Homework-1

CSE 276A Fall'24 — Introduction to Robotics

Pranav Malpure — **PID.: A69030926**

Akshar Tumu — **PID.: A69032632**

Setting

The setting of the robot will play an important role in all our future experiments. Considering the availability of the setting and its impact on the robot we chose that we will go ahead with a **carpetted** surface present in our housing dormitory. The other thing that will play a role in our experiments is the battery charge on the robot. While maintaining a constant battery charge is not in our hands, we chose to try keeping the **battery charge** corresponding to **5** LEDs on the battery to maintain a common ground for all future experiments. We decided to scale the waypoints by 0.5 to accommodate the model in the available space.

Calibration

Calibration involved figuring out the corresponding mapping between the commands given to the motors through the *setFourMotors* function and the actual rotation that they undergo on the surface. There were **two** variables to be tuned during this process, the **value given to the function**, and the **time** it requires to cover an exact distance of 1m. After a few initial observations, we found that this was not a linear relation, and hence we resorted to finding the time for every speed in multiples of 10, from 30 to 70. There was an **extra calibration error** which we encountered here. The robot did not go on a straight line, but diverted to the right, even when all the 4 commands given to the robot were equal implying that there was a syncing error in some of the motors. After a few trials and errors, we figured out that the front-left and back-left motors rotated a little more than the others. We added a small epsilon value to the right front and back motors, thus correcting the error. The table below summarizes our experiments and their observations:

Function Value Passed to all	Time	Real world speed(m/s) of robot	Epsilon
70	3.7	0.27	vfr:+3, vbr: +3
60	4.4	0.22	vfr:+2, vbr: +2
50	5.4	0.185	vfr:+2, vbr: +2
40	7.7	0.13	vfr:+2, vbr: +2
30	10	0.03	vfr:+2, vbr: +1

Table 1: Linear Velocity observations

Further, we did this for the rotation of the robot by 360 degrees and observed the required two variables. The rotation also included an added involvement of slipping and sliding, hence we decided to cap the values from 35 to 60 and observed in multiples of 5. We didn't observe a need to add any epsilon value to correct the errors in motor syncing in this case. The observations that we obtained are as follows:

Function Value Passed to all	Time	Real world ang. speed(rad/s) of robot
60	3.8	1.65
55	4.25	1.47
50	4.8	1.30
40	6.04	1.04
35	8	0.78

Table 2: Angular Velocity observations

Our objective is to obtain an **omega factor** from the above two observation tables, which relates the angular velocity of the wheel with the function value to be passed to the wheel motors. This is how we calculate this, taking the first observation as an example(70, 3.7, 0.27):

radius of the wheel = 0.03m

number of rotations of the wheel per 1 metre distance covered = $1/(2*\pi*0.03) = 5.30$

angular speed of the wheel = rotations$*2*\pi$/time = $5.30*2\pi/3.7 = 9.0$ rad/sec

omega factor = $70/9.0 \approx 7.77$

We do the above calculation for every case and these are the results:

Function Value:	70	60	50	40	30
Omega Factor:	7.77	7.92	8.1	9.24	9.008

Table 3: Omega Factor calculations

The next calibration check was how do these values scale with changing distance reducing to 0.5m instead of 1m. While the relation was not linear, on observing the time required for covering 0.5m and 180 degrees respectively for linear and angular velocities, we observed that there was a constant **delay** of 0.3sec, accounting for the in robot communication and actuator electronics delays. That is, the time required for 0.5 m distance for function value 40 was 3.6 seconds. This 3.6 seconds contains a delay of 0.3sec, so the motor actually ran for 3.3 seconds for 0.5m, so 6.6 seconds for 1m, and adding the delay of 0.3sec back, we get 6.9 seconds for 1m, which is quite close to 7 seconds we calculated earlier for 1m.
We repeated the above experiments for the other values and they matched as well!

Navigation Algorithm/Kinematic Model

We decided to go ahead with a constant angular and linear velocity algorithm. This also helps us to get the kinematic model and the matrix providing us with the relations between real-world linear/angular speeds and the function values to be passed to the motors.

The navigation algorithm is as such. We process the waypoints obtained in a loop. The robot aligns itself to the direction of the next waypoint by moving an angle calculated by the equation: $arctan((y_{next} - y_{current})/(x_{next} - x_{current})$. Then it traverses to the next waypoint linearly at a constant speed. After reaching the waypoint, we again align the robot to the required pose at the waypoint. We repeat this in a loop until we reach the final waypoint. In this process, we simultaneously keep updating the current pose of the robot by multiplying the real-world speeds we supply to the kinematic model for a time = linear distance/speed or angle/ang. speed.

We decided to go ahead with a speed of 0.13m/s for the linear case and a speed of 1.04 rad/sec for the angular case. The reason is that the omega factor obtained for these two sets of speeds gave the most consistent results. Now once we insert this into the kinematic model matrix, we get the corresponding ω_i's to be supplied to the four motors, with an omega factor that we calculated while calibration.
Given below is the kinematic model that we used:

$$\begin{bmatrix} vx \\ vy \\ \omega_z \end{bmatrix} = (r/4) * \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -1/(lx+ly) & 1/(lx+ly) & -1/(lx+ly) & 1/(lx+ly) \end{bmatrix} * \begin{bmatrix} \omega 1 \\ \omega 2 \\ \omega 3 \\ \omega 4 \end{bmatrix}$$

- vx - Velocity of the robot along X-axis
- vy - Velocity of the robot along Y-axis
- ωz - The angular velocity of rotation of the robot
- lx - Half of the distance between the two front/back wheels of the robot
- ly - Half of the distance between the centers of the front and back wheels on the left/right side
- r - Radius of the wheel

We calculated lx, ly and r from the dimensions of the robot:

- lx = 0.0675 m
- ly = 0.057 m
- r = 0.03 m

Without loss of generality, we assumed that the radius of each wheel is the same. Similarly, we assumed that lx is the same when calculated for front and back wheels and ly is the same when calculated for wheels on the left and right side. We also assumed that the robot is symmetrical along the X and y axes, giving us the said kinematic matrix. These are reasonable assumptions as the actual dimensions of the robot are close to being symmetrical.

Results

Upon implementing the proposed navigation algorithm to navigate the robot through the given waypoints, we observed a distance of 0.5m between the expected endpoint and the actual endpoint. This was occurring primarily due to the cumulative errors of the angular rotation of the robot during its alignment in the direction of the next waypoint or the required pose when it reaches a waypoint.

Possible sources of error were from the variability in electronic communication, play/backlash in the wheels, and surface friction. Another possible error is the change in battery level between calibrating and testing the robot.

Another result is the finding of the omega factor for our robot which is 9.24.

Possible Improvements

We intend to model the omega factor as a function that can take into consideration all the speeds' observations and approximately map all the velocities rather than just one accurately.

We also plan to improve the kinematic model to make it function as a closed loop with varying speeds depending on the error.

Youtube Unlisted Video Link

Click ***here*** for the link: https://youtu.be/vQpAe2chEek