Three Wheel Localization of a Holonomic Robot

Havish Netla

November 2019

1 Definitions

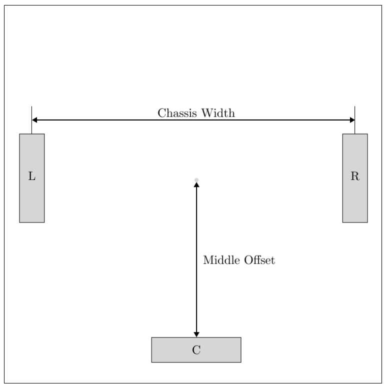
1. Holonomic Robot - A robot that can move in any direction
2. Odometry - Use of data to estimate change in position over time
3. Localization - A tool used to implement odometry

2 Introduction

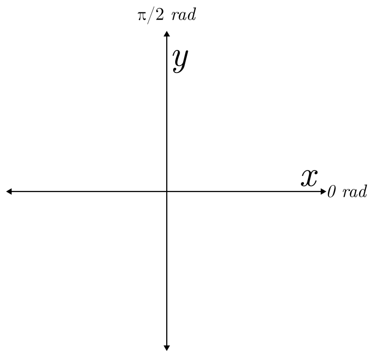
Three Wheel Localization is a style of odometry that utilizes three free spinning omni wheels to track a holonomic robot position relative to a starting point. This is very important in many fields of robotics in order to be able to avoid obstacles, follow path s, and have more data to build more resilient programs. In the scope of FIRST Tech Challenge, odometry is usefull for automation of routine tasks during the tele-operated period, smooth path planning and following during autonomous.

3 Setup

The setup that is optimized for accuracy and ease of use is one with three free spinning omni wheels, two on either side of the robot directly in the middle, and one in the back facing sideways also directly in the middle.



The coordinate system that will be used is a regular cartesian coordinate plane with $\frac{\pi}{2}$ facing forward.



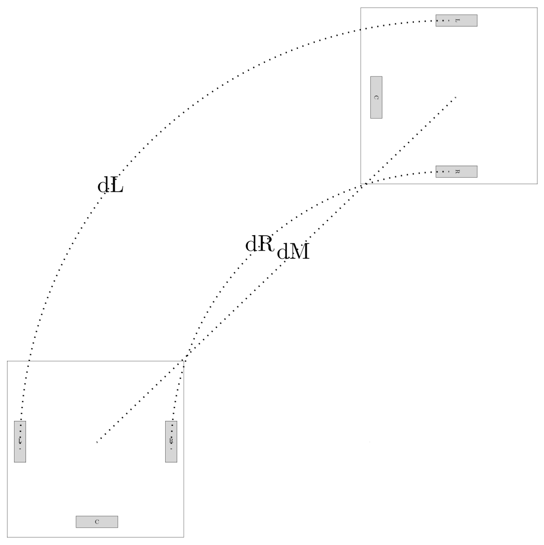
4 Derivation

The follwing derivation only applies to each individual update. Each update is very small and hence the robot will have traveled a very small arc.

From here on out dL will refer to the change in the left wheel, dR will refer to the change in the right wheel, and dM will refer to the change in the center wheel

4.1 Distance Traveled

Averaging dL and dR will produce dM.



Therefore we can use this formula to find dM

$$dM = \frac{dL + dR}{2}$$

4.2 Change in Angle

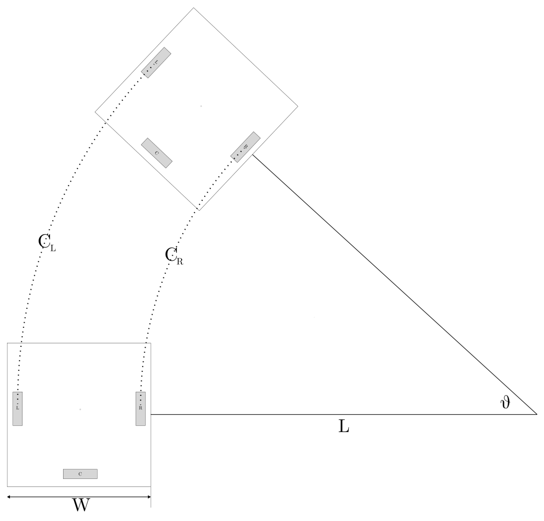


Figure 1: Robots turn angle

To find θ we start with knowing that a length of a circle segment is

$$C = \theta * R$$

In figure 1 there are two arcs, C_L and C_R. Therfore with the circle segment fomula we can derive

$$C_R = \theta L$$

$$C_L = \theta(L + W)$$

Next we can simplify C_L to

$$C_L = \theta L + \theta W$$

Then to

$$\theta L = C_L - \theta W$$

As shown before $\theta L = C_R$ so we can further simplify to

$$C_R = C_L - \theta W$$

$$\theta = \frac{C_R - C_L}{W}$$

4.3 Change in Position

Finding Δx and Δy requires incorparting everything derived so far, θ and dS

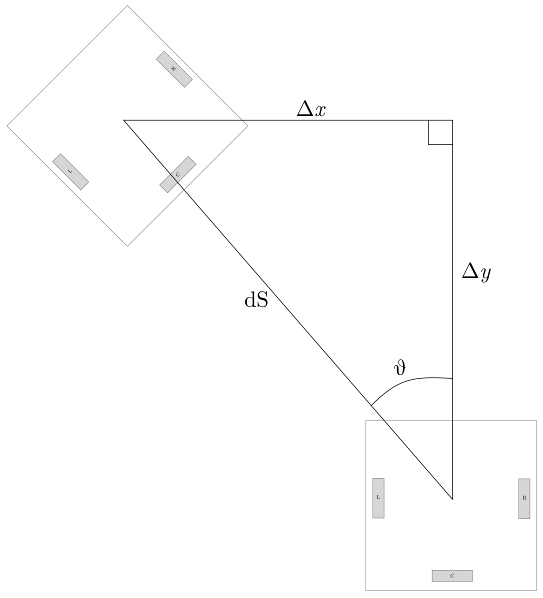


Figure 2: Robots change in position relative to dS and θ

To find Δx we can apply

$$\Delta x = dS * cos(\theta)$$

$$\Delta y = dS * sin(\theta)$$

These two equations fail to compensate for a holonomic robot because since the robot can strafe, Δx and Δy can change even while the left and right wheels are not moving. In order to compensate for this the middle wheel comes into play. We can draw a similar triangle to Figure 2 and derive that

$$dM * sin(\theta)$$

$$dM * cos(\theta)$$

need to be added to the Δx and Δy expressions. The final expressions will be

$$\Delta x = dS * cos(\theta) + dM * sin(\theta)$$

$$\Delta y = dS * sin(\theta) + dM * cos(\theta)$$

4.4 Compensation for Middle Wheel Offset

One issue with the odometry stup that we are using is that if we turn we would expect a positive change in the left wheel value, and a negative change in the right wheel value, and no change in the center wheel value, but because of our setup the center wheel will spin while we are spinning in place. To compensate for this we need to find the offset of the middle wheel from the center point of the robot. It is imperative that this distance is exact because the less percise it is the less accurate the odometry will become throughout the duration of use.

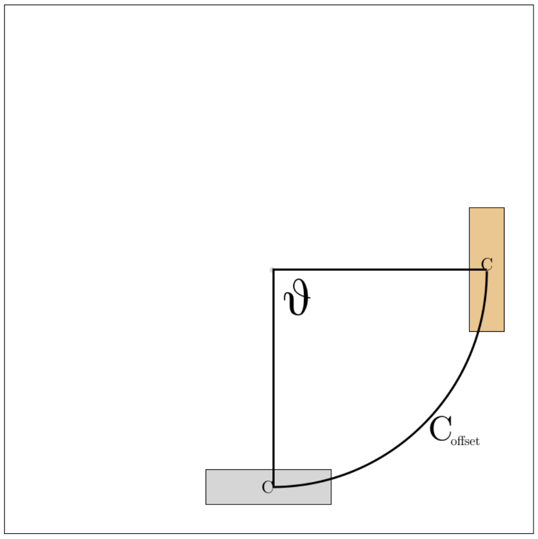


Figure 3: Orange represents the new location of the center wheel after a 90° turn

To compensate we need to find the length of C_{offset}. Utilizing the arc length formula we can find that

$$C_{offset} = \theta * offsetDist$$

5 Implementation

Implementation will be shown in Java

```
    public Pose2d update() {
            List<Double> wheelPositions =
                drive.getTrackingWheelPositions();
            if (!lastWheelPositions.isEmpty()) {
              double c = 11 * 2 * Math.PI;
              double dL = wheelPositions.get(0) -
                  lastWheelPositions.get(0);
              double dR = wheelPositions.get(1) -
                  lastWheelPositions.get(1);

              dTheta = ((dR - dL) / chassisWidth);

              double dM = wheelPositions.get(2) -
                  lastWheelPositions.get(2) - (c / (2 * Math.PI) *
                  dTheta);

              telemetry.addData("dl", dL);
              telemetry.addData("dR", dR);
              telemetry.addData("dM", dM);

              double dS = (dR + dL) / 2.0;

              telemetry.addData("dS", dS);

              double avgTheta = theta + dTheta / 2.0;

              double dY = dS * Math.sin(avgTheta) - dM *
                  Math.cos(avgTheta);
              double dX = dS * Math.cos(avgTheta) + dM *
                  Math.sin(avgTheta);
              telemetry.addData("dX", dX);
              telemetry.addData("dY", dY);

              // Update current robot position.
              x += dX;
              y += dY;
              theta += dTheta;

              poseEstimate = new Pose2d(x, y, theta);
              lastWheelPositions = wheelPositions;
```

```
        }
        lastWheelPositions = wheelPositions;

        return poseEstimate;
}
```