HW4

Commands:

1. **Download the package in src folder of catkin_ws.**
2. **Run the command** roslaunch vision_based_manipulation vb_publish_joint_pose_and_vel.launch **in the terminal.**

Steps:

1. Imported the Object URDF into the robot world.
2. Captured the reference image at initial position and Extracted features using Color Thresholding.
3. Traversed the robot to different location using position controller and captured the image and extracted the features.
4. Calculated the error between the current feature coordinates and reference feature coordinates.
5. Calculated the Image Jacobian and the End-effector Twist.
6. Calculated the Jacobian parameters using Forward Kinematics and subscribing joint angles from joint_state topic.
7. Calculated the Joint velocities using Inverse Jacobian by providing the end-effector velocity
8. Published the joint velocity into the Joint_velocity_controller.
9. Tried different values of lamda to obtain smooth servoing. Made Error zero for too small joint velocity values.(To prevent damping effect)

Output Screeenshots:

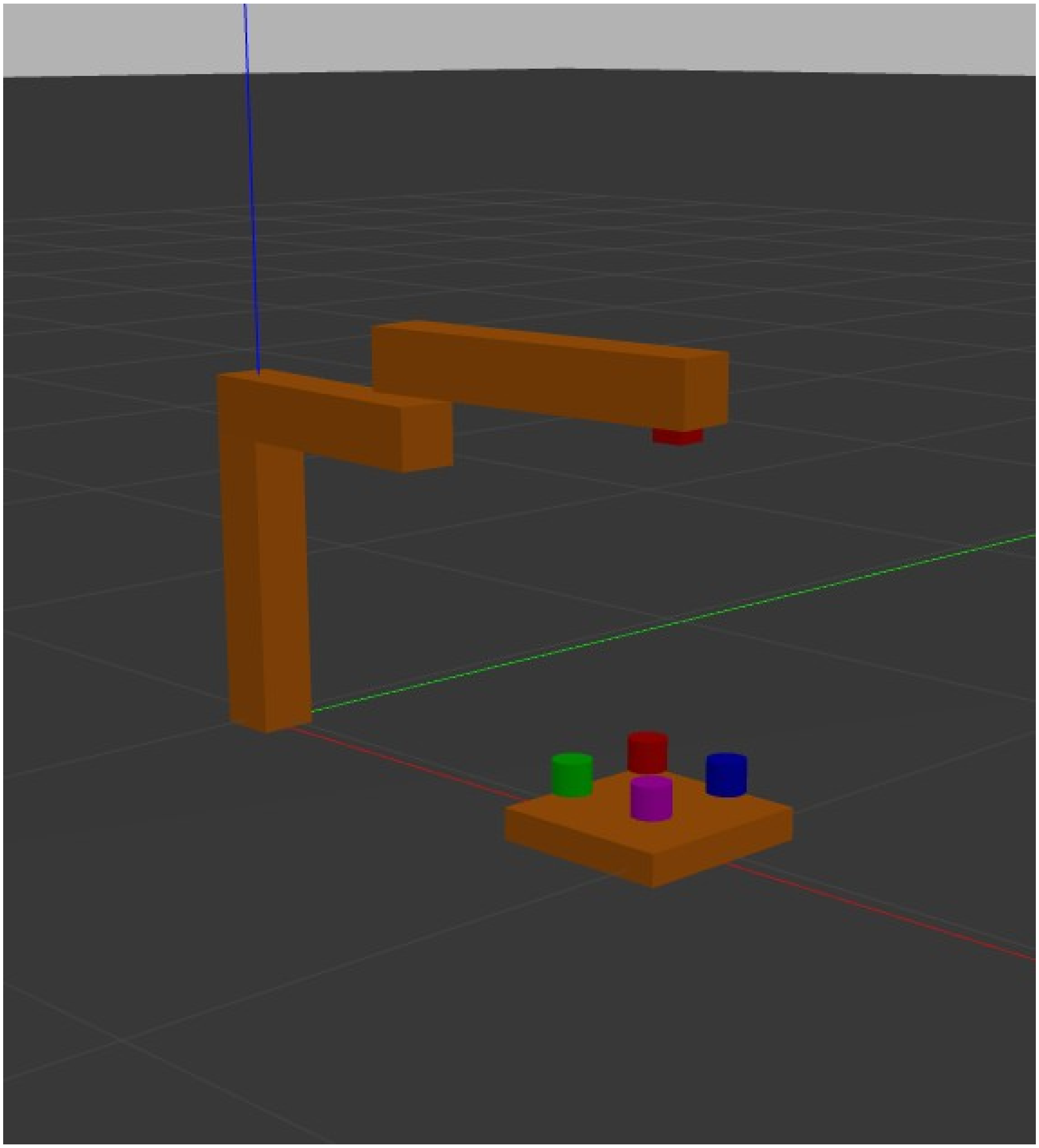


Figure 1: Robot and Object models in Gazebo

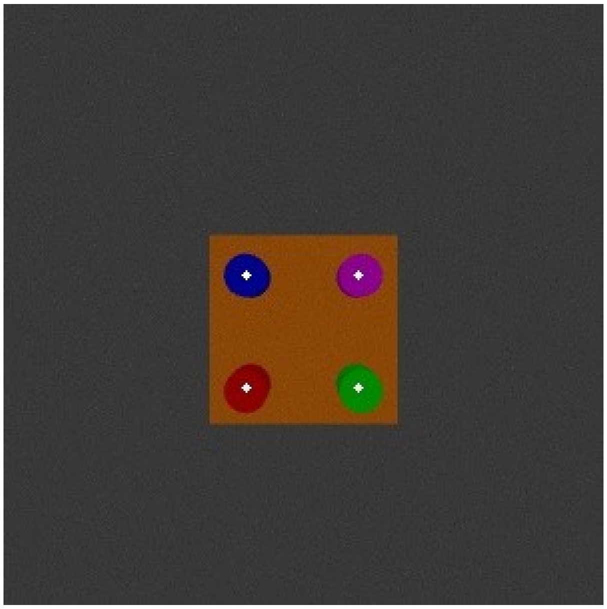


Figure 3: Refrence Image for Visual Servoing

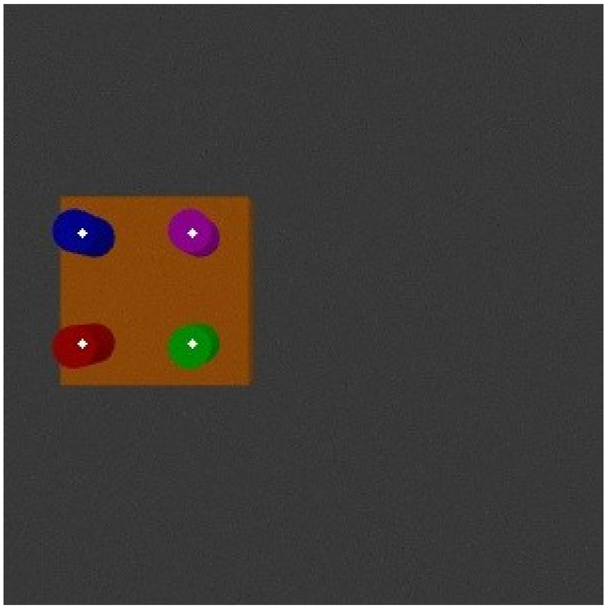
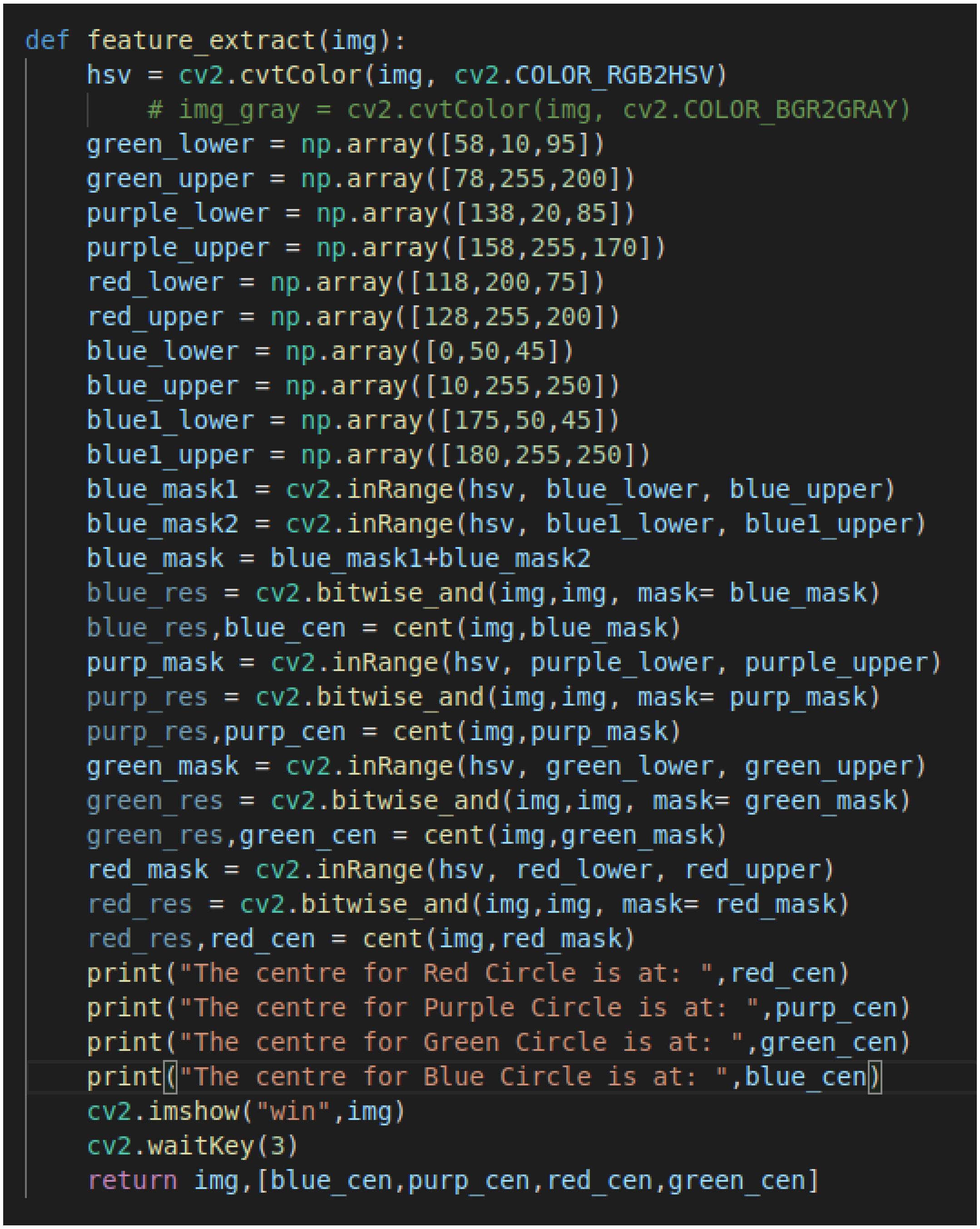


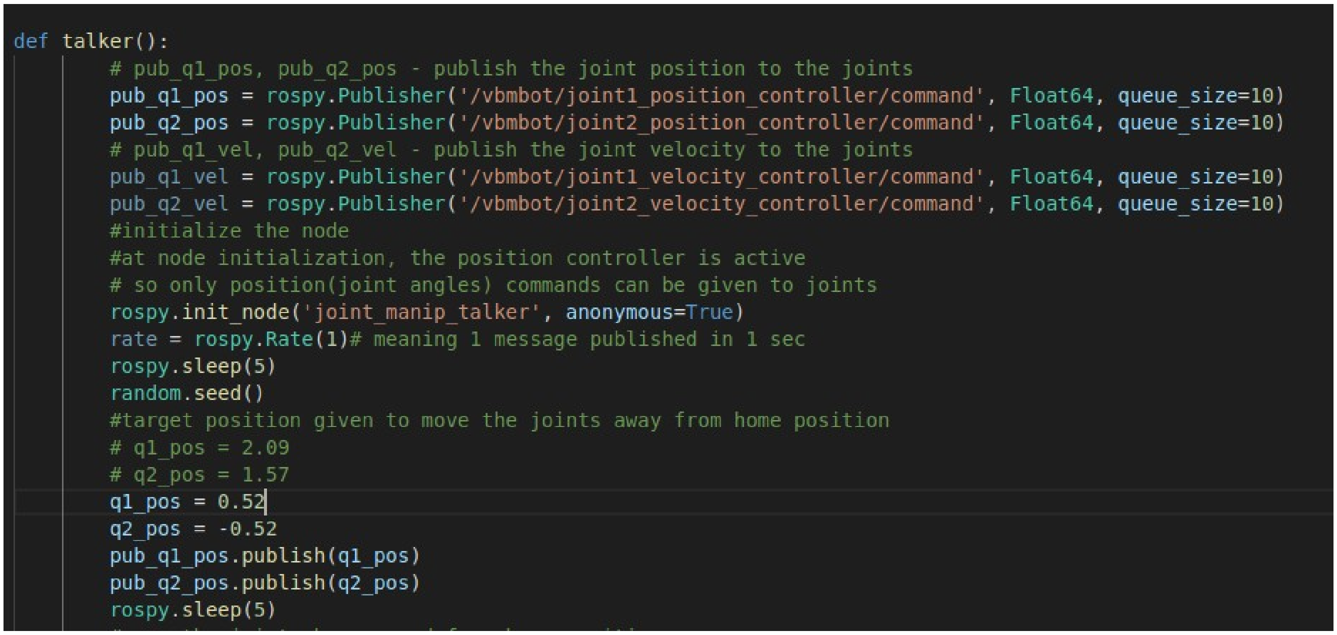
Figure 2: Image from different position

SNAPSHOT FOR FEATURE DETECTION:



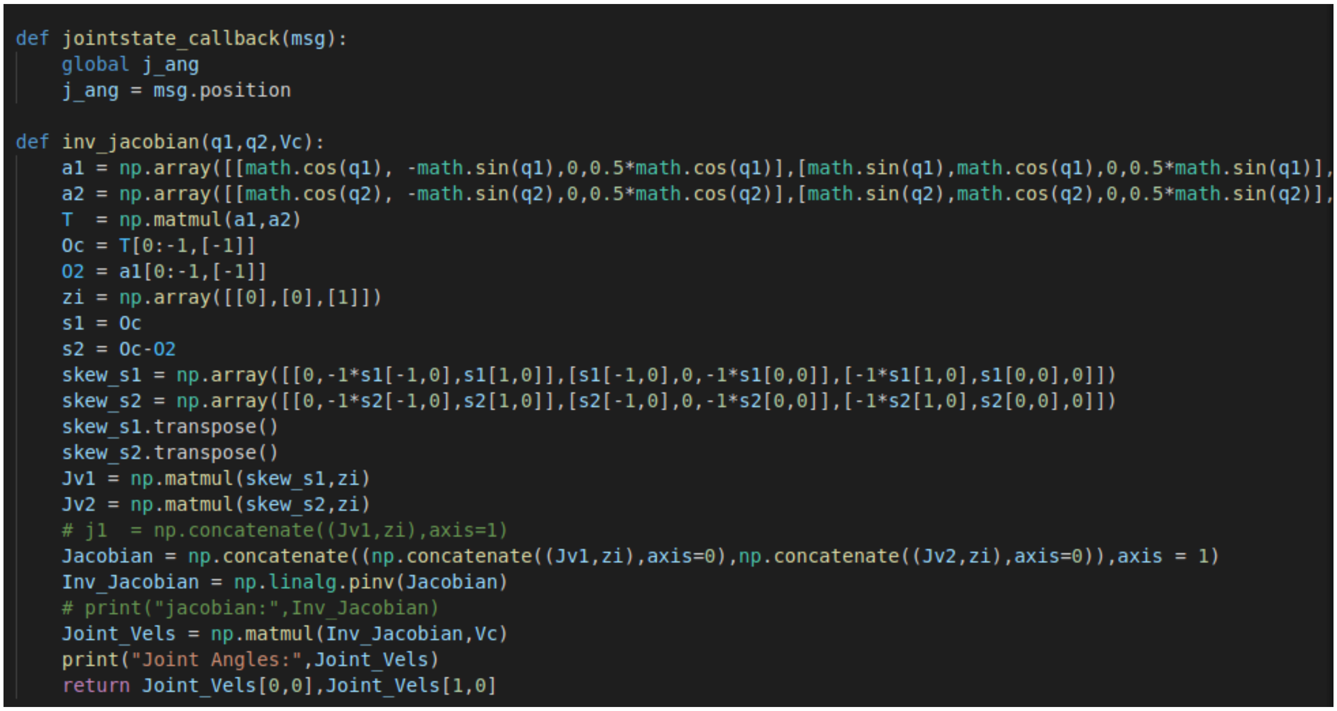
```python
def feature_extract(img):
    hsv = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
        # img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    green_lower = np.array([58,10,95])
    green_upper = np.array([78,255,200])
    purple_lower = np.array([138,20,85])
    purple_upper = np.array([158,255,170])
    red_lower = np.array([118,200,75])
    red_upper = np.array([128,255,200])
    blue_lower = np.array([0,50,45])
    blue_upper = np.array([10,255,250])
    blue1_lower = np.array([175,50,45])
    blue1_upper = np.array([180,255,250])
    blue_mask1 = cv2.inRange(hsv, blue_lower, blue_upper)
    blue_mask2 = cv2.inRange(hsv, blue1_lower, blue1_upper)
    blue_mask = blue_mask1+blue_mask2
    blue_res = cv2.bitwise_and(img,img, mask= blue_mask)
    blue_res,blue_cen = cent(img,blue_mask)
    purp_mask = cv2.inRange(hsv, purple_lower, purple_upper)
    purp_res = cv2.bitwise_and(img,img, mask= purp_mask)
    purp_res,purp_cen = cent(img,purp_mask)
    green_mask = cv2.inRange(hsv, green_lower, green_upper)
    green_res = cv2.bitwise_and(img,img, mask= green_mask)
    green_res,green_cen = cent(img,green_mask)
    red_mask = cv2.inRange(hsv, red_lower, red_upper)
    red_res = cv2.bitwise_and(img,img, mask= red_mask)
    red_res,red_cen = cent(img,red_mask)
    print("The centre for Red Circle is at: ",red_cen)
    print("The centre for Purple Circle is at: ",purp_cen)
    print("The centre for Green Circle is at: ",green_cen)
    print("The centre for Blue Circle is at: ",blue_cen)
    cv2.imshow("win",img)
    cv2.waitKey(3)
    return img,[blue_cen,purp_cen,red_cen,green_cen]
```

SNAPSHOT OF CODE FOR MOVING THE ROBOT TO DIFFERENT POSITION USING POSITION CONTROLLER



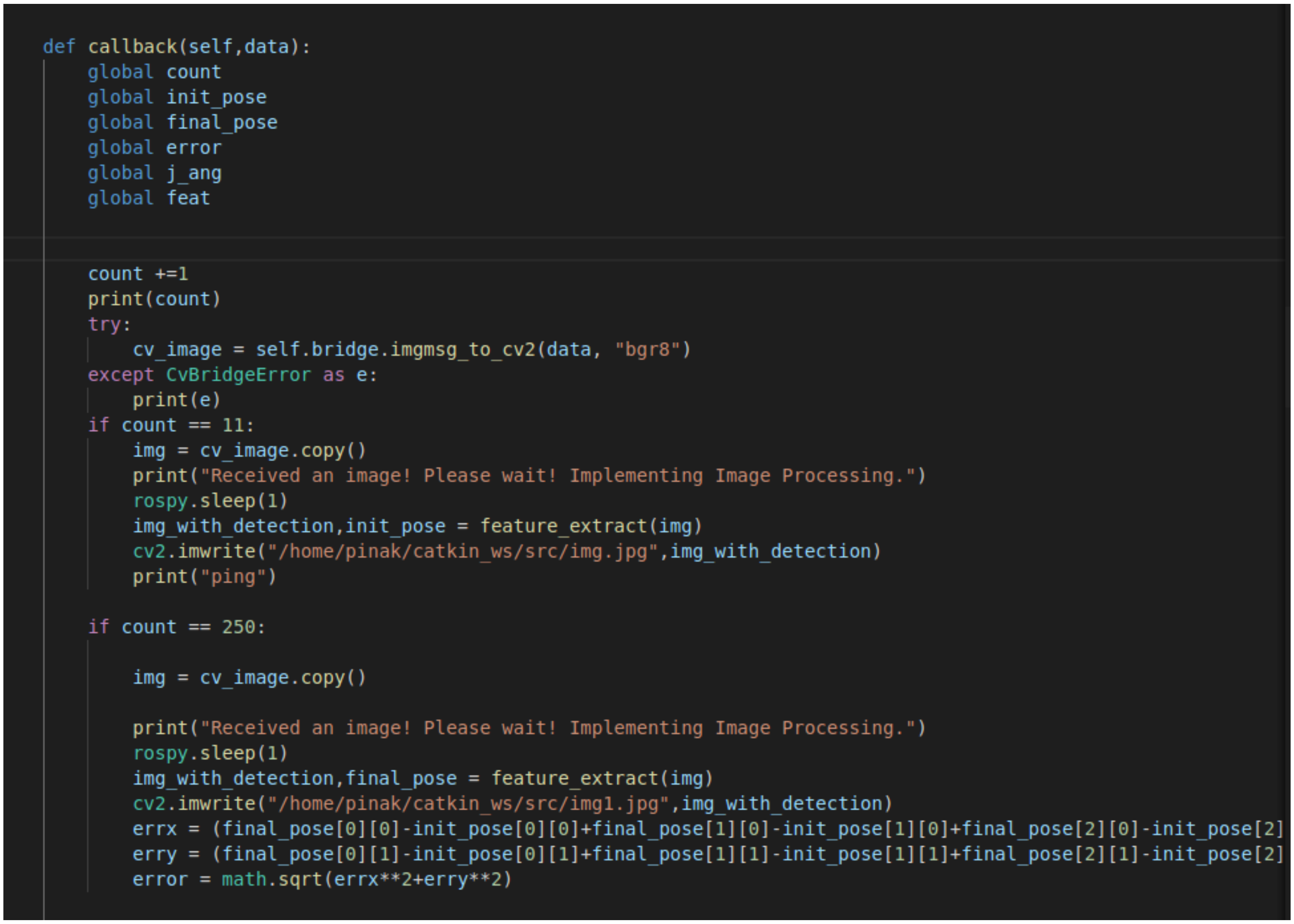
```python
def talker():
        # pub_q1_pos, pub_q2_pos - publish the joint position to the joints
        pub_q1_pos = rospy.Publisher('/vbmbot/joint1_position_controller/command', Float64, queue_size=10)
        pub_q2_pos = rospy.Publisher('/vbmbot/joint2_position_controller/command', Float64, queue_size=10)
        # pub_q1_vel, pub_q2_vel - publish the joint velocity to the joints
        pub_q1_vel = rospy.Publisher('/vbmbot/joint1_velocity_controller/command', Float64, queue_size=10)
        pub_q2_vel = rospy.Publisher('/vbmbot/joint2_velocity_controller/command', Float64, queue_size=10)
        #initialize the node
        #at node initialization, the position controller is active
        # so only position(joint angles) commands can be given to joints
        rospy.init_node('joint_manip_talker', anonymous=True)
        rate = rospy.Rate(1)# meaning 1 message published in 1 sec
        rospy.sleep(5)
        random.seed()
        #target position given to move the joints away from home position
        # q1_pos = 2.09
        # q2_pos = 1.57
        q1_pos = 0.52
        q2_pos = -0.52
        pub_q1_pos.publish(q1_pos)
        pub_q2_pos.publish(q2_pos)
        rospy.sleep(5)
```

SNAPSHOT OF CODE FOR IMPLEMENTING VISUAL SERVOING



```python
def jointstate_callback(msg):
    global j_ang
    j_ang = msg.position

def inv_jacobian(q1,q2,Vc):
    a1 = np.array([[math.cos(q1), -math.sin(q1),0,0.5*math.cos(q1)],[math.sin(q1),math.cos(q1),0,0.5*math.sin(q1)],
    a2 = np.array([[math.cos(q2), -math.sin(q2),0,0.5*math.cos(q2)],[math.sin(q2),math.cos(q2),0,0.5*math.sin(q2)],
    T  = np.matmul(a1,a2)
    Oc = T[0:-1,[-1]]
    O2 = a1[0:-1,[-1]]
    zi = np.array([[0],[0],[1]])
    s1 = Oc
    s2 = Oc-O2
    skew_s1 = np.array([[0,-1*s1[-1,0],s1[1,0]],[s1[-1,0],0,-1*s1[0,0]],[-1*s1[1,0],s1[0,0],0]])
    skew_s2 = np.array([[0,-1*s2[-1,0],s2[1,0]],[s2[-1,0],0,-1*s2[0,0]],[-1*s2[1,0],s2[0,0],0]])
    skew_s1.transpose()
    skew_s2.transpose()
    Jv1 = np.matmul(skew_s1,zi)
    Jv2 = np.matmul(skew_s2,zi)
    # j1  = np.concatenate((Jv1,zi),axis=1)
    Jacobian = np.concatenate((np.concatenate((Jv1,zi),axis=0),np.concatenate((Jv2,zi),axis=0)),axis = 1)
    Inv_Jacobian = np.linalg.pinv(Jacobian)
    # print("jacobian:",Inv_Jacobian)
    Joint_Vels = np.matmul(Inv_Jacobian,Vc)
    print("Joint Angles:",Joint_Vels)
    return Joint_Vels[0,0],Joint_Vels[1,0]
```



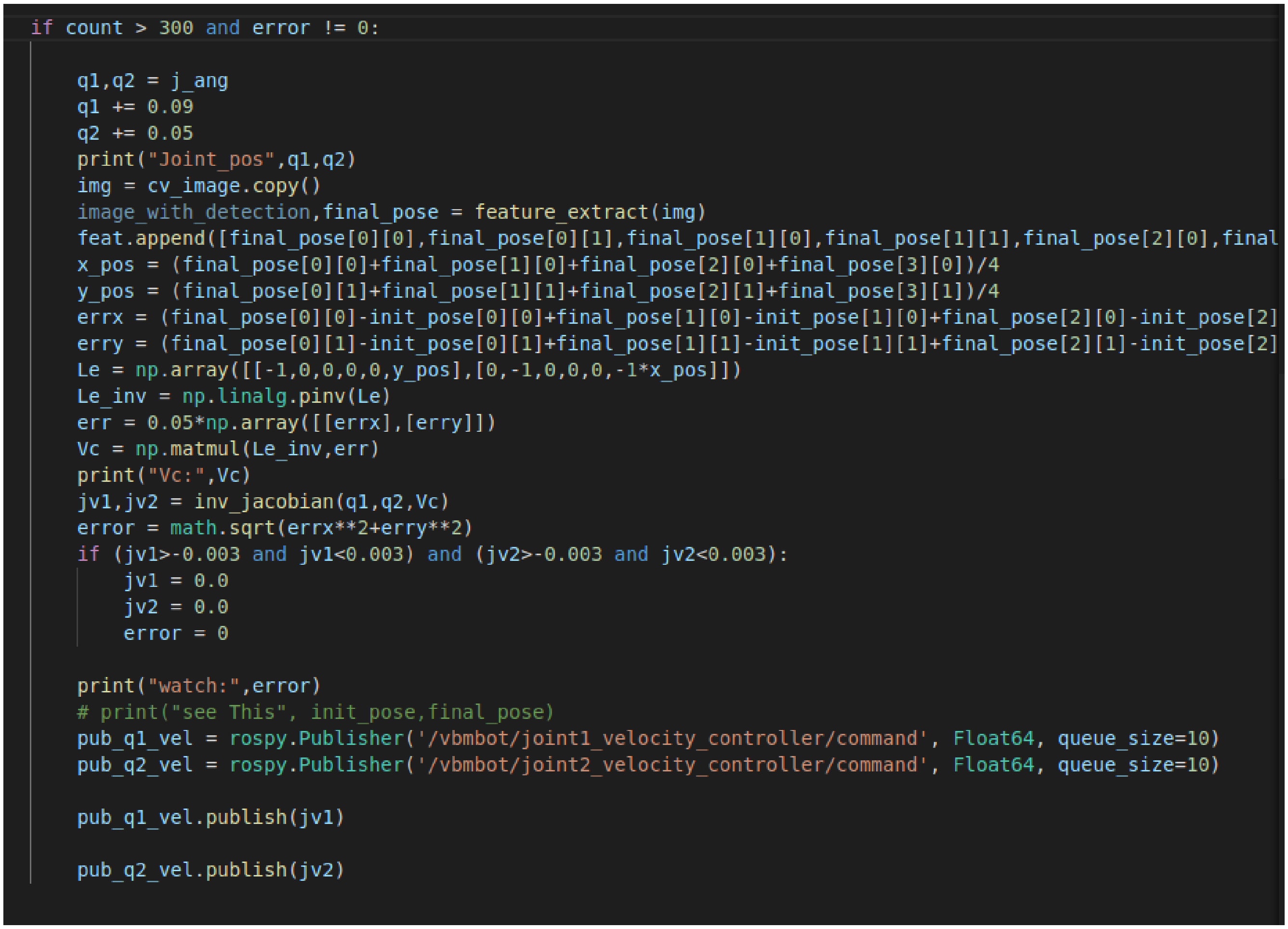
```python
def callback(self,data):
    global count
    global init_pose
    global final_pose
    global error
    global j_ang
    global feat


    count +=1
    print(count)
    try:
        cv_image = self.bridge.imgmsg_to_cv2(data, "bgr8")
    except CvBridgeError as e:
        print(e)
    if count == 11:
        img = cv_image.copy()
        print("Received an image! Please wait! Implementing Image Processing.")
        rospy.sleep(1)
        img_with_detection,init_pose = feature_extract(img)
        cv2.imwrite("/home/pinak/catkin_ws/src/img.jpg",img_with_detection)
        print("ping")

    if count == 250:

        img = cv_image.copy()

        print("Received an image! Please wait! Implementing Image Processing.")
        rospy.sleep(1)
        img_with_detection,final_pose = feature_extract(img)
        cv2.imwrite("/home/pinak/catkin_ws/src/img1.jpg",img_with_detection)
        errx = (final_pose[0][0]-init_pose[0][0]+final_pose[1][0]-init_pose[1][0]+final_pose[2][0]-init_pose[2]
        erry = (final_pose[0][1]-init_pose[0][1]+final_pose[1][1]-init_pose[1][1]+final_pose[2][1]-init_pose[2]
        error = math.sqrt(errx**2+erry**2)
```



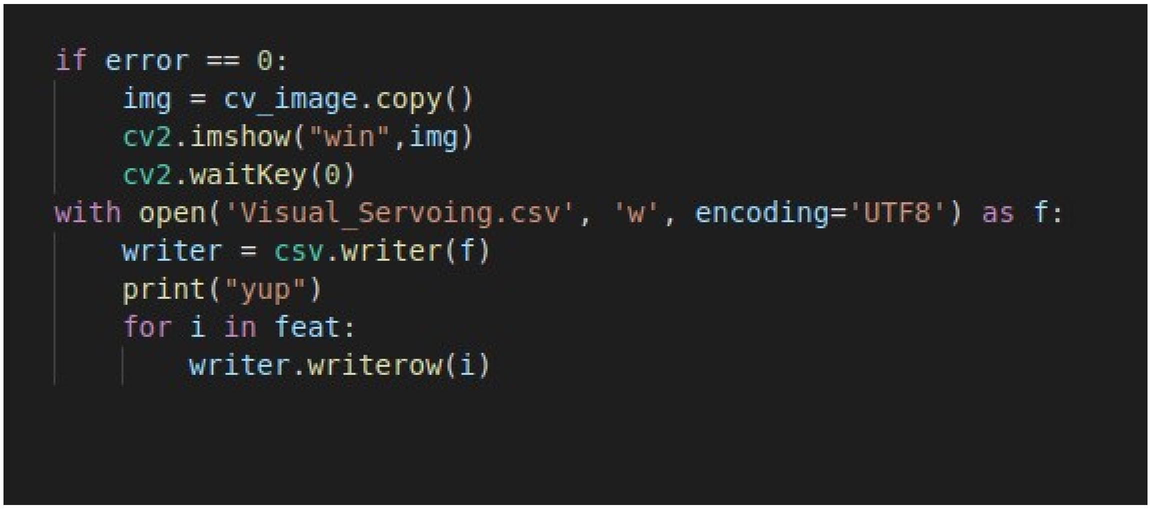
```
if count > 300 and error != 0:

    q1,q2 = j_ang
    q1 += 0.09
    q2 += 0.05
    print("Joint_pos",q1,q2)
    img = cv_image.copy()
    image_with_detection,final_pose = feature_extract(img)
    feat.append([final_pose[0][0],final_pose[0][1],final_pose[1][0],final_pose[1][1],final_pose[2][0],final
    x_pos = (final_pose[0][0]+final_pose[1][0]+final_pose[2][0]+final_pose[3][0])/4
    y_pos = (final_pose[0][1]+final_pose[1][1]+final_pose[2][1]+final_pose[3][1])/4
    errx = (final_pose[0][0]-init_pose[0][0]+final_pose[1][0]-init_pose[1][0]+final_pose[2][0]-init_pose[2]
    erry = (final_pose[0][1]-init_pose[0][1]+final_pose[1][1]-init_pose[1][1]+final_pose[2][1]-init_pose[2]
    Le = np.array([[-1,0,0,0,0,y_pos],[0,-1,0,0,0,-1*x_pos]])
    Le_inv = np.linalg.pinv(Le)
    err = 0.05*np.array([[errx],[erry]])
    Vc = np.matmul(Le_inv,err)
    print("Vc:",Vc)
    jv1,jv2 = inv_jacobian(q1,q2,Vc)
    error = math.sqrt(errx**2+erry**2)
    if (jv1>-0.003 and jv1<0.003) and (jv2>-0.003 and jv2<0.003):
        jv1 = 0.0
        jv2 = 0.0
        error = 0

    print("watch:",error)
    # print("see This", init_pose,final_pose)
    pub_q1_vel = rospy.Publisher('/vbmbot/joint1_velocity_controller/command', Float64, queue_size=10)
    pub_q2_vel = rospy.Publisher('/vbmbot/joint2_velocity_controller/command', Float64, queue_size=10)

    pub_q1_vel.publish(jv1)

    pub_q2_vel.publish(jv2)
```



```
if error == 0:
    img = cv_image.copy()
    cv2.imshow("win",img)
    cv2.waitKey(0)
with open('Visual_Servoing.csv', 'w', encoding='UTF8') as f:
    writer = csv.writer(f)
    print("yup")
    for i in feat:
        writer.writerow(i)
```

Following are the Plots of X,Y coordinates of 4 features.

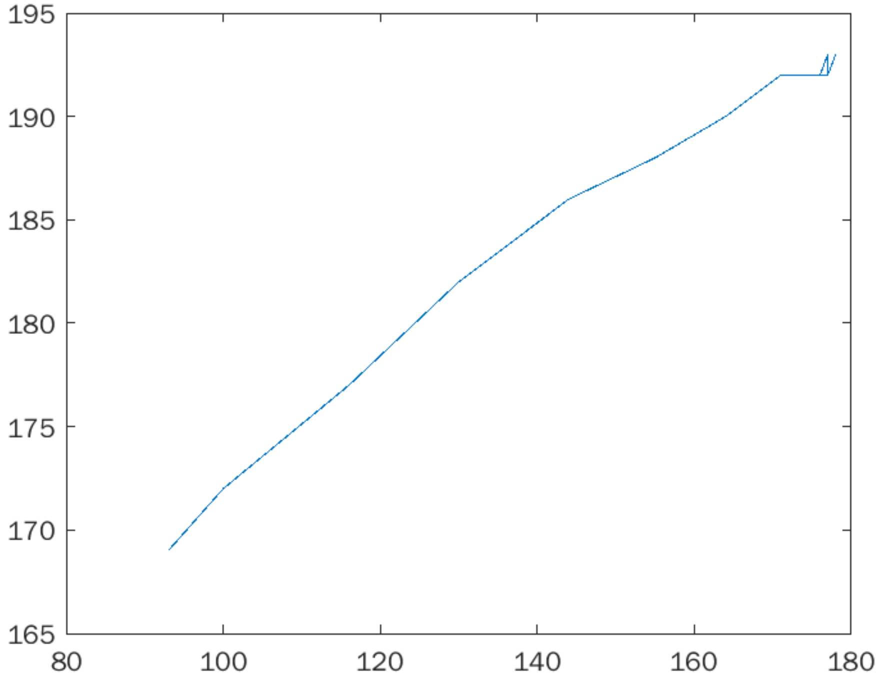


Figure 4: Feature 1 (Blue Circle Center)

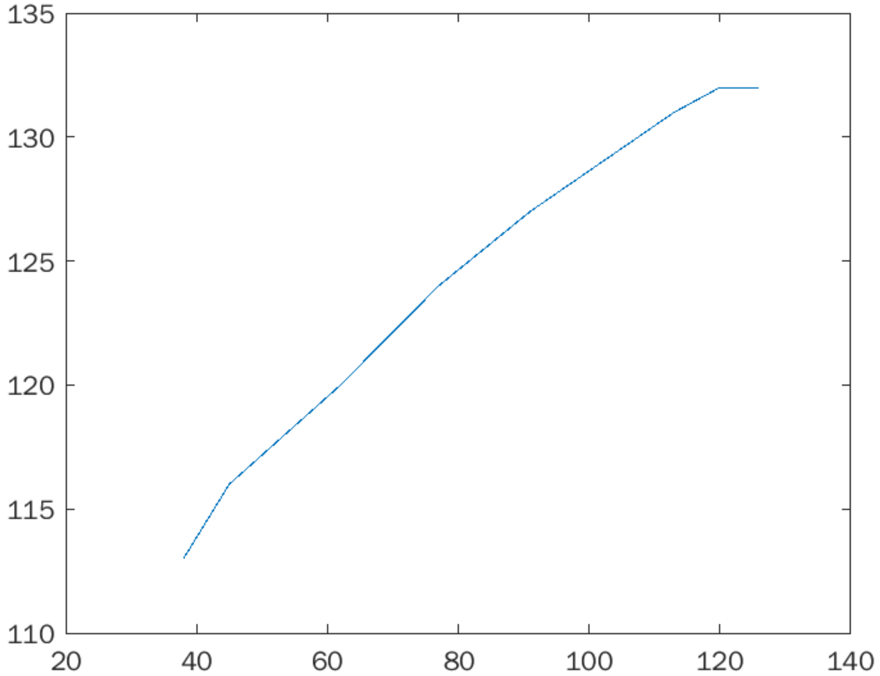


Figure 5: Feature 2 (Purple Circle Center)

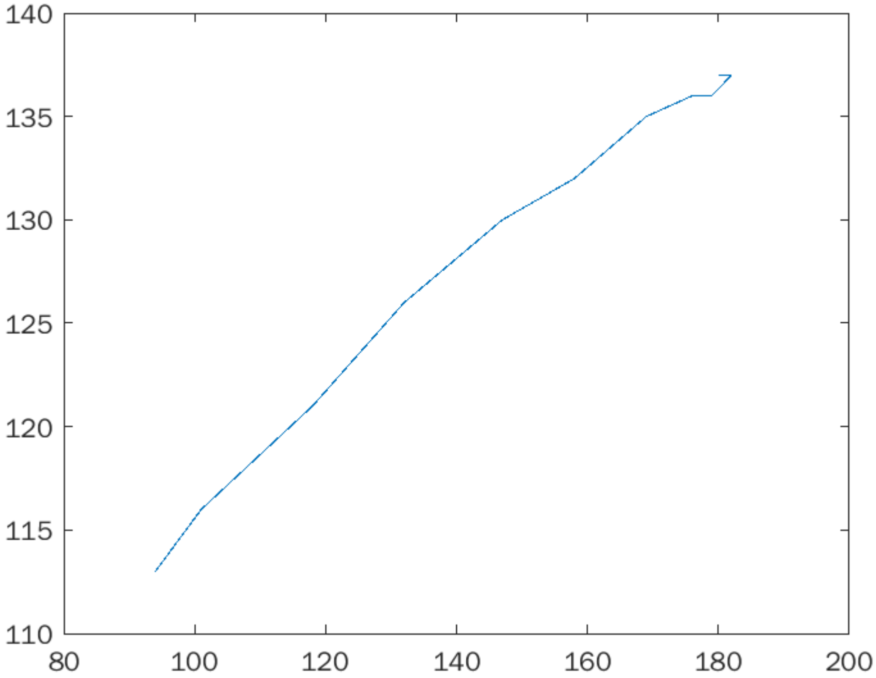


Figure 6: Feature 3 (Red Circle Center)

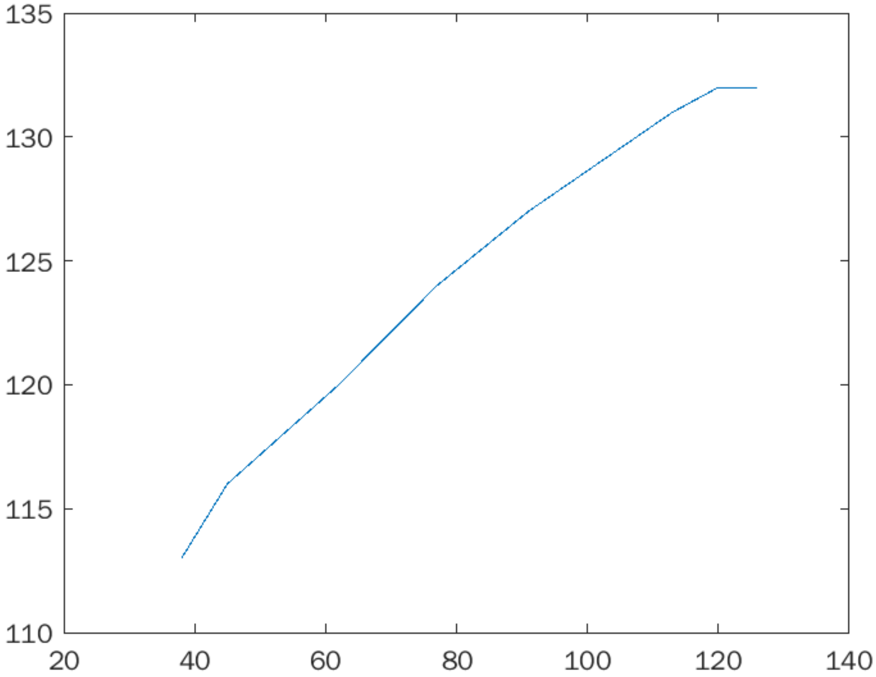


Figure 7: Feature 4 (Green Circle Center)