担当教員　小澤正宜先生

提出日　令和 6 年 6 月 10 日

ロボット工学実験

ROS と車両型ロボットの制御について

実施日　令和 6 年 5 月 18 日

学籍番号　120158

氏　　名　藤田崇太

共同実験者　古川

1. 目的

機械工学科 4 年生で学習した車輪型ロボットに任意の移動をさせるための理論を実機体に適用することを通じ，ロボットの動作を理論的に決定することを学ぶ．また，近年のロボット制作で利用されている ROS の意義と概要を学ぶ．

2. 理論

車両型ロボット turtlebot3 waffle-pi を ROS を用いて任意の位置に移動させる

1. ROS 及び ROS2 の意義 （AI により要約）

ROS（Robot Operating System）はロボットのソフトウェア開発を支援するツール群であり，ソースコードの再利用を容易にすることが目的だ．以前，ロボット研究者や開発者は自分で全てのソースコードを書かなければならなかったが，ROS の登場によって，ロボットの機能ごとにモジュール化されたソースコードを作成し，それを再利用できるようになった．これにより，開発効率が大幅に向上した．

初期の ROS は，1 台のロボットが安定した通信環境で動作することを前提に設計されていた．しかし，ROS の利便性が理解されるにつれて，その適用範囲は広がり，複数のロボット間での通信やパケットロスが発生する環境，計算リソースの少ないハードウェアにも対応するようになった．

ROS2 では，DDS（Data Distribution Service）が導入され，ユーザーが通信の優先順位を設定できるようになった．これにより，リアルタイム性が確保されるようになった．リアルタイム性とは，処理が決められた時間内に完了することが保証されることだ．これにより，産業用ロボットや商用アプリケーションなど，リアルタイム性が求められるシステムにも対応できるようになった．

ROS は主に Linux 上で開発が進められてきたが，ROS2 の利用範囲が拡大するにつれて，Windows もサポートされるようになった．ただし，2021 年時点では Linux の利用例が大半を占めており，Windows の利用はまだ限定的だ．しかし，今後は ROS2 の進化により，Windows での利用も増えると期待されている．

2. ROS の通信の特徴

ROS の通信プロトコルには Publish/Subscribe 型通信(Pub-Sub 通信と略される)が 使用されている．データ(Topic)を提供するノード(Publisher:出版者)は，提供したいデータをネットワークに参加しているほかのノードが自由に利用できるよう，提 供先を指定せずに発信する．データを利用したいノード(Subscriber:購読者)は，自 身が必要なデータを，発信元を指定せずに受け取る．通信先を指定せずにデータの やり取りを可能にしたことで，容易にロボットの構成を変更することが可能である．ROS の通信の簡略図を Fig. 1 に示す．

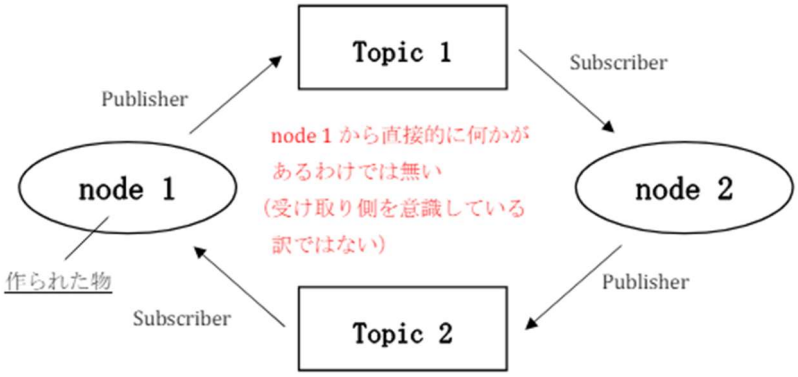

Fig 1 ROS 通信の概要図

3. 座標系

移動系ロボットを考える場合，絶対座標系と相対座標系が存在する．絶対座標系は地球の緯度経度のように原点や軸が定められているものである．ロボットが移動する環境は絶対座標系で与えられることが大多数である． 相対座標系とは使用者が任意の点を基準点として定め，そこからの距離や方向を示したものである．それぞれの座標系を Fig. 2 に示す.

(例)マニピュレータの手先の位置を表すとき，環境に原点を置く絶対座標系であれば台座の移動分と手先を動かした分の移動を考慮するが，台座に原点を置く相対座標系であれば手先を動かした分の移動のみの考慮となり，マニピュレータ自体の移動分は相対座標系には影響しない.

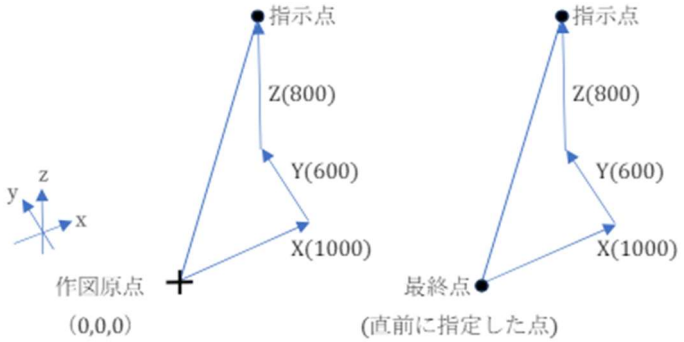

Fig 2 絶対座標系(左)と相対座標系(右)

4. **車輪型ロボットの旋回**（参考文献 1 ）

左右駆動輪の中心(車軸の中心)P の位置の速度 v，角速度 ω を用いてロボットの動きを考える．物体が半径 R の円周上を速度 v で移動する場合，v＝Rω が成り立つ．この式を用いて，左右の駆動輪の速度 vL，vR と中心の速度 v の関係を記述すると以 下のようになる．

Fig. 3 に回転中心から車軸の中心までの距離，左右車輪までの距離の関係を示す． 車軸の中心までの距離は R である．その場合，左車輪までの距離は R-W/2，右車輪 までの距離は R+W/2 になる．よって，各地点の速度と角速度の関係は，下記のように表すことができる．

$$v_L = \left(R - \frac{W}{2}\right)\omega \qquad \cdots(\mathrm{I})$$

$$v_R = \left(R + \frac{W}{2}\right)\omega \qquad \cdots(\mathrm{II})$$

$$v = R\omega$$

これらを v，ω でまとめると

$$v = \frac{v_R + v_L}{2} \qquad \cdots(\mathrm{III})$$

$$\omega = \frac{v_R - v_L}{W}$$

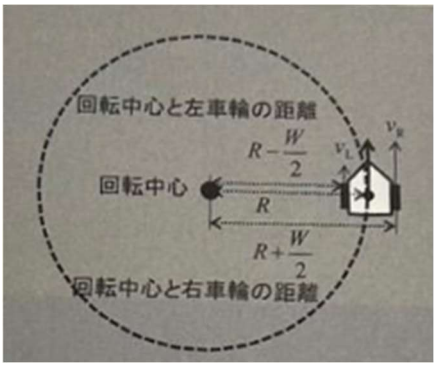

Fig 3 車輪型ロボットの旋回

3. 実験機材

本実験では以下の実験機材を使用する．

- turtlebot3 waffle-pi
- PC
- ubuntu 16．04
- ROS Kinetic

4. 実験手順及び結果

1 ROS の持つ Publish/Subscribe 型通信プロトコルの利用

1. ノードと topic の理解

以下の手順を実行することで，ノードの増減と topic の関係を理解する．

1. 端末(Terminal)を起動し(端末 1 とする)，「roscore」を実行する．
2. 別の端末(Terminal)を起動し(端末 2 とする)，「rostopic␣list」を実行する．
3. さらに別の端末を起動し(端末 3 とする)
「roslaunch␣turtlebot3_teleop␣turtlebot3_teleop_key．launch」を実行する
4. 端末 2 で再度「rostopic␣list」を実行する
5. 端末 3 で teleop_keyboard を終了する(Ctrl+c)
6. 端末 2 で再度「rostopic␣list」を実行する

結果：Fig．4 に示すような画面が出力された．Fig．3 の左の状態から 3 の手順を踏むことで Fig．4 の右のように ”/cmd_vel” が増え，その後終了すると Fig． 3 の左の状態に戻った．この変化から，ノードが増え topic が作られると Subscriber が topic を読み表示することが分かった．このとき，Subscriber は Publisher を区別して読んでいる訳では無い事を理解した．

Fig 4 手順 3 の実行前の結果(右)　手順 3~6 の結果(左)

2. ノードと topic の送受信関係の確認

接続状態を可視化する GUI ツールを使用して，ノードと topic がどのような接続 関係にあるかを理解する．

1. 端末(Terminal)を起動し(端末 1 とする)，「roscore」を実行する
2. 別の端末(Terminal)を起動し(端末 2 とする)，「rqt_graph」を実行する
3. さらに別の端末を起動し(端末 3 とする)，
「roslaunch␣turtlebot3_teleop␣turtlebot3_teleop_key.launch」を実行する
4. rqt_graph の表示を更新する
5. rqt_graph のプルダウンやチェックボックスを操作し，表示内容の変化を確認する

結果：Fig. 5，Fig．6 のような画面が出力された．Fig. 5 ではノードのみが表示されているが Fig．6 になると topic の内容も表示されるようになった.ノードのグラフが上の各種項目の選択によって表示される情報量が増減することが分かった.

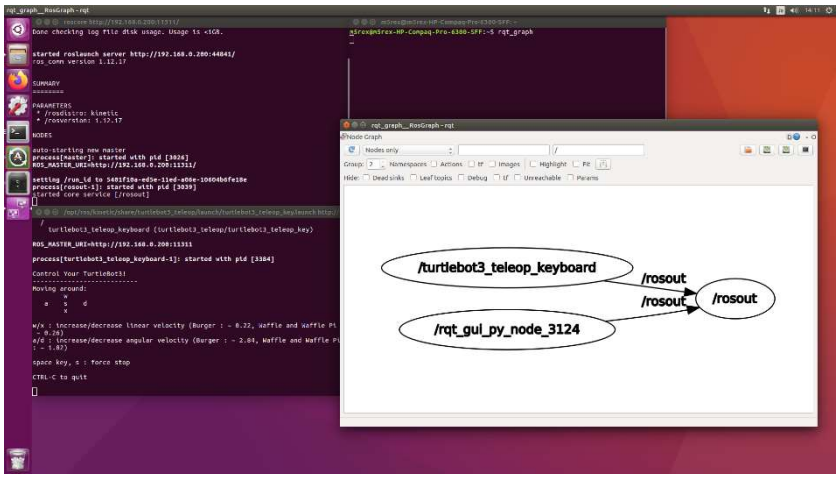

Fig 5 ノードのみの表示

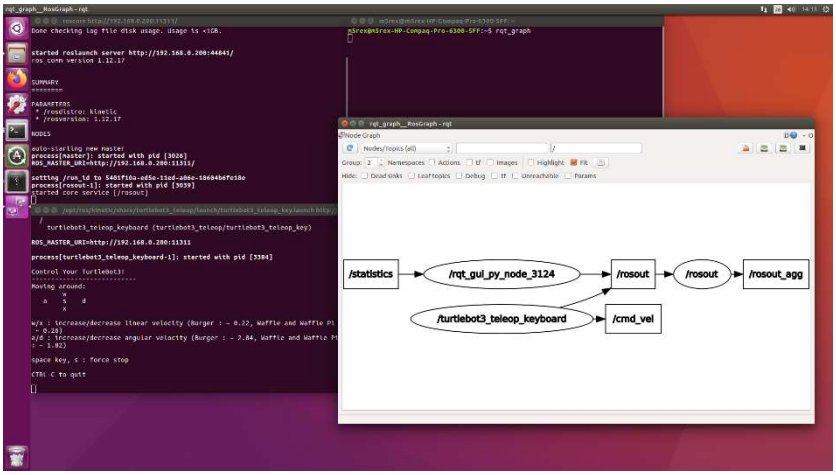

Fig 6 ノードとすべての topic の表示

3. Publish/Subscribe 型通信の理解

次の手順を実行することで，Publish/Subscribe 型通信の特性を理解する． 今回の座標系と回転の正負を Fig. 7 に示す.

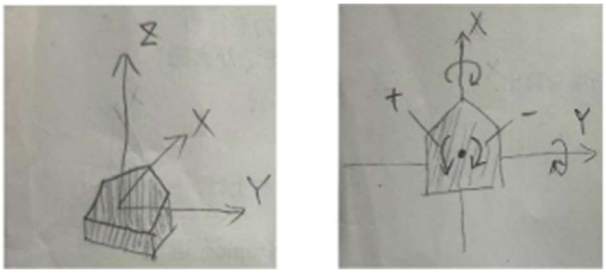

Fig 7 座標系

A) Publisher 1 に対して Subscriber 1 の場合

1. 端末(Terminal)を起動し(端末 1 とする),「roscore」を実行する
2. 別の端末(Terminal)を起動し(端末 2 とする)
「roslaunch␣turtlebot3_teleop␣turtlebot3_teleop_key.launch」を実行する
3. さらに別の端末を起動し(端末 3 とする)
「rostopic␣echo␣/cmd_vel」を実行する
4. 端末 2 を有効にし，Moving around:以下で示されたキーを入力する
5. 端末 3 の表示を確認する

結果：端末 2 で、キーボードによって移動の入力がされたとき、その都度新しく topic が作られ、それを端末 3 が Subscriber として読んでいる様子を確認できた.

B) Publisher 1 に対して Subscriber 多の場合

1. A.4.の続きから実施する
2. さらに別の端末を起動し(端末 4 とする)「rostopic␣echo␣/cmd_vel」を実行する
3. 端末 2 を有効にし，Moving around:以下で示されたキーを入力する
4. 端末 3 および端末 4 の表示を確認する

結果：端末 3，4 は同じタイミングで topic が表示された. これにより Publisher は特定の Subscriber におくっているのではないことが分かった. また Subscriber 同士もお互いのことを区別して認識しているのではないことも分かった.

C) Publisher 多に対して Subscriber 1 の場合

A.4.の続きから実施する

1. さらに別の端末を起動し(端末 4 とする)「rostopic␣pub␣-r␣10␣/cmd_vel␣geometry_msgs/Twist␣--␣'[1.0, ␣0.0, ␣0.0]''[0.0, ␣0.0, ␣0.0]'」を実行する(※/Twist␣の後ろはハイフン 2 つ)
2. 端末 3 の表示を確認する
3. 端末 4 で Ctrl+c を押し、rostopic␣pub を停止する

結果：linear[0.0, 0.0, 0.0]、angular[0.0, 0.0, 0.0]と linear[1.0, 0.0, 0.0]、angular[0.0, 0.0, 0.0]を繰り返す表示をした．なお，この実験中はキーボードによる入力を行っていない．このような動作をしたのは，端末 2，4 の Publisher の値を一つの Subscriber が読んでいる状況だったため読み取る値が不安定になったのではないかと考えられる．これにより Subscriber は Publisher をそれぞれ区別して認識していないことが分かった．

2　ロボットの移動制御

1.　計算を行わない場合の車輪ロボットの移動

コントローラからの操作により，waffle-pi が机の上に引かれた線に沿うよう移動を制御せよ．

1. waffle-pi、コントローラの電源を入れる
2. 十字キーの上下(移動速度)、左右(旋回速度)、右上のボタン(停止)を組み合わせ、狙った線に沿うようロボットを移動させる．

結果:移動速度がキーの押す回数(長押しの場合も含む)によって加速するため，速いスピードでライン上を進むのが難しかった．スピードが速吸ギルと旋回の際に必要なタイヤの摩擦力の大きさも考慮する必要があると思われる．しかし，低スピードであれば簡単にライン上を走らせることができた．

2　計算を行う場合の車輪ロボットの移動

事前に曲率に合わせた速度と角速度を計算してから線に沿うよう waffle-pi の移動を制御せよ．

1. waffle-pi の電源を投入する
2. 操作用 PC で端末を立ち上げ(端末 1 とする)、roscore を実行する
3. 別の端末を立ち上げ(端末 2 とする)、SSH(Secure Shell)接続で waffle-pi に接続する
   アカウント，パスワードは機体に記載されている「ssh {アカウント名}@{機体の IP アドレス}」を実行すること
   (例：ssh pi@192.168.0.100)
4. waffle-pi に接続した端末 2 で以下を実行する
   roslaunch␣turtlebot3_bringup␣turtlebot3_robot.launch
5. さらに別の端末を起動し(端末 3 とする)、遠隔操作ノードを実行する
   roslaunch␣turtlebot3_teleop␣turtlebot3_teleop_key.launch

6. 任意の曲率に合わせた速度と角速度を計算する
   速度は 2 つ以上選定して実施すること
7. 端末 2 を有効にし、4.で計算した速度、角速度になるよう入力を行う車輪を浮かして入力し，任意の速度になったら接地させる方法が実施しやすい

計算過程と結果

2 種のライン上を進む周速度は最低速度である v=0.01m/s で計算を行った．使用した式は「2．理論，4．車輪型ロボットの旋回の式」を使用した．これにより R=0.2m のとき ω=0.05rad/s ，R=0.4m のとき ω=0.025rad/s ，の計算結果が得られた．しかし，入力できる角速度が 0.1rad/s からであったため R=0.2 のとき(v ， ω)=(0.02 ， 0.1)，R=0.4 のとき(v ， ω)=(0.04 ， 0.1) で実験を行った．実験の様子は実験動画 1 に示す.

実験動画 1 に示すようにこれらの値の組み合わせで 2 種のライン上それぞれに沿って走行することができた. 本実験ではこれ以上の速度で走行させなかったため，ライン上を走行するのが最も早い速度は分からない.

3 授業内課題

1 ロボットが動作している状態の操作用 PC で新しい端末を立ち上げ,「rostopic list」を実行せよ
2 表示された topic の中から，waffle-pi の移動に関係ありそうな topic を選び,「rostopic echo ~~」で値を確認せよ
3 Waffle-pi を動かし，パラメータの変化を観察せよ
4 3 の値の変化から，ロボットに搭載されているセンサで移動量を算出する方法を検討せよ

結果： 2 では”/odom”を選択する．“odom”は“odometry”の略であり，この項目で実験動画 1 後半に示すようなパラメータを確認できる．様々なパラメータがあり，いくつかは実験により確認できた.

position : ロボットの絶対座標に対してロボットが向いている方向を示す
liner : ロボットのタイヤの並進速度を示す
angular : ロボットのタイヤの回転速度を示す
orientation : Z 軸の値が激しく変動したためロボットを原点とした加速度を GYRO センサで読み取った値を示していると考えられる.

4 Additional Work

(1) 机座標系で任意の点を 3 点選択せよ

(2) 決定した 3 点間をロボットに移動させて移動に関する情報を取得せよ

(3) ロボットの並進量および旋回量の算出，表示と 2.の計測を比較せよ

→時間が足らず全て行えなかった.

参考文献

1. 日本機械学会、”Robotics”、丸善出版株式会社、(2011)、P32～34　2024 年 6 月 9 日(日)閲覧
2. 1 週目 配付資料「ロボットの移動と姿勢・ROS の基礎」 2024 年 6 月 9 日(日)閲覧
3. M5R_No.33_森山拓海_ロボット実験 1 回目(レポート)　　2024 年 6 月 9 日(日)閲覧

[AI による要約]

2.理論の ROS 及び ROS2 の意義を AI による要約を行った.