モデル予測制御を用いた二輪車両ロボットの制御
―ニューラルネットワークによる制御則の学習―

2017SC007 伏屋主水之介　2017SC035 真野翼　2017SC042 盛田雄太
指導教員：大石泰章

1　はじめに

現在, 港や倉庫などで荷物を運搬する車両の大部分は人が運転している. 毎回コンテナなどの障害物の位置が異なる条件で, フォークリフトなどの運搬車両を自動運転することができれば人件費削減や夜間の作業を安全に行えるというメリットがある. 本研究では, 二輪車両ロボットに障害物を回避させつつ, 目標地点まで走行させる制御を考える. 二輪車両ロボットの制御には様々な方法が提案されてきた. 例えば, 不連続フィードバックによる方法 [1], サンプル値制御による方法 [2], 時間軸状態制御法による方法 [3][4] などがある.

我々は, モデル予測制御 [5] にもとづいてこの課題に取り組む. モデル予測制御は作業用ロボットの障害物回避など様々な研究で用いられており, 実現したい動作を元に目的関数を設計することで様々な制御を行うことが出来る.

実験機器としては, 二輪車両ロボット Zumo の使用を想定する. しかしモデル予測制御は計算負荷が高いため, Zumo 実装時の計算処理能力を考慮し, ニューラルネットワーク近似による計算量の低減を行う. 今回使用する Zumo の概要を図 1 に示す. ただし, 本研究では実際には Zumo は使わず, MATLAB を用いたシミュレーションのみを行った.

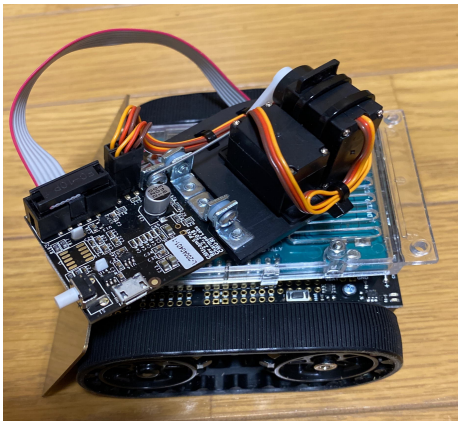

図 1 使用する二輪車両ロボット Zumo

2　問題設定

本研究では港や倉庫などにおけるフォークリフトなどの車両制御を想定する. 与えられた出発地点と目標地点に対してモデル予測制御を使用することにより適切な経路で車両を動作させる. 途中で障害物を検知するとそれを回避する新たな適切な経路と車両に対する入力を算出し実行する. これを繰り返すことで適切な経路を通り目標地点に到達する.

今回使用を想定する Zumo の大きさは, 縦 10cm, 横 10cm, 高さ 9cm であるため旋回半径を 5cm と考えてシミュレーションを行い, 車両は横滑りをしないものとする.

シミュレーションで考える環境を図 2 に示す. 車両の出発地点は $(x, y) = (0.4, 0.8)$ で目標地点を $(x_0, y_0) = (0.4, 0.4)$ とし, 車両は最初 x 軸正の方向を向いており, 目標地点でも x 軸正方向を向くものとする. また, 出発地点と目標地点の間に障害物を設ける.

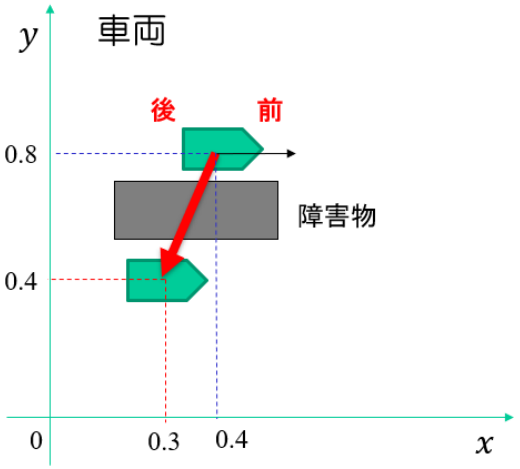

図 2 二輪車両ロボットのシミュレーション環境

3　車両のモデルと制御

3.1　制御モデルの設定

モデル予測制御とは有限時間の未来を制御対象のモデルにもとづいて予測し, 与えられた目的関数を最小化するような入力系列を求めてそのうちの最初の入力だけを与えることを繰り返す制御方法である. 本研究ではこれを利用し, 障害物の存在を考慮した目標地点への到達について制御を行う.

図 3 のような二輪車両の構造を考える. u_1 と u_2 はそれぞれ左の車輪の角速度, 右の車輪の角速度を表しており, どちらも制御入力として値を決められるものとする. r は左

右の車輪の半径である.

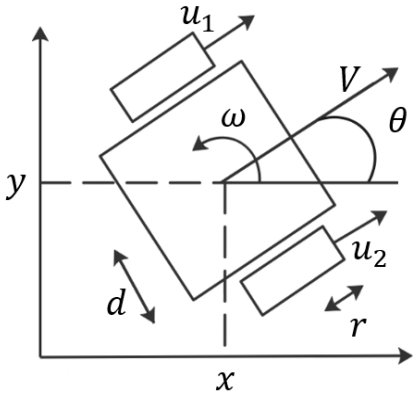

図 3 二輪車両ロボットのモデル

左右の車輪の速度は, それぞれ以下の式の V_r, V_l で定義される:

$$V_l = ru_1, \tag{1}$$

$$V_r = ru_2. \tag{2}$$

また, d を車両の重心から左右の車輪までの距離とすると, 車両の重心の速度 V と車両の重心の周りの旋回角速度 ω は

$$V = (V_r - V_l)/2, \tag{3}$$

$$\omega = (V_r - V_l)/(2d) \tag{4}$$

で与えられる. 式 (1), (2) を式 (3), (4) にそれぞれ代入し, 車体の速度, 旋回角速度と左右の車輪の角速度の関係を以下のように得る:

$$V = r(u_2 - u_1)/2,$$

$$\omega = r(u_2 - u_1)/(2d).$$

車両が走行する平面上に xy 直交座標系をとり, 車両の進行方向が x 軸正方向に対してなす角を θ とすると, 状態空間表現は以下のようになる:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} V + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega.$$

3.2 障害物がないときの目的関数

目的関数は車両の位置と目標地点までの距離, 車両と障害物の距離, 入力の大きさの評価を行うための関数であり, モデル予測制御では目的関数を最小化することで適切な入力を算出する. 障害物の回避ではポテンシャル法を用いたものがあり, 本研究ではこれに習って障害物の距離を考慮した目的関数を用いる.

目的関数 J は, J_1, J_2, J_3 の 3 項から成る. 以下, x_0, y_0 は目標地点の x 座標, y 座標を表している. またシミュレーション区間を刻み幅 h で分割したものをステップ, 現在時刻までのステップ数を n とし, p は予測ステップ数を表す. 以降の式にある $x((n+k)h|nh)$ は時刻 nh で予測した $n+k$ ステップ目における車両の x 座標であり, $y((n+k)h|nh)$ は時刻 nh で予測した $n+k$ ステップ目における車両の y 座標である. また, $\theta((n+k)h|nh)$ は, 時刻 nh で予測した $n+k$ ステップ目における車両の角度であり, 角度は x 軸正方向を基準に測るものとする.

u は左右の車輪の入力角速度 u_1, u_2 から成るベクトルであり, $u((n+k)h|nh)$ は時刻 nh で予測した $n+k$ ステップ目における入力であって

$$u((n+k)h|nh) = \begin{bmatrix} u_1((n+k)h|nh) \\ u_2((n+k)h|nh) \end{bmatrix}$$

である. 左右の車輪の角速度の制約条件を $-10[\mathrm{rad/s}] \leq u \leq 10[\mathrm{rad/s}]$ とする.

J_1 は車両と目標地点の座標との距離に関する項であり,

$$J_1 = \sum_{k=0}^{p-1} -C_1 \exp(-\sqrt{((x((n+k)h|nh) - x_0)^2 + (y((n+k)h|nh) - y_0)^2)/L_1^2}$$

のように定義される. ただし, L_1, C_1 は正の定数である.

J_2 は車両の入力に関する項であり,

$$J_2 = \sum_{k=0}^{p-1} u((n+k)h|nh)^T Ru((n+k)h|nh),$$

のように定義される. ただし, R は u に対する重みの行列である.

J_3 は目標地点の方向と機体の向きがなす角に関する項であり,

$$J_3 = \sum_{k=0}^{p-1} (\theta_0((n+k)h|nh) - \theta((n+k)h|nh))^2 C_2 \exp(-\sqrt{((x((n+k)h|nh) - x_0)^2 + (y((n+k)h|nh) - y_0)^2)/L_2^2)),$$

のように定義される. ただし, $\theta_0((n+k)h|nh)$ は時刻 nh で予測した $n+k$ ステップ目における目標地点へ向かう方向を表している. また, C_2, L_2 は正の定数である:

上記の J_1, J_2, J_3 を使い障害物がない場合の目的関数 J は

$$J = J_1 + J_2 + J_3, \tag{5}$$

となる.

3.3 モデル予測制御の動作手順

以下の章で使用するモデル予測制御の動作手順を以下に示す. 定義した目的関数を用いて

STEP1

予測区間内で目的関数を最小化する入力系列 $u(nh|nh)$, $u((n+1)h|nh), \ldots, u((n+p-1)h|nh)$ を計算する.

STEP2

入力系列のうち最初の入力 $u(nh|nh)$ を加え, 状態の更新を行う.

STEP3

シミュレーション終了時間まで STEP1 と STEP2 を繰り返す.

のように計算を行う.

3.4 障害物回避のアルゴリズム

ここで障害物が存在する場合について考える. 障害物回避のために文献 [6] に習ってペナルティ項 J_4 を用意し, 目的関数 J に加える:

$$J_4 = \sum_{i=1}^{l} \sum_{k=0}^{p-1} C_3 \exp(-$$

$$\sqrt{(x((n+k)h|nh) - x_i)^2 + (y((n+k)h|nh) - y_i)^2}/L_3^2). \quad (6)$$

障害物についてのペナルティは 360 度を 20 分割し, それぞれにセンサを配置して検知することを想定している. 検知した障害物の数を l とし, x_i, y_i はそれぞれの障害物の x 座標, y 座標とする. C, L は正のスカラーである. 式 (6) を式 (5) に加えることで, 障害物回避を考慮した目的関数となる:

$$J = J_1 + J_2 + J_3 + J_4. \quad (7)$$

また, 以降のシミュレーションで使うパラメータを表 1 に示す.

3.5 障害物がある場合のシミュレーション

以下に障害物がある場合のシミュレーション結果を表示する. また, 以降のシミュレーション結果における障害物は半径 0.05 の黒点とし赤丸を出発地点, 青丸を目標地点とする.

表 1 パラメータ

記号	名称	値
J	目的関数の初期値	0
R	入力に対する重み	diag(0.01, 0.01)
L_1	J_1 の影響範囲への重み	2
C_1	目的地の座標に対する重み	5500
L_2	J_2 の影響範囲への重み	0.04
C_2	姿勢角に対する重み	100
L_3	J_3 の影響範囲への重み	0.1527
C_3	障害物に対する重み	8000
L_4	J_4 の影響範囲への重み	0.3
C_4	右手法による引力に対する重み	900

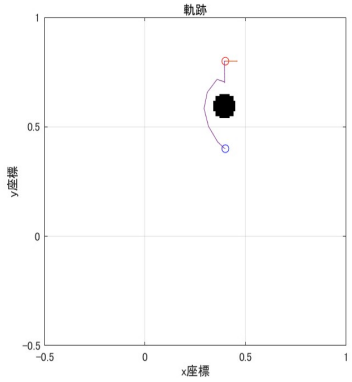

図 4 一つの障害物でのシミュレーション結果

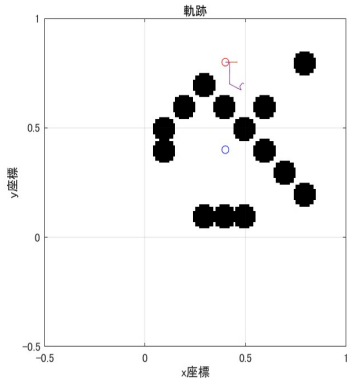

図 5 途中で停止した場合のシミュレーション結果

図 4 のような単純な障害物配置の場合, 出発地点 $(0.4, 0.8)$ から目標地点 $(0.4, 0.4)$ まで障害物を回避しながら適切な経路で到達していることが分かる.

図 5 のように出発地点 (0.4, 0.8) から目標地点 (0.4, 0.4) に到達する前に停止してしまう場合がある. これは, 目的関数をこれ以上減少させることができなくなってしまったことが原因であると考える. このような場合を解決するために次章で提示する右手法を採用する.

4　目的関数上での右手法の運用

4.1　右手法について

前述したように目標地点に到達する前に停止する状態からの脱出が必要であり, この解決方法に右手法を活用する. 右手法とはスタートとゴールが外側に面している場合, 壁に右手をつけて歩くとゴールに辿り着くという考え方である. J_4 までを使用したアルゴリズムでは, 壁状に大きく広がる障害物などの場合, J_1 と J_4 が打ち消しあう事でこれ以上目的関数を減少させる入力が計算できず, その点で停止してしまう. この点から脱出するために車両から見て左手側に引きつける項を目的関数に加える. これにより壁沿いに左方向に向かう事で目的関数が減少するため, 上記の状態から脱出し目標地点までモデル予測制御を継続して行うことが可能となる.

4.2　右手法の手順

右手法の動作手順を以下に示す.

STEP1

センサーが障害物を検知し, 検知した障害物の左手側に引きつける項を目的関数に加える.

STEP2

目的関数の最小化を行って制御すると, 左手側に引き寄せられ障害物の左側へ車両が移動する.

4.3　右手法のアルゴリズム

左手側に引きつけるために目的関数に加える項を J_5 とする. 時刻 nh で予測した $n+k$ ステップ目における車両の x 座標を $x((n+k)h|nh)$, y 座標を $y((n+k)h|nh)$ とし, i 番目の障害物の x 座標を x_i, y 座標を y_i とする. C_3, L_3 は正の定数である. まず, 70° 回転する行列を

$$L = \begin{bmatrix} \cos\frac{7}{18}\pi & -\sin\frac{7}{18}\pi \\ \sin\frac{7}{18}\pi & \cos\frac{7}{18}\pi \end{bmatrix},$$

を用いて,

$$g = L\left(\begin{bmatrix} x_i \\ y_i \end{bmatrix} - \begin{bmatrix} x(nh) \\ y(nh) \end{bmatrix}\right) + \begin{bmatrix} x(nh) \\ y(nh) \end{bmatrix}.$$

と定める. ベクトル g は車両をひきつけたい点の座標を表す. g の x 成分と y 成分 g_x, g_y を用いて, 次を定義する.

$$W = -C_4 \exp\left(\sqrt{(x((n+k)h|nh) - g_x)^2 + (y((n+k)h|nh) - g_y)^2}/L_4^2\right).$$

また, Q を

$$Q = \sqrt{(x((n+k)h|nh) - x_0)^2 + (y((n+k)h|nh) - y_0)^2}.$$

とし, J_5 を

$$J_5 = \sum_{i=1}^{l} \sum_{k=0}^{p-1} QW.$$

とする. ここで W に Q をかけるのは車両と目標地点の距離に応じて W の影響を変化させるためである.

式 (7) に J_5 を加えた

$$J = J_1 + J_2 + J_3 + J_4 + J_5. \tag{8}$$

により目標地点に到達する前に停止してしまう状態から脱出することができる. 以降のシミュレーションでは式 (8) を使用する.

4.4　右手法を導入したシミュレーション

図 6 に右手法を導入したシミュレーション結果を示す. 右手法を目的関数に加えたことにより出発地点 $(0.4, 0.8)$ から目標地点 $(0.4, 0.4)$ まで車両が移動できている. これより目標地点に到達する前に車両が停止する状態から脱出できたことが分かる.

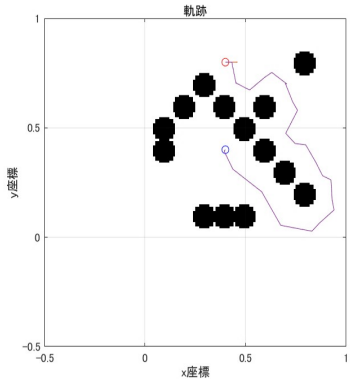

図 6　右手法の導入による二輪車両ロボットのシミュレーション結果

5　目的関数の切り替え

5.1　目的関数の使用方法

4 章では障害物を検知した際に右手法を利用していたが, 5 章では右手法を用いない関数と右手法を用いる関数の二つを切り替えて制御することを考える. このような方法を考えるのは, 4 章で使用した関数では袋小路の場合に J_1 と J_5 が打ち消し合ってしまうため目標地点まで到達できない

場合があり, この問題を解決するためである.

MATLAB への実装は Lumelsky ら [7] が提案した BUG2 というアルゴリズムを元に行う. この BUG2 は

STEP1

出発地点と目標地点を直線で結び, その直線に沿って目標地点まで移動する.

STEP2

障害物を検知すると右手法で障害物にそって移動することで回避し, その後直線を離れた位置よりも目標地点近くで STEP1 で結んだ直線に戻った場合にその直線に沿って目標地点まで移動する.

のような手順で行われる.

使用する目的関数を以下に示す.

$$J = J_1 + J_2 + J_3 + J_4, \tag{9}$$

$$J = J_3 + J_4 + J_5. \tag{10}$$

ここで式 (9) は STEP1 における直線方向への移動として, 式 (10) は STEP2 における右手法として考えることができる. よって, これらを切り替えて使用することで問題の解決に当たる.

5.2 実験結果

以下に目的関数を切り替えない場合と切り替えた場合のシミュレーション結果を示す.

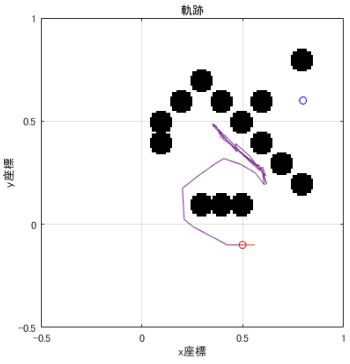

図 7 目的関数を切り替えない場合の結果

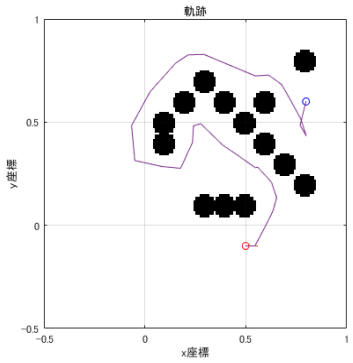

図 8 目的関数を切り替えた場合の結果

上記のような複雑な障害物配置において目的関数を切り替えない場合は適切な制御ができないために目標地点へ到達できていない. しかし, 目的関数を適切に切り替えることによって車両が目標地点へ到達できることが分かった.

6 制御則の近似

6.1 近似の必要性

Zumo 車両にプログラムを実装する場合, モデル予測制御の計算を Arduino 上でリアルタイムに処理するのは計算負荷が大きく不可能である. そのためあらかじめ車両の状態と入力を紐づけしておき, そのときの車両の状態に応じてただちに求められる入力を生成できるようにする. この方法を取ることにより, リアルタイムの制御を可能とする.

6.2 近似方法

近似はニューラルネットワーク (Neural Network; NN) を用いて行う. NN への入力として, 状態を 26 次元のベクトル z で表現したものをデータとして用いる. すなわち $z = [x, y, \theta, x_0, y_0, \theta_0, 1/d_1, 1/d_2, ..., 1/d_{20}]$. ここで, x, y, θ は車両の状態, x_0, y_0, θ_0 は目標状態, d_1, d_2, ..., d_{20} は障害物への距離であり, 車両の周りを 20 分割して, それぞれの方向ごとに障害物を検知して得た距離である. ただし検知範囲外だった場合, 距離は無限大であるとする. NN の出力は, 二次元ベクトルの制御入力 u とする. これらの入出力データを用いて NN の教師あり学習を行う.

6.3 データ生成

データ生成時の各パラメータは表 1 と R 以外同じものを使用した. ここで R のみ $\mathrm{diag}(0.5, 0.5)$ へと変更したが, これは入力の重みを増やすことにより目的地付近で減速するようになり, 目的地で止まろうとする教師入力を多く収集することを意図したものである. 学習用のデータについて

述べる. 目標状態を座標平面内の (0.4, 0.4) 上で x 軸の正方向を向くものとし, 開始状態 ($-0.1 \leq x \leq 1.1$, $-0.1 \leq y \leq 1.1$, $-\pi \leq \theta \leq \pi$) で障害物に埋まっていないものからランダムなものとした. また障害物配置は図 4 のものと同様の配置とする. 以上の環境でモデル予測制御を繰り返しデータを収集する.

シミュレーションを 1000 回行い z, u を 29448 ステップ分収集した. ただし NN の出力 u は 10 で除算し正規化を行っている.

6.4 NN の学習

NN には様々なモデルが提案されているが今回は図 7 のような三層フィードフォワード NN を用い, MATLAB の nftool を利用し近似を行った. 中間層は 100 個のニューロンからなり活性化関数はシグモイド関数を用いる. ここで中間層の i 個目のニューロンへの入力 a_i は, 入力層の j 番目のニューロンに対する重みを w_{ij} として,

$$a_i = \sum_{i=1}^{26} w_{ij} z_j + b_i$$

として表せる. すなわち出力は, $f(a_i) = 2/(1 + e^{-2a_i}) - 1$ である. 出力層も同様に i 個目のニューロンへの入力は中間層の入力層の j 番目のニューロンに対する重みを w_{ij} として

$$a_i = \sum_{i=1}^{100} w_{ij} z_j + b_i$$

であり, 出力は活性化関数として恒等関数を用いて, $f(a_i) = a_i$ である. 学習は反復 810 回のバックプロパゲーションにより行い, データ内の 15% をテストデータとした. 学習の結果, 平均二乗誤差は 4.5 であった.

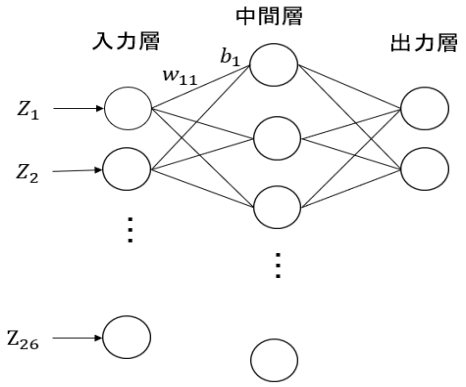

図 9 学習に使用した NN の構成

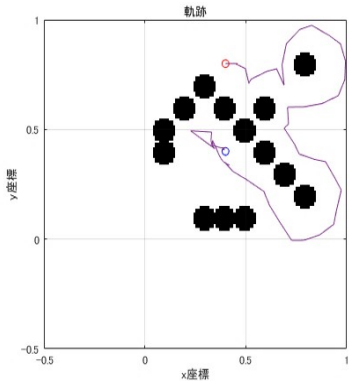

図 10 NN で近似したシミュレーション結果

図 10 より目標地点付近に到達できていることが確認できる. この時, 1 ステップ当たりの計算時間は 9.0×10^{-2} であり, 1 ステップの時間間隔が 0.2 秒であるから十分な実用性があるといえる.

7 おわりに

本研究ではモデル予測制御を用いて障害物の回避をシミュレーションで行った. また, 車両停止時の脱出方法に右手法を用いて解を導き, NN で近似を行い計算負荷の減少を実現した.

今後の課題に実機実装と近似精度の向上が考えられる. 近似性能向上に障害物の距離を畳み込み層に通し隣接する 4 つの検知距離ごとで畳み込みを行うことを考えている.

参考文献

[1] R. W. Brockett: Differential Geometric Control Theory, pp. 181-191, Boston : Birkhauser (1983).

[2] M. Yamada, S. Ohta, Y. Syumiya and T. Funahashi: Transactions of the Society of Instrument and Control Engineers, Vol. 38, No. 4, pp. 369-378 (2002).

[3] M. Sampei: Proc. of the IEEE Conference on Decision and Control, pp. 1120-1121 (1994).

[4] 塚原正人・山田学・舟橋康行: 「障害物を回避する非ホロノミック車両システムの適応制御」. 第 53 回自動制御連合講演会, pp. 1256-1261 (2010).

[5] 足立修一: 「モデル予測制御の基礎」. 日本ロボット学会誌, Vol. 32, No. 6, pp. 499-502 (2014).

[6] 小山健太郎・野中謙一郎: 「障害物回避と切り返し点の自動調節によるモデル予測車庫入れ制御」. 計測自動制御学会論文集, Vol. 50, No. 1 (2014).

[7] V.J. Lumelsky and A.A. Stepanov: Path-Planning Strategies for a Point Mobile Automaton Moving Admist Unknown Obstacles of Arbitrary Shape, Algorithmica, 2, 403/430 (1987)