```
# Import all the required modules
import cv2
import numpy as np
import movements
from picamera import PiCamera
from picamera.array import PiRGBArray

# Constants and Variable Declaration
threshold = 50      # in pixels
ball_captured = 0   # odd -> capture orange ball, even -> score a goal
distance = 1.25     # in cm
turn_angle = 2      # in degree
search_angle = 5    # in degree
rotation_time = 0   # for number of times rotated
direction = 'r'     # l - left, r - right
res = (608, 368)    # resolution for the frame
kernel = np.ones((5, 5), np.uint8)

# HSV Range for required objects
orange = np.array([[0, 135, 135], [32, 255, 255]])
goal = np.array([[154, 100, 100], [175, 255, 255]])

camera = PiCamera()      # To initialize the PiCamera
camera.resolution = res  # set the resolution of the camera
camera.rotation = 180    # to rotate the frames by 180 degrees
camera.framerate = 16    # Set the frame rate
rawCapture = PiRGBArray(camera, size=res)
movements.wp.delay(10)   # Wait for the Camera to initialize


def search():     # Function to turn the bot for searching the bot
    # global variable declaration
    global search_angle, direction, rotation_time

    if rotation_time > 360:
        rotation_time = 0

    rotation_time += search_angle
    movements.turn(direction, search_angle)


def tracking():   # Function to process the captured frame

    # define all the global variables
    global ball_captured, direction, orange, goal

    # Setting values based on the ball capture
    if (ball_captured % 2) == 0:
        color_range = orange
        threshold_y = 300
    else:
        color_range = goal
        threshold_y = 250

    # to start receiving the  frames form the camera
    for image in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
        # save the image as a numpy array
        frame = image.array
        # clear the buffer memory
        rawCapture.truncate(0)

        # convert the frame to HSV co-ordinates
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

        # mask -> to apply filter to the image based on color range
        mask = cv2.inRange(hsv, color_range[0], color_range[1])
        # erode -> to remove small blobs in the image
        mask = cv2.erode(mask, kernel, iterations=1)
        # dilate -> to sharpen the edges
        mask = cv2.dilate(mask, kernel, iterations=1)

        # contours -> set of points which are in white
        contours = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2]
        if contours:
            # maximum area -> the ball
            object = max(contours, key=len)
            # calculating the x-center and y-center
            (x, y) = ((max(object[:, :, 0])+min(object[:, :, 0]))//2, (max(object[:, :, 1])+min(object[:, :, 1]))//2)

            # to check for captured condition
            if y >= threshold_y:
                # to close/open the gate depending on the condition
                movements.gate(ball_captured)
                ball_captured += 1
                break

            # to check if the  ball is on the left side
            elif x < ((res[0]//2) - threshold):
                direction = 'l'
                movements.turn(direction, turn_angle)
                print("Left")

            #  to check if the ball is on the right side
            elif x > ((res[0]//2) + threshold):
                direction = 'r'
                movements.turn(direction, turn_angle
)
                print("Right")

            # if the ball is within the threshold region
            elif y < threshold_y:
                movements.move('f', distance)
                print("moving towards object")
        else:
            # search for the ball
            print("Searching")
            search()


# loop indefinitely
while 1:
    tracking()


```