

移動ロボット走行制御コマンド系および走行制御系

"Spur" の考え方と "YP-Spur" の実装 rev. 4 [2010.08.19]

筑波大学 知能ロボット研究室 Robot Platform Project(RPP) 渡辺敦志 (WATANABE Atsushi)

1. はじめに

現在, センサ情報を処理して自己の動作を決定・行動する, 自律移動ロボットの研究・開発が盛んに行われている. この自律移動ロボットのシステムは, Fig. 1 に示すようなブロック構造をもつ.

- センサ情報処理ブロックでは, 目的に従って, 自己位置推定値や, センサ情報, 事前知識から行動を決定し, 動作指示を移動体制御ブロックに渡す.
- 移動体制御ブロックは, 与えられた動作指示に移動体が従うように, アクチュエータの動作を決定して, アクチュエータ制御ブロックに渡す. 主に力学・運動学の知識に基づいて設計される.
- 自己位置推定ブロックでは, アクチュエータからの内界センサ情報に基づき, ロボットの自己位置を計算で求める. 運動学の知識に基づいて設計される.
- アクチュエータ制御ブロックは, アクチュエータが与えられた動作を実現するように, 電気信号を生成してハードウェアに出力する. 主に電磁気学・メカトロニクス・力学の知識に基づいて設計される.

このシステムは, 各ブロックの機能を分離して設計することが可能で, 切り分け方によっては, 柔軟な可汎性と, 高い再利用性を実現できる. 筑波大学 知能ロボット研究室で開発された移動ロボットプラットフォーム"山彦"では, 図中の点線で示すように, 移動体制御・自己位置推定より下のブロックが切り離して構築されており, Spur コマンドシステムを通して行動指示や自己位置取得を行っている. 本稿は, 特に移動体制御に注目して, 知能ロボット研究室 Robot Platform Project を中心に整備している, 移動ロボット走行制御コマンド系および走行制御系"YP-Spur"の実装と今後の展開について述べる.

また, 本稿は, 2010 年度第 1 回 山彦シンポジウム「特別セッション: 山彦ロボットプラットフォームの考え方」のために書き起こしたものであり, 本研究室で 1980 年代から 1990 年代にかけて開発され, その後現在まで改良されてきた研究成果に基づいている. 移動体制御および走行制御コマンドシステムの考え方・設計は文献^{[1][3][4]}に詳しい. また, より精密な移動体モデルを考慮した移動体制御と, 安定性などに関する考察も含めた横断的な情報が, 文献^[5]にまとめられている. 最新の変更やコマンド一覧などは, 知能ロボット研究室内部限定 Robot Platform Project^{*1)} のページを参照するとよい.

^{*1)} 筑波大学 知能ロボット研究室 内部限定 Robot Platform Project
<http://www.roboken.esys.tsukuba.ac.jp/internal/platform/>

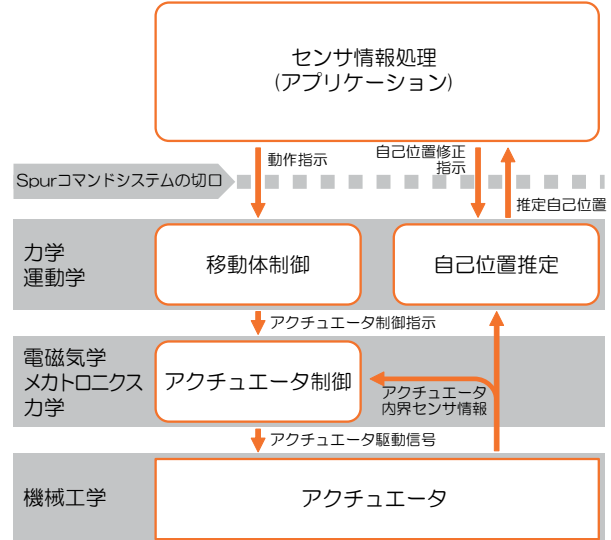


Fig. 1 自律移動ロボットの抽象的アーキテクチャの例

なお, 本文中で角丸の枠で囲われた部分は, YP-Spur パーバージョン 0.13.2 の実装で用いている式を表す. また, Spur の設計の基本となる考え方を, 太字で示している.

2. 移動ロボット"山彦"の制御の特徴

移動ロボットプラットフォーム"山彦"シリーズは, 独立に制御される 2 つの駆動輪を持つ, 独立二輪操舵 (Power Wheeled Steering/PWS) 型台車である. Figure 2 に示す "山彦 LR-1" も独立に制御される 2 つの駆動輪を持つ, PWS 型台車である. この PWS 型台車では, 2 つの駆動輪の回転数を制御することで, 動作の自由度を, 台車の前進成分と, 操舵成分に分離することが可能である. 具体的には, 台車の前進成分と操舵成分をそれぞれ $v(t)$, $\omega(t)$ とすると, PWS 台車のキネマティクスは, 左右の車輪の回転数 $\omega_r(t)$, $\omega_l(t)$, 車輪径 R_r , R_l , 左右車輪の間隔 (トレッド) T に対して, 次式の線形変換で表せる.

$$\begin{pmatrix} v(t) \\ \omega(t) \end{pmatrix} = \begin{pmatrix} \frac{R_r}{2} & \frac{R_l}{2} \\ \frac{R_r}{T} & -\frac{R_l}{T} \end{pmatrix} \begin{pmatrix} \omega_r(t) \\ \omega_l(t) \end{pmatrix} \quad (1)$$

また, 移動体の位置姿勢を (x, y, θ) とすると, 移動体のキネマティクスは, $v(t)$, $\omega(t)$ が与えられたとき次式で表せる.

$$\begin{cases} \theta(t) = \int_0^t \omega(\tau) d\tau + \theta(0) \\ x(t) = \int_0^t v(\tau) \cos[\theta(\tau)] d\tau + x(0) \\ y(t) = \int_0^t v(\tau) \sin[\theta(\tau)] d\tau + y(0) \end{cases} \quad (2)$$

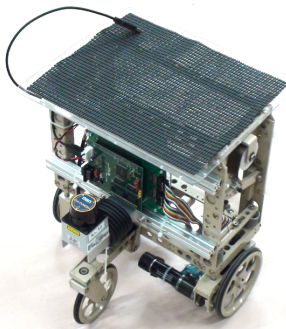


Fig. 2 移動ロボットプラットフォーム "山彦 LR-1"

PWS 型台車は、基本的に床や地面などの平面上を走行することが想定されている。このとき、平面上でのロボットの自己位置は、2 自由度の位置と 1 自由度の角度で表される。ところが、PWS 型台車は、自己位置の 3 自由度と、アクチュエータ（ここではモータ）の 2 自由度が、ノンホロノームな拘束関係をもつことが知られている。つまり、2 自由度のアクチュエータに到達目標値を与えて駆動したとき、最終的な 3 自由度の自己位置は、初期値と最終的な目標値の間における値の変化の軌跡によって決まる。従って、"山彦"に目標の動作をさせるためには、単に最終的な目標の位置や姿勢を指示するのではなく、最終的な目標の位置・姿勢を実現するような途中の経路を計画して指示する必要がある。その計画した経路を受け取って、経路通りに移動体を制御しようとする目的で開発されたのが、"Spur"コマンド系・制御系である。

3. YP-Spur とは

これまで、移動ロボット走行制御のコマンド系・走行制御系である Spur システムは、様々な実装がなされてきた。

MICHI 1980 年代後半に、油田助教授（当時）を中心に、有限会社コンピュータアプリケーションズに依頼して実装された、Spur の前身となる走行制御コマンド系・走行制御系である。基本的な走行制御系の構造は同じだが、コマンド系が異なる。1 枚の MC6809CPU ボード上に実装されていた。なお、"山彦 9 号"はこの時代から使用されていた。

Spur 1990 年頃に飯田氏を中心に開発された走行制御系である。1 枚の 68000CPU ボード上に、固定小数演算で実装されていた。

L-Spur 1990 年代中盤に宮井氏らを中心に開発された走行制御系で、2009 年度に旧来のコントローラボードを廃棄するまで使用されていた。Spur を浮動小数演算が可能な T805CPU の、Liberos に移植したもの。長い間使用された枯れたシステムであり、これ以降の"Spur"の多くは、Spur および L-Spur を基に実装されている。1 枚のトランスピュータ T805CPU ボード (T-LoCo ボード) 上に実装されていた。

vxv_tools 2001 から 2005 年頃にかけて上村氏によって開発された、走行制御・測域センサ制御などを含む、教育・研究用ロボット「ビーゴ」用ライブラリである。Spur に該当するシステムは、1 枚の SH2 マイコンボードに実装されている。

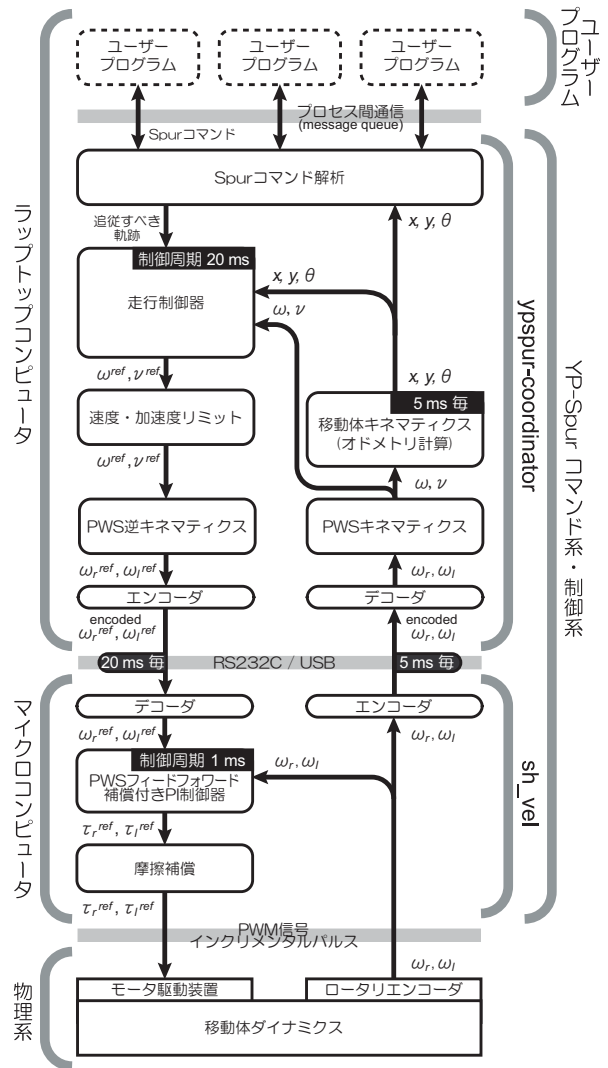


Fig. 3 YP-Spur コマンド系・制御系のブロック図

SHSpur 2005 から 2007 年頃にかけて竹内氏によって開発された走行制御系である。走行制御系をラップトップコンピュータ上に実装し、SH2 マイコンボード上にはモータの速度制御系のみを実装している。その後、各種の改良がなされ、統一した管理がされないまま使用されてきた。

MRspur 2008 から 2009 年に、富士ソフト株式会社が、次世代ロボット知能化技術開発プロジェクトの一環として知能ロボット研究室の協力の下で開発した走行制御系である。基本的な構成は SHSpur と同様である。

YP-Spur は、2010 年にロボットプラットフォーム整備のために発足した、知能ロボット研究室 Robot Platform Project において、研究室内の各所で別々に改良がなされていた SHSpur を統合して、新たに開発をスタートした走行制御系である。制御系の修正、バージョン管理方法の統一、通信プロトコル確認コマンドの追加などを経て、現在も開発が進められている。北陽電機株式会社の、測域センサ URG シリーズに搭載されている SCIP2.0 通信規格と互換のハードウェア情報問い合わせコマンド (VV コマンド) が実装されており、移動ロボットシステム上で、統一的にハードウェア情報を取得可能である。

4. YP-Spur のシステム構成

YP-Spur のコマンド系・制御系は、ラップトップコンピュータ上のプロセスと、マイクロコンピュータ上のプログラムに分けて実装されている。Figure 3 は、YP-Spur のコマンド系・制御系のブロック図を示している。

ラップトップコンピュータ上のプロセス ypspur-coordinator は、Spur コマンドの解析と走行制御を行う。ユーザープログラムは、専用のライブラリを用いて、プロセス間通信で ypspur-coordinator に走行制御や自己位置取得などのコマンドを送信し、結果を受け取る。ypspur-coordinator 中の走行制御器は、最近に与えられた走行制御コマンドに応じて、その制御器の構成を切り替えることで、様々な走行制御コマンドに対応している。走行制御コマンドごとの、制御器の構成については、5. 章で説明する。

マイクロコンピュータ上のプログラム sh_vel は、ラップトップコンピュータ上で動作している ypspur-coordinator から、起動時などに送信された制御パラメータに従い、逐次送信される目標値に追従するように、各駆動輪の角速度を PI 制御する。PI 制御器には、ある車輪を駆動したときに、PWS の動特性によって他の車輪に及ぼす影響などを打ち消すための、フィードフォワード補償器が内蔵されている。この PWS の動特性に対するフィードフォワード補償器については、文献^[2]に詳しい。また文献^[5]にも解説がまとめられている。ただし、現状では各ロボット用の標準パラメータには、PWS の動特性に関する項が設定されていない。

5. YP-Spur の走行制御

YP-Spur の走行制御コマンド系には以下の 6 種類の、軌跡追従制御コマンド、位置制御コマンド、速度制御コマンドが用意されている。

軌跡追従制御コマンド	
直線追従制御	Spur_line
円弧追従制御	Spur_circle
位置制御コマンド	
角度制御コマンド	Spur_spin
直線上での停止コマンド	Spur_stop_line
速度制御コマンド	
停止コマンド	Spur_stop
速度・角速度指令コマンド	Spur_vel

以降の章では、それぞれのコマンドにおける、走行制御器の構成について解説する。なお、そのほかの、パラメータ設定・取得コマンドや、自己位置の設定・取得コマンドに関しては、本稿では詳しく触れない。また、速度制御コマンドは、単に目標速度・角速度を 0 や任意の値に設定するコマンドなので、説明は省く。

なお、これらのコマンドが実行される際には常に、Fig. 3 中の速度・加速度制限部で、後述の各走行制御器が出力する速度・加速度、角速度・角加速度が制限される。目標の速度を

$v^{ref}(t + \Delta t)$ は、現在のロボットの速度を $v(t)$ とすると、許容速度 v_{max} 、許容加速度 a_{max} を超えないよう、次式を満たすようにクリップしている。

$$|v^{ref}(t + \Delta t)| < v_{max} \quad (3)$$

$$\left| \frac{v^{ref}(t + \Delta t) - v(t)}{\Delta t} \right| < a_{max} \quad (4)$$

ただし、現在の時刻を t 、制御周期を Δt とする。また、目標回転角速度 $\omega^{ref}(t + \Delta t)$ についても、許容回転角速度 ω_{max} 、許容回転角加速度 α_{max} を超えないよう、次式を満たすようにクリップしている。

$$|\omega^{ref}(t + \Delta t)| < \omega_{max} \quad (5)$$

$$\left| \frac{\omega^{ref}(t + \Delta t) - \omega(t)}{\Delta t} \right| < \alpha_{max} \quad (6)$$

6. 軌跡追従制御コマンド

軌跡追従制御コマンド実行時の YP-Spur の走行制御器は、時刻 t において、与えられた軌跡に対して、以下の $\eta(t)$ 、 $\phi(t)$ 、 $\omega_{diff}(t)$ を全て 0 にするような、フィードバック制御器を構成する。

ロボットと軌跡の距離	$\eta(t)$
軌跡に対するロボットの向き	$\phi(t)$
軌跡の追従に必要な角速度とロボットの角速度の差	$\omega_{diff}(t)$

この制御器は、フィードバックゲインを K_η 、 K_ϕ 、 K_ω 、目標の角加速度(制御出力)を $(\frac{d}{dt}\omega)^{ref}(t)$ とすると、以下の式で表される。

$$\left(\frac{d}{dt}\omega\right)^{ref}(t) = -K_\eta \eta(t) - K_\phi \phi(t) - K_\omega \omega_{diff}(t) \quad (7)$$

実際にはこの制御系は離散系であるので、その制御周期を Δt 、現在のロボットの角速度を $\omega(t)$ 、目標の角速度を $\omega^{ref}(t)$ とすると、走行制御器は以下の式で実装されている。

$$\begin{aligned} \omega^{ref}(t + \Delta t) \\ = \omega(t) + \Delta t (-K_\eta \eta(t) - K_\phi \phi(t) - K_\omega \omega_{diff}(t)) \end{aligned} \quad (8)$$

ただし、 $\eta(t)$ は、距離の制限値 η_{max} を超えないよう、式 (9) を満たすようにクリップしている。また、 $\phi(t)$ は、式 (10) を満たすように表現している。

$$|\eta(t)| < \eta_{max} \quad (9)$$

$$-\pi < \phi(t) < \pi \quad (10)$$

ロボットの目標の並進速度 $v^{ref}(t)$ は、コマンドを通してユーザから指示された値 $v^{ref'}(t)$ から、以下の式で与えている。

$$v^{ref}(t) = v^{ref'}(t) - \text{sign}\left(v^{ref'}(t)\right) |\omega(t)| \quad (11)$$

これは、角速度が大きいときに、速度を減少させることで、ロ

ボットに働く遠心加速度を抑え、タイヤのスリップを防止する効果がある。

なお、現在は式 (11) が実装されているが、これはロボットの速度が比較的遅く、ほぼ一定の値で使用されていたときに策定されたものである。物理的には、ロボットがある程度のスピードで走行しながら、回転 (旋回) する時は、横方向に遠心力が発生し、内側の車輪が浮く現象が発生する。これは制御性を悪くすると同時に車輪の回転による自己位置計算に大きな誤差を発生させる。このため、横方向の遠心加速度に予め制限を与えておき、それを超えるときは、速度を制限する必要がある。制限する横加速度を $\alpha \cdot g$ ($\alpha = 0.1 \sim 0.3$ 程度, g は重力加速度) とすると、与えられた $\omega(t)$ に対する制限速度は、

$$v_{\max_by_w}(t) = \frac{\alpha \cdot g}{|\omega(t)|} \quad (12)$$

となり、速度の絶対値はこの値でクリップする必要がある。

6.1 直線追従制御

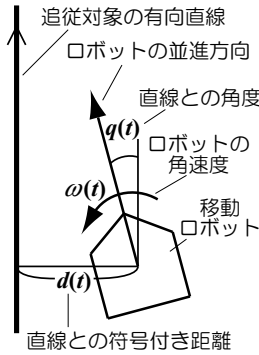


Fig. 4 YP-Spur の実装における、直線追従制御のパラメータ

YP-Spur の直線追従制御は、Fig. 4 に示す直線との符号付き距離 $d(t)$ 、直線との角度 $q(t)$ 、ロボットの角速度 $\omega(t)$ を用いて、以下の値を、制御式 (8) に与えることで実装している。

$$\begin{cases} \eta(t) = d(t) \\ \phi(t) = q(t) \\ \omega_{diff}(t) = \omega(t) \end{cases} \quad (13)$$

実際に、移動ロボット"山彦 LR-1"を、上記の制御で直線追従させた際の軌跡を、Fig. 5, 6 に示す。これらは、移動ロボット"山彦 LR-1"を用いて、YP-Spur 標準ロボットパラメータバージョン 0.1.2 を使用して、許容最大速度 0.3m/s、許容最大角速度 1.5rad/s の条件で実験した結果である。若干のオーバーシュートをしながら、直線に追従していることが分かる。ところで、移動体のキネマティクスの式 (2) より、 $\theta(t) \simeq 0$ 、かつ $v(t)$ が v^{ref} で一定のとき、ロボットと軌跡の距離は、

$$y(t) = \int_0^t v(\tau) \sin[\theta(\tau)] d\tau \simeq v^{ref} \int_0^t \theta(\tau) d\tau$$

つまり、対象の直線を x 軸とする座標系で考えれば、

$$\frac{d}{dt} \eta(t) \simeq v^{ref} \phi(t)$$

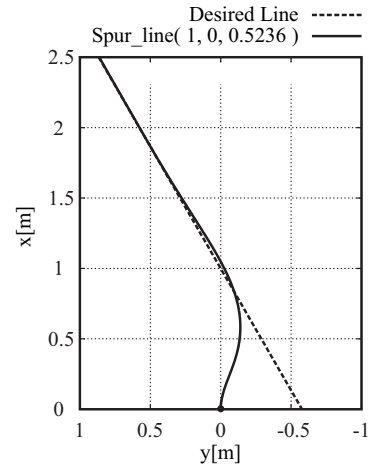


Fig. 5 直線追従制御の実行例: (0, 0) から 0 度方向を向いてスタートし、(1, 0) を通る 30 度方向を向いた直線に追従

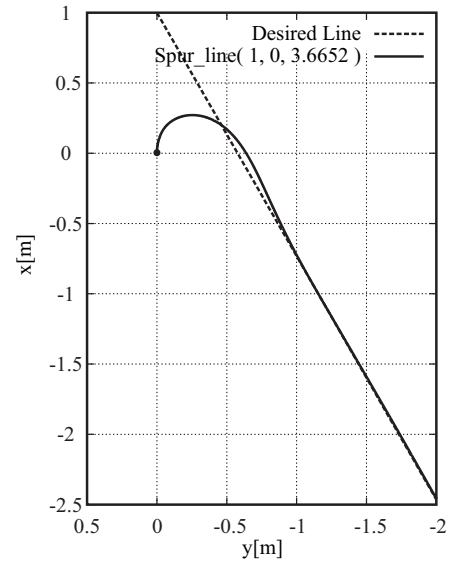


Fig. 6 直線追従制御の実行例: (0, 0) から 0 度方向を向いてスタートし、(1, 0) を通る 210 度方向を向いた直線に追従

の関係をもつ。また、

$$\frac{d}{dt} \phi(t) = \omega_{diff}(t) = \omega(t)$$

も満たす。従って、ロボットの角速度 $\omega(t)$ が、 $\omega^{ref}(t)$ に十分な速さで追従するとき、式 (7) で表される制御系の挙動は前の条件下で、以下の斉次線型常微分方程式で表される。

$$\frac{d^3}{dt^3} \phi(t) = -K_\eta v^{ref} \phi(t) - K_\phi \frac{d}{dt} \phi(t) - K_\omega \frac{d^2}{dt^2} \phi(t) \quad (14)$$

式 (14) は、この直線追従制御系において、ロボットがほとんど目標の直線と同じ方向を向いていて、並進速度が正で一定、かつロボットの角速度が目標に十分な速さで追従するとき、正の K_η 、 K_ϕ 、 K_ω を、極の実数部が正になるように与えることで、ロボットの姿勢が安定化されることを示している。一方、この直線追従制御はロボットが目標の軌跡から大きく外れている場合の安定性は考慮されていない。制御パラメータや追従対象の直線の与え方によっては、ロボットが直線に追従す

るまでに時間がかかったり、発振・発散して経路から外れていく場合がある。条件式 (9, 10) で、フィードバック値をある程度クリップしているため、適切に制御パラメータを選べば、ロボットは目標の軌跡から外れていても、過渡的な動作を経て直線に追従することが多い。ただし、特にロボットの許容回転角速度や許容回転角加速度の制限が強い場合に系が発振することが多いので、注意が必要である。

また式 (14) は、ロボットの進んだ距離 x に対するロボットの姿勢 $\phi'(x)$ を用いて、次のように表すことができる。

$$\begin{aligned} \frac{d^3}{dt^3} \phi'(x) = & -\frac{K_\eta}{(v^{ref})^2} v^{ref} \phi'(x) - \frac{K_\phi}{(v^{ref})^2} \frac{d}{dt} \phi'(x) \\ & - \frac{K_\omega}{v^{ref}} \frac{d^2}{dt^2} \phi'(x) \end{aligned} \quad (15)$$

ただし、

$$\phi'(x) = \phi\left(-\frac{x}{v^{ref}}\right)$$

式 (15) は、ロボットの走る軌跡は並進速度の影響を受け、例えば、速度が速くなると、実質的に軌跡追従制御のフィードバックゲインが小さくなり、与えられた直線に収束するまでの距離が長くなることを示している。これは、並進速度が大きいときに大きな角速度を与えた場合に、ロボットに働く遠心加速度が非常に大きくなってタイヤがスリップする現象を防ぐ目的もある。よって並進速度が大きいときには、軌跡追従制御が収束するまでに、比較的長い距離がかかることを考慮して目標の軌跡を与える必要がある。

この直線追従制御系のフィードバックゲインの決定は、経験的に行う事が多い。式 (15) から軌跡追従の時定数を求めることで、ゲインの妥当性を確認することができる。また、筆者は文献 [6] で、直線との角度が大きい非線形領域での挙動も含めてくり返しシミュレーションを行い、ヒューリスティックな手法でゲインを最適化する手法を報告しており、これを用いてフィードバックゲインを決定することもできる。

6.2 円弧追従制御

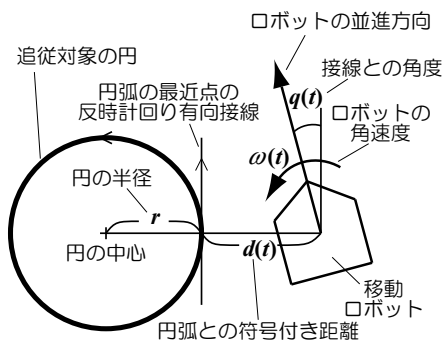


Fig. 7 YP-Spur の実装における、円弧追従制御のパラメータ

YP-Spur の円弧追従制御は、Fig. 7 に示す追従対象の円の半径 r 、円弧との符号付き距離 $d(t)$ 、円弧の最近点における有向接線とロボットの角度 $q(t)$ 、ロボットの角速度 $\omega(t)$ を用いて、以下の値を、制御式 (8) に与えることで実装している。半径 r が負の場合は反時計回り、正の場合は時計回りの円を考える。

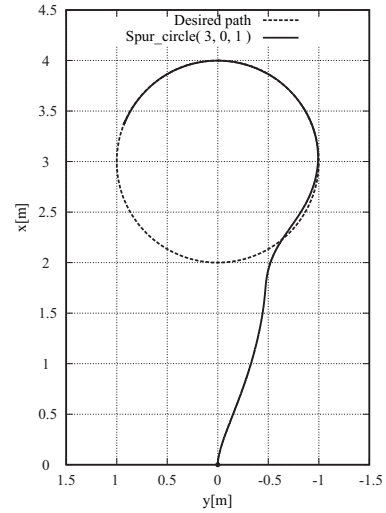


Fig. 8 円弧追従制御の実行例: (0, 0) から 0 度方向を向いてスタートし、(2, 0) を中心とする半径 0.3m の円弧に追従

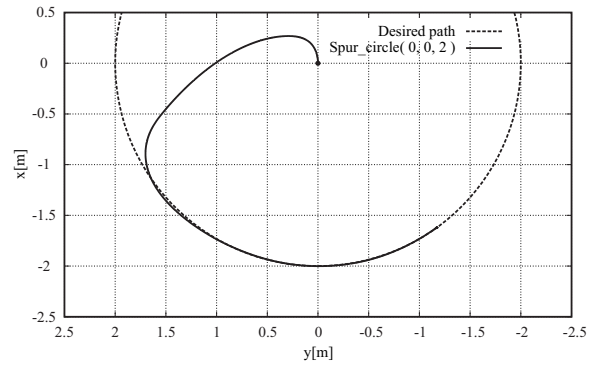


Fig. 9 円弧追従制御の実行例: (0, 0) から 0 度方向を向いてスタートし、(0, 0) を中心とする半径 2m の円弧に追従

$$\begin{cases} \eta(t) = d(t) \\ \phi(t) = q(t) \\ \omega_{diff}(t) = \omega(t) - \omega_{required} \end{cases} \quad (16)$$

ただし、

$$\omega_{required} = \begin{cases} -\omega_{max} & \left(\frac{v(t)}{r+d(t)} \right) < -\omega_{max} \\ \frac{v(t)}{r+d(t)} & \left(\left| \frac{v(t)}{r+d(t)} \right| \right) < \omega_{max} \\ \omega_{max} & \left(\frac{v(t)}{r+d(t)} \right) > \omega_{max} \end{cases}$$

式中の $\omega_{required}$ は、追従すべき円の中心からロボットの距離を半径とする、同心円を移動するために必要なロボットの角速度である。

実際に、移動ロボット"山彦 LR-1"を、上記の制御で円弧追従させた際の軌跡を、Fig. 8,9 に示す。これらは、移動ロボット"山彦 LR-1"を用いて、YP-Spur 標準ロボットパラメータバージョン 0.1.2 を使用して、許容最大速度 0.3m/s、許容最大角速度 1.5rad/s の条件で実験した結果である。大きく円弧を描くような軌跡で、円弧に追従していることが分かる。

また、円弧追従制御系についても、直線追従制御系の場合と

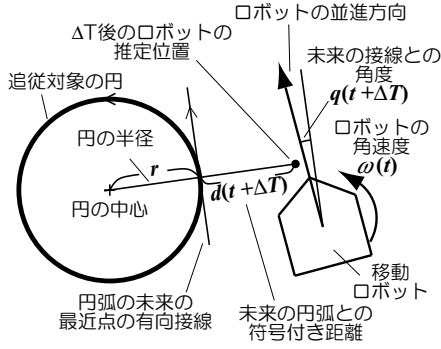


Fig. 10 過去の Spur の実装における, 円弧追従制御のパラメータ

同様に, ロボットがほとんど目標の直線と同じ方向を向いていて, 並進速度が正で一定, かつロボットの角速度が目標に十分な速さで追従するとき, 正の K_η , K_ϕ , K_ω を, 式 (15) の極の実数部が正になるように与えることで, ロボットの姿勢が安定化されることが分かる.

過去の Spur の実装では, Fig. 10 に示すように, 円弧との符号付き距離と円弧の最近点における有向接線とロボットの角度を, それぞれ ΔT 後の時刻における推定値 $d(t+\Delta T)$, $q(t+\Delta T)$ を用いて, 以下の式で表す値を制御式 (8) に与えて制御していた.

$$\begin{cases} \eta(t) = d(t+\Delta T) \\ \phi(t) = q(t+\Delta T) \\ \omega_{diff}(t) = \omega(t) \end{cases} \quad (17)$$

この方法では $\omega(t)$ を 0 にするようなフィードバックがかかるため, ΔT が小さいとき, ロボットの追従する円軌跡の径が大きくなる. また, ΔT が大きすぎると径が小さくなり, ΔT の値を適切に選ぶ必要があった.

7. 位置制御コマンド

位置制御コマンド実行時の YP-Spur の走行制御器は, ロボットの位置・姿勢を, 与えられた位置・姿勢の目標値で停止させるように, フィードバック制御を行う. ただし, 2. 章で述べたように, 平面上の PWS 型台車の 3 自由度の位置・姿勢を, 任意の状態から目標値に一致させるためには, 非線形な移動体のキネマティクスを考慮した経路計画を行う必要があり, 単純なフィードバック制御での実現は困難である. そこで, Spur の位置制御コマンドでは, 3 自由度の位置・姿勢のうち, 1 自由もしくは 2 自由度分の成分のみを目標値に一致させる制御を行っている. 軌跡追従制御コマンドと位置制御コマンドを組み合わせることで, 3 自由度の位置・姿勢を目標値にほとんど一致させることが可能になる.

7.1 角度制御コマンド

角度制御コマンド実行時の YP-Spur の走行制御器は, ロボットの向き $\theta(t)$ を, 目標の向き θ^{ref} に停止させるように, ロボットの角速度 $\omega^{ref}(t)$ を制御出力としたフィードバック制御を行う. このとき, 許容最大角加速度 α_{max} で減速したときに, 目標の向きで速度 0 になるような角速度 $\omega^{ref}(t)$ を計算

で求めて出力する.

$$\begin{cases} \omega(t+t_0) = 0 \\ \theta(t) + \int_t^{t+t_0} \omega(t) dt = \theta^{ref} \\ \frac{d}{dt} \omega(t) = \alpha_{max} \end{cases}$$

以上を満たせばよいので, このフィードバック制御は次式で表される.

$$\omega^{ref}(t+\Delta t) = \text{sign}(\theta(t) - \theta^{ref}) \sqrt{2\alpha_{max}|\theta(t) - \theta^{ref}|} \quad (18)$$

また, ロボットの目標の並進速度 $v^{ref}(t)$ は 0 を与えている. 角度制御コマンドの実行中, ロボットの位置に関しては制御されておらず, 外から力が加わればその位置が変化する場合がある.

7.2 直線上での停止コマンド

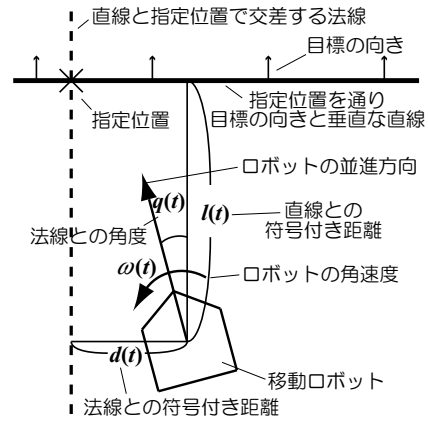


Fig. 11 YP-Spur の実装における, 直線上での停止コマンドのパラメータ

直線上での停止コマンド実行時の YP-Spur の走行制御器は, Fig. 11 に示すロボットの位置を, 指定された位置を通り, 目標の向きと直交する直線上で, 目標の向きを向いて停止するようにロボットの目標角速度 $\omega^{ref}(t)$, 目標並進速度 $v^{ref}(t)$ を制御出力としたフィードバック制御を行う. 目標並進速度の算出方法は, 角度制御コマンドにおける目標角速度と同様である.

また, 停止する直線との距離が遠い場合には, 直線追従と同様の制御を行う. 具体的には, 目標並進速度 $v^{ref}(t)$ を式 (19) で与え, 停止する直線の法線との符号付き距離 $d(t)$, 法線との角度 $q(t)$, ロボットの角速度 $\omega(t)$, 停止する直線との符号付き距離 $l(t)$ を用いて, 式 (20) で示す値を制御式 (8) に与えることで実装している.

$$v^{ref}(t + \Delta t) = \text{sign}(l(t)) \sqrt{2a_{max}|l(t)|} \quad (19)$$

$$\begin{cases} \eta(t) = \begin{cases} d(t) & v^{ref}(t + \Delta t) < v_{max} \\ 0 & \text{otherwise} \end{cases} \\ \phi(t) = q(t) \\ \omega_{diff}(t) = \omega(t) \end{cases} \quad (20)$$

一方、直線上での停止コマンドの実行中、停止する直線との距離が近いときには、ロボットの位置は停止する直線の水平方向には制御されておらず、外から力が加わればその位置が変化する場合がある。ただし、停止する直線との距離が十分長ければ、ロボットは初め直線追従と同様に動作するので、指定した位置の点に向かって進む。この場合、停止する直線に近づいた際に外乱が働かなければ、最終的には指定した位置の近傍で停止することが期待できる。

8. おわりに

本稿では、移動ロボット走行制御コマンド系および走行制御系、"Spur"の考え方と、2010年度現在の知能ロボット研究室の標準走行制御系である、"YP-Spur"の実装について報告した。本稿で説明した内容の多くは、研究室の先人たちが残した多くの論文・ソースコードなどの財産を参考にしたものである。

現在の"Spur"は、計算機技術の発展に伴って、走行制御系の大部分をラップトップコンピュータ上で動作させる構造に変化した。実際の使用例でも、軌跡追従制御の代わりに、画像処理や測域センサデータ処理で得られた情報に基づいた制御を実装して動作させるなど、柔軟に移動ロボットのシステムを

構築することが可能になっている。今後も、利用形態や他の技術の進歩にあわせて、走行制御コマンド系および走行制御系を整備していくことは、移動ロボットに関する研究を行っていく上で極めて重要である。

参考文献

- [1] S. Yuta, Y. Kanayama. *An implementation of MICHl - A locomotion command system for intelligent mobile robot*, in Proc. of International Conference on Advanced Robotics, pp. 127-134, 1985
- [2] S. Iida, S. Yuta, *Control of Vehicle with Power Wheeled Steerings Using Feed-forward Dynamics Compensation*, in Proc. of Annual Conference on the IEEE Industrial Electronics Society, pp. 2264-2269, 1991
- [3] 飯田重喜, 油田信一. 車輪型移動ロボットのための走行制御コマンド系と軌跡制御方式, in Proc. of 第1回日本ロボット学会ロボットシンポジウム, pp. 85-90, 1991
- [4] 飯田重喜. 車輪型移動ロボットの走行制御システムに関する研究, 筑波大学大学院博士課程工学研究科 学位請求論文, 1991
- [5] 坪内孝司. 車輪移動体の制御, in Proc. of 日本ロボット学会主催第43回講習会 ロボット工学入門シリーズ<移動技術編>『移動ロボットのやさしい解説』, pp. 58-68, 1995
- [6] 渡辺敦志. 繰り返しシミュレーションとヒューリスティック最適化による制御系のパラメータ決定手法～広い速度域に対応した移動ロボットの直線追従パラメータ生成～ In Proc. of the 2009年度第2回山彦シンポジウム

訂正内容

rev. 1	2010.07.21	初稿
rev. 2	2010.07.23	p.4, 6 安定性の議論を修正
rev. 3	2010.07.27	p.2, Fig. 3 中に制御・通信周期を追加
rev. 4	2010.08.19	p.2, Fig. 3 中の制御・通信周期を修正

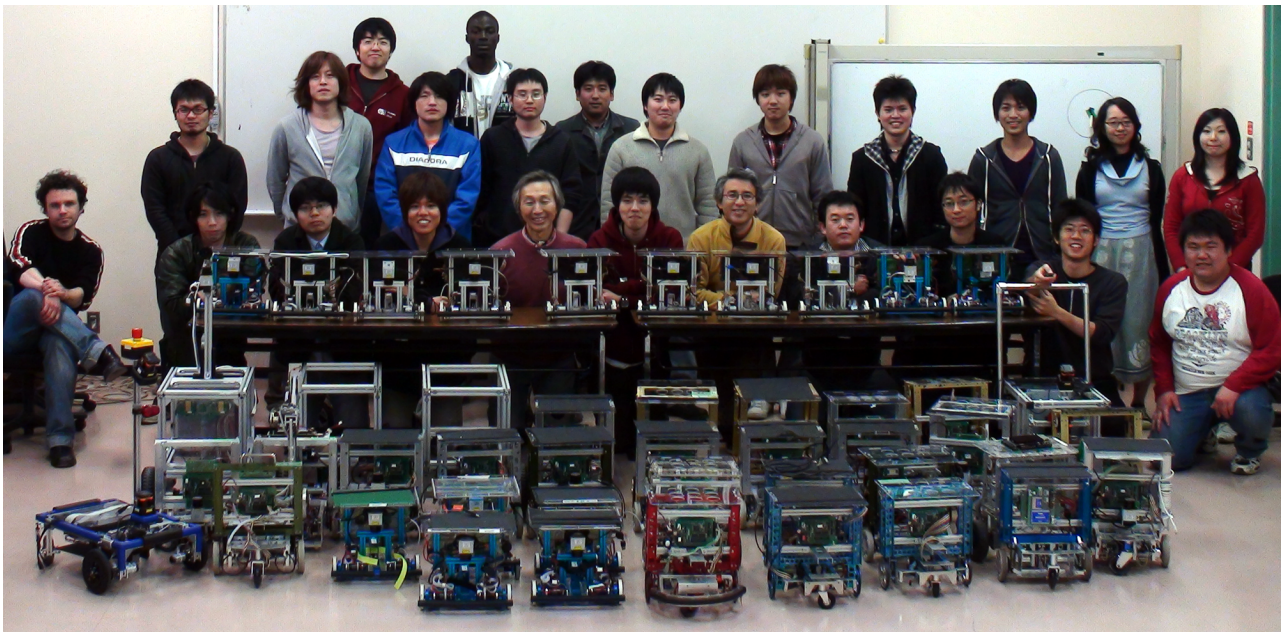


Fig. 12 2010年度プラットフォーム整備のための作業日, "山彦"シリーズと"ビーゴ"シリーズ, および筑波大学 知能ロボット研究室メンバー