicada de lo que parece, esto se puede observar claramente en la imagen segmentada, ya que aunque el contorno de la taza es claro, se siguen notando partes del dinosaurio que no se encuentran en el rango definido y otras piezas de la imagen que cumplen con el rango también se observan como manchas blancas, esto dependiendo de la aplicación podría ser algo muy poco conveniente.

Si se atravesara un objeto que cumple con la misma paleta de colores definida también sería identificada por el nodo y eso supondría probablemente otro tipo de problemas. Para eliminar estos casos es necesario un procesamiento más especializado que podría conllevar el uso de más mascaras para eliminar este error.