第6章

複数のロボット指によるグラスプレス・マニピュレーションの計画

6.1 はじめに

グラスプレス・マニピュレーションの適用範囲を拡大する上での課題の１つに，対象物の操作計画問題が挙げられる．初期位置から目標位置までのあやつり (Fig. 6.1) を自動的に計画することができるようになれば，作業教示の手間の大幅な低減が期待できるが，グラスプレス・マニピュレーションでは計画を行う上でいくつかの困難な点が存在する．

通常のピックアンドプレイスの場合，いったん把持してしまえば，対象物の運動はロボットの運動と自明に１：１対応する．したがって，計画は基本的に幾何学的な障害物回避問題に帰着される．しかしながら，グラスプレス・マニピュレーションの場合，対象物を把持していないため，その運動は，ロボットの運動だけでなく，環境との接触点における摩擦など力学的条件にも影響される．したがって，計画を行うためには幾何学的条件だけではなく力学的条件をも考慮する必要がある．また，グラスプレス・マニピュレーションでは，対象物を把持していないために操作が不可逆になる場合（押せても引けない，など）があり，これが問題をさらに複雑にしている．

本章では，前章で示した指の制御モードの決定手法を利用した，複数のロボット指によるグラスプレス・マニピュレーションの計画手法を提案する．これにより，多様なグラスプレス・マニピュレーションを，外乱に対してもロバストに行えるような指の動作計画の実現を目指す．

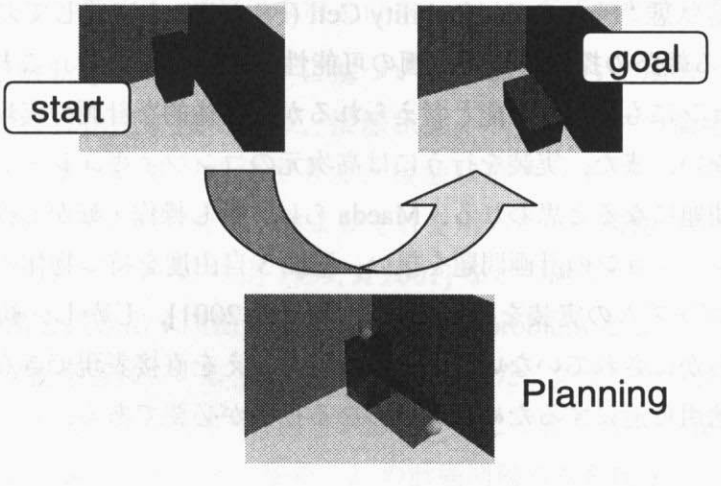


Fig. 6.1: Planning of Graspless Manipulation

6.2 従来研究

現状では，一般的なグラスプレス・マニピュレーションの計画問題は，単に計算量が多いというのみならず，計画アルゴリズムを実装する上で問題をどのように表現すればよいか，という点もまだ明らかではない．そのため，実際にグラスプレス・マニピュレーションの範疇に入る計画問題を扱った研究のほとんどは，ロボットを考慮せずに対象物の運動のみの計画を扱ったもの，もしくは特定の操作方法に限定して議論をしているものである．前者の例としては，[吉川 1992a, 余 1995a, 相山 1999, 栗栖 1999, Marigo 2000, Ji 2001] が挙げられる ([栗栖 1999, Marigo 2000] は，さらに操作方法も転がしに限定している)．後者の例としては，押し操作に限定したもの [栗栖 1995, Lynch 1996b, 吉川 1996c]，転がし操作に限定したもの [沢崎 1991, 山下 2001, 栗栖 2002] がある．

寺崎らは，指先に回転機構を有する平行2指ハンドを対象に，把持による操作と押し操作 (寺崎らは「滑らし操作」と呼んでいる)，および回転機構を利用した転がし操作を含む物体操作の計画を行っている [寺崎 1994, 寺崎 1995]．実装された計画手法は完成度の高いものであるが，幾何情報に基づく動作計画に重点が置かれており，一般的なグラスプレス・マニピュレーションの計画を実現するために必要な力学解析は扱われていない．

Erdmann は2つの線分状の掌によるあやつり ("Two-Palm Manipulation") を対象に，平面3自由度を持つ物体の滑らせ・転がしなどの操作を含む計画を行っている [Erdmann 1998] が，これも限定的な操作の計画だと言えるであろう．

摩擦などの力学的問題を考慮した上で，操作方法を限定しない計画問題を扱った例としては，以下の研究がある．Trinkle らは Whole Arm Manipulation を対象に接触点での滑り・転がりを考慮した安定状態 "First-Order Stability Cell (FS-cell)" を定義しており [Trinkle 1995]，FS-cell だけを通る経路の探索による計画の可能性を示唆している．これはグラスプレス・マニピュレーションにも応用が可能と考えられるが，具体的な計画の実現手法については明らかにされていない．また，実装を行うには高次元のコンフィギュレーション空間の構築が必要で計算量が問題になると思われる．Maeda らは，押し操作・転がし操作を含むグラスプレス・マニピュレーションの計画問題を扱い，平面3自由度を持つ物体の1本指での操作を対象に計画アルゴリズムの実装を行っている [Maeda 2001]．しかし，複数の指による操作の計画方法が明らかにされていないこと，指の持ち替えを直接表現できないことから，より一般的な作業に適用可能にするためにはさらなる拡張が必要である．

6.3 計画の概要

6.3.1 計画の基本方針

本章では，操作方法を限定しない一般的なグラスプレス・マニピュレーションの自動計画を行うことを目指す．しかし，素直に計画問題を実装すると，計算量が爆発することが予想される．そこで，ここでは操作を現実的な時間で計画することのできる計画器を実現するために，以下のような2段階で計画を行うアプローチをとることを考える．

1. 幾何学的情報に基づく大まかな操作コストの見積もりによって，操作の候補となる探索領域を絞る．
2. 限定された探索領域の中で，力学解析に基づく詳細な操作計画を行う．

1. は，全体の計算量を減らすために，例えば環境の幾何学的情報から，接触状態ネットワーク [平井 1988] を利用して，探索領域を限定する．この場合，まずはロボットの存在は考えず，対象物と環境の間の接触状態遷移のみを考える．接触状態ネットワークに対し，適当なコスト評価を与えて探索を行うことによって，初期状態から目標状態までの接触状態遷移の候補が得られる [余 1995a, 相山 1999].

しかしこの接触状態遷移は，ロボットを考慮した詳細な力学解析を行わない限り，最適であるかどうかはもとより実現可能であるかどうかも不明である．そこで，接触状態ネットワークから（最適解だけではなく）複数の有望な接触状態遷移候補を抽出することで，簡略化された接触状態ネットワークを得る．そして，この簡略化されたネットワークに対して，(2) の詳細な計画を行うことで，最終的な解を得る，という方法が考えられる．

以上のように接触状態ネットワークに基づいて計画を行った場合，個々の接触状態に関しては対象物の自由度が減るため，接触状態ネットワークの簡略化を適切に行うことで，全体の計算量の低減が期待できる．接触状態ネットワークの生成については [平井 1988, Xiao 2001] などの研究が，およびそれを用いた（対象物の運動のみの）計画に関しては，[吉川 1992a, 余 1995a, 相山 1999, Ji 2001] などの研究がある．また，グラフ上から複数の有望な経路を抜き出す問題は k shortest paths problem として定式化され効率的な計算アルゴリズムが提案されている [Eppstein 1998]．また，グラフから多様な経路を抜き出す方法も提案されている [藤田 2002]．したがって，ここではそれらの成果を利用することとし，本章では探索領域が限定された後の，2. の計画問題のみを扱う．

なお，一般的なロボットの動作計画問題では，幾何計算（障害物回避）が主たる問題となる [比留川 1994] が，これについてはすでに極めて多数の研究が行われている [Latombe 1991, 太田 2001]．そこで本章では，グラスプレス・マニピュレーションに特有である力学的問題の解決に焦点を絞り，通常の動作計画問題と共通する，障害物回避などの幾何学的問題の扱いは最小限にとどめることとする．

6.3.2　問題設定

本章で扱う計画問題では，(Fig. 6.2) のような多指ハンドによるグラスプレス・マニピュレーションを想定して，以下の条件を仮定する．

1. 対象物，ロボットの指先，環境は剛体である．
2. マニピュレーションは準静的に行われる．
3. クーロン摩擦が，対象物と環境の間，および対象物とロボット指の間に存在する．
4. 静止摩擦係数と動摩擦係数は等しい．
5. 摩擦円錐は凸多面錐 [平井 1999] によって近似できる．
6. ロボット指は対象物と摩擦あり点接触する．
7. すべての接触は複数の点接触で近似できる（2.3 節参照）．
8. 不完全な接触[張 1996]は存在しない．
9. ロボットの指力の接触法線成分には上限がある．
10. ロボットの各指は位置制御モードもしくは力制御モードのどちらかに設定される．

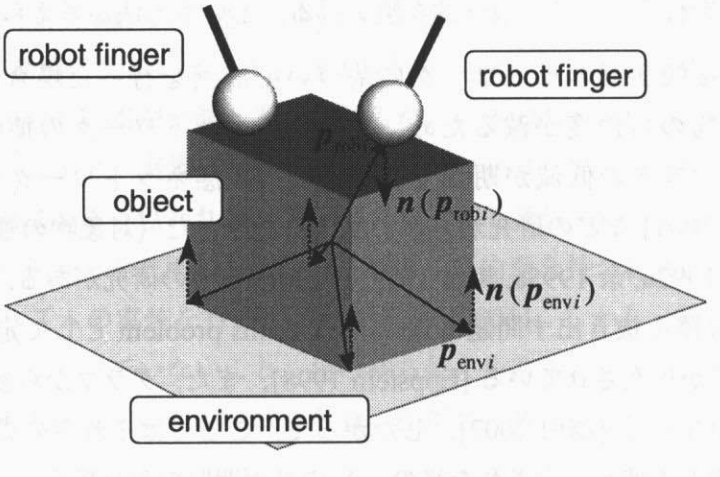


Fig. 6.2: Object in Graspless Manipulation

11. 位置制御モードの場合，ロボットの指は，その摩擦円錐内の任意の力を（受動的に）発生することができる．
12. 力制御モードの場合，ロボットの指は位置・力のハイブリッド制御[Raibert 1981] される．ここでは，ロボット指は接触法線方向には指令された力を能動的に発生し，接線方向には摩擦円錐内の任意の力を受動的に発生できるとする．
13. ロボット指は，対象物と摩擦あり点接触する球形の剛体で近似される．
14. 指の対象物上での滑り・転がりは扱わない．対象物上の指位置を変化させる場合は，持ち替えを行う．
15. 指と指の間隔はある一定距離以下であるとし（ハンドの機構による制約条件の近似），この範囲内であれば他の物体と干渉しない限り持ち替えを行うことができるとする．

下線部は 2.3 節における仮定に加えて新たに導入したものである．記号等は 2.3 節で定義されたものを用いる．

通常の動作計画手法でカバーされるような幾何学的問題の取り扱いを最小限とするため，ロボット指のリンクなどは無視し，指先だけを考える．これによって，グラスプレス・マニピュレーション特有の，力学的問題に焦点を絞る．ただし，リンクの存在を無視する代わりに，最低限の考慮として，指と指の間隔が一定距離以上にならないような制約を入れている（条件 15）．なお，指先を球で近似する（条件 13）のは，障害物との干渉判定を単純化するためである．

解くべき問題は以下の通りである．

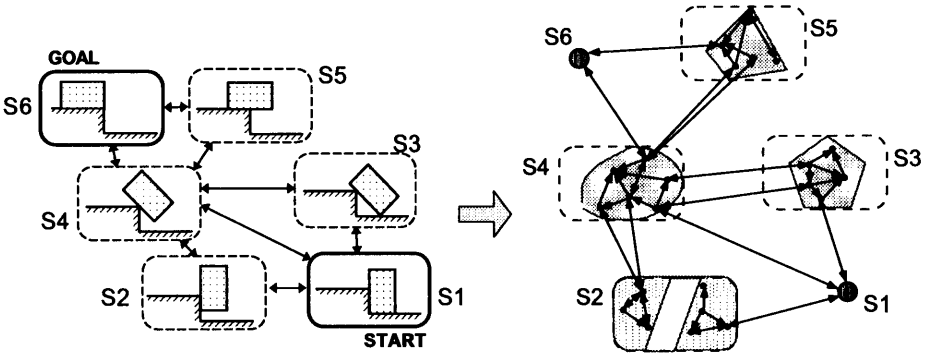
> 対象物を初期コンフィギュレーションから目標コンフィギュレーションまで移動させるための，指先位置および指の制御モード（位置制御／力制御）の系列を求める．

また，力制御指については目標指力も併せて求める．

6.3.3 計画手法の概要

提案する計画手法では，対象物と指の自由度を合わせて考えたコンフィギュレーション空間をグラフで表現することによって，最終的にマニピュレーションの計画問題を，グラフ探索問題に帰着させることとする．計画手法は以下の要素から構成される．

1. 指先位置が与えられた際の，指の制御モードを考慮した作業の確実性の評価（5 章）
2. 同一接触状態内（以後特に明記しない限り，「接触状態」とは環境と対象物の間の接触



i) Contact State Network ii) Manipulation-Feasibility Graph

Fig. 6.3: Generation of Manipulation-Feasibility Graph

状態を指すこととする）でのマニピュレーションを表現するグラフの生成（6.4 節）

3. 接触状態遷移の表現のためのグラフ間の連結（6.5 節）
4. 生成されたグラフの探索による計画の実行（6.6 節）

1. では，コンフィギュレーション空間内の 1 点に対してマニピュレーション中にその点を通過可能であるかどうかの判定を，操作の確実性の評価を元に行う．その際に，同時に指の制御モード（および力制御指に関しては指令指力）を決定する．これは 5 章の内容に対応する．2. では，1 つの接触状態について，コンフィギュレーション空間を離散化してノードを生成し，それらをアークで接続する．本論文ではこのようにして作られるグラフを「操作可能性グラフ」と呼ぶ．3. では，2. で生成された複数のグラフの間をさらにアークで接続して 1 つの大きなグラフにする．4. では，最終的に生成された操作可能性グラフを探索することによって計画を行う．操作可能性グラフの生成のイメージを Fig. 6.3 に示す．

6.4 同一接触状態内でのグラフ生成

6.4.1 コンフィギュレーション空間の導入

対象物と指の自由度を合わせて考えたコンフィギュレーション空間を考える．なお，ここでは説明のためにコンフィギュレーション空間の概念を用いるが，実際には陽にコンフィギュレーション障害物やコンフィギュレーション自由空間を計算することは行わない．いま，対象物の位置・姿勢が3次元空間内の $d(\leq 6)$ 自由度を持つとする．また，指の本数を n とする．本来，コンフィギュレーション空間は $d+3n$ 次元となる（指は球と仮定していることに注意）が，これでは次元がかなり高い．そこで，計画における指のコンフィギュレーションを，対象物表面上に限定する (Fig. 6.4)．これにより，コンフィギュレーション空間の次元を $d+2n$ に下げることができる．

しかし，このコンフィギュレーション空間は通常の障害物回避問題の場合と同様な方法では探索することはできない．なぜなら，

- 対象物を把持していないため，コンフィギュレーション空間内のある点からある点への移動ができても，その逆が不可能であることがある．
- 持ち替えが発生するため，指の自由度に関してはコンフィギュレーション空間内での不連続な移動が生じる．

という問題があるためである．もちろん，後者の問題は，コンフィギュレーション空間の次元を下げるために，指位置を対象物平面上に限定したことに起因するものである．しかし，

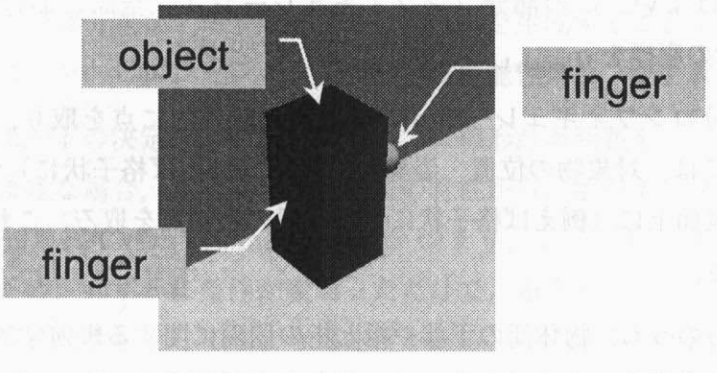


Fig. 6.4: Robot Fingers on the Object

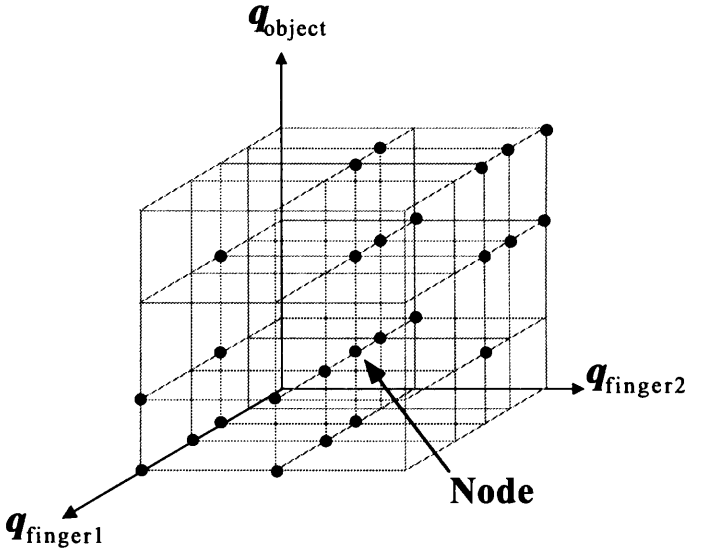


Fig. 6.5: Generation of Nodes

前者はグラスプレス・マニピュレーションにおける本質的な問題と言える.

これらの問題を解決するため, ここでは コンフィギュレーション空間を離散化してグラフのノードとし, その間の遷移が可能な場合のみノード間をアークでつなぐことにより, 有向グラフでコンフィギュレーション空間を近似表現することとする.

6.4.2 コンフィギュレーション空間の離散化によるノード生成

ここではまず, 同一接触状態内でのノード生成を行う. 接触状態が指定されているので, このときの対象物の自由度を $d(\leq 6)$ として, $(d+2n)$ 次元の部分コンフィギュレーション空間を考えればよい. この部分コンフィギュレーション空間における対象物の座標を $\boldsymbol{q}_{\text{object}} \in \Re^d$, 指の座標を $\boldsymbol{q}_{\text{finger1}}, \cdots, \boldsymbol{q}_{\text{fingern}} \in \Re^2$ とおく.

まず, この部分コンフィギュレーション空間内に離散的に点を取り, これをノード候補とする. 具体的には, 対象物の位置・姿勢について (例えば格子状に) サンプル点を取る. また, 対象物の表面上に (例えば格子状に) 指配置の候補点を取る. これらの組み合わせがノード候補となる.

そして, これらのうち, 物体間の干渉や指と指の間隔に関する幾何学的な制約条件 (6.3.2 節の仮定を参照) を満たすもののみを, ノードとして採用する (Fig. 6.5).

6.4.3 アークの生成

ノード間を接続するアークとして，対象物の変位を表すアークと持ち替えを表すアークの2種類が存在する．これらのアークの生成の方法を以下に述べる．

対象物の変位を表すアークとは，対象物上の指位置を変化させずに対象物のみを動かすことに相当し，コンフィギュレーション空間上では対象物の自由度方向に隣接しているノード間を接続する．

準静的なマニピュレーションを実現するためには，過大な内力を生じさせることなく，また操作の確実性が十分あることが必要である．そこで，5章で示した指の制御モードの決定手法を用いて，適切に指の制御モード（および力制御指の指令指力）を決定したときの，操作の確実性の評価値 z を調べる．この z の値が，ある基準値 $z_{\min}$ 以上になるようにする．このアーク上のすべての点で $z \geq z_{\min}$ である必要があるので，アーク上を細かく P 区間に分割し，各分割点について（指の制御モードを適切に決定したときの）z の値を計算し，すべての点において $z \geq z_{\min}$ であった場合のみ，ノード間をアークで結ぶ．滑りのない遷移の場合は双方向のアークで結んでよいが，滑りがある場合は，片方向にしか操作ができない場合（押すことはできるが引くことはできない，など）が存在するので，滑りの方向ごとに別々に調べて有向アークで結ぶ (Fig. 6.6 (a)).

持ち替えを表すアークとは，対象物の位置・姿勢を変化させることなく1本の指の対象物上の位置を変化させることに相当する．持ち替えによって対象物上の指位置は不連続に変化するので，アークはコンフィギュレーション空間内で隣接していないノード間にも生成する必要がある．したがって，まず持ち替えを行う指を離した状態での操作の確実性 z の値を調べる．$z \geq z_{\min}$ のときは自由に持ち替えが可能なので，このノードと，持ち替える指以外の座標が共通であるノードすべてとの間を双方向のアークで接続する (Fig. 6.6 (b)).

以上のようにして遷移可能なノード間すべてにアークを生成することで，1つの接触状態内でのグラスプレス・マニピュレーションを表す操作可能性グラフが生成される．

なお，指の制御モードの決定において，本論文では線形計画問題をシンプレックス法で解いているが，このような場合，シンプレックス法の特性から，解のチャタリングが起こる可能性があることが知られている [Shimoga 1996]．つまり，シンプレックス法は解領域の頂点を解として返すため，わずかに条件が変わっただけで，頂点から頂点への解のジャンプが発生し，不必要な不連続性を生じてしまう可能性がある．このような状況への対応策としては，グラフ探索によりいったん解が得られた後，再度，シンプレックス法以外の安定な方法により，指の制御モード（および指令指力）の決定をやり直す，という方法が考えられる．

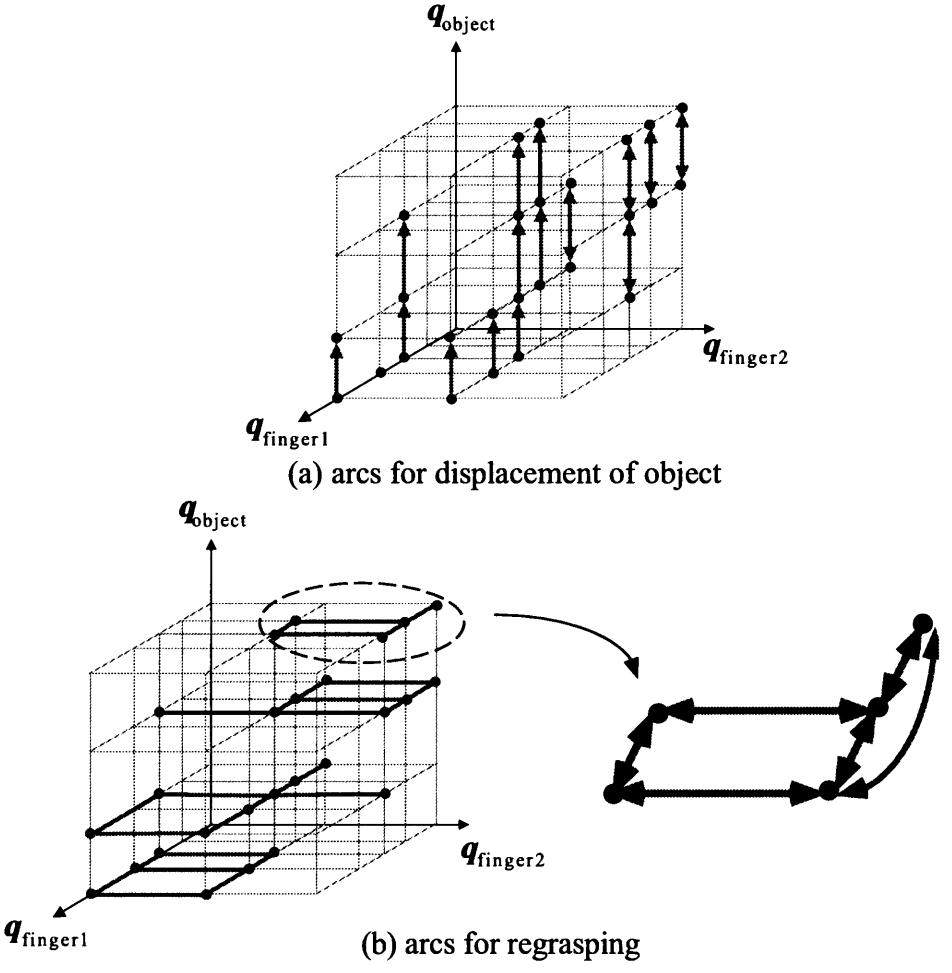


Fig. 6.6: Generation of Arcs

この場合，シンプレックス法の代わりに例えば勾配射影法 (gradient projection method) を用いる方法が提案されている [Klein 1990].

6.5 グラフ間の結合による接触状態遷移の表現

6.4 節で生成されたグラフは，同一接触状態内でのマニピュレーションを表現するものであった．グラスプレス・マニピュレーションは複数の接触状態間にまたがる場合があり，その計画のためには，接触状態ごとのグラフ間を連結して 1 つの大きな操作可能性グラフを生成する必要がある．

2 つの異なる接触状態の操作可能性グラフの間で連結できるノードは，対象物および指のコンフィギュレーションが全く同じであるノードである．これらは接触状態が遷移する瞬間の全く同じ状態を表しているので，これらを無条件でコスト 0 の双方向のアークで接続すればよい．

すべての操作可能性グラフの間でこの操作を行うことで，複数の接触状態にまたがる大きな操作可能性グラフが生成できる．

6.6 グラスプレス・マニピュレーションの計画

6.6.1 操作可能性グラフの探索

生成された操作可能性グラフの探索により，グラスプレス・マニピュレーションの計画を行う．計画における初期および目標コンフィギュレーションは，指位置に関しては規定しないため，操作可能性グラフ上ではそれぞれ複数のノードに対応する．このため，これらのノードにコスト0のアークで接続された仮想的な初期ノード・目標ノードを生成する(Fig. 6.7).

操作計画は，初期ノードから目標ノードまでの最小コスト経路を，操作可能性グラフの探索により求めることで行える．前節までのグラフ生成は，実際には探索する前に全体を構築するのではなく，探索を行う過程で必要な部分だけグラフを生成すればよい．グラフ探索のためには，生成される各アークに適当なコストを割り振る必要があるが，ここでは

1. 操作の確実性が低くなるところは通らない．
2. 持ち替え回数を最少にする．
3. できるだけロボット指の移動量を小さくする．
4. できるだけ操作の確実性を大きくする．

という方針をとることにする．

グラスプレス・マニピュレーションは対象物を把持していないため，操作の確実性を十分確保することは，操作の失敗を防ぐ上で重要である．また，指の持ち替えは，操作にかかる時間や持ち替え中の対象物の安定性の問題から，避けたほうが望ましい．アーム・ハンド機

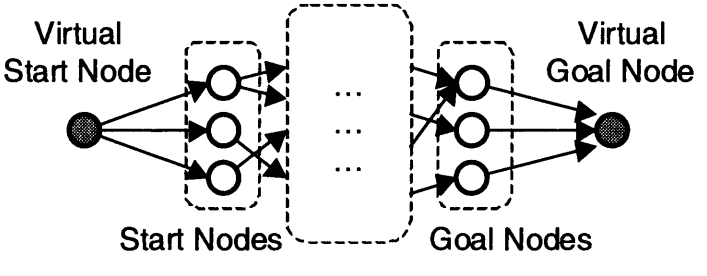


Fig. 6.7: Virtual Start/Goal Node

構を用いる場合は，持ち替えが無ければハンドを動かさずにあやつりを行える利点もある．指の移動量については，可動範囲や操作にかかる時間の問題を考えると，基本的には小さい方が望ましいと言える．以上のことから，上記の方針は妥当であると考えられる．ただし，上の条件から順に優先度が高いとし，ここではこれを実現するための適切なコスト設定を考える．

まず，操作の確実性については，アーク生成時に基準値 $z_{\min}$ を設定していることによって，すでに対応済みである．

次に，ノード n と n' を結ぶ，対象物変位方向のアークに対して割り当てるコストを，

$$c(n,n') = \max_i \sum_{j=1}^{P} \left(1 + \frac{X_{\text{stab}}}{z_j}\right) \|\Delta \boldsymbol{q}_{\text{finger}\,i,j}\| \tag{6.1}$$

とする．ただし $\|\Delta \boldsymbol{q}_{\text{finger}\,i,j}\|$ はこのアークを P 区間に分割したときの，j 番目の区間におけるロボットの i 番目の指先の（絶対的な）変位の大きさ，z_j はそのときの（適切に制御モードを選択したときの）操作確実性の値である．また X_{stab} は，$X_{\text{stab}}/z_{\min}$ が 1 より十分小さくなるように決めた定数である．(6.1) 式は，各指の変位が大きくなるほど，また，操作の確実性が小さくなるほど，大きな値をとることになる．

滑りのある変位に対応するアークの場合，滑りの方向によって最適な指力が変化するため，逆方向のアークでは (6.1) 式は異なる値をとる（場合によっては，その操作は不可能になる）ということに注意されたい．

また，ノード n と n' を結ぶ，持ち替え方向のアークに割り当てるコストは，

$$c(n,n') = X_{\text{regr}} \tag{6.2}$$

とする．ここで X_{regr} は持ち替えそのもののコストを表す定数とし，式 (6.1) の値より十分大きくなるようにとる．

以上のようにコストの割り当てを行うことにより，持ち替えの回数の最少化を優先し，その上で（式 (6.1) の意味で）ロボットにかかる負荷の最も小さい経路を，グラフ探索で求めることができる．

6.6.2 A^* 探索のためのヒューリスティクス

グラフの探索をより短時間に行うため，A^* 探索 [Pearl 1984, 古川 1997] を用いる．A^* 探索とは，ゴールまでの見積もりコスト

$$f(n) = g(n) + h(n) \tag{6.3}$$

に基づいて探索を行う方法である．ここで，$g(n)$ はスタートからノード n までの経路のコスト，$h(n)$ はノード n からゴールまでの最短経路の見積もりコストであり，したがって $f(n)$ はノード n を通る最短経路の見積もりコストを意味する．以下に，A^* 探索の手順を示す [Pearl 1984].

1. スタートノード s を OPEN リストに入れる．
2. OPEN リストが空ならば，探索失敗として終了．
3. f が最小となるノード n を OPEN リストから削除して CLOSED リストに入れる．
4. もし n がゴールノードならば，n から s までポインタを逆にたどって解を求め，探索成功として終了．
5. そうでなければ，n を展開してその次のノードをすべて生成し，それらのノードに n へ戻るポインタ情報を付け加える．n の次のすべてのノード n' について，
   (a) もし n' がまだ OPEN リストにも CLOSED リストにも入っていなければ，$h(n')$ を見積もって $f(n') = g(n') + h(n')$ を計算し，n' を OPEN リストに入れる．
   (b) もし n' がすでに OPEN リストもしくは CLOSED リストに入っていればそのポインタを，最小の $g(n')$ を与える経路を指すように変更する．
   (c) もし n' のポインタの変更が行われ，かつそれが CLOSED リスト上にあった場合は，再度 n' を OPEN リストに入れる．
6. ステップ 2 に戻る．

ここで，$h(n')$ はノード n' におけるヒューリスティック関数の値であり，ノード n' からゴールノードまでの最適経路のコストの見積もりを表す．また，$g(n')$ はノード n' までの経路の実コストを表し，以下のように定義される．

$$g(n') = g(n) + c(n, n')$$
$$g(s) = 0$$

ただし，$c(n, n')$ はノード n から次のノード n' へ移動するためのコストである．

A^* 探索では，許容的 (admissible)，すなわち実際のコストを越えないようなコストの予測（ヒューリスティック関数）を探索に用いることで，最適性を保ったまま高速化を図ることができる．また，コストの予測が正確であればあるほど，A^* 探索は効率的になる．つまり，設計するヒューリスティック関数の質いかんで，A^* 探索の性能が決まることになる．

本章で扱うマニピュレーション計画問題では，持ち替えに対して大きなコストを与えている．したがって，探索の性能を上げるためには，持ち替えの必要な状態を検知し，ヒューリスティック関数によって適切なコスト予測をすることが重要であると言える．そこで，ここ

ではヒューリスティック関数として，以下のような許容的な関数を設計した．

$$h(n) = \begin{cases} \max_i \|\Delta \boldsymbol{q}^*_{\text{finger}\,i}\| & \left(\begin{array}{c}\text{現在の指配置で目標位置に到達}\\ \text{しても指が環境と干渉しない場合}\end{array}\right) \\ n_{\text{viol}} X_{\text{regr}} & \left(\begin{array}{c}\text{現在の指配置で目標位置に到達}\\ \text{すると指が環境と必ず干渉する場合}\end{array}\right) \end{cases} \tag{6.4}$$

ここで $\|\Delta \boldsymbol{q}^*_{\text{finger}\,i}\|$ は，現在の指配置のままで目標位置に到達した際の，i 番目の指の変位の予測値（最小値）を表す．また，n_{viol} は，現在の指配置のままで対象物を目標位置に到達させた場合に，環境と干渉する指の本数である．現在の指配置のままで対象物を目標位置に到達させたとすると指が環境と干渉する場合は，それらの指は必ず持ち替えをしなければならない，という事実を利用して，探索の高速化を図っている．

ここで，(6.4) 式のヒューリスティック関数は単調である．すなわち，successor(n) をノード n と接続されているノードの集合としたときに，

$$h(n) \leq c(n, n') + h(n') \text{ for } \forall n, n' | n' \in \text{successor}(n) \tag{6.5}$$

である（証明は付録 A.3 参照）．したがって，経路最大 (pathmax) の式 [古川 1997] を用いずとも，A^* 探索中に同じノードを複数回展開する必要がなく [Pearl 1984]，効率的に探索を行える．

なお，ヒューリスティックスを評価するための指標として，有効分岐数 b^* がある．これは，ある問題に対して A^* で展開されたノードの総数が N で解の深さが d としたとき，

$$N + 1 = 1 + b^* + (b^*)^2 + \cdots + (b^*)^d \tag{6.6}$$

となる数である [古川 1997]．ヒューリスティック関数の導入によって，有効分岐数をどれくらい減らせたか（1 に近づけることができたか）が，その良し悪しを評価する 1 つの指標となる．

6.6.3 A^* 探索の改良

本章で提案する計画アルゴリズムにおいては，各ロボット指の適切な制御モードを決定する部分（5 章）の計算量がかなり大きい．A^* 探索は，ヒューリスティック関数を利用して，展開する必要のあるノード数を減らすことによって計算量を低減するものであるが，本章で扱う計画問題において計算量を低減するためには，単に展開するノード数を減らすだけでなく，制御モード決定の計算回数を減らすことが有効である．

制御モード決定の計算回数を減らすためには，ロボット指の制御モードの計算を必要とする，$c(n, n')$ の計算の回数を抑えることが必要である．そこで，通常の A^* 探索のように，展開したノードの次のノードすべてについて $f(n') = g(n) + c(n, n') + h(n')$ を計算するのではなく，必要なときだけ $f(n')$ を計算するようにすることを考える．

1. スタートノード s を OPEN リストに入れる．
2. OPEN リストおよび SEMI-OPEN リストがともに空ならば，探索失敗として終了．
3. f が最小となるノード n を OPEN リストの中から調べる．
4. SEMI-OPEN リストの中から $f'(\tilde{n}) < f(n)$ となるすべてのノード $\tilde{n}$ について，$f(\tilde{n})$ を計算し，SEMI-OPEN リストから削除して OPEN リストに入れる．$f(\tilde{n}) < f(n)$ ならば $n \leftarrow \tilde{n}$ とする．
5. f が最小となるノード n を OPEN リストから削除して CLOSED リストに入れる．
6. もし n がゴールノードならば，n から s までポインタを逆にたどって解を求め，探索成功として終了．
7. そうでなければ，n を展開してその次のノードをすべて生成し，それらのノードに n へ戻るポインタ情報を付け加える．n の次のすべてのノード n' について，
   (a) もし n' がまだ OPEN リストにも SEMI-OPEN リストにも CLOSED リストにも入っていなければ，$h(n')$ を見積もって，$f'(n') = g'(n') + h(n')$ を計算し，n' を SEMI-OPEN リストに入れる．
   (b) もし n' がすでに OPEN リストもしくは SEMI-OPEN リストに入っていれば，そのポインタを，最小の $g(n')$ を与える経路を指すように変更する．最小の $g(n')$ を与える経路を決定するのに必要であれば，$f(n')$ を計算する．このとき，n' が SEMI-OPEN リストに入っていたならば，削除して OPEN リストに入れる．
   (c) もし n' がすでに CLOSED リストに入っていれば，（ヒューリスティック関数の単調性より）無視する．
8. ステップ 2 に戻る．

以上の手続きの概略のフローチャートを Fig. 6.8 に示す．ここで，$g'(n')$ はノード n' までの最適経路の実コストの予測値を表し，以下のように定義される．

$$g'(n') = g(n) + c'(n, n')$$
$$g'(s) = 0$$

ただし，$c'(n, n')$ はノード n から次のノード n' へ移動するためのコストの予測値であり，$c'(n, n') \leq c(n, n')$ とする．ここでは以下のように設定した．

$$c'(n, n') = \begin{cases} \max_i \sum_{j=1}^{P} \|\Delta \boldsymbol{q}_{\text{finger}\,i,j}\| & \text{（対象物変位に対するアークの場合）} \\ X_{\text{regr}} & \text{（持ち替えに対応するアークの場合）} \end{cases} \tag{6.7}$$

以上の手続きでは，通常は $f'(n') = g(n) + c'(n, n') + h(n')$ のみを計算し，必要なときだけ $f(n') = g(n) + c(n, n') + h(n')$ を計算することになる．これにより，容易に計算でき

る $c'(n,n'), h(n')$ を利用することにより，計算コストの高い $c(n,n')$ の計算回数を抑えることができる．結果として，探索にかかる時間を低減できる．

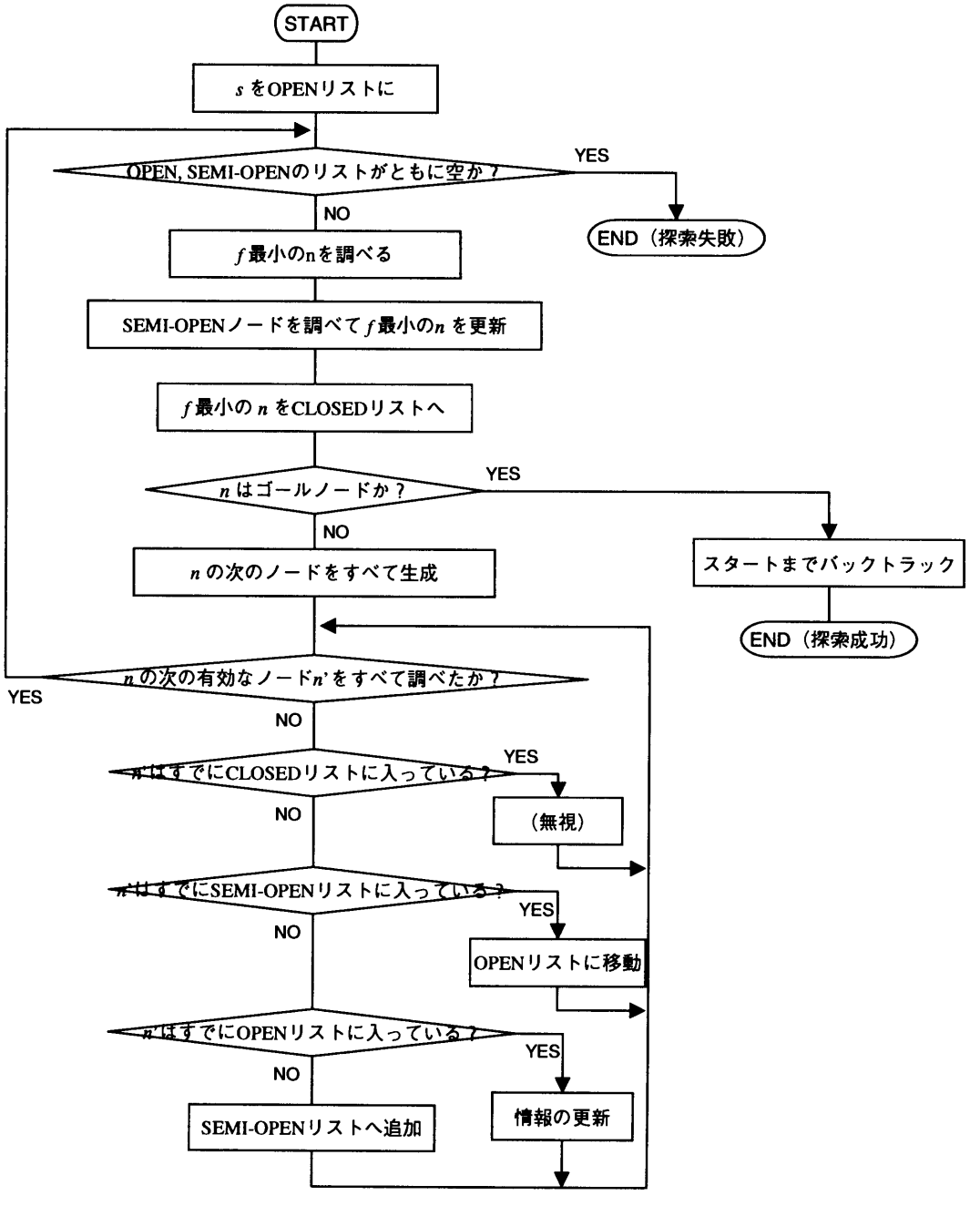


Fig. 6.8: A Modified A^* Algorithm

6.7 計画例

以上の計画アルゴリズムをC言語を用いて実装した．本節では，それを用いた計画の例を示す．なお，以下に示す計算時間は，いずれもPentium4-2.8GHzを搭載したLinux PC上での実行にかかった結果である．

本節では，2本のロボット指によるグラスプレス・マニピュレーションを考える．各指は半径1の球で表現する．また，重力加速度を9.8，各接触点での摩擦円錐は，凸6面錐で近似 ($s = 6$) し，各指の力の上限は $\boldsymbol{f}_{\max} = [10, 10]^T$ とする．指同士の間隔は9以下でなければならないとし，また，対象物と指の間の摩擦係数を0.7，対象物と環境との間の摩擦係数を0.5と設定する．

また，操作の確実性の値の計算において，6次元単位超球を，それに外接する超多面体で近似する．ここでも，この超多面体の頂点としては，以下の76点をとった ($N_{\text{vert}} = 76$).

$$\boldsymbol{l}_i = \begin{cases} k[\pm 1,\ 0,\ 0,\ 0,\ 0,\ 0]^T \\ k[0,\ \pm 1,\ 0,\ 0,\ 0,\ 0]^T \\ k[0,\ 0,\ \pm 1,\ 0,\ 0,\ 0]^T \\ k[0,\ 0,\ 0,\ \pm 1,\ 0,\ 0]^T \\ k[0,\ 0,\ 0,\ 0,\ \pm 1,\ 0]^T \\ k[0,\ 0,\ 0,\ 0,\ 0,\ \pm 1]^T \\ k\left[\pm\frac{1}{\sqrt{6}},\ \pm\frac{1}{\sqrt{6}},\ \pm\frac{1}{\sqrt{6}},\ \pm\frac{1}{\sqrt{6}},\ \pm\frac{1}{\sqrt{6}},\ \pm\frac{1}{\sqrt{6}}\right]^T, \end{cases} \tag{6.8}$$

ここで $k = 2\sqrt{3 - \sqrt{6}} \approx 1.48$ である（付録A.2参照）．式(3.1)の $\boldsymbol{R}$ については，対象物の質量 M_o と慣性テンソル $\boldsymbol{J}_o$ を使う，式(3.2)を適用した．

また，計画に関するパラメータについては，$X_{\text{regr}} = 10^2$，$X_{\text{stab}} = 10^{-2}$，$z_{\min} = 0.5$，$P = 3$ とした．

6.7.1 滑らせ操作の計画

提案したアルゴリズムによって，Fig. 6.9のような，平面上に置かれた立方体対象物の滑らせ動作の計画を行う．これは，対象物を初期位置から目標位置まで，1次元の並進運動をさせるための，ロボット指の動作を求める問題である．ここでは対象物を距離9だけ並進させることとした．

コンフィギュレーション空間の離散化については，対象物上の各面を 7×7 のグリッドに切ることによって，配置可能点を設定した．また，対象物の自由度については31段階に離散

Fig. 6.9: Sliding of a Cuboid

化した．これにより，理論的には最大

$$_{7\times7\times6}C_2 \times 31 = 1,335,201$$

のノードから成る操作可能性グラフを探索することになる．

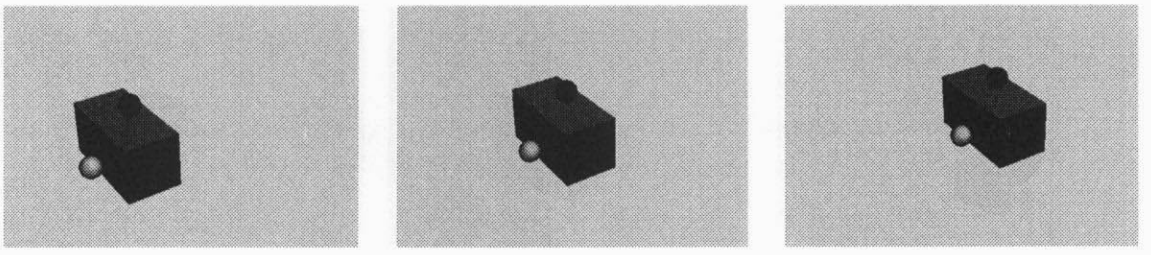
直方体の滑らせ操作 (1)

操作対象物を，大きさ $5 \times 5 \times 10$ の直方体とし，その質量は 1 ($M_o = 1$) で均質であるとする．

横に寝かせたこの直方体を平面上で滑らせる操作の計画 (Fig. 6.9) を行った（この計画問題を Sliding 1 と呼ぶことにする）．この場合，Fig. 6.10 のように，片方の指（指 1）で対象物の背面を位置制御で押し，もう片方の指（指 2）で上面を力制御で押さえつけて動かす，という操作が生成された．このときの対象物上の指位置は，指 1 が $[0,\ -2.5,\ -1.25]^T$，指 2 が $[0,\ 0.625,\ 2.5]^T$ である．このとき計画された指力の指令値を Fig. 6.11 に示す．ただし，指令値がゼロの場合は，その指が位置制御モードにあることを表す（以降のグラフでも同様）．計画にかかった時間は約 533 CPU 分，探索中に展開されたノード数は 32,538，有効分岐数は 1.3374 であった．

なお，指力の上限を 5 と小さく設定して同じ操作を計画したところ（問題 Sliding 1' とする），Fig. 6.12 のように，両方の指で，対象物の背面を位置制御で押す操作が生成された．このときの対象物上の指位置は，指 1 が $[-3.75,\ -2.5,\ 0]^T$，指 2 が $[3.75,\ -2.5,\ 0]^T$ である．生成された操作は Lynch の stable push [Lynch 1996b] に対応するものである．計画にかかった時間は約 203 CPU 分であり，探索中に展開されたノード数は 23,195，有効分岐数は 1.3213 であった．

なお，これらの例における操作の確実性の評価値の推移を Fig. 6.13 に示す．操作の確実性の値は，対象物を滑らせ始める瞬間に小さくなり，操作が完了して静止状態になると元に



(a) start (b) intermediate (c) goal

Fig. 6.10: Sliding 1: Planned Operation

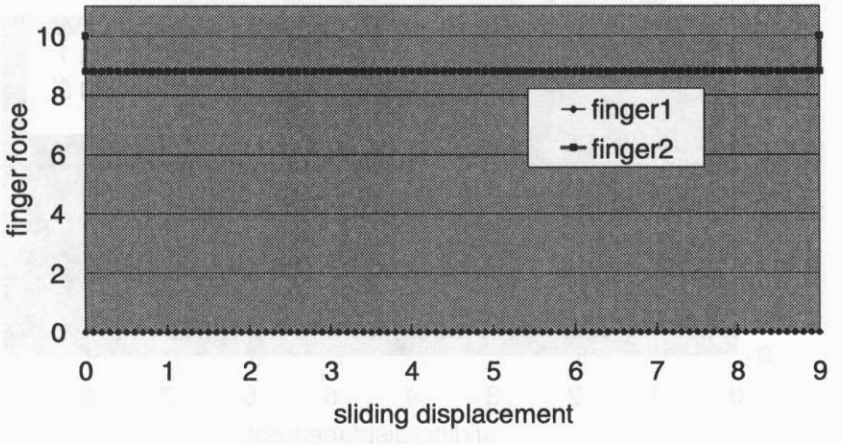


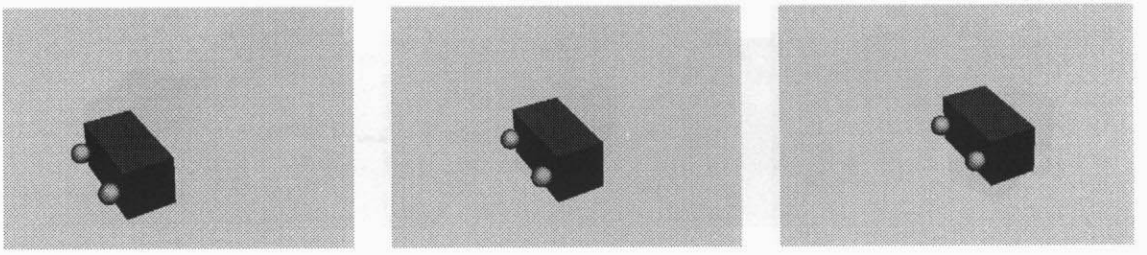
Fig. 6.11: Sliding 1: Planned Finger Forces

戻ることが分かる．また，指力の制限が厳しい場合は，操作の確実性がより低いマニピュレーションとなっていることが見てとれる．

以上の結果からは，本章で提案している計画アルゴリズムを用いると，大きい指力を発生可能な場合は，内力を大きくして把持に近い状態を作って動かす操作（Type A の操作 [相山 1996a], 2.2.3 項を参照）が生成され，また指力が弱いときには，内力を伴わない操作（Type B の操作 [相山 1996a]）が生成されることがわかる．

直方体の滑らせ操作 (2)

次に，同じ対象物を長手方向に滑らせる操作を計画した（問題 Sliding 2）ところ，Fig. 6.14 のように，片方の指（指 1）で対象物の背面を位置制御で押し，もう片方の指（指 2）で上面を力制御で押さえつけて動かす，という操作が生成された．このときの対象物上の指位置



(a) start (b) intermediate (c) goal

Fig. 6.12: Sliding 1': Planned Operation

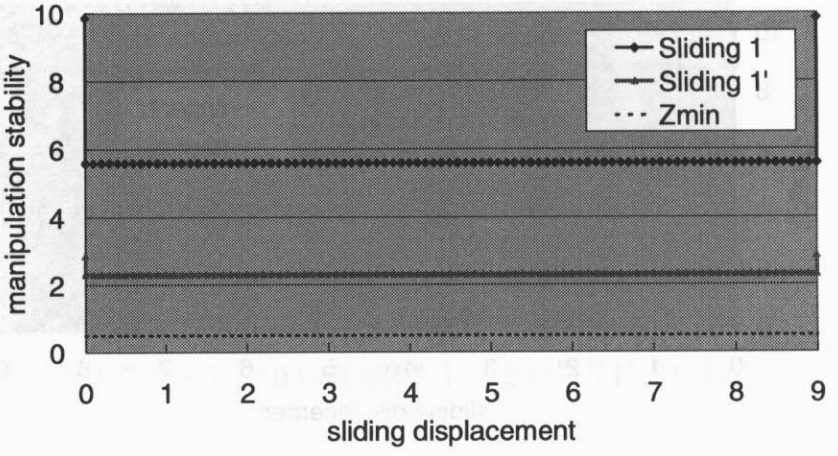
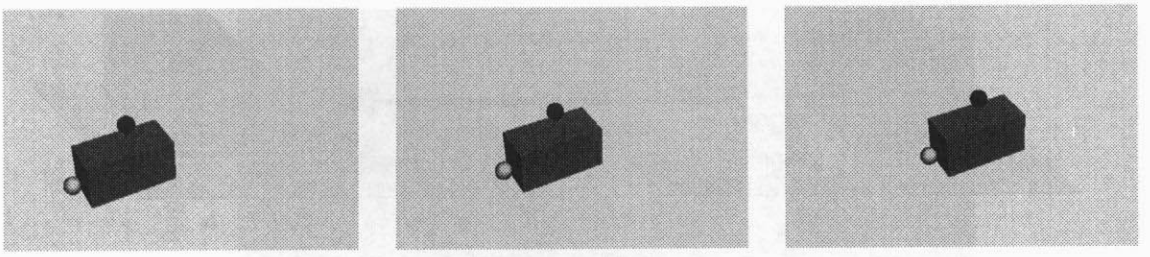


Fig. 6.13: Sliding 1: Manipulation Stability

は，指 1 が $[-0.625,\ -5,\ -1.25]^T$，指 2 が $[-1.875,\ 1.25,\ 2.5]^T$ である．

指 1, 2 の x 座標がゼロでなく，ずれた位置に配置されているのは，計画のための評価関数の値がまったく同じになる指配置が複数あるためである．つまり，その中でたまたま最初に見つかったのが，上記の指配置であった，ということである．評価関数が同じになる指配置が複数できてしまうのは，操作の確実性の評価指標がミニマックス解として与えられていること（式 (3.4)）に起因する．つまり，最も「弱い」方向についての外乱に対するロバスト性が同じであれば，他の方向については差があっても，指標の値に影響しないためである．

なお，このとき計画された指力の指令値を Fig. 6.15 に示す．また，操作中の操作の確実性の指標値の推移を Fig. 6.16 に示す．計画にかかった時間は約 524 CPU 分，探索中に展開されたノード数は 34,691，有効分岐数は 1.3405 であった．



(a) start (b) intermediate (c) goal

Fig. 6.14: Sliding 2: Planned Operation

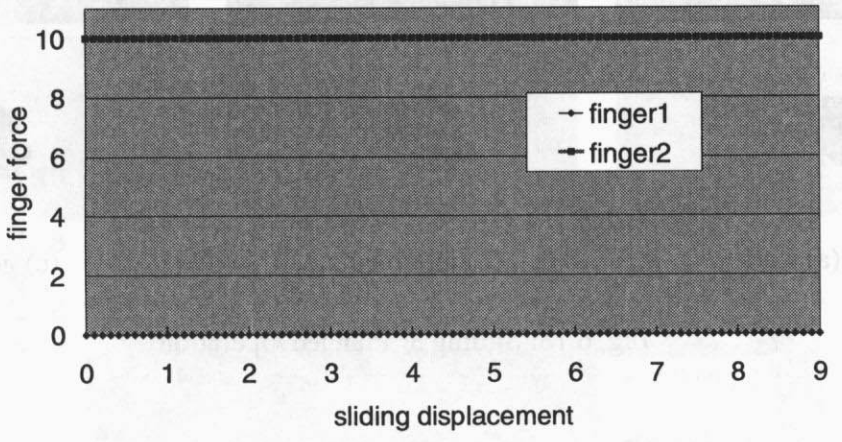


Fig. 6.15: Sliding 2: Planned Finger Forces

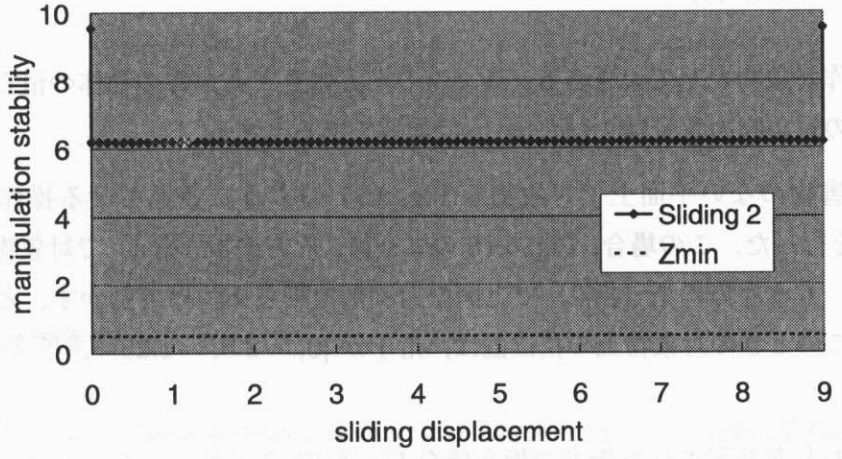


Fig. 6.16: Sliding 2: Manipulation Stability

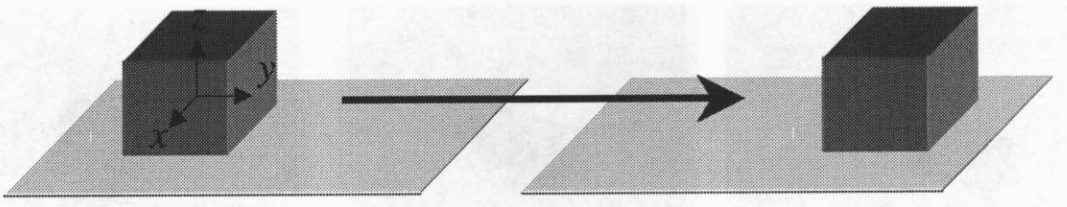
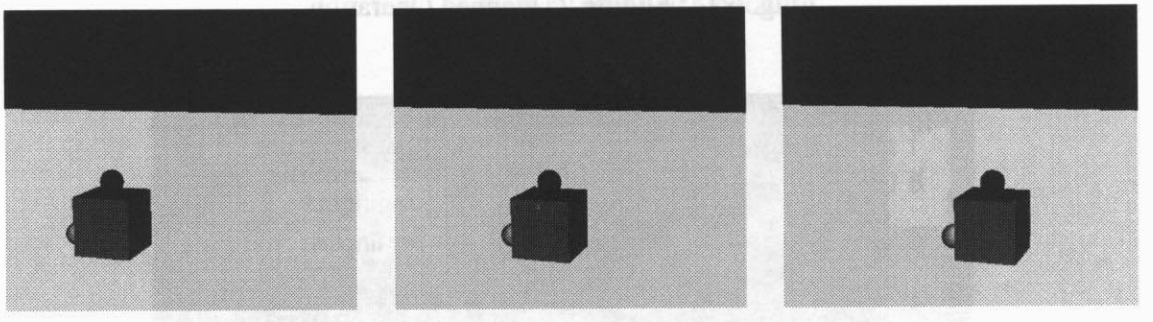


Fig. 6.17: Sliding of a Cube



(a) start (b) intermediate (c) goal

Fig. 6.18: Sliding 3: Planned Operation

立方体の滑らせ操作 (1)

次に，操作対象物として大きさ $5 \times 5 \times 5$ の立方体を考え，この物体の滑らせ操作の計画を行う．この対象物の質量は 1 ($M_o = 1$) で均質であるとする．

まず，障害物のない平面上で対象物を Fig. 6.17 のように並進させる操作の計画（問題 Sliding 3）を行った．この場合，Fig. 6.18 のように，片方の指（指 1）で対象物の背面を位置制御で押し，もう片方の指（指 2）で上面を力制御で押さえつけて動かす，という操作が生成された．このときの対象物上の指位置は，指 1 が $[0,\ -2.5,\ -1.25]^T$，指 2 が $[0,\ 0,\ 2.5]^T$ である．

なお，このとき計画された指力の指令値を Fig. 6.19 に示す．また，操作中の操作の確実性の指標値の推移を Fig. 6.20 に示す．計画にかかった時間は約 1030 CPU 分，探索中に展開されたノード数は 63,323，有効分岐数は 1.3694 であった．

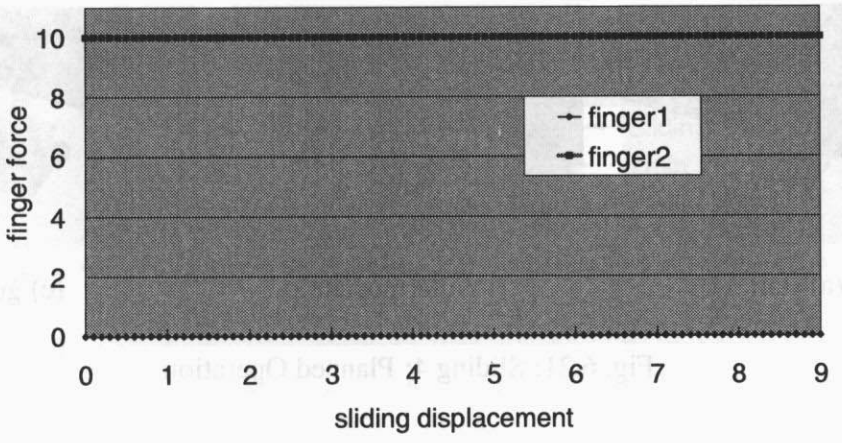


Fig. 6.19: Sliding 3: Planned Finger Forces

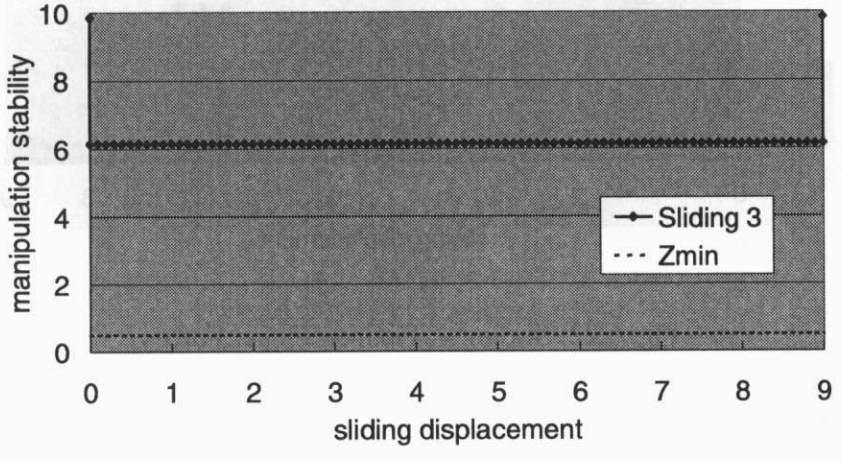
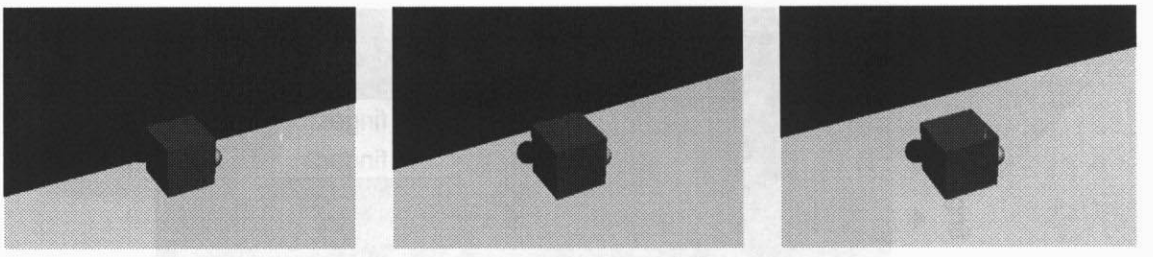


Fig. 6.20: Sliding 3: Manipulation Stability

立方体の滑らせ操作 (2)

次に，対象物の背後に障害物がある場合について計画を行った（問題 Sliding 4）．この場合，前の例のように物体の背後に指を配置することができないので，代わりに Fig. 6.21 のように，対象物を両側からつまんで滑らせる動作が生成された．このときの対象物上の指位置は，指 1 が $[2.5,\ -1.25,\ 0.625]^T$，指 2 が $[-2.5,\ 0,\ -1.25]^T$ である．また，各指の指令力，および操作の確実性の指標の値の推移を Fig. 6.22，6.23 に示す．この場合は，指 1 を力



(a) start (b) intermediate (c) goal

Fig. 6.21: Sliding 4: Planned Operation

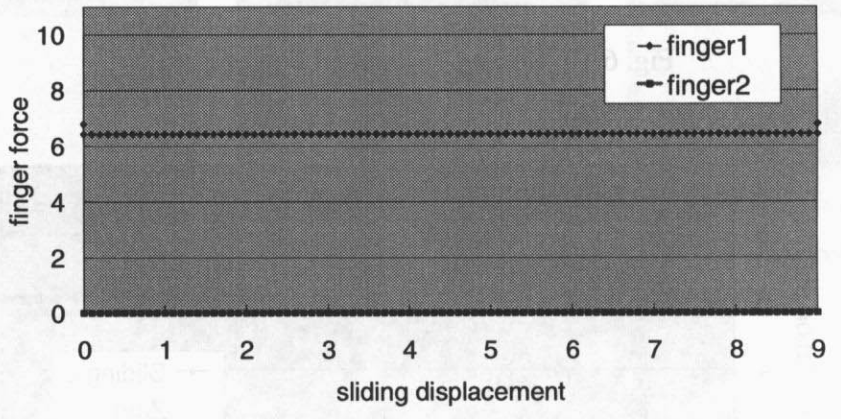


Fig. 6.22: Sliding 4: Planned Finger Forces

制御，指2を位置制御とするような動作が計画されている．生成された動作の指配置が対称的でないのは，操作のコストがまったく同じになる指配置が複数あり，その中の1つだけが探索によって返されているからである．

なお，この計画において展開されたノードの数は，1,335,201個のうち22,785個で，有効分岐数は1.3205，要した時間は504 CPU分であった．

6.7.2 転がし操作の計画

次に，ここでは平面上での直方体対象物の転がし操作 (Fig. 6.24) を考える．対象物を初期姿勢から目標姿勢まで倒す動作を実現するためのロボット指の動作を求める問題である．

滑らせ操作のときと同様，直方体の大きさは $5 \times 5 \times 10$，質量は1 ($M_o = 1$) で均質であるとする．コンフィギュレーション空間の離散化については，前項と同様，対象物上の各面

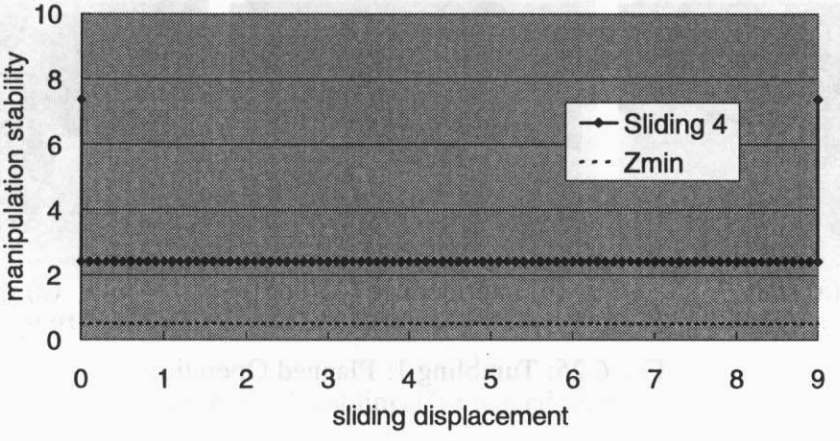


Fig. 6.23: Sliding 4: Manipulation Stability

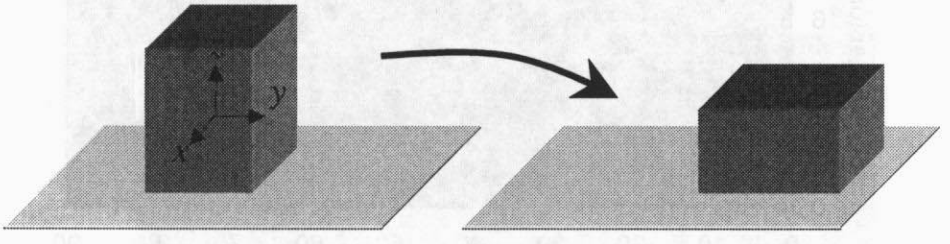


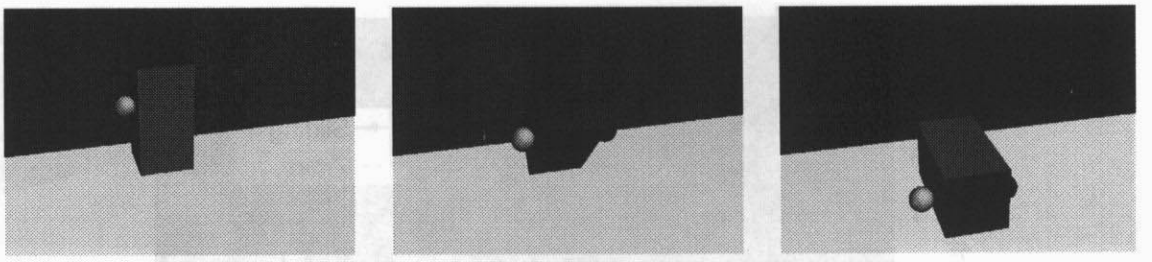
Fig. 6.24: Tumbling of a Cuboid

を 7×7 のグリッドに切ることによって，配置可能点を設定し，また，対象物の自由度についても同じく 31 段階に離散化した．

直方体の転がし操作 (1)

平面上で対象物を転がして倒す操作の計画を行った．対象物の背後だけに障害物がある場合（問題 Tumbling 1），Fig. 6.25 のように，両側から対象物を挟んで，持ち替えなしで転がす操作が生成された．このときの対象物上の指位置は，指 1 が $[2.5,\ 1.25,\ 1.25]^T$，指 2 が $[-2.5,\ 1.25,\ 1.25]^T$ である．また，各指の指令力を Fig. 6.26 に示す．この場合，片方の指を力制御，もう片方の指を位置制御とする結果が得られた．

なお，この計画において展開されたノードの数は，1,923 個で，有効分岐数は 1.2056，要した時間は 84 CPU 分であった．



(a) start (b) intermediate (45[deg]) (c) goal

Fig. 6.25: Tumbling 1: Planned Operation

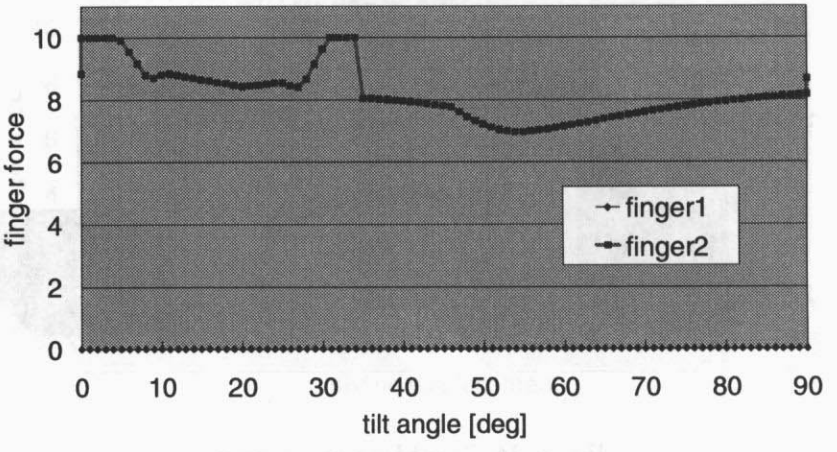


Fig. 6.26: Tumbling 1: Planned Finger Forces

一方，指力の上限を 5 と小さく設定して同じ操作を計画したところ（問題 Tumbling 1' とする），Fig. 6.27 のように，前の例よりも回転軸から遠いところをつまんで転がす操作が生成された．このときの対象物上の指位置は，指 1 が $[2.5,\ -1.25,\ 2.5]^T$，指 2 が $[-2.5,\ -1.25,\ 2.5]^T$ である．これは，発生可能な指力が小さくなったために，重力に対抗するようなモーメントを大きく取れるよう，指位置を変化させたものと考えられる．このとき，計画にかかった時間は約 245 CPU 分であり，探索中に展開されたノード数は 3,358，有効分岐数は 1.2312 であった．

なお，これらの例における操作の確実性の評価値の推移を Fig. 6.28 に示す．操作の確実性は，対象物を倒し終わる瞬間が最も低くなることが分かる．また，指力の制限が厳しい Tumbling 1' の方が，一部を除いて操作の確実性の値がより低くなっていることが見てとれる．