1. Localization

*Dead reckoning* is a way to keep track of the position of a mobile robot with reference to a starting position (coordinate frame). See Section 5.4: Navigation by Odometry in the textbook for a detailed discussion. Figure 1 shows world coordinate frame OXY and the moving robot frame (meaning the coordinate frame is attached to the moving robot), OX’Y’ at the starting time when OX’Y’ on top of OXY.

The accuracy of the reckoning relies heavily on the accuracy of the sensors (motor encoders and the Gyro sensor) and how fast the sensor data is sampled to update the position and orientation. In this lab, we limit the motion of the robot to pure translations and rotations to simplify and improve the accuracy of robot tracking. Figure 2 shows examples of pure translation and rotation motion of the robot. For instance, let the starting location of the robot be [𝑥, 𝑦, 𝜃].

If the robot makes a Δ𝜃 *pure rotation* in the clockwise direction, the new location of the robot becomes [𝑥, 𝑦, 𝜃 + Δ𝜃]. Note that the x and y coordinates remain unchanged.  
If the robot moves Δ𝑥 distance forward from the original location [𝑥, 𝑦, 𝜃], the new robot coordinates become [𝑥 + Δ𝑥, 𝑦, 𝜃]. Note the orientation of the robot remains unchanged.

*Pure translation after a pure rotation:*

If the robot makes Δ𝜃 rotation and 𝑙 translation in the +X direction (see Figure 3), the change of robot coordinates with respect to OXY, are Δ𝑥 = 𝑙 cos Δ𝜃, Δy = 𝑙 sin Δ𝜃. Similarly, we can keep updating the robot location if we account for the position and orientation change of the robot to know the robot position with respect to the starting location (i.e., the world coordinate frame).

*Figure 1: Robot and world coordinate frames*

Q1: *Rotation and Translation One after the other*

In this question, you will implement the position tracking algorithm on the EV3 robot and verify in the robot virtual world environment. We can calculate the position change using the motor encoder information and rotation using the gyro heading.

* 1. a)  Implement Tasks to interact with the motors and display. Use 20 Hz and 10 Hz task frequencies. Main Task runs at 50 Hz.
  2. b)  Create a PositionTrack Task that runs at 50 Hz. Implement the following pseudocode in this Task.

Do forever at 50 Hz – convert to the corresponding sleep time

prevRightMotorEncoder <= rightMotorEncoder

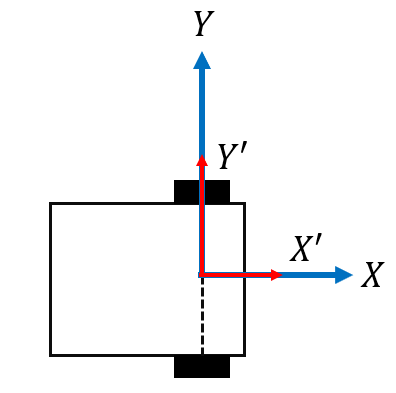
rightMotorEncoder <= read motor encoder value of the right

motor

wheelAngleChange <= convert to radians (rightMotorEncoder-

prevRightMotorEncoder)

2



currentHeading <= convert to radians (-1\* read gyro

heading)

Once the rotation has ended, find the x and y as follows:

x <= x + r\* wheelAngleChange\*cos(currentHeading)

y <= y + r\* wheelAngleChange\*sin(currentHeading)

Important:

* 1. prevRightMotorEncoder, rightMotorEncoder, turnRatio, motorPower variables

are integers

* 1. wheelAngleChange, x, y, currentHeading variable are floats
  2. r is the wheel radius, 28 mm.
  3. x, y are given in mms. The currentHeading is given in radians.
  4. (-1) multiplication is to change the sign of rotation as counterclockwise rotation is

considered positive in coordinate systems.

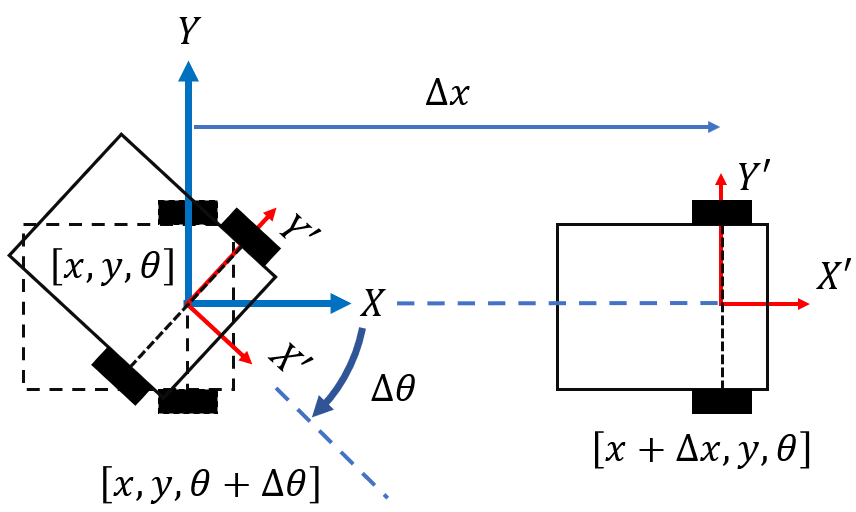
c) Implement a state machine to **move the robot forward (without turning) by turning right wheel 5 times (1800 encoder counts)**. In the display Task, display x,y, and current heading. Check the accuracy of the location tracking algorithm by running the program on the Metric Distance Utility challenge under Utility Tables of Robot Virtual Worlds (RVW).

*Figure 2: Pure translation and rotation motions*

Q2. *Testing the Algorithm*In this question, we will move the robot in a square shape lap and measure the accuracy of

location tracking. Assume robot wheel radius is 28 mm. a) Design a state machine to

3



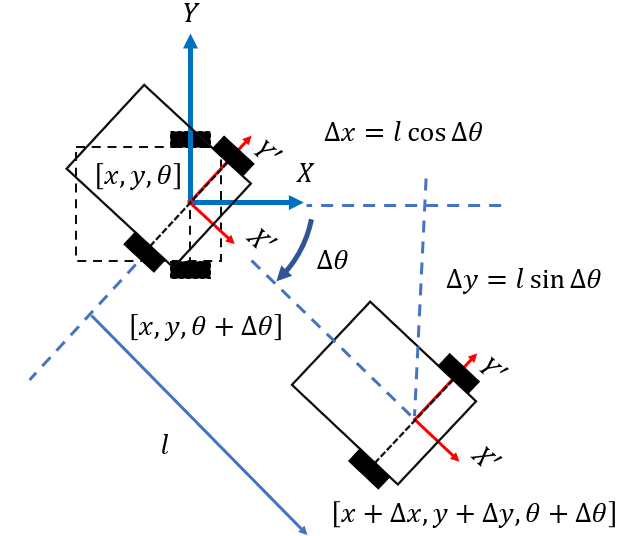
* 1. Move 1800 encoder counts forward
  2. rotate 90 degrees counterclockwise
  3. move 720 encoder counts forward
  4. rotate 90 degrees counterclockwise
  5. move the robot 1800 encoder counts forward
  6. rotate 90 degrees counterclockwise
  7. move 720 encoder counts forward, and

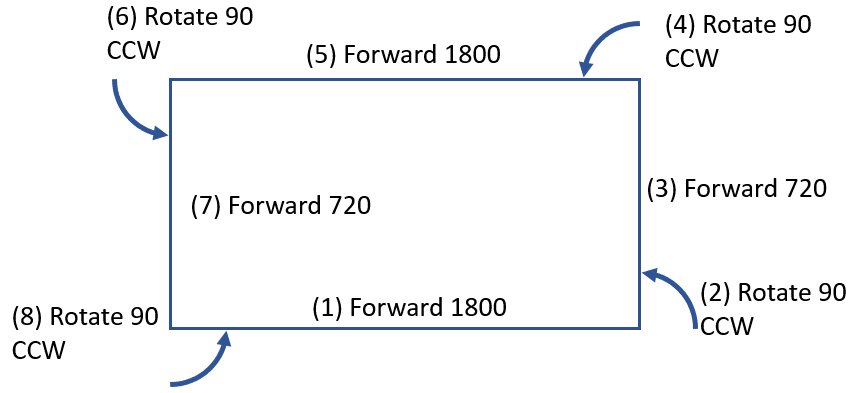
• rotate 90 degrees counterclockwisein order. Figure 4 shows the order of motion. Ideally, the robot should be where it started after all the movements completed. That is, the co- ordinates should read (0,0) for x and y.

* 1. b)  Implement state machine in RobotC and test on the Metric Distance Utility challenge under Utility Tables of Robot Virtual Worlds (RVW).
  2. c)  Discuss your observations.
     1. Does the robot arrive in the starting position and orientation? If not, explain why.
     2. Does the robot location tracking information (i.e., x,y, and heading) agree with the final robot location? If not, explain why.

*Figure 3: Consecutive pure rotation* 𝛥𝜃 *and pre translation* 𝑙*.*

4





*Figure 4: Consecutive pure rotation* 𝛥𝜃 *and pre translation* 𝑙*.*