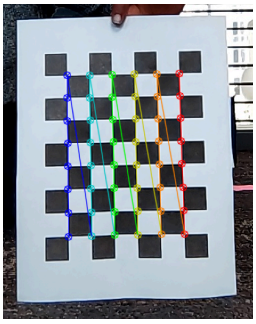
ROBOTICS : 276A HW2 REPORT

PID1: A69034019 Name: Anandhini Rajendran

PID2: A69033245 Name: Yueqi Wu

Video link: https://www.youtube.com/watch?v=8Xk9zRY5riA

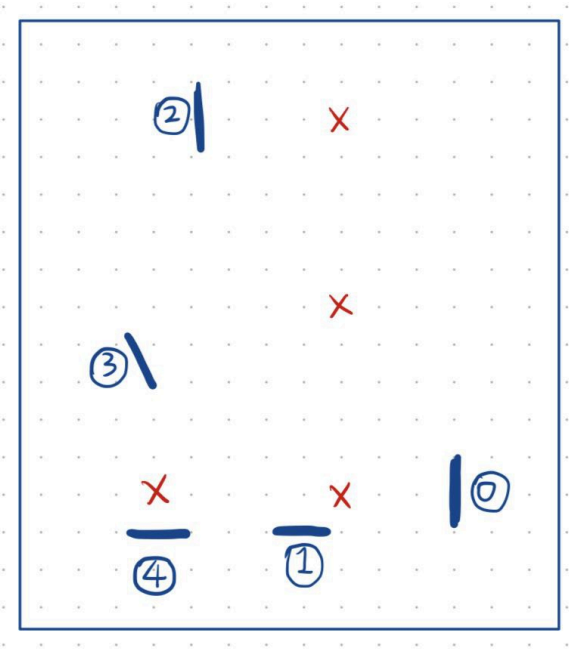
1. Camera Calibration
    a. This is the photo that the robot captures when we were doing calibration for the camera.



    b. Then we use this image in opencv to do the calibration. We get the following matrices.
        i. Camera Matrix:
        [[1.37513711e+03 0.00000000e+00 9.63968702e+02]
        [0.00000000e+00 1.36959868e+03 4.52219384e+02]
        [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
        ii. Distortion Coefficients:
        [[1.08675231e+00 -1.08970459e+01 -3.99173376e-02 -4.19008681e-03
        7.84687460e+01]]
        iii. Rotation Vectors:
        (array([[-0.05068273], [0.05367279], [1.54975435]]),
        array([[-0.03401715], [0.06748636], [1.55867044]]),
        array([[-0.14743508], [0.05101935], [1.55415503]]))
        iv. Translation Vectors:
        (array([[2.81993904], [1.85224341], [40.35201124]]),
        array([[2.67138361], [1.70467353], [40.01496111]]),
        array([[2.76566677], [1.86153047], [40.16621172]]))
    c. Updated tag size and camera calibrated parameters in april detection.cpp

2. Setting up landmarks

Logic for each landmark: uses where the robot faces when it stops after a constant 0.8 s interval. Our setting ensures that the corrections are made in each step and no path goes unchecked.



a. 0 : from 0,0 to 1,0, the current_state as provided by the robot has an error in the x value. So we use an april tag facing the robot at [1.5,0] to correct the x a direction error. The z axis of the camera points in the direction of positive x (world frame) which makes it convenient to correct x error.
b. 1 : from 1,0 to 1,2 the robot reaches 0.8 ,1.7 in the original pid path. So we need a correction in both x,y. After starting from 1,0 it turns and stops to see tag 1 (as shown in figure). Updates y according to the world coordinates of landmark 1 and the z predicted by april tag. Updates x according to the world coordinates of landmark 1 and the x predicted by april tag.
c. 2: to correct the robot state to (1,2,pi) - sometimes the robot has the wrong orientation.
d. 3,4: from (1,2) to (0,0) the value for x had a huge error (not much error in y due to previous corrections) : so we update the value of x using the z for 3 and x for 4 (due to frame orientation changes) and landmark position in world frame.

3. Estimation of robot poses:

a. Approach1: Transformation Matrix
   i. Code for transformation matrix and get the current position of the robot:

```
def transformation_matrix(x, y, z, pitch):
        """Creates a 4x4 transformation matrix with translation and pitch
(Y-axis rotation)."""
        return np.array([
        [np.cos(pitch), 0, np.sin(pitch), x],
```

```
        [0, 1, 0, y],
        [-np.sin(pitch), 0, np.cos(pitch), z],
        [0, 0, 0, 1]
        ])
    def get_curr_pos(tag_world, tag_camera):
        # Unpack tag_world and tag_camera
        x_w, y_w, z_w, pitch_w = tag_world
        T_tag_w = seslf.transformation_matrix(x_w, y_w, z_w, pitch_w)
        x_c, y_c, z_c, pitch_c = tag_camera
        T_tag_c = self.transformation_matrix(x_c, y_c, z_c, pitch_c)
        # Invert T_tag_w and compute T_c_w
        T_tag_w_inv = np.linalg.inv(T_tag_w)
        T_c_w = np.dot(T_tag_c, T_tag_w_inv)
        # Extract x, y, z translation in the world frame
        x = -T_c_w[0, 3]
        y = -T_c_w[1, 3]
        z = -T_c_w[2, 3]
        return [x, y]
```

ii. We are using april tag world coordinates and april tag prediction coordinates: converting the quaternion representation to obtain pitch,roll and yaw. We are only using the pitch and x,z coordinates since its 2d motion.

iii. Calculating the transformation matrices to get the coordinates of robot in world frames

1. 3d transform using (x,y,w) :The issue was that the value of y changes a lot for a little change in w(the pitch) when we rotate the robot in the same spot. Which means we can't do accurate corrections for y value if the robot is slightly tilted. So we moved to the next approach of using 2d transform and keeping w independant.
2. 2d transform using (x,y) and keeping pitch independent.: Also gave a similar issue to 3d : the y values were not consistent for small changes in orientation.
3. We realized that the z and x axis of the camera keeps changing and it's not fixed. Which implies the transformation matrix for april tag predictions with respect to camera is different for different landmarks.
4. Which makes it difficult to use the same april tag prediction values for the function to give the final robot orientation. This is due to the z,x axis orientation changing depending on the april tag position.

b. Approach 2:

i. Using different transforms for x,y coordinates for each landmark.

ii. The transformation logic is explained in april tag setup logic.

iii. We do the update for the current state based on the camera axes orientation and april tag prediction vector along with the april tag landmark vector.

4. Control loop mechanism:

Updating the current state of the robot using AprilTag calibrated values in the PID controller.

a. Updating current state with calibrated state after a constant 0.8 s interval. Calibration of time interval : we tried values : 0.6, 0.8, 0.9, 1.1, 1.2 to see optimal stopping points for the robot.
b. To update the current state we use rclpy.spin_once to see the april tag only once.
c. Breakpoints occur when the error of current state and waypoint is less than a threshold.

5. Next Improvements:

a. Use Tf frame tree to update the current state and do transformations.
b. In the case where the robot doesn't see april tags, we need to address this.
    i. Possible ideas
        1. Put dummy tags which do no correction
        2. Rotate 360 to find the april tags.
        3. Continue with current motion