Mobile Robotics - Homework 2

Kevin Robb

October 26, 2021

1 Differential drive kinematics on $SE(2)$

This robot has wheel radius r and track width w.

In the world frame W, the robot has a pose described by the parameters

$$t_{WR} = (x_{WR}, y_{WR}) \in \mathbb{R}^2$$
$$\theta_{WR} \in \mathbb{R}.$$

The equations of motion for this vehicle are

$$\begin{pmatrix} \dot{x}_{WR} \\ \dot{y}_{WR} \\ \dot{\theta}_{WR} \end{pmatrix} = \begin{pmatrix} R(\theta_{WR}) & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \frac{r}{2}(\dot{\phi}_r + \dot{\phi}_l) \\ 0 \\ \frac{r}{w}(\dot{\phi}_r - \dot{\phi}_l) \end{pmatrix} \tag{1}$$

where $\dot{\phi}_r$ and $\dot{\phi}_l$ are the angular speeds of the left and right wheels, with positive values corresponding to forward motion. The rotation matrix $R(\theta_{WR})$ is defined as

$$\begin{aligned} R &: \mathbb{R} \to SO(2) \\ R(\theta) &= \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}. \end{aligned} \tag{2}$$

This creates the rotation matrix for a particular angle θ.

We will be reformulating Eq. (1) on the Lie group $SE(2)$, which is expressed as

$$SE(2) \cong \left\{ \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \in \mathbb{R}^{3\times 3} \;\middle|\; R \in SO(2), t \in \mathbb{R}^2 \right\}. \tag{3}$$

We will apply Lie group theory to perform forward simulation of the vehicle's trajectory.

1.a

We have the mapping

$$\Psi : \mathbb{R}^2 \times \mathbb{R} \to SE(2)$$
$$\Psi(t, \theta) = \begin{pmatrix} R(\theta) & 0 \\ 0 & 1 \end{pmatrix}, \tag{4}$$

which sends each pose (t, θ) to the element of $SE(2)$ obtained by replacing the angle θ by its corresponding rotation matrix $R(\theta)$.

Note that

$$\Psi(0, 0) = \begin{pmatrix} R(0) & 0 \\ 0 & 1 \end{pmatrix} = I, \tag{5}$$

so the derivative map

$$d\Psi_{(0,0)} : T_{(0,0)}(\mathbb{R}^2 \times \mathbb{R}) \to \text{Lie}(SE(2)) \tag{6}$$

sends each tangent vector $(\dot{t}, \dot{\theta})$ to a tangent vector in $T_1(SE(2)) \cong \text{Lie}(SE(2))$, the Lie algebra of $SE(2)$.

We can compute the derivative map as

$$d\Psi(\dot{t}, \dot{\theta}) = \begin{pmatrix} dR_\theta(\dot{\theta}) & \dot{t} \\ 0 & 0 \end{pmatrix}$$

where $dR_\theta(\dot{\theta})$ is defined as

$$dR_\theta(\dot{\theta}) = \begin{pmatrix} -\sin\theta & -\cos\theta \\ \cos\theta & -\sin\theta \end{pmatrix} \cdot \dot{\theta},$$

which is the directional derivative of R at θ in the direction of $\dot{\theta}$.

Now if we take this at $x = y = \theta = 0$,

$$d\Psi_{(0,0)} = \begin{pmatrix} 0 & -\dot{\theta} & \dot{x} \\ \dot{\theta} & 0 & \dot{y} \\ 0 & 0 & 0 \end{pmatrix}.$$

This makes sense, because the Lie algebra of $SO(2)$ is the set of Skew-Symmetric matrices, and the Lie algebra of $\mathbb{R}^2$ is $\mathbb{R}^2$. This fits the expected form, and has a final row of zeros to maintain the required structure for elements of $SE(2)$.

1.b

Supposing the robot starts at the origin, $(0,0) \in \mathbb{R}^2 \times \mathbb{R}$, and given the wheel speeds $(\dot{\phi}_l, \dot{\phi}_r) \in \mathbb{R}^2$, we can derive the mapping $\dot{\Omega}$ which sends the pair of wheel speeds to the robot's velocity in the Lie group $SE(2)$ at I.

$$\dot{\Omega} : \mathbb{R}^2 \to \text{Lie}(SE(2)) \tag{7}$$

We can evaluate Eq. (1) in general to form a simple set of three equations,

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \frac{r}{2}(\dot{\phi}_r + \dot{\phi}_l)\cos\theta \\ \frac{r}{2}(\dot{\phi}_r + \dot{\phi}_l)\sin\theta \\ \frac{r}{w}(\dot{\phi}_r - \dot{\phi}_l) \end{pmatrix},$$

which we can solve at $(x, y, \theta) = (0, 0, 0)$ for

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \frac{r}{2}(\dot{\phi}_r + \dot{\phi}_l) \\ 0 \\ \frac{r}{w}(\dot{\phi}_r - \dot{\phi}_l) \end{pmatrix}.$$

We can then substitute these expressions into our result of part (a) to obtain

$$\dot{\Omega}\left(\dot{\phi}_l, \dot{\phi}_r\right) = \begin{pmatrix} 0 & -\frac{r}{w}(\dot{\phi}_r - \dot{\phi}_l) & \frac{r}{2}(\dot{\phi}_r + \dot{\phi}_l) \\ \frac{r}{w}(\dot{\phi}_r - \dot{\phi}_l) & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

1.c

The previous part assumed a robot pose in the world frame of $T_{WR} = I$. Now we instead suppose it is an arbitrary pose $T_{WR} = X \in SE(2)$. We can then create a closed-form expression of the map

$$V : SE(2) \times \mathbb{R}^2 \to T(SE(2)) \tag{8}$$

where we know that $T(SE(2))$ will be a vector of velocities in $\mathbb{R}^3$.

We can introduce an auxiliary frame F which is fixed w.r.t the world frame and is aligned with the robot such that the pose of the robot in this frame is $T_{FR} = I$. If we do this, we can see that regardless of the robot's pose, our solution for $\dot{\Omega}$ will give the velocities in the frame F, so we only need to transform this into the world frame to find our general solution. We can treat the given pose X *as this transformation*, so then our general solution becomes

$$V(X, \dot{\phi}_l, \dot{\phi}_r) = X \cdot \dot{\Omega}\left(\dot{\phi}_l, \dot{\phi}_r\right).$$

1.d

If we now fix a specific choice of wheel speeds $(\dot{\phi}_l, \dot{\phi}_r) \in \mathbb{R}^2$, the map V that we derived in part (c) simplifies to a function

$$V_{(\dot{\phi}_l, \dot{\phi}_r)}(X) = V(X, \dot{\phi}_l, \dot{\phi}_r) \in T_X(SE(2)) \tag{9}$$

that assigns a velocity to each pose $X \in SE(2)$. i.e., each choice of wheel speeds determines a vector field $V_{(\dot{\phi}_l, \dot{\phi}_r)}$ on $SE(2)$.

This fact allows us to examine our result from part (c) in a new way. We can see that for some choice of wheel speeds, $\dot{\Omega}\left(\dot{\phi}_l, \dot{\phi}_r\right)$ is an element of the Lie algebra of $SE(2)$. We are also given that the pose X is an element of the group $SE(2)$. Lastly, we acknowledge that every element of $SE(2)$ is invertible, due to its structure and the invertibility of $SO(2)$, and thus $SE(2)$ is a subgroup of the general linear group $GL(3)$. As a result of this relationship, the left-invariant vector fields look the same in both groups; moreover, the formula we derived in Lab 1, problem 2 will hold true in $SE(2)$ as well.

> For an element $A \in GL(n)$, the corresponding left-translation map is $L_A(X) = A \cdot X$, its derivative map is $dL_A = A$, and the left-invariant vector field determined by some element $\Omega \in \text{Lie}(GL(n))$ is $V_\Omega(X) = d(L_X))_I(\Omega) = X \cdot \Omega$.

We can compare this general result for a left-invariant vector field in $GL(n)$ to our result for $V(X, \dot{\phi}_l, \dot{\phi}_r)$ in part (c) and see that we have precisely the same formulation; we identify the velocities at some pose X w.r.t. the identity via an element of the Lie algebra, $\dot{\Omega}\left(\dot{\phi}_l, \dot{\phi}_r\right)$.

We can further demonstrate that our $V_{(\dot{\phi}_l, \dot{\phi}_r)}$ is indeed a left-invariant vector field by checking against the property

$$dL_g V(x) = V(gx) \tag{10}$$

for a left-invariant vector field V on a group G, where g is a group element and x identifies the Lie algebra.

Substituting in the derivative map and the vector field we have derived in previous parts yields the following. For some group element

$$g = \begin{pmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{pmatrix},$$

the derivative map is

$$
\begin{aligned}
d\Psi_g &= \begin{pmatrix} -\dot{\theta}\sin\theta & -\dot{\theta}\cos\theta & \dot{x} \\ \dot{\theta}\cos\theta & -\dot{\theta}\sin\theta & \dot{y} \\ 0 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} -\frac{r}{w}\left(\dot{\phi}_r - \dot{\phi}_l\right)\sin\theta & -\frac{r}{w}\left(\dot{\phi}_r - \dot{\phi}_l\right)\cos\theta & \frac{r}{2}\left(\dot{\phi}_r + \dot{\phi}_l\right)\cos\theta \\ \frac{r}{w}\left(\dot{\phi}_r - \dot{\phi}_l\right)\cos\theta & -\frac{r}{w}\left(\dot{\phi}_r - \dot{\phi}_l\right)\sin\theta & \frac{r}{2}\left(\dot{\phi}_r + \dot{\phi}_l\right)\sin\theta \\ 0 & 0 & 0 \end{pmatrix}
\end{aligned}
$$

and

$$
\begin{aligned}
V(gX) &= g \cdot X \cdot \dot{\Omega} = g \cdot I \cdot \dot{\Omega} \\
&= \begin{pmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & -\frac{r}{w}(\dot{\phi}_r - \dot{\phi}_l) & \frac{r}{2}(\dot{\phi}_r + \dot{\phi}_l) \\ \frac{r}{w}(\dot{\phi}_r - \dot{\phi}_l) & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} -\frac{r}{w}\left(\dot{\phi}_r - \dot{\phi}_l\right)\sin\theta & -\frac{r}{w}\left(\dot{\phi}_r - \dot{\phi}_l\right)\cos\theta & \frac{r}{2}\left(\dot{\phi}_r + \dot{\phi}_l\right)\cos\theta \\ \frac{r}{w}\left(\dot{\phi}_r - \dot{\phi}_l\right)\cos\theta & -\frac{r}{w}\left(\dot{\phi}_r - \dot{\phi}_l\right)\sin\theta & \frac{r}{2}\left(\dot{\phi}_r + \dot{\phi}_l\right)\sin\theta \\ 0 & 0 & 0 \end{pmatrix}.
\end{aligned}
$$

We can see that both these quantities match, and thus left-invariance holds. In words, the tangent space at a certain pose can be achieved by multiplying the pose by the tangent space at the identity. ■

1.e Forward kinematics

At time $t = 0$, the robot is at pose $X_0 \in SE(2)$. We drive its wheels at constant velocity $(\dot{\phi}_l, \dot{\phi}_r) \in \mathbb{R}^2$. We can find a closed-form formula for the curve

$$\gamma : \mathbb{R} \to SE(2) \tag{11}$$

that reports the robot's pose $\gamma(t)$ at time t.

We know that the exponential map can be used to propagate forward motion:

$$X_t = \gamma(t) = X_0 \exp(t\dot{\Omega}).$$

1.f Inverse kinematics

We now suppose at time $t = 0$ the robot is at pose $X \in SE(2)$ and we want to drive it to pose $Y \in SE(2)$. Using the result from part (e), we can determine what constant wheel speeds $(\dot{\phi}_l, \dot{\phi}_r) \in \mathbb{R}^2$ should be commanded so that the robot arrives at pose Y at time $t = T$.

We can use the result of part (e) to see that

$$\begin{aligned} Y &= X\exp(T\dot{\Omega}) \\ X^{-1}Y &= \exp(T\dot{\Omega}) \\ \log(X^{-1}Y) &= T\dot{\Omega} \\ \dot{\Omega} &= \frac{1}{T}\log(X^{-1}Y) \end{aligned}$$

This tells us what we need to know, since the turtlebot receives a twist as a command, where we only need dx and $d\theta$. Referring to what the entries of $\dot{\Omega}$ represent, we can see that

$$\begin{aligned} dx &= \dot{\Omega}_{1,3} \\ d\theta &= \dot{\Omega}_{2,1} = -\dot{\Omega}_{1,2} \end{aligned}$$

We could, if desired, then extract from $\dot{\Omega}$ our solutions for $\dot{\phi}_l$ and $\dot{\phi}_r$.

$$\begin{aligned} \dot{\phi}_l &= \frac{1}{r}dx - \frac{w}{2r}d\theta \\ \dot{\phi}_r &= \frac{1}{r}dx + \frac{w}{2r}d\theta \end{aligned}$$

2 Turtlebot3 Kinematics

Followed the instructions in the repository for setting up the simulation. ✓

2.a Kinematic parameters and control limits

The robot has the following constraints given:

$$\begin{aligned} w &= 0.16 \text{ m} \\ r &= 0.033 \text{ m} \\ v_{\max} &= 0.22 \text{ m/s} \end{aligned}$$

From this maximum linear speed $v_{\max}$, we can find the maximum wheel speed

$$\dot{\phi}_{\max} = \frac{v_{\max}}{r} = \frac{0.22}{0.033} = 6.667 \text{ rad/s}.$$

2.b Command velocity as a Twist

The desired velocity is sent as a Twist message, which is an element of the Lie algebra $SE(3)$. Due to the turtlebot's motion constraints, most components of this message will be zero, maintaining only the forward velocity and the yaw rate. Specifically, a commanded velocity message will take

the form

$$\dot{\mu} = \begin{pmatrix} \dot{x}_R & 0 & 0 & 0 & 0 & \dot{\theta}_R \end{pmatrix}^T$$

To make our robot follow a circular path of radius $R = 1.5$ m CCW at a linear velocity of $v = 0.2$ m/s, we will need to send the command

$$\dot{\mu} = \begin{pmatrix} 0.2 & 0 & 0 & 0 & 0 & 0.1333 \end{pmatrix}^T.$$

We arrive at these values by taking $\dot{x}_R$ to be the provided forward velocity and using $\dot{\theta}_R = \dot{x}_R / R$ to find the yaw rate. To check if this is dynamically feasible, we need to extract the wheel speeds corresponding to these commands, and ensure both are within the range $\left(-\dot{\phi}_{\max}, \dot{\phi}_{\max}\right)$

Using our results for inverse kinematics, we can find the commanded wheel speeds

$$\dot{\phi}_l = \frac{1}{r} dx - \frac{w}{2r} d\theta = 5.73$$
$$\dot{\phi}_r = \frac{1}{r} dx + \frac{w}{2r} d\theta = 6.38$$

This is thus dynamically feasible, since neither speed value exceeds $\dot{\phi}_{\max} = 6.667$.

2.c Check in simulation

We can easily simulate this command by first starting the environment with an empty world, and then publishing our command in the provided format.

```
roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
rostopic pub /cmd_vel geometry_msgs/Twist -r 10 '[0.2, 0, 0]' '[0, 0, 0.1333]'
```

The robot is able to complete a CCW circle of radius 1.5 m and return to its start point with this command. ✓

A video of this running has been uploaded to Canvas alongside this document.

3 Visual Servoing

This exercise involves driving the turtlebot to a target pose specified by an AprilTag using a simple ROS node.

We subscribe to the "/tag_detections" topic to receive the AprilTag's pose in the robot's camera's frame of reference. We could use this coordinate system to avoid needing the robot's position in the world frame by simply assuming the robot's current position is always at the origin, which for this system (and projected down to the xy-plane) is true, and simply continue sampling the AprilTag's pose until it matches the inverse of our desired goal pose relative to it; unfortunately, the robot is

unable to detect the AprilTag once it is within 0.5 m, meaning we can't use this method to achieve a goal pose closer than that.

Instead, we use the robot's starting pose as the origin of the world frame, and measure the AprilTag's position once. This gives us the transformation between the camera and the tag frames. We can use the tf listener to obtain the transform between the robot's base and camera frames. We can manually create an affine matrix that transforms the tag to the desired goal pose (using the provided instructions for this relation). Thus, we can compose all these translations to obtain the goal pose.

Using these and our result for the inverse kinematics from (1f), we can find the necessary motor commands to run the robot at a constant speed to reach the target. We simply generate and send the command, sleep for the chosen amount of time T, then send a halt command. **NOTE: this model assumes we are capable of moving at the requested speed for the entire duration, which is not the case in the simulation or with a real robot; as a result, the robot will not achieve the exact correct position with this method, since it takes the motors time to accelerate to the requested speeds.** I programmed a more complicated method which used odometry measurements to ensure it halted as close to the goal pose (without going too deep into making good localization code), but this seemed outside the scope of the expectations for this problem, so I changed back to this simple form that aligns more with the rest of this assignment.

I also want to mention that since this code relies on timing by necessity, when I have gazebo, rviz, and my node running in addition to a screen recording software, everything lags and runs more slowly, further preventing the robot from moving as far as it should. I've submitted a video taken from my phone of the screen, which I know is bad, but the program was able to run a lot better without the burden of screen recording software.

I have made a GitHub repository for this assignment as requested. My git username is "kevin-robb", and the repository can be found here. I've added Tarik as a collaborator.

My videos have been uploaded to Canvas as well.