CSE 276A - Homework 2 - Report

Visual Feedback Control

Wang, Shou-Yu[A69030868][a], Huang, Yao-Ting[A69030676][a]

[a] *University of California, San Diego.*

October, 2024

Abstract

In this assignment, we used the April Tag as the anchors in the environment, so the rb5 robot can estimate its state and thus we can use feedback control on the robot.

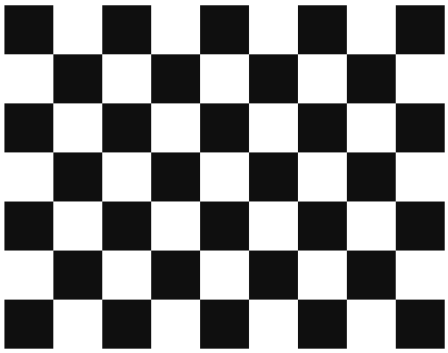

Figure 1: Chessboard image

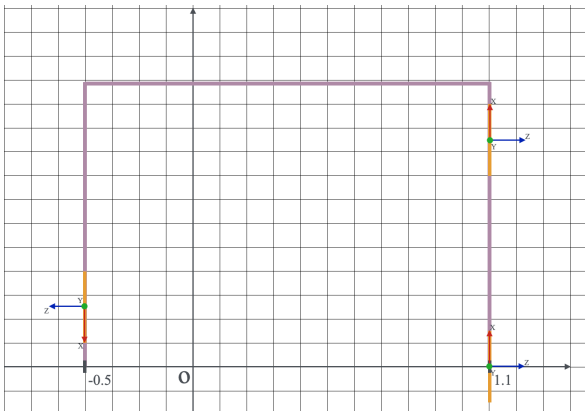

Figure 2: April Tag's setup. The yellow part indicate where we set the April Tag and its corresponding coordinates. The gray lines indicates the wall.

1. Result

Check out the video of our car's navigation at link.

2. Calibration

2.1. Camera calibration

First we took lots of picture containing the chessboard, shown in Fig.1, and we used those images to calibrate the parameters of camera's intrinsic model and it's distortion coefficients with cv.findChessboardCorners and cv.findChessboardCorners functions.

2.2. April Tag calibration

We stuck three April Tags in the surrounded wall, shown in Fig. 2 to maximize the possibilities that the robot having at least one April Tag in its camera view. After setting up the environment, we placed the rb5 in the origin and run the april detection node to get the transform matrix of April Tag's frame with respect to camera's frame, $\mathrm{T}_{\mathrm{tag}}^{\mathrm{camera}}$.

Since, we already know the $\mathrm{T}^{\mathrm{car}_{\mathrm{tag}}}$, we can compute the transform matrix between camera and car with

$$\mathrm{T}_{\mathrm{camera}}^{\mathrm{car}} = \mathrm{T}_{\mathrm{tag}}^{\mathrm{car}}\ \mathrm{T}_{\mathrm{tag}}^{\mathrm{camera}\,-1}$$

3. Navigation Algorithm

3.1. Closed-Loop Controller

This time we designed a closed-loop controller, shown in Fig.3. The controller will take into account both the next waypoint and the estimated position and direction of the robot. Each time we received the update message from the sensor node, we will update the robot's position and direction.

3.2. Navigation Algorithm

To solved the problem that the robot may happen to not able to see the April Tag during navigation, we came up with a mechanism to make sure the robot can see the April Tag. That is, if the robot's position and direction had not been updated by

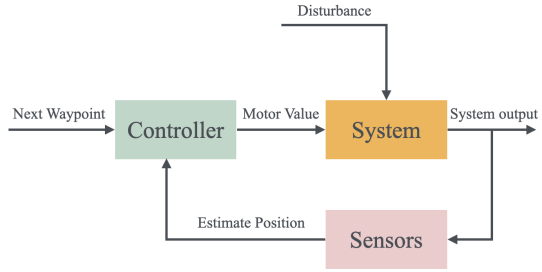

Figure 3: This figure shows the schematic of our closed-loop controller.

sensor node for 5 seconds, the robot will start rotating at current spot. In this way, the robot can have April Tag in its camera view and can thus update it's state. The rest algorithm is like the previous report's steps. The whole navigation algorithm shown in below. To smooth the trajectory, we discarded the movement that is too little, that is, once we calculated a next movement, we will examine the noise of the next move and the direction also, to make sure rb5 can move in a descent behavior without encountering some imperfection problem like sleeping wheels or the signal is too small for the motor controller.

1. Initialize the car's position and direction in world frame.
2. Calculate the difference of position $(\Delta x, \Delta y)$ and direction $\Delta\phi$. Divide those three by a proper transition time and get the velocities.
3. We move the car first and then rotate it to the desired direction.
4. Transform the velocities from world frame to car's frame.
5. Convert the velocities to the motors' speeds by Kinematic model.
6. After sending the motor signal, update the car's position and direction.
7. Each time we received the pose estimation message from sensor node, we update the robot's position and direction.
8. Rotate a current position if the robot's position and direction had not been updated by sensor node for 5 seconds.
9. Repeat for each given waypoint.

4. Performance

4.1. Total movement

Out total movement is about 0.05 m and 0.5 degree. The average error is 0.2 m

4.2. Trajectory

We record the estimated position every 0.5 second. The trajectory shown in Fig. 4.

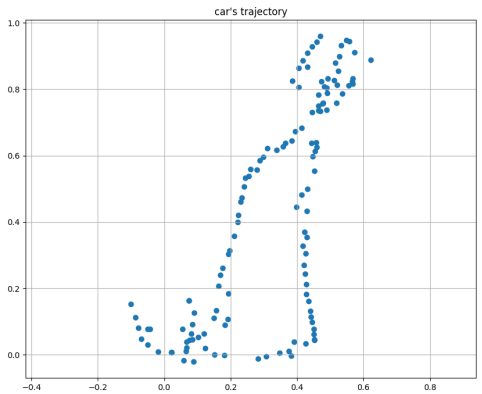

Figure 4: This figure shows the schematic of our closed-loop controller.