Report

HW2

Manoj Gayala (A69032570) **Viroopaksh Chekuri (A69032651)**

Video Link: https://youtu.be/QP5shURLXHI

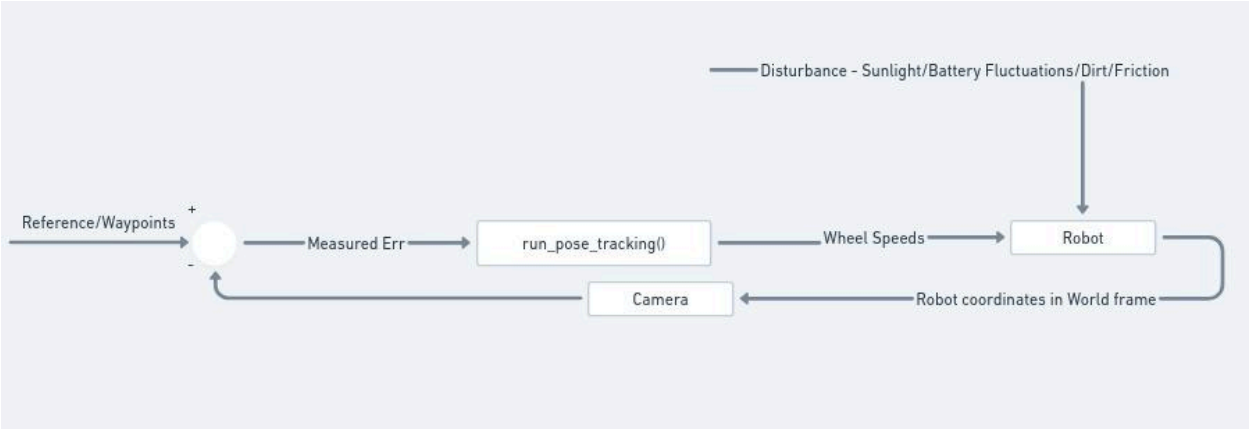
Algorithm

On a high level, we reuse the base code we developed for the first assignment and introduce the camera feedback to close the loop. Specifically we used the lateral offset from the april tag to detect if the robot is moving off the path expected and corrects it while moving towards the waypoint. The rotation doesn't use camera calibration since we realized the yaw measurement using the /tf messages has some calibration issues which we don't want to introduce in our current working model.

Calibration

We set the KP_CORRECTION = 15.0 which is a coefficient for the lateral error we see wrt the line connecting the camera and the april tag. So this ensures that based on the error either the left or right wheel speeds are increased or decreased to ensure the error becomes 0. The fluctuations in battery are taken into consideration and it works well when the battery is 4 points. Other calibrations are same as mentioned in hw1, where the K_L = 0.09 and K_R = 0.10 for each wheels since intrinsically the left wheel is a little faster. We also cap the angular speed in case of rotation to 0.345.

Closed Loop Controller



As you can see the initial input to the system is the waypoints. We measure the error which is the distance between the current position and waypoint to reach and also the orientation difference. Then we run the pose tracking where the input would be the

position of robot and waypoints. It returns the wheel speeds which are calibrated based on the feedback from the camera. This is sent to robot which also takes the disturbance into account to move and give the final world coordinates. Now these coordinates would change the position of april tag wrt camera that is used to close the loop and error is measured wrt these new locations and the loop continues.

Landmark Description

We placed 3 landmarks (april tags - family tag36h11) at the positions (2, 0, pi) and (1, 3, -pi/2) and (-0.5, -0.5, arctan(2)). The reason we placed these tags here is to ensure that the robot uses these tags to correct its orientation when it moves in a straight line. We observed that placing the tags in other locations have camera calibration issues due to which we need to correct the values of translation and rotation /tf values, so we didn't take that approach.

Car Pose Estimation

As mentioned above, we use the april tag to understand the lateral error of how much the car has moved to the left or right compared to the straight line joining the camera center and the april tag center. Since there could be some noise in measuring the lateral offset, we handle it by having a coefficient for the same and couple this with actual L and R values for the car so that effect doesn't completely effect the trajectory of the car. This solution is scalable for a general scenario since we are not giving too much power to either dead reckoning logic or camera based loop feedback.

No Landmark Scenario

In cases when the landmark is not visible, the car just spins at the same location until it finds the tag in sight and starts moving towards that direction. But this doesn't occur most of the times since we ensure that the lateral offset ensures that the car could always face the april tag. And during rotation, it follows predefined set of commands so it doesn't get effected due to lack of april tags detected during motion.

Smoothness

The car movement is sometimes smooth while sometimes it is not. I think this is correlated with the speed the robot moves. We plan to reduce the speed of robot in future assignments to make it more smoother. The rotation is relatively smoother than last assignment after few changes. But there are no sudden direction changes or jumps when we run the robot. The lateral correction is relatively smooth since the value of lateral error decreases steadily as the robot correct its path.

Total Moving Distance

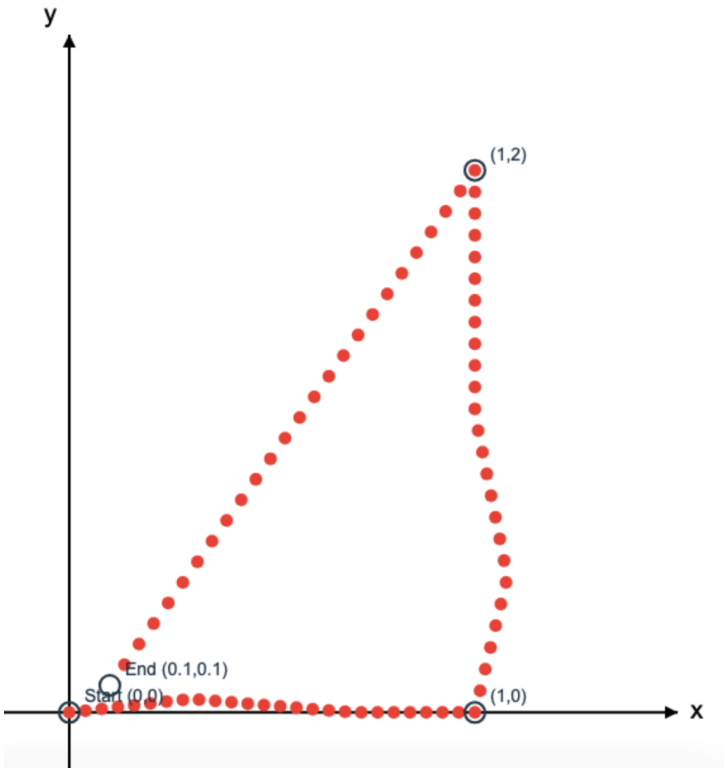
A rough measurement gives us a total translational distance of (1+2+2.236 = 5.236m) and total rotational distance of (pi + pi = 2pi radians) and given the radius of wheel is around 0.05m, the total is 5.550 metres + some extra distance covered due to adjustment of the pose wrt tag which sums upto a total of around **6 meters**.

Localization Error

The localization error in this setup refers to how far is the robot compared to its expected point. For each point (x, y, w) in the waypoints the average error is as follows,

- (0, 0, 0) => The error is 0 since this is where we start the robot from.
- (1, 0, 0) => The translation error is **0.05m** and the rotation error is around **5 degrees** (very minimal)
- (1, 2, pi) => The translation error is **0.05m** and the rotation error is around **10 degrees** (it turned slightly more than 180 degrees)
- (0, 0, 0) => The translation error is around **0.1m** (it stopped a little away from (0,0) and we think it is due to battery fluctuations) and the rotation error is around **5 degrees**.

2D Trajectory of Car



Reflection

Our overall performance seems to be as expected. As mentioned above, the performance includes both translation and rotation. The translation seems to be working as expected but the rotation could be improved. As you can observe in our localization error, the error in rotation could be fatal to overall performance since a slight difference in angle could lead the robot to an entirely different location than the expected position. This is due to the fact that the gap between two lines at some small angle 0.2 radians could increase as per **0.2K** meteres where K is the distance. The yaw error should also be calibrated and taken into consideration to ensure higher performance.

Coordinate System

Here are the coordinate systems,

World Coordinate System: This is the global reference frame for the environment which is defined at the corner of the robot where the first tape is marked. We ensured that this would be our first waypoint for simplicity. All the coordinates of waypoints are measured in this environment. The origin is (0, 0, 0) and the x-axis is the horizontal line toward east and the y-axis is the vertical line towards the north and z-axis is the upward line.

Vehicle Coordinate System: This is the position of the objects in the environment wrt the robot. This is set on the robot's base and in our case it would be at (0.1, 0.05, 0) wrt the world coordinates in the beginning based on our measurement of robot coordinates. This moves as the robot moves and we apply translational and rotational transformations to get the actual coordinates of objects wrt the world coordinate system.

Sensor Coordinate System: The sensor in this case is the camera and all the /tf messages we receive are in this coordinate system. The interesting observation here is that the z-axis measures how far the object is wrt the camera. The positive values on x-axis mean that the object is to the right wrt camera and negative means it is to the left. The y-axis is positive is it is above and negative if below.

Start Pose: The object is placed at the (0, 0, 0) where the third coordinate is the orientation which is 0 degrees.