Demonstration of Robot Kinematics

Harrison Ursitti, harrison.ursitti@unh.edu, CS 733

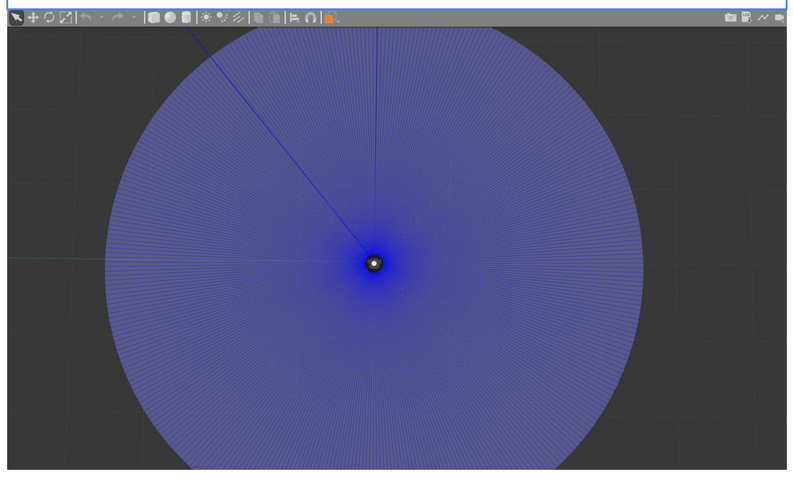
Technologies

- TurtleBot3:
  - A differential drive robot, with two powered wheels on one axle.
  - Controlled through two variables, v and ω
  - v controls linear velocity of the robot
  - ω controls angular velocity of the robot
- ROS:
  - A common protocol for communication to robot control systems
  - Publisher and subscriber model, where processes publish and subscribe to activity tied to topic names

Purpose

- Demonstration of TurtleBot3 control
  - Publish Twist messages to give the robot movement commands
  - Subscribe to Odometry messages to determine robot position
- Control Translations
  - Forward For Distance
    - Driving forward to specified distance
  - Arc Drive
    - Rotate to angle while keeping turn radius and speed
  - Velocities From Wheel Rotation
    - Determine v and ω from wheel φ□ and φ_r

TurtleBot3 in Gazebo



Python rospy Example

```
def drive_straight_odometry(self, speed, distance):
    """Drives the Turtlebot forward for the given distance at the given
    speed. The finish condition relies on odometry information."""
    self.get_logger().info("Driving straight for {0} m at {1} m/s using odometry".format(distance, speed))

    twist_msg = Twist()

    twist_msg.linear.x = speed

    initPos = self.odom.pose.pose.position

    euc = self.efromq(self.odom.pose.pose.orientation.x, self.odom.pose.pose.orientation.y, self.odom.pose.pose.orientation.z, self.odom.pose.pose.orientation.w)

    nextPos = Point()

    nextPos.x = (math.cos(math.fabs(euc[2])) * distance) + math.fabs(initPos.x)
    ySin = math.sin(math.fabs(euc[2]))
    ySin = ySin if euc[2] > 0 else -ySin
    nextPos.y = (ySin * distance) + math.fabs(initPos.y)

    #print(math.cos(math.fabs(euc[2])))
    #print(math.sin(math.fabs(euc[2])))

    print(initPos)
    print(nextPos)

    def finish_condition():

        print(self.odom.pose.pose.position)
        print(math.fabs(self.odom.pose.pose.position.x - nextPos.x))
        print(math.fabs(self.odom.pose.pose.position.y - nextPos.y))

        at_goal = nextPos.x - self.odom.pose.pose.position.x < LINEAR_THRESHOLD if self.sign(nextPos.x - initPos.x) > 0 else self.odom.pose.pose.position.x - nextF

        at_goal = at_goal and (nextPos.y - self.odom.pose.pose.position.y < LINEAR_THRESHOLD if self.sign(nextPos.y - initPos.y) > 0 else self.odom.pose.pose.posit
```

Solution

- Forward For Distance
  - Simple velocities: v = some constant, ω = 0
  - End Condition: $x_1 > x_0$ + distance
- Velocity From Wheel Position
  - v□ = φ□ * wheel_radius * 2 * π
  - $v_r = \varphi_r$ * wheel_radius * 2 * π
  - v = v□ + v_r, ω ~ v_r - v□
  - End Condition: None
- Arc Drive
  - Δx = r * α
  - v = v
  - ω = v * α / Δx
  - End Condition: α□ ≈ α

Outcome

- TurtleBot3 Simulation
  - Instead of using the physical model, a simulated TurtleBot3 was controlled using the same systems as the physical model would
  - Gazebo, a robot model software, hosted a TurtleBot3 model with running control software
- Demonstrated movement path
  - Created path that demonstrates straight forward, drive arc, and rotation movement