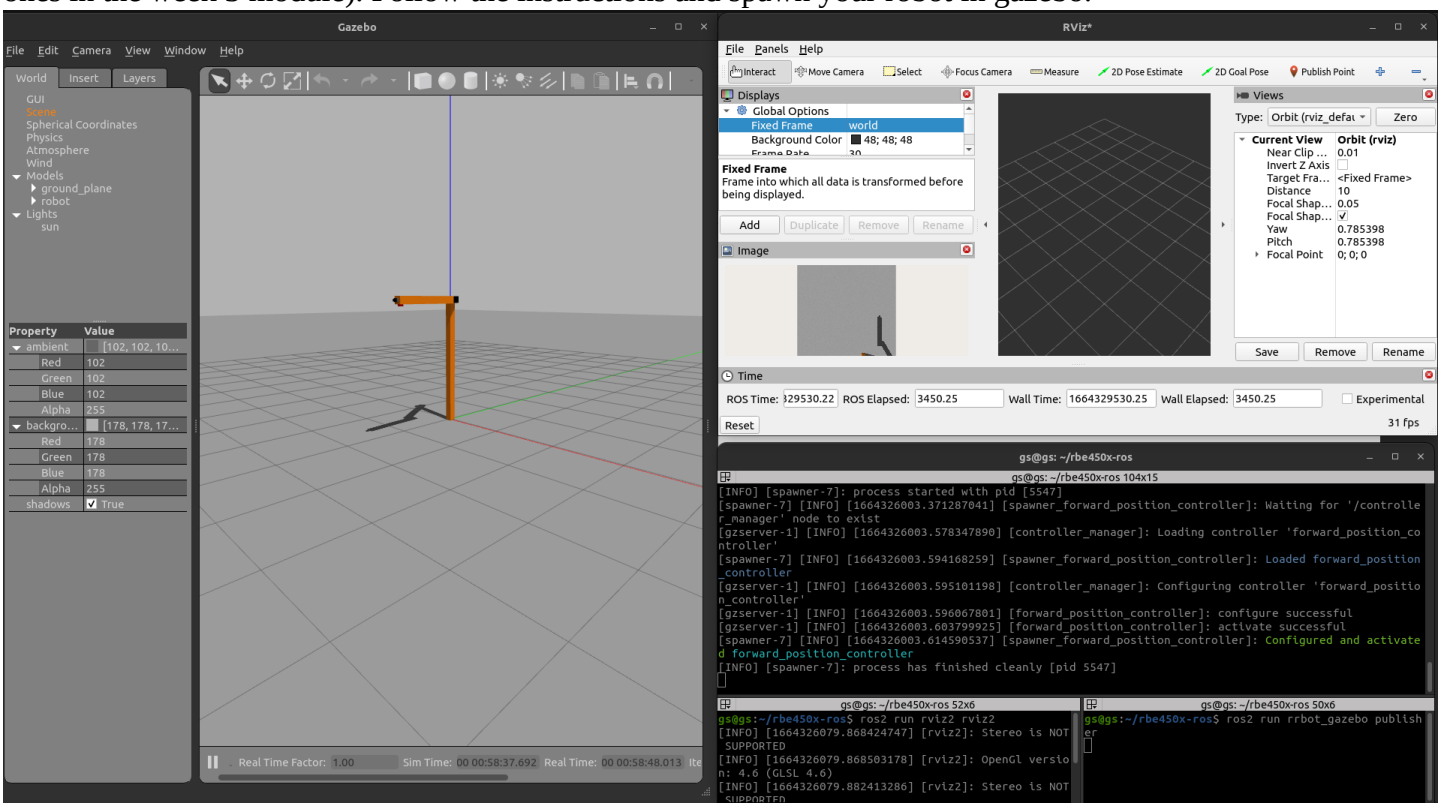
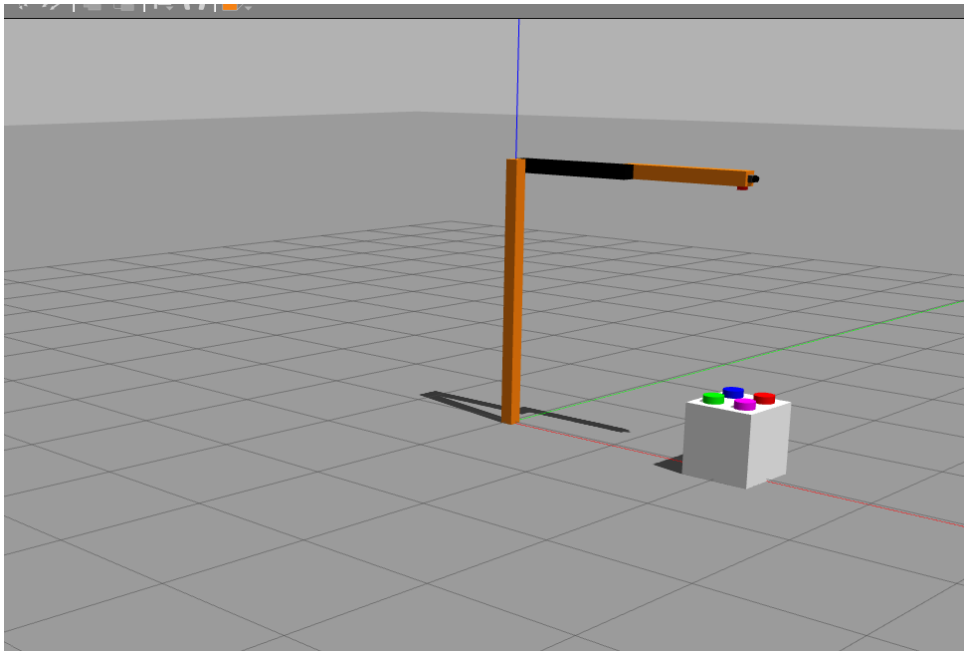
Gokul Srinivasan 500594698

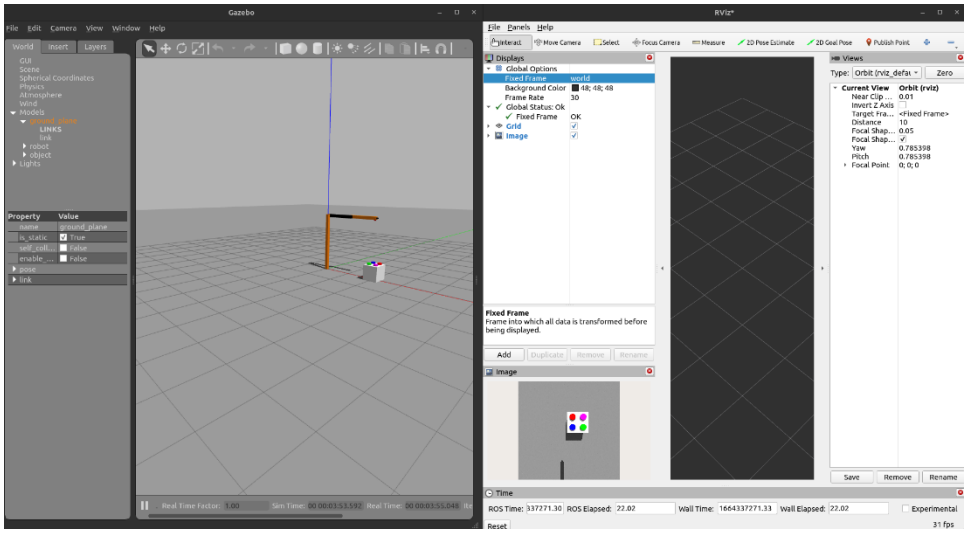
RBE 450X - Vision Based Manipulation
HW - 03

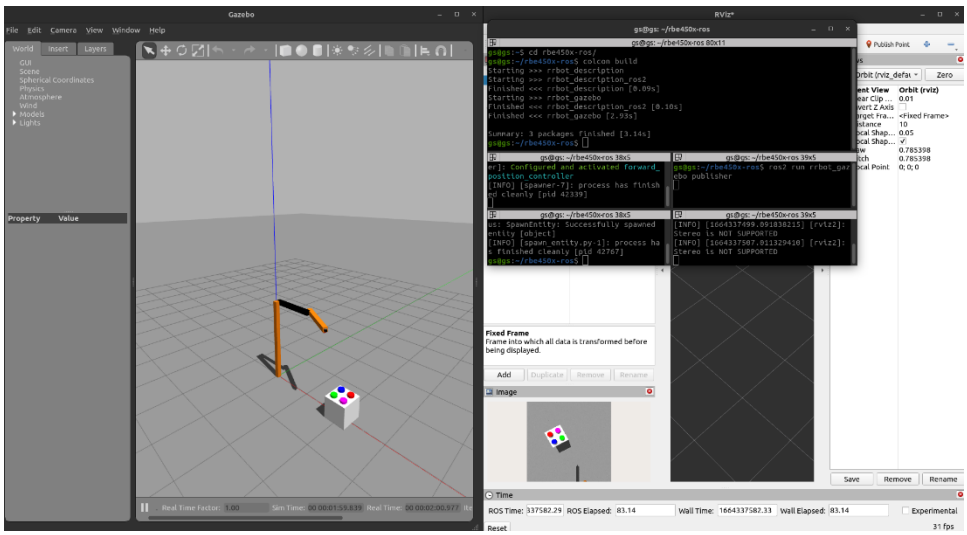
Step 1 (1.5 pts): Download the gazebo robot package from the week 4 module in Canvas (be careful, not the ones in the week 3 module). Follow the instructions and spawn your robot in gazebo.

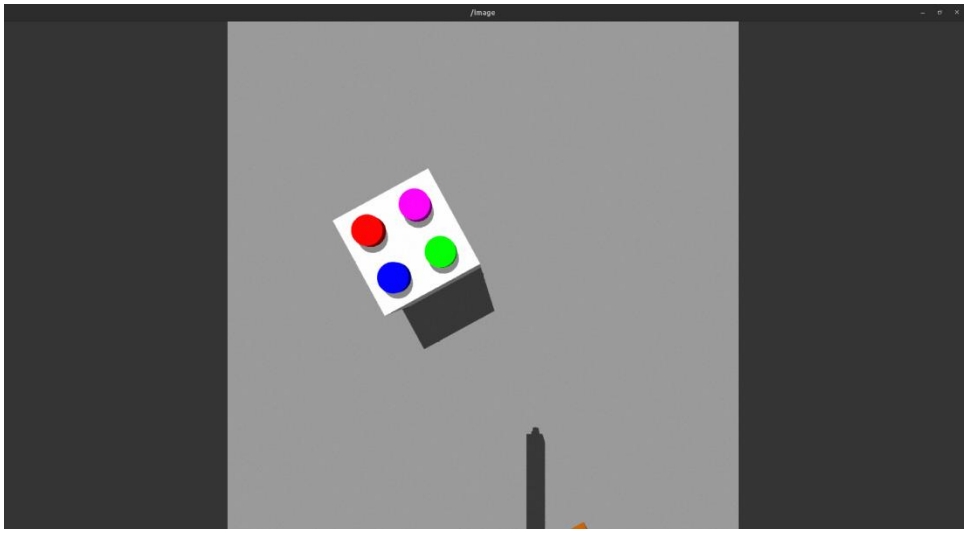


Step 2 (4 pts): From Canvas, download the URDF file of the plate that has four colors as below:

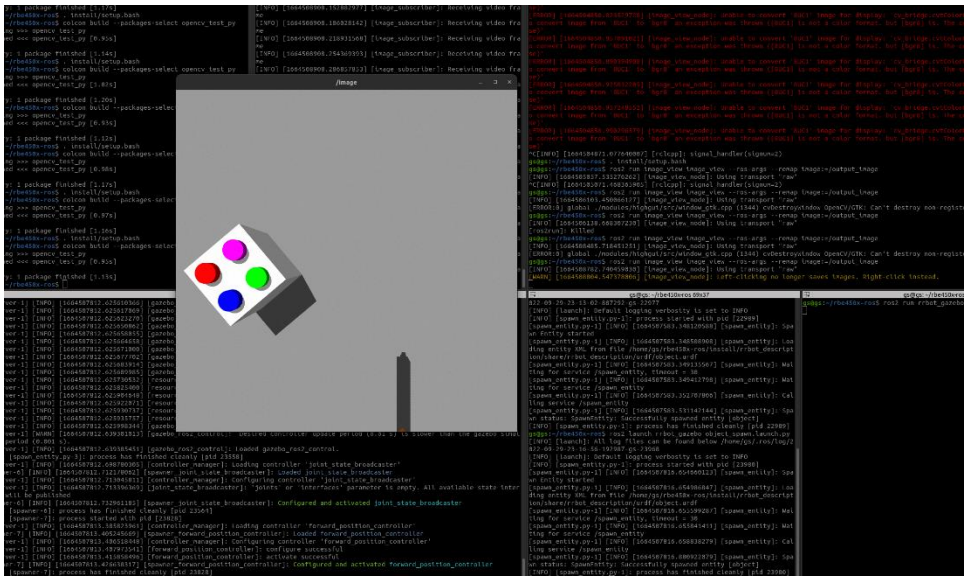



Gazebo
RViz*
File Edit Camera View Window Help
File Panels Help
Displays
Global Options
Fixed Frame
Image
Fixed Frame
Frame into which all data is transformed before being displayed.
Time
ROS Time: 337271.30 ROS Elapsed: 22.02 Wall Time: 1664337271.33 Wall Elapsed: 22.02
Reset
31 fps


Gazebo
RViz*
Fixed Frame
Frame into which all data is transformed before being displayed.
Time
ROS Time: 337582.29 ROS Elapsed: 83.14 Wall Time: 1664337582.33 Wall Elapsed: 83.14
Reset
31 fps

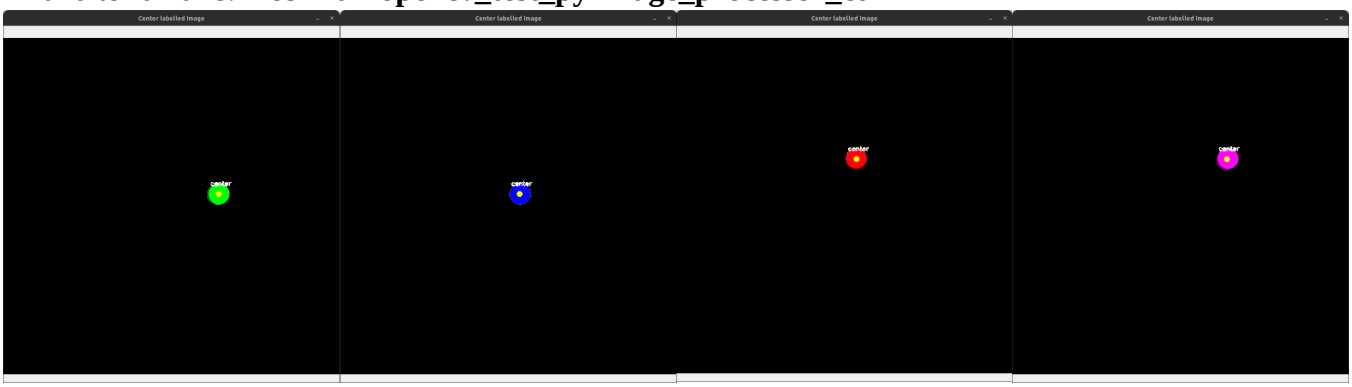

/Image

Step 3 (1.5 pts): Image_View

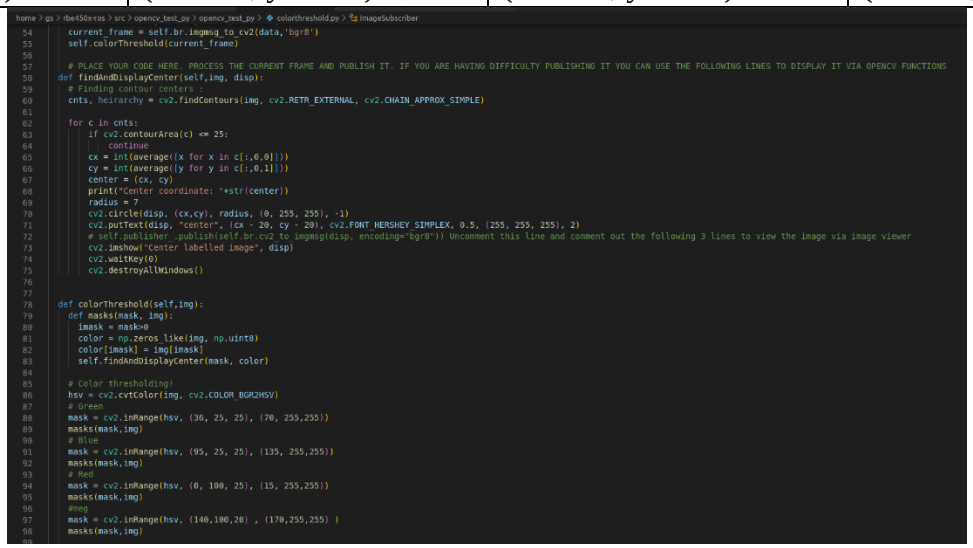


Step 4 (2 pts): Now modify these test files so that you apply color thresholding to the image using OpenCV functions

Command to run this: **`ros2 run opencv_test_py image_processor_ct`**



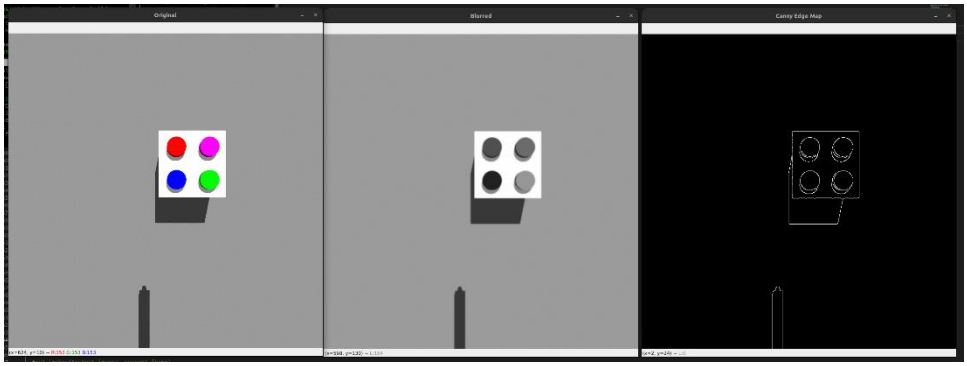
(x = 510, y = 369)	(x = 428, y = 369)	(x = 428, y = 287)	(x = 510, y = 287)

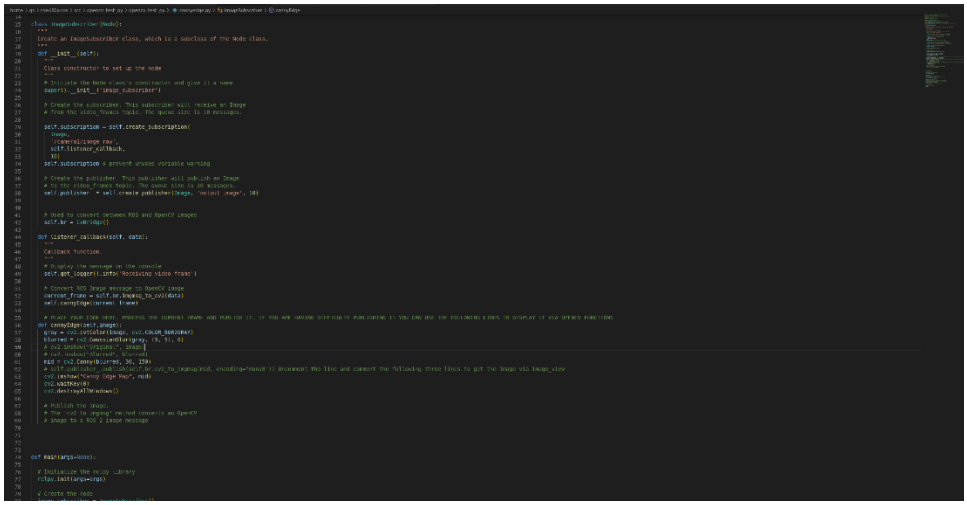


```
    current_frame = self.br.imgmsg_to_cv2(data,'bgr8')
    self.colorThreshold(current_frame)

    # PLACE YOUR CODE HERE. PROCESS THE CURRENT FRAME AND PUBLISH IT. IF YOU ARE HAVING DIFFICULTY PUBLISHING IT YOU CAN USE THE FOLLOWING LINES TO DISPLAY IT VIA OPENCV FUNCTIONS
  def findAndDisplayCenter(self,img, disp):
    # Finding contour centers
    cnts, heirarchy = cv2.findContours(img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    for c in cnts:
        if cv2.contourArea(c) <= 25:
            continue
        cx = int(average([x for x in c[:,0,0]]))
        cy = int(average([y for y in c[:,0,1]]))
        center = (cx, cy)
        print("Center coordinate: "+str(center))
        radius = 7
        cv2.circle(disp, (cx,cy), radius, (0, 255, 255), -1)
        cv2.putText(disp, "center", (cx - 20, cy - 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)
        # self.publisher .publish(self.br.cv2_to_imgmsg(disp, encoding="bgr8")) Uncomment this line and comment out the following 3 lines to view the image via image viewer
        cv2.imshow("Center labelled image", disp)
        cv2.waitKey(0)
        cv2.destroyAllWindows()


  def colorThreshold(self,img):
    def masks(mask, img):
      imask = mask>0
      color = np.zeros_like(img, np.uint8)
      color[imask] = img[imask]
      self.findAndDisplayCenter(mask, color)

    # Color thresholding!
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    # Green
    mask = cv2.inRange(hsv, (36, 25, 25), (70, 255,255))
    masks(mask,img)
    # Blue
    mask = cv2.inRange(hsv, (95, 25, 25), (135, 255,255))
    masks(mask,img)
    # Red
    mask = cv2.inRange(hsv, (0, 100, 25), (15, 255,255))
    masks(mask,img)
    #Yeeg
    mask = cv2.inRange(hsv, (140,100,20) , (170,255,255) )
    masks(mask,img)
```

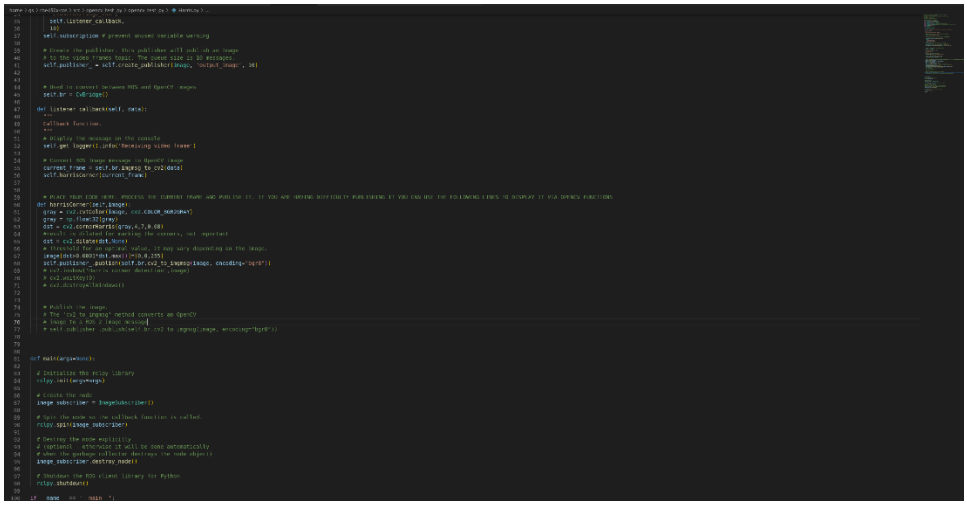
Step 5 (1 pts): Apply canny edge detection algorithm to the captured image using OpenCV

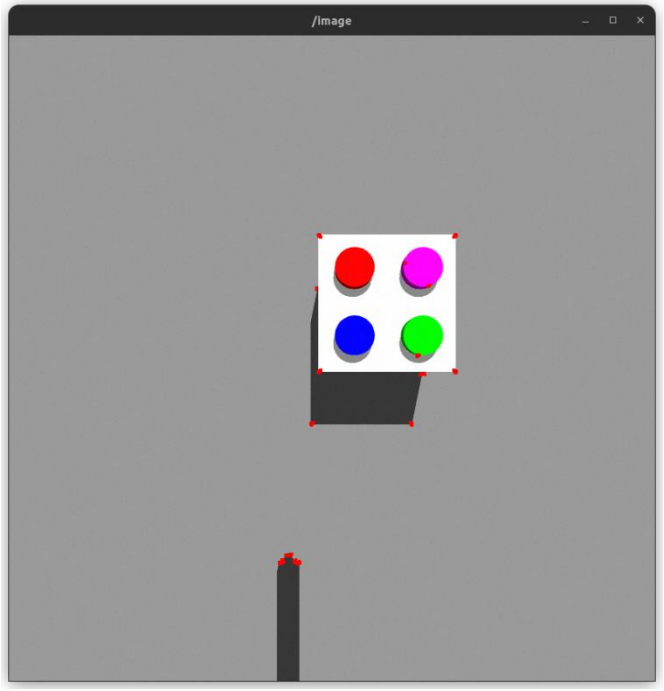
Command to run this: **`ros2 run opencv_test_py image_processor_ced`**





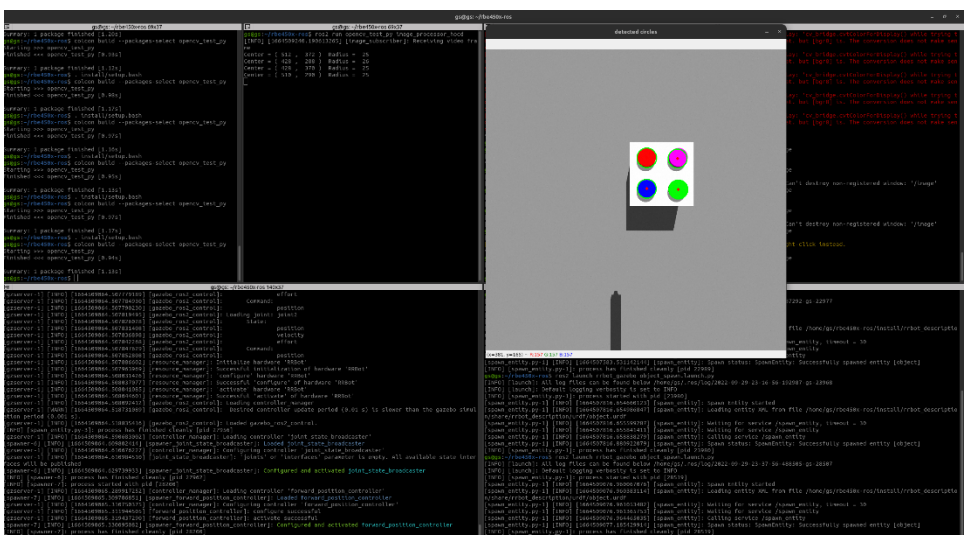
BONUS: Step 6 (2 pts): Apply Harris corner detection algorithm to the captured image using OpenCV functions. Command to run this: **`ros2 run opencv_test_py image_processor_hcd `**





BONUS: Step 7 (2pts): Apply Hough circles algorithm to detect the circles and find their centers.

Command to run this: **`ros2 run opencv_test_py image_processor_hocd`**



Center = (512 , 372) Radius = 25
Center = (428 , 288) Radius = 26
Center = (428 , 370) Radius = 25
Center = (510 , 290) Radius = 25

```
home > ros > ros2Ubuntu > src > opencv_test_pkg > opencv_test_pkg > HoughCircles.py > ImageSubscriber > houghCircles
    Class constructor to set up the node
    '''
    # Initiate the Node class's constructor and give it a name
    super().__init__('image_subscriber')

    # Create the subscriber. This subscriber will receive an Image
    # from the video frames topic. The queue size is 10 messages.

    self.subscription = self.create_subscription(
      Image,
      '/camera1/image_raw',
      self.listener_callback,
      10)
    self.subscription # prevent unused variable warning

    # Create the publisher. This publisher will publish an Image
    # to the video frames topic. The queue size is 10 messages.
    self.publisher_ = self.create_publisher(Image, 'output_image', 10)


    # Used to convert between ROS and OpenCV images
    self.br = CvBridge()

  def listener_callback(self, data):
    '''
    Callback function.
    '''
    # Display the message on the console
    self.get_logger().info('Receiving video frame')

    # Convert ROS Image message to OpenCV image
    current_frame = self.br.imgmsg_to_cv2(data)
    self.houghCircles(current_frame)

    # PLACE YOUR CODE HERE. PROCESS THE CURRENT FRAME AND PUBLISH IT. IF YOU ARE HAVING DIFFICULTY PUBLISHING IF YOU CAN USE THE FOLLOWING LINES TO DISPLAY IT VIA OPENCV FUNCTIONS

  def houghCircles(self,img):
    cimg = img.copy()
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    circles = cv2.HoughCircles(img,cv2.HOUGH_GRADIENT,1,40,param1=50,param2=40,minRadius=0,maxRadius=0)
    circles = np.uint16(np.around(circles))
    for i in circles[0,:]:
        # draw the outer circle
        cv2.circle(cimg,(i[0],i[1]),i[2],(0,255,0),2)
        # draw the center of the circle
        cv2.circle(cimg,(i[0],i[1]),2,(0,0,255),3)
        print("Center = (", i[0], ", ", i[1], ")", " Radius = ", i[2])
    self.publisher_.publish(self.br.cv2_to_imgmsg(cimg, encoding="bgr8")) #Comment this line and uncomment the following lines to view the image through cv2
    # cv2.imshow('detected circles',cimg)
    # cv2.waitKey(0)
    # cv2.destroyAllWindows()


    # Publish the image.
    # The `cv2_to_imgmsg` method converts an OpenCV
    # image to a ROS 2 image message
    # self.publisher_.publish(self.br.cv2_to_imgmsg(cimg, encoding="bgr8"))


def main(args=None):

  # Initialize the rclpy library
  rclpy.init(args=args)

  # Create the node
  image_subscriber = ImageSubscriber()
```