Roboterkinematik

Für diese Aufgaben wird ein in Python geschriebener Roboter-Simulator eingesetzt, der im Rahmen der Vorlesung eingeführt wird.

Aufgabe 1

Realisieren Sie die folgenden beiden kinematischen Grundfertigkeiten des Roboters:

- curveDrive(v, r, $\Delta\theta$)
- straightDrive(v, l)

curveDrive bewegt den Roboter solange auf einem Kreisbogen mit Radius r und Geschwindigkeit v, bis sich seine Orientierung um $\Delta\theta$ geändert hat. straightDrive bewegt den Roboter eine gerade Strecke der Länge l mit der Geschwindigkeit v. Definieren Sie eine Folge von Geschwindigkeitsbefehlen, die die gewünschte Bewegung umsetzt. Beachten Sie dabei, dass Geschwindigkeit und Rotationsgeschwindigkeit beim Roboter begrenzt sind.

Gehen Sie zunächst davon aus, dass der Roboter die gewünschten Geschwindigkeiten korrekt umsetzt (motionNoise auf 0 setzen) und realisieren Sie Kreis- und Rechteckfahrten und einen Fahrspurwechsel.

Gehen Sie nun davon aus, dass die gewünschten Geschwindigkeiten nicht korrekt umgesetzt werden (motionNoise werden auf die voreingestellten Werte gesetzt). Was beobachten Sie?

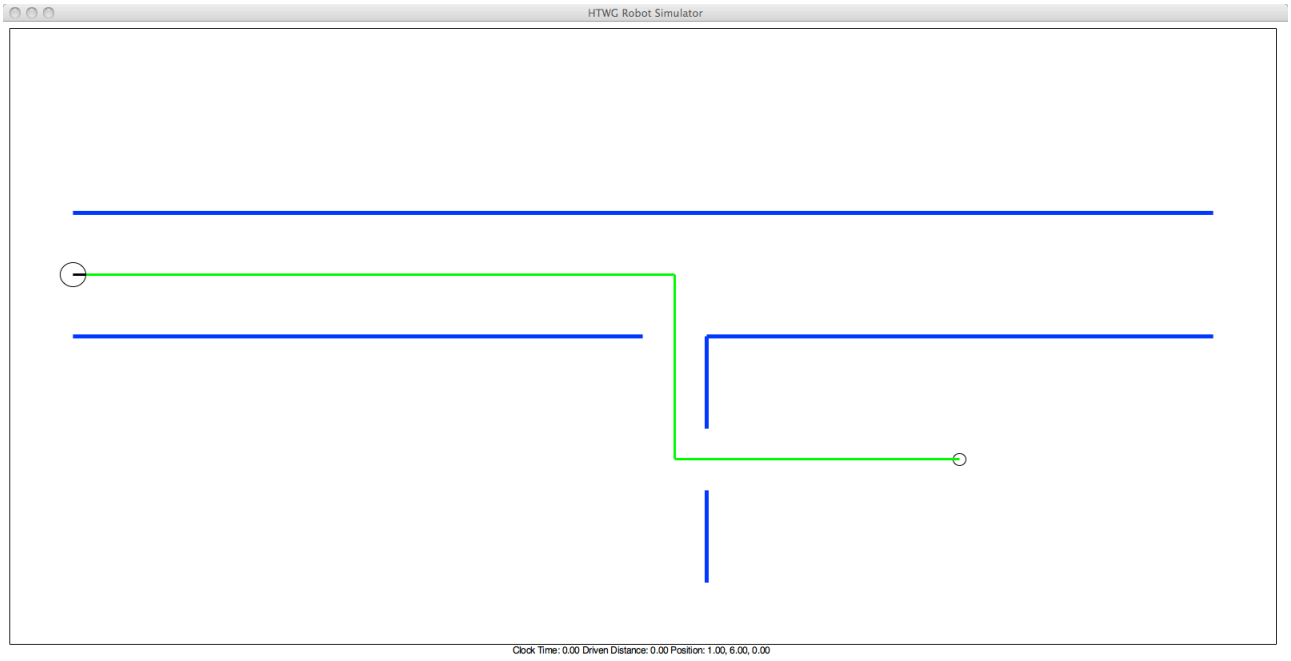
Aufgabe 2 Lokalisierungsbasierte Steuerung

Für diese Aufgabe soll die tatsächliche Position des Roboters mit World.getTrueRobotPose() abgefragt werden. Koordinaten beziehen sich immer auf das globale KS.

a) Realisieren Sie einen Linienverfolger followLine(p1, p2), der eine Strecke von Punkt p1 nach Punkt p2 möglichst genau verfolgt. Realisieren Sie zunächst einen P-Regler, der den Abstand zur Linie auf 0 regelt. Was beobachten Sie? Verbessern Sie Ihre Regelung, indem Sie einen PD-Regler einsetzen.

b) Realisieren Sie eine Steuerung gotoGlobal(v, p, tol), die den Roboter auf den Punkt p mit der Geschwindigkeit v zusteuert, wobei der Punkt p lediglich mit einer gewissen Toleranz tol errreicht werden muss.

c) Realisieren Sie einen Linenverfolger followPolyline(v, poly), der einen Polygonzug poly mit einer Liste von Koordinaten mit der Geschwindigkeit v abfährt. Die Geschwindigkeit v soll möglichst konstant gehalten. Sobald der Roboter einen Eckpunkt mit einer gewissen Toleranz erreicht hat, fährt er bereits auf den nächsten Eckpunkt zu. Testen Sie Ihren Linienverfolger in typischen Szenarien.



Aufgabe 3 Odometriebasierte Steuerung

Für diese Aufgabe wird die über Odometrie geschätze Position des Roboters verwendet. Verwenden Sie dazu die Methode OdometryPoseEstimator.getPose().

a) Realisieren Sie eine Steuerung gotoLocal(v, p, tol), die den Roboter auf den lokalen Punkt p (Punkt im Roboter-KS) mit der Geschwindigkeit v zusteuert. Dabei muss der Punkt p lediglich mit einer gewissen Toleranz tol errreicht werden.

b) Realisieren Sie eine Wandverfolgung. Positionieren Sie den Roboter so, dass er entweder eine linke oder eine rechte Wand „sieht“.
Benutzen Sie die Funktion SensorUtilities.extractLinesFromSensorData, die aus Sensordaten eine Liste von Liniensegmente extrahiert.

c) Freiwillig: die Wandverfolgung aus b) kann benutzt werden, um aus eienm Labyrinth herausfinden („Rechte-Hand-Regel“ oder „Linke-Hand-Regel“).