# 動的計画法による不確実性を考慮した修復動作を含んだ行動計画

○齊藤貴大　小林祐一　成瀬達也　（静岡大学）

## Planning of pushing manipulation using Dynamic Programming considering uncertainty and cost of recovery motion

∗T. Saito and Y. Kobayashi and T. Naruse (Shizuoka University)

**Abstract**− This paper presents a planning method of pushing manipulation by a mobile robot. It is sometimes very useful if the robot can take recovery action, namely, re-approaching and re-pushing, when it turns out to be ineffective to keep current pushing motion. The proposed planning framework is based on the idea of mode switching, where three modes; approaching, pushing and re-pushing, are considered. The pushing motion is first built with dynamic programming, which provides value function of the state. Based on the value, planning of re-approaching to the object and re-pushing is conducted using a value iteration algorithm extended to state space with uncertainty. The proposed planning framework was evaluated in simulation, and it was shown that it provides more effective behaviour of the robot by recovery motion at an appropriate timing.

**Key Words:** Motion generation, Hybrid system, Mobile robot.

## 1　はじめに

近年様々な分野でロボットが活躍しているが，今後のロボットの活躍の場はますます広がると考えられ，完全に自律的に行動できることが求められる．ロボットの自律行動には，ロボット自身が動くだけでなく対象とする物体を操作するため動作も含まれてくる．自律ロボットの物体操作を考慮すると，ロボットの利便性を高めるが，モデルと実システムの実際の挙動との間に大きなギャップが生まれるため動作計画と制御の問題はより複雑になる．

モデルの不完全性に対する一つの手法として強化学習が挙げられる．強化学習では，ロボットの試行錯誤により未知の環境に適応した行動を取得することができ，ロボットの自律行動に関する研究の多くで用いられている [1]．しかし，強化学習は一般的に多くの試行錯誤を必要とするため，計算時間と試行回数が膨大となる．

この問題に対し，動作を複数のモードに分けて学習を行い，計算時間を短縮する研究がある [2]．これらの研究では物体へ接近するモードと物体を押すモードの組み合わせにより，物体操作を実現している．一般に物体を操作する場合，接触・非接触，あるいは滑り・非滑りといった状態の境界が存在する．各状態においてはダイナミクスは連続だが，モードが切り替わる場合には離散的に異なるダイナミクスとなり，このように連続したダイナミクスが離散的に切り替わる問題をハイブリッドシステムという [3]．このようにモードの切替を考慮することで，最終的にタスクを達成するという情報のみで行動を生成する場合よりも計算量が少なく，かつ効率的な動作が得られることが期待できる．また，物体の押し動作において制御のずれ等により物体を確実に目標位置へ運ぶことは難しいため，押し直しを図る必要がある．移動ロボットのモード切り替えによる物体操作の研究 [4] もなされているが，ここでは押し直しにおける不確実性は十分に考慮されていない．

ロボットが移動や物体操作をするときには，状態遷移の不確実性によって行動のたびに目標とする状態との誤差が生じると考えられる．誤差は蓄積されて大きくなり，目標とするタスクを達成できない状態に陥る可能性もある．このような不確実性を扱った研究として，文ら [6] の研究がある．文らは，ロボットの移動と観測の不確かさのモデルを基に，安全かつ効率的な走行や観測位置を決める手法を提案している．ここでは，移動の不確かさを恣意的に与えているが，本手法ではモンテカルロ法 [5] の考えを基に，確率的に移動位置を求めるため，不確実さをより適切に表現ことができる．複数台のロボットを用いて物体操作を行い，タスクを達成しやすくしている研究も行われているが，必要なときに協調できる相手が必ずしも存在するとは限らない [7][8][9][10]．そこで，不確実さにロボット単体で対処する方法を考える．

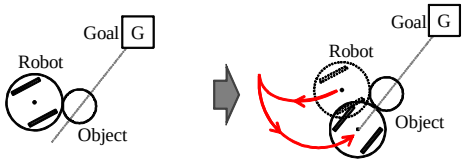


Fig. 1: An example of re-pushing behavior.

ロボットが操作中にタスクが達成不可能であると判断した際には，図 1 のように行動の修正を図る．このような修復動作には物体操作で有効である．例えば，把持により物体操作するようなロボットでは，把持した物体が不安定で落下が予測できる場合には，物体を一度離し持ち直すという動作が考えられる．このことにより，ロボット単体でもタスクの確実性を高めることを考える．本研究では，移動ロボットによる物体位置の不確実性を考慮した押し直し動作を含む動作計画を提案する．

## 2　問題設定

### 2.1　二輪ロボットの運動学

本研究では二輪ロボットによる物押し動作を行う．ロボットによる物体の把持は行わず，押し動作のみによって対象を到達時間の期待値を最小にすることを目的と

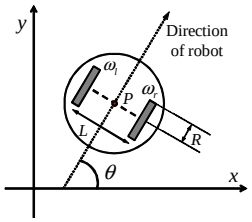


Fig. 2: Model of two-wheeled robot.

する．

対象物は円形で，初期位置はロボットから離れている．ロボットおよび対象の状態は常に観測することが可能であるとし，障害物や他のロボットは存在しない．また，車輪と接地面の間に横滑りは生じないとする．ロボットが対象物を目標領域へ運ぶのに失敗した場合，より長い時間を要するタスクを再試行しなければならない．例えば，ロボットが目標領域を通り過ぎてしまった場合，ロボットは大きく姿勢を変え，さらに押し動作を行うための時間を要する．従って，ロボットは到達できないリスクを考慮することで全体の所要時間の期待値を最小にする必要がある．しかし，ロボットと対象物は円形であるため，接触を維持したまま物体の位置誤差を修正をすることが難しい．そこで，状況に応じて一度対象物から離れ，押し直しを行うことでリスクを減らすことができる．このような押し直し動作を考慮して初期状態から目標とする状態（対象物体を目的位置に運搬した状態）に達するまでの時間を最小とするように，行動計画を行なう．

本研究では，円形の二輪ロボットのモデルを用いてシミュレーションを行う．二輪ロボットのモデルを図 2 に示す．各変数は以下のとおりである．

- $x, y$：点 $P$ の座標 [mm]
- $\theta$：ロボットの進行方向と $x$ 軸がなす角度 [rad]
- $\omega_l$,$\omega_r$：車輪の回転角速度 [rad/sec]
- $R$：車輪の半径 [mm]
- $L$：車軸の長さ [mm]

### 2.2　衝突時の対象物の変位モデル

シミュレーションにおいて，円形のロボットと対象物が衝突した際の対象物の挙動を設定する方法について述べる．図 3 に示すように，ロボットが対象物に距離 $d$ だけ干渉したとする．このとき，対象物はロボットと物体の中心を結ぶ線分の延長線上，かつロボットに接している位置に移動する．また，ロボットの姿勢は対象物の移動前の物体の位置を $(x_{ob}, y_{ob})$，ロボットと対象物の中心を結ぶ線分が x 軸と成す角度を $\alpha$ とすると，衝突後の物体の位置 $(x_{oa}, y_{oa})$ を次式で表す．

$$x_{oa} = x_{ob} + d\cos\alpha \quad (1)$$

$$y_{oa} = y_{ob} + d\sin\alpha \quad (2)$$

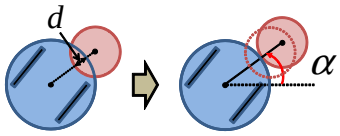


Fig. 3: Displacement model.

## 3　動的計画法による動作計画

### 3.1　動的計画法

本研究では，ロボットが行なう行動を動的計画法によって決定する．動的計画法とは，多段階決定問題を Bellman の最適方程式に従って解く方法である．動的計画法を用いる利点として，得られる結果が最適性を持つということがある．本研究でいう動的計画法は，強化学習の分野におけるものであり，主に価値反復のことを指す．

動的計画法は完全なモデル（報酬や遷移確率）が与えられている場合に適用することが可能である．Bellman 方程式を更新規則とした反復処理を行なうことで最適価値関数を求め，最適方策を獲得することができる．

ここで $s \in \mathcal{S}$， $a \in \mathcal{A}$ はそれぞれ離散空間上での状態と行動を表す．$P^a_{ss'}$ は現在の状態 $s$ から遷移先の状態 $s'$ へ行動 $a$ をとることによって遷移する確率（状態遷移確率）であり，$R^a_{ss'}$ は現在の状態 $s$ から遷移先の状態 $s'$ へ行動 $a$ をとることによって遷移したときに得られる評価の期待値である．行動計画の目的は，累積報酬 $\sum_{k=1}^{\infty} \gamma^{k-1} R^a_{ss'}$ を最大化するような制御方策 $a = \pi(s)$ を取得することである．$\gamma$ は割引率と呼ばれ，どれだけ先の報酬まで予測して評価に組み入れるかを決める減衰定数である．繰り返し計算のステップ数を $k$ とすると，方策評価は次式を更新規則とすることで行なう．

$$V_{k+1}(s) = \max_a \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V_k(s)] \quad (3)$$

ここで，$k \to \infty$ で系列 $V_k$ は $V^\pi$ に収束する．この操作を全ての状態 $s$ に対して行なうことで，状態価値関数 $V^\pi(s)$ が得られる．$\max_a$ は状態価値関数を最大（もしくは最小）とするような行動 $a$ を探すという意味を持つ．以下に価値反復のアルゴリズムを示す．

Algorithm 1: Algorithm of Dynamic Programming

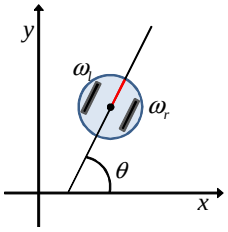


Fig. 4: Non-contact

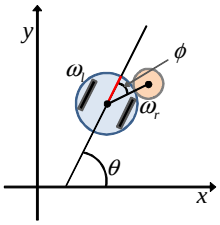


Fig. 5: Contact

The initialized with the value of any $V$.
$V(s) \leftarrow 0$
Repeat :
For each $s \in S$
$v \leftarrow V(s)$

$$V(s) \leftarrow \max_a \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V(s)] \quad (4)$$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$
$\Delta < \epsilon$
output the policy $\pi$
$\pi(s) = \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V(s)]$

### 3.2 非接触モード

非接触モードとはロボットが対象に接触していない状態にある，つまりロボットが対象に近づくモードである．

$$x = (x_r, y_r, \theta)^T, u = (\omega_l, \omega_r)^T \quad (5)$$

ここで，$x_r, y_r$ はロボットの位置，$\theta$ はロボットの姿勢を表す．また，$\omega_l$,$\omega_r$ は車輪の回転速度である．非接触状態における価値反復では，対象物と目的地位置がなす直線上で，目的地位置とは反対の位置で対象物に接する状態を目標状態とする．これは押し直し時における行動でも同様である．図 4 に非接触モード時の状態を示す．

### 3.3 接触モード

接触モードとはロボットが対象に接触している状態にある，つまりロボットが対象を押すモードである．

$$x = (x_r, y_r, \theta, \phi)^T, u = (\omega_l, \omega_r)^T \quad (6)$$

また，$\phi$ はロボットに対する対象物の方向を表している．接触状態における価値反復では，目的地位置を $(x_g, y_g)$ としたとき，$(x_g, y_g)$ を中心とする円形領域内に対象物の中心位置が入る状態を目標状態とする．図 5 に非接触モード時の状態を示す．

## 4 不確実性を考慮した動作計画

押し動作において，対象をまっすぐ押すことができれば対象はそのまま直進して運ぶことができる．しかし実際に押し動作を行う際には，床と車輪の間の摩擦や，車輪の制御等において計算機上には存在しないずれが生じると考えられる．また，本研究では動作計画を行なうために動的計画法を用いているので，状態は離散化されている．そのため，到達位置として与えた

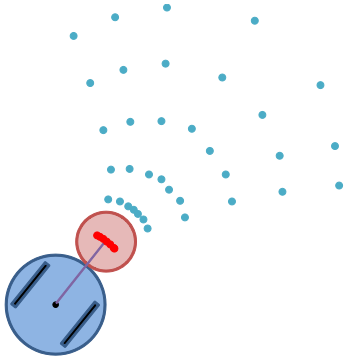

Fig. 6: Example of variation.

目標値が存在する離散化された状態には達するが，必ずしも目標として設定した値にはならない．対象の位置やロボットの位置にばらつきがあっても，それが離散化された同じ状態内であれば次に従う方策は同じものである．つまり同じ状態から押し動作を行う場合でも，離散化された状態の中のばらつきにより対象が到達する位置は異なってくる．図 6 は赤い点群は同じ状態として与えられているが，方策に従い押し動作を行うと青い点群のように対象の位置のばらつきが大きくなっていくことを示している．

そこで，現在の状態から対象を目的の位置まで押すことができないと判断したとき，あるいは，押し直しを行なうことで確実にタスクを達成できると判断したときに，接触状態から非接触状態に移り，別の位置から改めて接触状態に移る動作を行なうことを考える．

押し直し動作は，非接触モードと同じ行動とみなすことができ，同様に動的計画法によって方策を導出し行動経路を選択する．押し直し後押し動作を再開する位置は，目標位置と対象を結ぶ直線上であり，対象に接し目標位置方向を向いた状態とする．

### 4.1 押し直しの判断

押し直しを行なうにあたり，押し直しが必要か，押し直しを行なうならばどのタイミングが良いかといった判断を行う必要がある．ここでは，押し直し位置の算出方法について説明する．

本手法では，モンテカルロ法 [5] の考えを基に対象物の不確実性を粒子を用いて表現する．まず現在の状態内においてランダムに粒子を配置し，それぞれの粒子に対して，接触モードで得た方策に従い粒子を遷移させる．これにより，対象の可到達領域を予測することができ，予測した情報を用いて押し直しを図る位置を決定する．

ロボット，対象物，目的地の関係は，

- 対象物と目的地との距離　$r$
- 対象物とロボットとの角度　$\phi$
- 目的地とロボットとの角度　$\psi$

によって表現することができる（図 7）．

これらの 3 つの変数は，計算過程で適応するためにそれぞれを離散化する必要がある．物体とゴールとの

距離を $r_i$（$i = 1, \cdots, I$），ロボット正面から見た時の対象物の位置を $\phi_j$,（$j = 1, \cdots, J$ ），ロボット正面からゴールまでの角度を $\psi_k$（$k = 1, \cdots, K$）で表す．したがって，不確実さを含めたロボットと対象物と目的地の状態は，$(r_i, \phi_j, \psi_k)$ で表すことができる．

押し直し動作のとる基準は，押し続けた場合と押し直した場合の所要時間の期待値を比較することで決定する．ある状態 $(r_i, \phi_j, \psi_k)$ において $N$ 個の粒子を散布し，$c_{i'j'k'}$ 個の粒子は押し動作により $\Delta t$ 秒後に状態 $(r_{i'}, \phi_{j'}, \psi_{k'})$ へと遷移する．一方，押し直しを行う場合，$N$ 個の粒子のうち，$c_{j''k''}$ 個の粒子は押し直し動作により $\Delta l$ 秒の時間を要し，状態 $(r_i, \phi_{j''}, \psi_{k''})$ へと遷移する．押し直しの場合，対象物と目的地との位置関係は変化しないため，遷移後の距離も変わらない．このときの押し動作の場合と押し直し動作の場合の所要時間の期待値を比較し，期待値が小さい方の行動を記憶する．

押し直し判断の考え方を説明するために，図 8 を用い，二次元の場合について説明する．現在の粒子の状態を $(r_i, \phi_j)$ とすると離散化された状態空間では，図 8 のグレーの粒子として表現されている．この粒子を押し続けた場合，$\Delta t$[s] 後の粒子の遷移先は青色の粒子である．図 8 に示すように一般的にロボットから見た時の角度 $\phi$ は大きくなるが目的地との距離 $r$ は短くなる．一方，押し直しを行った場合は，$\Delta l$[s] 後に図 8 の黄色の粒子のように遷移する．押し直した場合，対象物の絶対位置は変化しないため目的地との距離 $r$ は変化しないが，ロボットから見た時の角度 $\phi$ は小さくなるように遷移する．これらの押し続けた場合と押し直した場合の粒子の到達時間の期待値の総和 $V_1$，$V_2$ を比較し，より小さいものを選択していく．

以上のような考え方から，各状態について次式で与えられる価値関数 $V(r_i, \phi_j, \psi_k)$ を動的計画法により計算し，各状態ごとの最適な行動を求める．得られた方策から，押し直しを考慮し最短時間でタスクを達成させることが可能である．

$$V(r_i, \phi_j, \psi_k) = \max_{i=1,2} V_i \tag{7}$$

$$V_1 = \sum_{i'=1}^{I} \sum_{j'=1}^{J} \sum_{k'=1}^{K} \frac{c_{i'j'k'}}{N} (\Delta t + V(r_{i'}, \phi_{j'}, \psi_{k'}))$$

$$V_2 = \sum_{j''=1}^{J'} \sum_{k''=1}^{K'} \frac{c_{j''k''}}{N} (\Delta l + V(r_i, \phi_{j''}, \psi_{k''}))$$

## 5 実験

### 5.1 実験条件

本手法を用いて，押し直しを考慮した物体の押し動作をシミュレーションにより行なった．ロボットと対象物はそれぞれ半径を $R_r, R_o$ とする円形形状であり，ロボットと対象物，およびゴールの位置は常に観測可能であるとする．状態は，$s = [x, y, \theta, \phi]^T$ であり，非接触モード時には $\phi$ は考慮しない．行動は制御入力 $u = [\omega_l, \omega_r]^T$ で与えられる．以下にシミュレーションを行なうための各設定を示す．

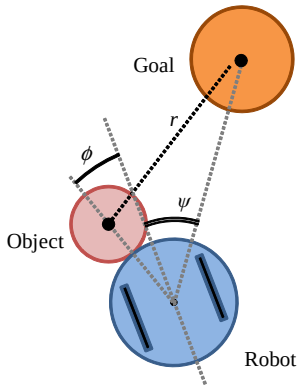


Fig. 7: Positional relationship.

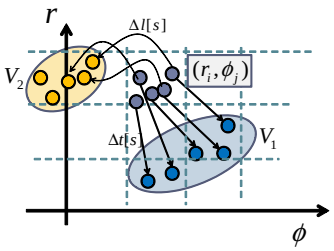


Fig. 8: The concept of re-pushing.

- ロボットの半径 $R_r$：25 mm
- 対象物の半径 $R_o$：15 mm
- 車輪の半径 $R$：20 mm
- 車軸の長さ $L$：50 mm

Table 1: 状態空間と離散化の定義

| | 状態空間範囲 | 分解能 |
|---|---|---|
| $x$ [mm] | [0 250] | 26 |
| $y$ [mm] | [0 250] | 26 |
| $\theta$ [deg] | [0 360] | 36 |
| $\phi$ [deg] | [-30 30] | 13 |

### 5.2 価値反復による状態価値の計算および方策の確認

まず始めに，価値反復によって非接触モードと接触モードそれぞれの状態価値を計算する．この計算結果を用いてロボットの行動計画を行なう．

#### 5.2.1 非接触モード

対象物に接近するための行動を獲得する．初期状態から対象物に接する位置にゴール（対象を運ぶ目的地）の反対側となる状態を価値反復を行なう際に最も価値が高くなる状態とする．対象物と目的地は以下のように設定した．これにより目標位置は以下に示す $(x_p, y_p, \theta_p)$ となる．

- 対象物の位置 $(x_o, y_o)$：(20, 14)
- 目的地 $(x_g, y_g)$：(20, 20)
- 非接触モード時の目標位置 $(x_p, y_p, \theta_p)$:(20, 10, 90)

図 9 に動的計画法による価値計算結果を示す．これは $\theta = 90$[deg] のときの価値計算結果である．ここで

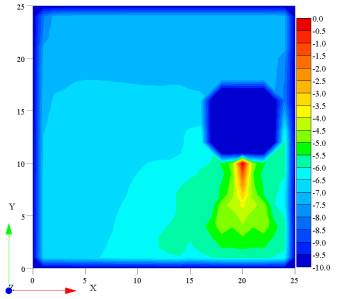

Fig. 9: Value calculation results.

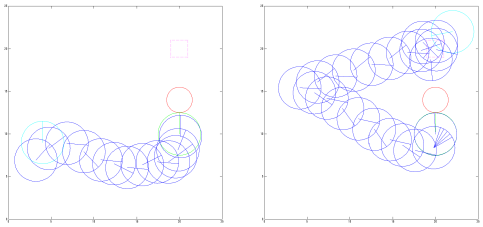

Fig. 10: Examples of approaching operations.

は，ロボットが状態の定義域外に出ることのないように外側の価値を低く設定している．また，非接触モードではロボットと対象物が接触することはないという前提条件があるため，対象物を設置した位置周囲でも価値を低く設定している．

また，図 10 に得られた方策に従って複数の初期状態から対象物に接するまでのロボットの動作の軌跡を示す．ロボットの軌跡を青色で，対象物の位置を赤色で表している．緑色の円はロボットの目標位置である．複数の状態から目標とする状態まで遷移することができており，非接触モードの行動方策が正しく得られていることがわかる．

### 5.2.2 接触モード

次に，対象物を押すための行動計画を獲得する．対象物がゴール（対象を運ぶ目標位置）に到達した状態を最も価値が高くなるように価値反復を行なう．ただし，ゴールの位置は点ではなく目的地 $(x_g, y_g) = (20, 20)$ を中心とした 20 mm の円形の範囲とする．図 11 に対象物を押した軌跡を示す．青色がロボットの軌跡，赤色が対象物の軌跡である．

図 11 の左の例では，対象を目的地まで運ぶことができていることがわかる．しかし，図 11 の右の例では，対象物を目的地まで運ぶことができていない．これは，動的計画法による計算のために状態を離散化していることによると考えられる．実験ではロボットから見た対象の位置を $\phi$ で表し，-30~30 deg の範囲を 13 分割している．つまり，$\phi$ が同じ値でも最大で 4 deg のばらつきが生じることになる．このばらつきが，押し動作を失敗に導く要因となる．より状態を細かく設定し，状態数を増やすことで改善されると考えられるが，本研究では押し直しを図るため，単一のモードで最後まで運べる必要はない．

## 5.3 押し直しを考慮した押し動作の生成

実際に押し直し動作を行う前に，粒子を用いて可到達領域を推定し，押し直しを図る基準を決定する．可到達領域の推定の例として，接触モードのある初期位置における結果を示す．粒子の初期位置として状態 $(x, y, \theta, \phi)$

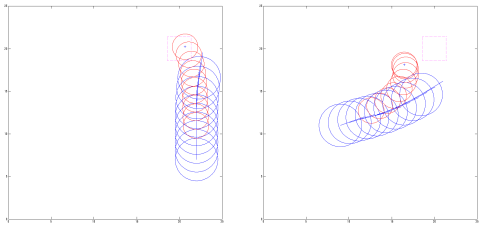

Fig. 11: Examples of push operations.

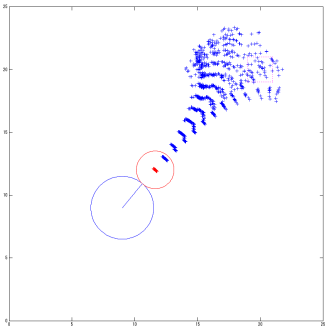

Fig. 12: Variation in the position of the object.

$= (9, 9, 50, -1)$ における対象物の位置にランダムに散布する．その後，各粒子ごとに方策に従い押し動作を行い，数ステップ後の対象の位置を予測する．ここでは粒子 200 個を散布し，対象物が到達するであろうと考えられる位置を示した結果が図 12 である．赤い点群は対象の初期位置，青い点群は押し動作後の対象の位置を表している．

図 12 から，同じ状態内から動作を始めても，僅かなばらつきにより到達領域が大きく異なっていることがわかる．この粒子の遷移結果を用いて，押し直しを行なう位置を決定する．

押し直し位置を決定するために，前節で説明した式 (7) を用いて動的計画法により各状態 $(r_i, \phi_j, \psi_k)$ ごとに価値を計算し，方策を決定した．状態空間と離散化を以下のように定義した．

Table 2: 状態空間と離散化の定義

| | 状態空間範囲 | 分解能 |
|---|---|---|
| $r$ [mm] | [0 200] | 20 |
| $\phi$ [deg] | [-90 90] | 67 |
| $\psi$ [deg] | [-50 50] | 11 |

ここで $\phi$ は，不等間隔で離散化している．問題の性質上，ロボット正面からの対象物の角度 $\phi$ が小さいほど，$\phi$ 方向の遷移が小さくなる傾向にある．$\phi$ の離散化の粒度を等間隔に細かくする方法もあるが，その場合状態数が増え，計算時間の増加につながる．従って，$\phi$ の大きさに応じて，離散化の粒度を変化させる手法をとる．今回は，$\phi > 50$ で 9[deg]，$\phi > 20$ で 3[deg]，$\phi < 20$ では 1[deg] の粒度で離散化を行った．また，状態空間の定義域外へ遷移してしまう場合，強制的に押し直しを行うことで対処した．動作生成の際には，各モードで得られた最適方策を用いる．

動的計画法による価値計算結果を図 13 に示す．図 13 の右図は $\psi = 0$[deg]，左図は $\psi = -30$[deg] の時の各

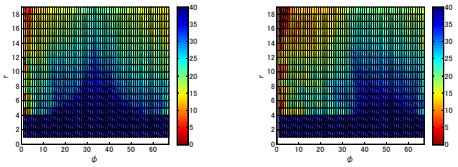

Fig. 13: Result of value calculation.

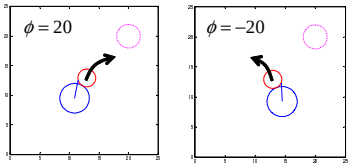


Fig. 14: Examples of object's state.

状態における時間コストを示している．各グラフの縦軸は対象物と目的地の距離 $r$，横軸は対象物とロボットとの角度 $\phi$ である．値はそれぞれ離散値になっている．従って，横軸の中央値が $\phi = 0$[deg] に相当する．図 13 のグラフでは，対象物と目的位置との距離 $r$ が大きく，物体のばらつき $\phi$ が大きいほどコストが高くなっていることが分かる．また，左図については $\phi$ の中央値より右側のコストが低くなっていることが分かる．これは，$\phi > 0$[deg] の部分に相当する．$\psi = -30$[deg] かつ $\phi > 0$[deg] の状態は，目的地と対象物がロボット正面から見たときに右側にある状態であり，$\phi < 0$[deg] よりも対象物を目的地へ運ぶことに適した状態である．図 14 にその例を示す．左図は，$\psi = -30$[deg] かつ $\phi = 20$[deg] の状態，右図は，$\psi = -30$[deg] かつ $\phi = -20$[deg] の状態である．図 14 から，左図の方がよりタスク達成にかかる時間コストが低くなることが分かる．このことから，直感的にも妥当な結果が得られたと言える．

このとき得られた最適方策は，押し直しの判断となる．得られた方策を用いて，押し直しを考慮した押し動作を行った．図 15 に対象物を押した軌跡を示す．黒色がロボットの軌跡，赤色が対象物の軌跡である．図 15 の右図は，比較のために押し直しの閾値を設定した場合の押し動作の結果である．このときの閾値は，対象物がロボット正面から 30[deg] 以上ずれた場合に押し直しを行うものとした．本手法では押し直しを 3 回，合計 20 ステップで目的地に対象物を運ぶことに成功した．一方，閾値を設定した場合では，目的地に対象物を運ぶのに，押し直しを 2 回，合計 26 ステップを要した．これは，対象物がロボット正面から大きくずれた位置から押し直しを行っているため，ステップ数が増加している．本手法では，常に対象物がロボット正面に位置するように適度に押し直しを行っているため，ステップ数の増加を防ぐことができたと考えられる．

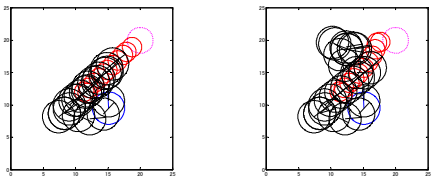

Fig. 15: Result of comparative experiment.

## 6 結言

本報告では，モード切り替えと効率的な押し直しを考慮した二輪ロボットのための物体押し動作の生成手法を提案した．二つのモードに分割し，それぞれについて動的計画法により動作計画を行った．また，不確実性を考慮した動作計画により効率的な押し直し判断の基準を決定した．シミュレーションによって押し直しの基準を固定したものとの比較実験を行い，本手法の有効性を確認した．今後の展望として，押し直しを考慮した押し動作の定量的な評価や，対象物の形状や状態変位モデルを変更した場合の本手法の有効性を確認する予定である．また，押し直しだけでなく複数の移動ロボットを用いた協調動作や分担動作などの生成を目指す．

## 参考文献


1) T. Kondo and K. Ito : "A Study on Designing Controller for Peg-pushing Robot by using Reinforcement Learning with Adaptive State Recruitment Strategy" , SICE Annual Conference, 2003.
2) Yuichi Kobayashi and Shigeyuki Hosoe: "Planning-space shift motion generation:variable-space motion planning toward flexible extension of body scheme", Journal of Intelligent and Robotic Systems, Vol.62, No.3-4, pp.467- 500, 2010.
3) Arjan J. van der Schaft and Hans Schumacher: An Introduction to Hybrid Dynamical Systems, Springer, 2000.
4) 関口拓生，小林祐一:"逐次最小二乗法を用いた連続行動空間での最適行動の学習"，第 12 回システムインテグレーション部門講演会講演論文集 (SI2011)，pp.1618-1620, 2011.
5) S. Thrun, W. Burgard, D. F. : Probabilistic Robotics, The MIT Press, 2005.
6) 文仁赫，三浦純，白井良明："不確かさを考慮した観測位置と移動のオンライン計画手法"，日本ロボット学会誌，Vol.17，No.8，pp.1107-1113，1999.
7) Takahiro Otani and Makoto Koshino: "Applying a path planner based on rrt to cooperative multirobot box-pushing", Artificial Life and Robotics, Vol.13, No.2, pp.418-422, 2009.
8) 山本元司，黒田創明，毛利彰："複数台ロボットによる協調作業経路計画"，日本ロボット学会誌，Vol.11，pp.217-226，1993.
9) Seiji Yamada and Junya Saito: "Adaptive action selection without explicit communication for multi-robot box-pushing", Proceedings of the IEEE Transactions on Systems, Man, and Cybernetics, Part C, Vol.31, No.3, pp.398-404, 2001.
10) Arnab Ghosh，Avishek Ghosh，Arkabandhu Chowdhury，Amit Konar and Ramadoss Janarthanan : "Multi-robot cooperative box-pushing problem using multi-objective particle swarm optimization technique"，Proceedings of the Second World Congress on Information and Communication Technologies（WICT 2012），pp.272-277，2012.