```python
# print('Program started')
import vrep
import numpy as np
import cv2
import cv2.aruco as aruco
import sys, time, math
from platform import python_version

##############Comunicação com V-REP##########################
serverIP = '127.0.0.1';
serverPort = 19999; #Esta porta do servidor está sempre aberta
vrep.simxFinish(-1);
clientID=vrep.simxStart(serverIP,serverPort,True,True,5000,5);
#############################################################

#------- ROTATIONS https://www.learnopencv.com/rotation-matrix-to-euler-angles/

# Checks if a matrix is a valid rotation matrix.
def isRotationMatrix(R):
    Rt = np.transpose(R)
    shouldBeIdentity = np.dot(Rt, R)
    I = np.identity(3, dtype=R.dtype)
    n = np.linalg.norm(I - shouldBeIdentity)
    return n < 1e-6

# Calculates rotation matrix to euler angles
# The result is the same as MATLAB except the order
# of the euler angles ( x and z are swapped ).
def rotationMatrixToEulerAngles(R):
    assert (isRotationMatrix(R))

    sy = math.sqrt(R[0, 0] * R[0, 0] + R[1, 0] * R[1, 0])

    singular = sy < 1e-6

    if not singular:
        x = math.atan2(R[2, 1], R[2, 2])
        y = math.atan2(-R[2, 0], sy)
        z = math.atan2(R[1, 0], R[0, 0])
    else:
        x = math.atan2(-R[1, 2], R[1, 1])
        y = math.atan2(-R[2, 0], sy)
        z = 0

    return np.array([x, y, z])
```

```
##############################################################################

#-- Update fps
def update_fps_read():
    global t_read, fps_read
    t           = time.time()
    fps_read    = 1.0/(t - t_read)
    t_read      = t

def update_fps_detect():
    global t_detect, fps_detect
    t           = time.time()
    fps_detect  = 1.0/(t - t_detect)
    t_detect      = t

t_read      = time.time()
t_detect    = t_read
fps_read    = 0.0
fps_detect  = 0.0

##############################################################################

windowName = "Imagem-Processada" #Name of the window created
#cv2.namedWindow(windowName, cv2.WINDOW_NORMAL)#Setting the name ande type of
window
#cv2.setWindowProperty(windowName, cv2.WND_PROP_FULLSCREEN,
cv2.WINDOW_KEEPRATIO)#setting fullscreen

#-- Define Tag
id_to_find = 2
marker_size = 45 #-cm

#-- Get the camera calibration
calib_path = ''
camera_matrix = np.loadtxt(calib_path+'cameraMatrix.txt', delimiter = ',')
camera_distortion = np.loadtxt(calib_path+'cameraDistortion.txt', delimiter =
',')

#-- 180 deg rotation matrix around x axis
R_flip = np.zeros((3,3), dtype=np.cfloat)
R_flip[0,0] = 1.0
R_flip[1,1] = -1.0
R_flip[2,2] = -1.0
```

```
#-- variables of control
ti=0

eyawp = 0 #erro passado yaw
exp = 0 #erro passado x
eyp = 0 #erro passado y
ezp = 0 #erro passado z
```