Experiment 2

Inverse Kinematics and Simple Motions

Time to Complete: See course outline | Marks: 30 (prelab 25, lab 5)

Purpose

In this experiment, the three-wheel drive (3WD) mobile robot to be used through out this course is introduced. The ELE719 Python module `Robot3WD` for controlling the motion of this robot is also introduced.

Objectives

By the end of this experiment, you will be able to:

1. Obtain the inverse kinematics equation for this robot.
2. Extend the `Robot` object class in the `Robot3WD` module with additional class methods that implement the inverse kinematics equation and simple motions for the robot.
3. Program the robot to perform simple motions using the extended object class.

2.1 Preparatory Work

Consider the kinematics model of the 3WD robot in Figure 2.1, where the **fixed** base frame is $\mathcal{F}^0 = o_0 x_0 y_0$, and the robot frame is $\mathcal{F}^R = o_R x_R y_R$. The robot frame $\mathcal{F}^R$ is always fixed on the robot, but its position and orientation will change w.r.t. the base frame $\mathcal{F}^0$ as the robot moves around. The position of the robot with respect to $\mathcal{F}^0$ is given by (x, y) and its orientation by the angle θ. The vector $p = [x, y, \theta]^T$ represents the current *pose* of the robot. The *linear* velocities of the right, left and back wheels are denoted by the vectors v_1, v_2 and v_3 respectively. These vectors point to the direction of the *linear* motion produced by *counter-clockwise* rotation of the respective wheel. Let the radius of the wheel be r, the angular displacement of wheel i be ϕ_i, then the angular velocities of the wheels are $\dot{\phi}_i = v_i/r,\ i = 1, 2, 3$.

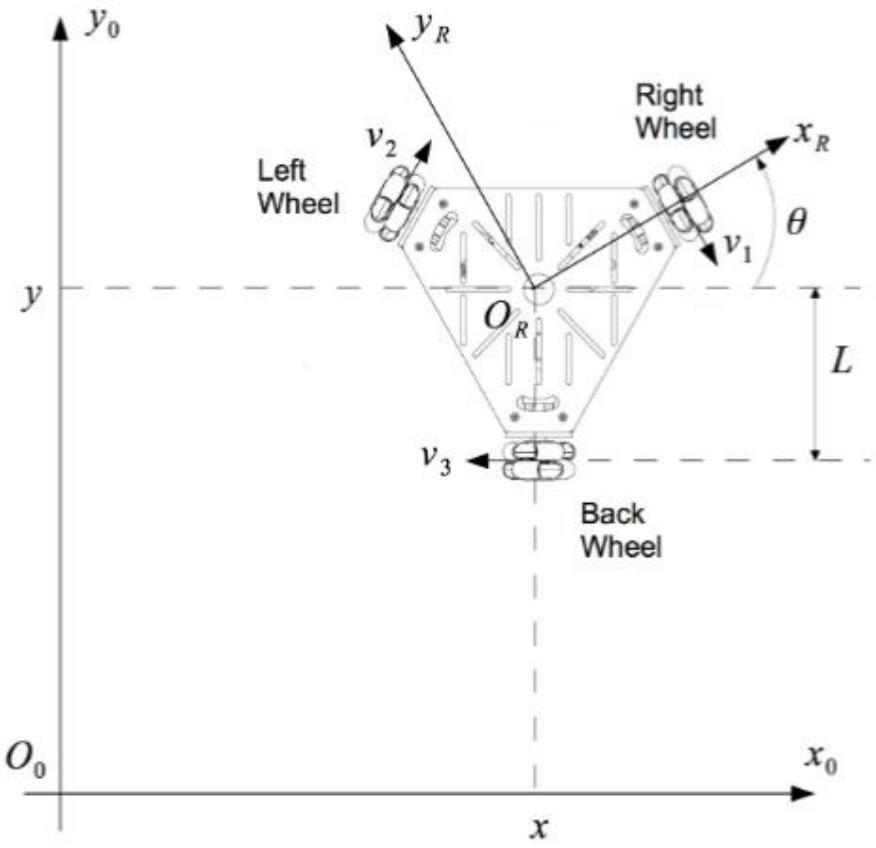

Figure 2.1: 3WD robot

Using the given notations,

1. Show that the inverse kinematics equation for the 3WD robot is given by:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} r\dot{\phi}_1 \\ r\dot{\phi}_2 \\ r\dot{\phi}_3 \end{bmatrix} = \begin{bmatrix} \sin(\theta) & -\cos(\theta) & -L \\ \cos(\frac{\pi}{6}+\theta) & \sin(\frac{\pi}{6}+\theta) & -L \\ -\cos(\frac{\pi}{6}-\theta) & \sin(\frac{\pi}{6}-\theta) & -L \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{2.1}$$

or,

$$\dot{\Phi} = M(\theta)^{-1}\dot{p} \tag{2.2}$$

where $\dot{\Phi} := [\dot{\phi}_1, \dot{\phi}_2, \dot{\phi}_3]^T$, $\dot{p} := [\dot{x}, \dot{y}, \dot{\theta}]^T$, and

$$M(\theta)^{-1} := \frac{1}{r}\begin{bmatrix} \sin(\theta) & -\cos(\theta) & -L \\ \cos(\frac{\pi}{6}+\theta) & \sin(\frac{\pi}{6}+\theta) & -L \\ -\cos(\frac{\pi}{6}-\theta) & \sin(\frac{\pi}{6}-\theta) & -L \end{bmatrix}.$$

2. Define a new python class `myRobot` by extending the `Robot` class in the `Robot3WD` module to provide an additional class method `inverse_kinematics()` that implements the inverse kinematics equation (2.1). At this point, you do not need to know the details of the `Robot` class or the `Robot3WD` module. All you need to do here is to follow the instructions below to implement Eq. (2.1) with Python code.

   Start up the Ubuntu Studio VM, login in and go to the folder: `ele719/controllers` and edit the file `myRobot.py`. You will find the following Python code at the top of the file:

```
from math import pi, sin, cos, sqrt, exp
from numpy import array, dot

from Robot3WD import Robot

class myRobot(Robot):
    def __init__(self, sampling_period, wheel_radius=None, L=None):
        Robot.__init__(self, sampling_period, wheel_radius, L)
```

   The above code simply defines a new object class called `myRobot` which *inherits* the properties and methods of the object class `Robot` defined in the `Robot3WD` modules. In other words, the new object class `myRobot` will have access to all the properties and methods of the `Robot` class. To extend the `Robot` class, simply add the code for the `inverse_kinematics()` method to implement Eq. (2.1) into the file `myRobot.py`. A template for this method has been included in `myRobot.py`:

```
def inverse_kinematics(self, p_dot, theta):
    L = self._L
    wheel_radius = self._wheel_radius
    #... (Fill in rest of the code here) ...
    return wheel_angular_velocities
```

3. Further extend the `Robot` class by providing additional methods to perform the following simple motions (defined with respect to the base frame $\mathcal{F}^0$): (a) move forward with linear velocity v, (b) move backward with linear velocity v, (c) move left with linear velocity v, (d) move right with linear velocity v, (e) rotate counter-clockwise (CCW) with angular velocity ω, and (f) rotate clockwise (CW) with angular velocity ω.

   As an example, consider the motion: move left with linear velocity v. Since the motion are defined with respect to $\mathcal{F}^0$, moving left with velocity v simply means moving in the negative x direction of $\mathcal{F}^0$ at velocity v, or in the positive x direction with velocity $-v$ (see Figure 2.1). Thus, the required velocity vector of the robot is $\dot{p} = [-v, 0, 0]^T$ and the required angular velocities of the wheels to execute this motion can be obtained using the inverse kinematics equation (2.1). Therefore, this motion can be easily implemented using the following method:

```
def move_left(self, v, theta):
    p_dot = [-v, 0.0, 0.0]
    PHI_dot = self.inverse_kinematics(p_dot, theta)
    self.set_angular_velocities(PHI_dot)
```

   The above code first defines the velocity vector $\dot{p}$, uses the `inverse_kinematics()` method defined earlier to determine the required angular velocities of the wheels $\dot{\Phi} = [\dot{\phi}_1, \dot{\phi}_2, \dot{\phi}_3]^T$, and then uses the `set_angular_velocities()` method from the `Robot3WD` module to move the robot.

   Using the above code as a guide, implement the remaining simple motions with the following methods in `myRobot.py`:

```
def move_forward(self, v, theta):
    #... (Fill in rest of the code here) ...

def move_backward(self, v, theta):
    #... (Fill in rest of the code here) ...

def move_right(self, v, theta):
```

```
        #... (Fill in rest of the code here) ...

    def rotate_CCW(self, w, theta):
        #... (Fill in rest of the code here) ...

    def rotate_CW(self, w, theta):
        #... (Fill in rest of the code here) ...
```

4. Using the sample program `lab2.py` in Section 2.2 as a guide, develop a Python program, `lab2demo.py`, for the robot to execute the following motions *in the given order*:

   (a) Move forward with velocity of 0.2 m/sec. for 2.5 sec., followed by

   (b) Move right with velocity of 0.2 m/sec. for 2.5 sec., followed by

   (c) Move backward with velocity of 0.2 m/sec. for 2.5 sec., followed by

   (d) Move left with velocity of 0.2 m/sec. for 2.5 sec., followed by

   (e) Rotate counter-clockwise with velocity of 2 rad/sec. for 2.5 sec., followed by

   (f) Rotate clockwise with velocity of 2 rad/sec for 2.5 sec..

   You program should be very similar to `lab2.py`, with the exception that the following statements in `lab2.py`:

   ```
   from Robot3WD import Robot
   ...
   myrobot = Robot(sampling_period)
   ```

   be replaced by

   ```
   from myRobot import myRobot
   ...
   myrobot = myRobot(sampling_period)
   ```

   A template for this program, `lab2demo.py`, can be found in `ele719/controllers/lab2demo`. You may wish to test the method for each motion separately before running all of them together.

2.2 Virtual Laboratory Session

The robot to be used in the laboratory experiments for this course is a three-wheel drive robot with 3 omni-directional wheels. Each wheel of the robot is actuated by its own DC motor with an encoder providing feedback on the angular displacement of the wheel. This robot also has 6 infrared sensors capable of measuring distances in the range of 4 to 30 cm. These sensors will be used in later experiments for simple obstacle avoidance path planning. An inertia measurement unit (IMU) is also available to provide information on the orientation of the robot.

A Python module, Robot3WD, has been developed for this course to provide transparent access to the various hardware components of the robot. Using this and other supporting Python modules, the robot can be controlled using very simple and straight forward code. The entire Robot3WD Python module contains hundreds lines of Python, C and assembly language code. However, from a user's point of view, only several special functions (class methods) in the library are relevant. In particular, the hardware of the 3WD robot can be accessed through the Python object class called `Robot` and its associated methods to be described in a moment.

You are now ready to explore the `Robot3WD` module. Follow the video instruction on D2L to start up the Webots Robot Simulator. Go to the "`File`" menu and click on "`Open World`". Navigate to the `ele719/worlds`" folder and select the world file `lab2.wbt`. Expand the "`DEF OMNI_3_WHEEL Robot`" node and navigate down to select the "`controller`" field. Click on the "`Select`" button and select "`lab2`" in the window that pops up. Click on the "`Edit`" button to open the controller program "`lab2.py`" in the editor window. Consider the controller program `lab2.py` (where line numbers are added for explanation purpose and are not part of the code): Line 1 of the code loads the definition of the `Robot` defined in the `Robot3WD` module. Lines 2 and 3 load the `math` and `time` modules. Line 5 defines the sampling period to be used by the controller. Line 7 sets the run time of the control loop to 5 seconds. Line 9 defines the variable `w` which is the angular velocity for the wheels and sets it to 8 rad/sec. Line 11 defines an instance of the `Robot` object, assigns it to `myrobot` and sets the sampling period for the controller to 0.02 second (i.e. a sampling frequency of 50 Hz). Line 15 uses the `Robot` class method `initialize()` to initialize the robot. Note that the `initialize()` method requires the initial orientation of the robot, θ_0, as an argument. (Recall that, from Figure 2.1, the orientation of the robot is the angle of rotation θ from the base frame $\mathcal{F}^0$ to robot frame $\mathcal{F}^R$.) In this case the initial orientation is assumed to be 30°, but it has to be converted to radian before being passed to the `initialize()` method.

Once the robot is initialized, the motors and IMU will be calibrated and several threads will begin to run in the background to provide readings from the wheel encoders, IR sensors and the IMU. Lines 20 to 26 is the control loop which will execute for `run_time` = 5 sec-

```
1 from Robot3WD import Robot
2 import math
3 import time
4
5 sampling_period = 0.02
6
7 run_time = 5.0
8
9 w = 8.0
10
11 myrobot = Robot(sampling_period)
12
13 theta_0 = 30.0*math.pi/180.0
14
15 myrobot.initialize(theta_0)
16
17 elapsed_time = 0.0
18 start_time = time.time()
19
20 while (elapsed_time < run_time):
21     myrobot.get_readings_update()
22     print('angular velocities of wheels = ', myrobot.angular_velocities)
23     print('ir sensors raw values = ', myrobot.ir_sensors_raw_values)
24     print('robot orientation = %6.1f degrees.\n' % (myrobot.orientation*180.0/math.pi))
25     myrobot.set_angular_velocities([w, w, w])
26     elapsed_time = time.time() - start_time
27
28 print('Elapsed Time = ', elapsed_time)
29
30 myrobot.stop()
31 myrobot.close()
```

Figure 2.2: lab2.py

onds. The first line of the control loop (Line 21) calls the `get_readings_update()` method to retrieve current readings of the wheel encoders, IR sensors and the IMU. The angular velocities of the wheels, readings of the IR sensors and orientation of the robot are defined as *properties* of the `Robot` class and can be accessed as in Lines 22 to 24. In Line 22 the angular velocities of the 3 wheels are returned as an array of 3 elements and printed as follows: the angular velocity of the right wheel is first element (index 0), the left wheel is the second element (index 1) and the back wheel is third element (index 2). In Line 23, the raw values of the IR sensor readings are returned as an array of 6 elements and printed as follows: The raw reading of sensor 0 is returned as the first element (index 0), sensor 1 as the second elements (index 1), etc. The orientation of the robot (i.e. the angle θ in Figure 2.1) is measured by the IMU and can be obtained and converted to degrees and printed as in Line 24.

As discussed in the lecture, the linear and angular velocities of the robot can be con-

trolled by setting the angular velocities of its wheels to the required values. Once the required wheel velocities are determined, they can be sent to the robot for execution using the `set_angular_velocities()` method as follows:

```
>>> myrobot.set_angular_velocities([phi1_dot, phi2_dot, phi3_dot])
```

where `phi1_dot`, `phi2_dot` and `phi3_dot` are the required angular velocities of the right, left and back wheels respectively (see Figure 2.1). In the above code, Line 25 sets the angular velocity of all three wheels to `w` = 8 rad/sec. This causes the the robot itself to rotate clockwise at a uniform angular velocity. To stop the robot, one can set all its angular velocities to 0, or simply use the `stop()` method as in Line 30. Finally, the `close()` method should be used to terminate the robot control program as in Line 31.

Now that you are familiar with the `Robot3WD` module, review the program `lab2demo.py` developed in in Part 4 of the Preparatory Work to check if changes are required.

Task 2.1 *Simple Motions*
Start up the Webots Robot Simulator, open the world file `lab2.wbt` and change the controller to `lab2demo` and run the controller program `lab2demo.py`. Use an initial orientation of $\theta_0 = 30°$ (or $\frac{\pi}{6}$ radian) in the controller program. To be successful, the robot should return close to its starting position and orientation.

2.3 What to Submit

A report is not required for this experiment, but the following are required:

1. Part 1 of the Preparatory Work.
2. Python code developed for this experiment: `myRobot.py` and `lab2demo.py`.

Archive the prelab work and Python code together into a .tar file (using your last name as the name of the tar file) and email it as an attachment to your TA. For example,

```
cd
cd ele719/controllers
tar cvf lastname-lab2.tar prelab2.pdf myRobot.py lab2demo/lab2demo.py
```