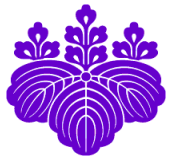


筑波大学 知能ロボット研究室
University of Tsukuba, Intelligent Robot Laboratory

指定した位置へ移動・直線追従

Contents

1 指定した位置への移動

次に出来そうな事

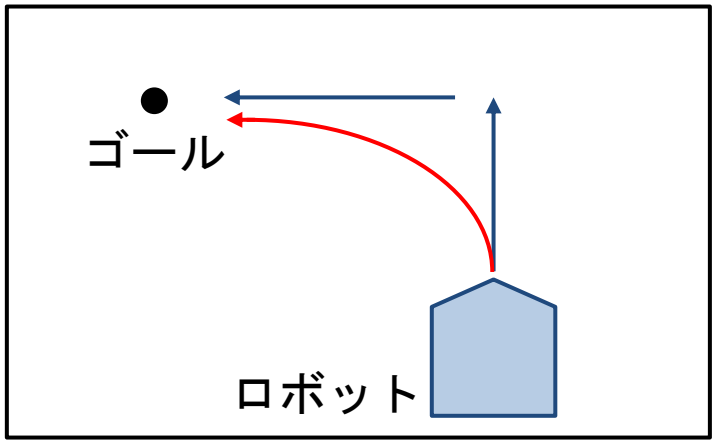
- 今までで出来るようになった事
  - 速度と角速度の指令方法
  - 現在の自分の位置(自己位置)　etc.

→指定した位置に移動することなら出来そう

指定した位置への移動方法

- 移動の仕方の一例（⟶）

→ゴールの方を向きつつも移動したい…(⟶)

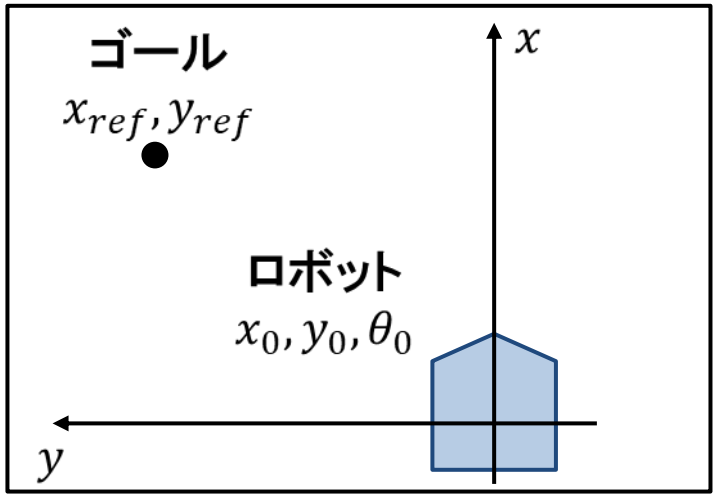


- どうすれば実現できそう？
  - 速度：ゴールとロボットの位置が縮まったら小さくする
  - 角速度：ゴールの方を向いたら小さくする

P制御

P制御(比例制御)

目標値と現在の値との差に比例した制御



速度と角速度の計算例

$$v = k_v \cdot \sqrt{(x_{ref} - x_0)^2 + (y_{ref} - y_0)^2}$$

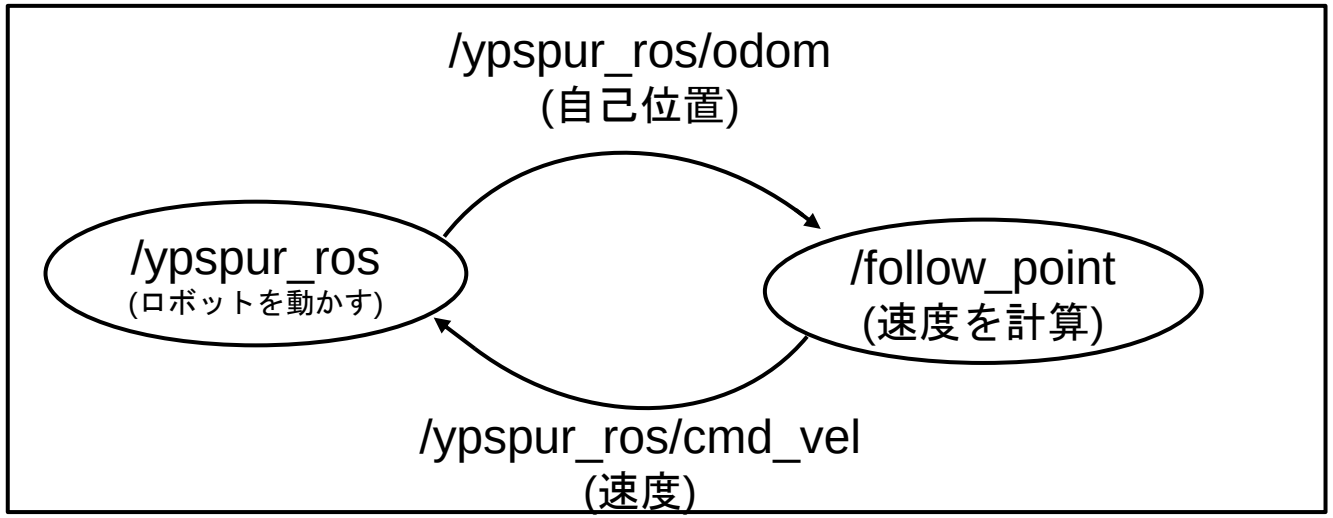
$$\omega = k_\omega \cdot \left(\tan^{-1} \left(\frac{y_{ref} - y_0}{x_{ref} - x_0} \right) - \theta \right)$$

※k_v, k_ωの値は調整が必要

6 プログラム上での記述

- 先程の式をプログラムにすると…
  (follow_point.cpp内86, 90行目参照)

- 以下のような関係性であることで実際にゴールへ向かって動いてくれる



実行方法

実行方法はいつもと同じように行う

1. ターミナル① `$ roscore`
2. ターミナル②

```
$ rosrun ypspur_ros ypspur_ros_param_file:=/home/<user>/researches/programs/platform/yp-robot-params/robot-params/<ロボットの種類>.param
```

3. ターミナル③

```
$ rosrun yamasemi2023_follow follow_point
```

↑配布したプログラムだと(2, 2)の位置に向かってくれる

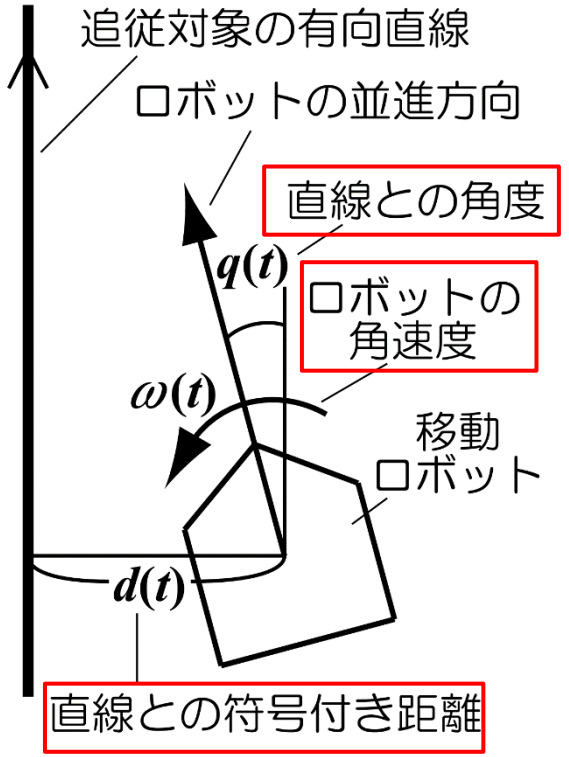
2 直線追従

直線追従の考え方

- 指定した直線上を走らせるには以下の3つを考慮したい

  ・ロボットと直線との距離を0に近づけたい
  ・直線の方向にロボットが向いてほしい
  ・最終的には角速度を0にしたい

→これらを考慮するような制御をすれば直線追従できそう

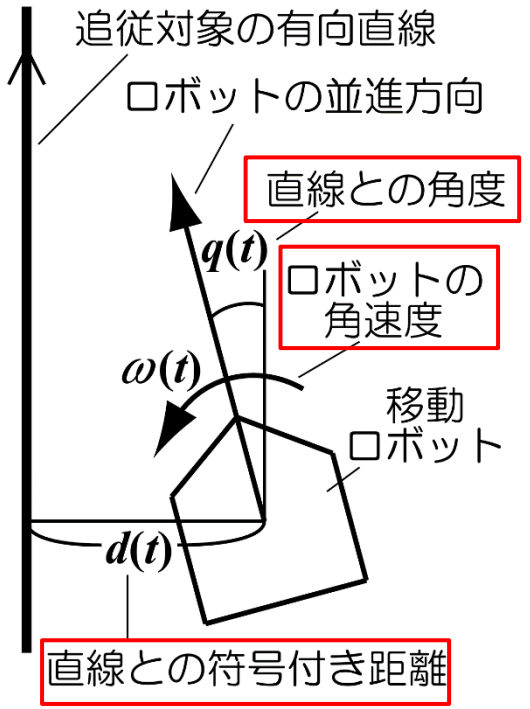


制御のための計算

- 止まる必要がないので速度は自由

- 角速度の計算にさっきの3つの要素を加える

$$\omega(t+\Delta t) = \omega(t) + \Delta t\left(-k_1 d(t) - k_2 q(t) - k_3 \omega(t)\right)$$



※正確には更に考慮する事項はある。
　以下を参考にしてみるとよい。
https://at-wat.github.io/ROS-quick-start-up/files/Vehicle_and_Motion_Control.pdf
https://www.roboken.iit.tsukuba.ac.jp/platform/wiki/_media/ypspur-0.13.2.pdf

11 演習課題

- 課題1

  指定した座標(x, y)に到達するプログラムを作成し，(2, 2)へと移動させせよう

  ただし，$k_v = 0.1, k_\omega = 1.6$とする

  ※指定した座標に到達したかを判定する部分に注意

- 課題2

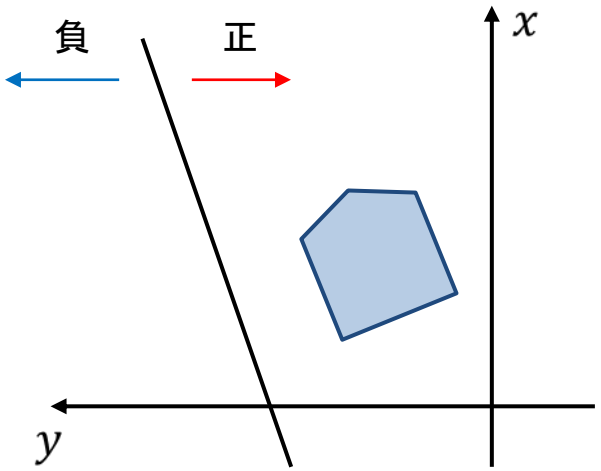
  直線追従を行うプログラムを作成し，y=x+1の直線上を走行させせよう

  ただし，$k_1 = 8, k_2 = 3, k_2 = 2$とする

11 オプション課題

- 符号付き距離は直線から見てx軸側にある点との距離が正となる
  →直線の傾きの絶対値が$\frac{\pi}{2}$より大きい時には計算に注意が必要

- 上記のような直線にも追従できるプログラムを作成せよ



P制御で考えられる問題とPID制御

- 使う環境が異なると目標とする値に届かないことがある
  (「オフセット」と呼ばれる一定の差が残ってしまうような状態)
  →一定時間差が残っているのを把握するため積分の項を入れる(PI制御)

- 急激に出力に変化が起こった時にそれを抑えたい場面が存在
  →前の時刻と比べて急激に出力が上がったことを知るため微分の項を入れる(PD制御)

- これらを両方考慮したものがPID制御

参考

- ゴールの指定に使用したメッセージ(PoseStamped)

http://docs.ros.org/en/noetic/api/geometry_msgs/html/msg/PoseStamped.html

- PID制御を噛み砕いて説明したページ

https://www.rkcinst.co.jp/technical_commentary/13090/

- 直線追従について

https://at-wat.github.io/ROS-quick-start-up/files/Vehicle_and_Motion_Control.pdf

https://www.roboken.iit.tsukuba.ac.jp/platform/wiki/_media/ypspur-0.13.2.pdf