行動蓄積データを用いた多関節アームの卓上物体把持計画

Manipulator Trajectory Planning for Reaching an Object on a Table Using Accumulated Behavior Data

○守屋佑亮（信州大学）　　山崎公俊（信州大学）

Yusuke Moriya, Shinshu University　　11t1078h@shinshu-u.ac.jp
Kimitoshi Yamazaki, Shinshu University　　kyamazaki@shinshu-u.ac.jp

This paper describes trajectory planning for a manipulator to grasp an object placed on a table. The more obstacles are on the table, the more calculation costs are needed to find a course for avoiding the obstacles. To overcome this problem, we use accumulated behavior data, which is a set of planning results that succeeded reaching task under various situation. One behavior data is composed of obstacles arrangement and manipulation trajectory. The manipulator acts on the information that is decided by comparing data arrangement and present arrangement. For evaluating this planning, we performed experiments in different conditions on a simulator.

***Key Words*:** Basic Study, Support, Learning Data

1.　緒言

ロボットが室内で人間の生活支援を行う時，料理の配膳や食器の片付けなど，テーブル上で作業する機会が多々ある．これらの作業時，テーブル上に毎回同じ物体が同じ個数，同じ配置で置かれているとは限らない．そのような作業空間で目標となる物体を把持するためには，周囲にある物体との衝突を避けるような手先の経路とそれを実現するためのアームの関節角度列を毎回求めることが必要である．これらを求めるに当たって，アームの各リンクの形状と，周囲の物体の形状を取得し，新たな関節角度が与えられる度にそれらを用いた干渉チェックを行う必要があるが，リンクの数や障害物が増えるほど，また計画された関節角度列の量が多いほど，経路探索にかかる計算負荷は増大する．すなわち，物体の個数が多くなる可能性が高いテーブル上では，動作計画にかかる計算負荷が大きくなる．

こういった問題に対して，簡単な動作のまとまりを複数作成し，それらの組み合わせで一連の動作を行うことにより，計算量を軽減する研究が報告されている[1][2]．しかし，これらの動作計画は，動作の組み合わせを決める処理が必要であることから，計算量がまだ多いと予想される．

本研究の目的は，更なる計算負荷を軽減する方式を示すことである．アームの初期姿勢と手先の目標姿勢のみが与えられたとき，周囲の物体に衝突することなく目標手先姿勢を実現するようなアーム関節角度列を得る．このとき，過去の動作経験を用いることでアームと障害物との干渉チェックを省略する方式を提案する．

提案方式では，様々な障害物の配置パターンに対してあらかじめ動作計画を行った結果を利用する．障害物配置マップとそれを回避する関節角度列の 1 セットを行動データと呼び，それを大量に集めたものを行動蓄積データと呼ぶものとする．実際の動作計画では，現在の障害物配置が過去のどの行動データと似ているかを検索し，その検索結果に紐づけされた関節角度列を読み出すことで，衝突を起こさない動作を得る．すなわち，動作計画では関節角度列計算と干渉チェックを一切行わないため，障害物配置の検索処理のみに依存したアームの動作計画が可能になり，計算量の軽減が見込まれる．

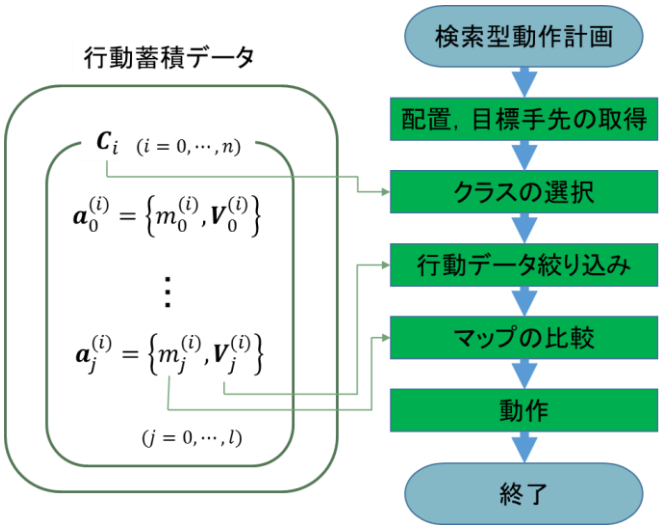


Fig.1 A flow of trajectory planning

2.　物体配置マップを用いたアームの動作計画

テーブル上にいくつかの物体が置かれている状況を想定する．そのうち一つを把持目標物とし，残りの物体を障害物とみなす．障害物にぶつからずに対象物へハンドを近づけ，把持をおこないたい．

図 1 に，本研究で提案する動作計画の流れを示す．図中の行動蓄積データは，行動データ $\boldsymbol{a}$ の集まりである．行動データは，テーブル上に任意に置かれた物体の配置マップ m と，その配置のもとで目標物を把持するために求められたアームの関節角度列 v の 2 つの要素からなる．目標物に複数の把持方法が存在する場合には，それぞれにおいて関節角度列を求め，それらをまとめて $\boldsymbol{V}$ とする．$m, \boldsymbol{V}$ の組を 1 つの行動データとし，様々な物体配置に対してこれを取得する．また，それぞれの関節角度列 v において，最終関節角度のアームの姿勢での手先姿勢を求め，その手先姿勢に関して $\boldsymbol{C}_i(0 \leqq i \leqq n)$ のクラスへの分類を行う．この分類は，動作計画において全ての行動データを検索の対象とするのではなく，あるクラス

のみに絞りこむことを可能とするためのものであり，検索時間を短縮する効果がある．

行動蓄積データを用いた動作計画を，検索型動作計画と呼ぶ．検索型動作計画は，以下のような手順で行う．

(1) テーブル上の物体配置（現在配置）に対するマップと，その環境での目標手先姿勢を取得する．
(2) 目標手先座標に近い行動データが含まれているクラス C_k を，行動蓄積データから選択する．
(3) クラス C_k 内の全ての v における最終手先姿勢を順に引用し，それと目標手先姿勢を比べ，姿勢がある程度似た行動データを選出する．
(4) (3)で選出した複数の行動データから，マップ m を順に引用し，それと現在配置マップを比較する．最も近いと判断されたマップ m_k を含む行動データ a_k を選出する．
(5) a_k に含まれる V_k 中で，最も目標手先座標に近い最終手先座標を持つ関節角度列 v_k を最終的な結果とする．

3. 行動蓄積データの取得手法

行動蓄積データは行動データの集合体である．行動データを構成する物体配置マップと関節角度列について，それぞれの取得手法を説明する．

3.1 物体配置マップ

テーブル上に置かれた複数の物体のうち，把持したい物体を目標物，それ以外の物体を障害物とみなし，物体配置マップを生成する．本稿では，マップはシミュレーションにより仮想的に生成するが，実環境を対象にした場合であっても，3次元距離画像センサなどで得たデータを用いれば，同様のマップを得ることは容易である．

物体配置マップの生成方法は次のようである．まず，テーブル天板を基準面として，物体が存在するテーブル上の空間を単位立方体に分割し，ボクセル空間を定義する（図 2(a)）．初期値としてすべてのボクセルに 0 を代入しておく．次に，障害物と重なるボクセルを探し，そこに 1 を代入する（図 2(b)）．こうして得られるボクセルの集合を物体配置マップと呼ぶ．

3.2 関節角度列

アームは多関節構造とし，各リンクや手先の幾何形状は既知であるとする．まず，物体配置マップを生成したのち，目標物の把持が可能な手先姿勢を決める．次に，動作計画法の 1 つである BiRRT 法[3]を利用し，最終手先姿勢に到達が可能かつ移動中に他の物体やアームのリンク同士の干渉がないような関節角度列を求める．ここで，目標物の把持方法を限定（例えば，マグカップであれば，取手のみを把持箇所にするなど）してしまうと，障害物との隣接関係やアームの可動域の関係から，把持を実現可能な物体配置が大きく限定される．そこで，ある物体配置での目標物に対して複数の把持方法を与え，それぞれに対応する関節角度列をまとめたものを，その時の物体配置マップに紐付けする．

4. 検索型動作計画

行動蓄積データを用いた動作計画の詳細を，2 章で説明した流れに従って示す．以下のそれぞれの節は，2 章の検索型動作計画の説明の(1)~(5)に対応している．

4.1 現在配置マップの取得

3 章で説明した物体配置マップと同様に，現在の物体配置についてもボクセルマップを取得する．これを現在配置マップと呼ぶ．また，目標物に対する目標手先姿勢を定める．

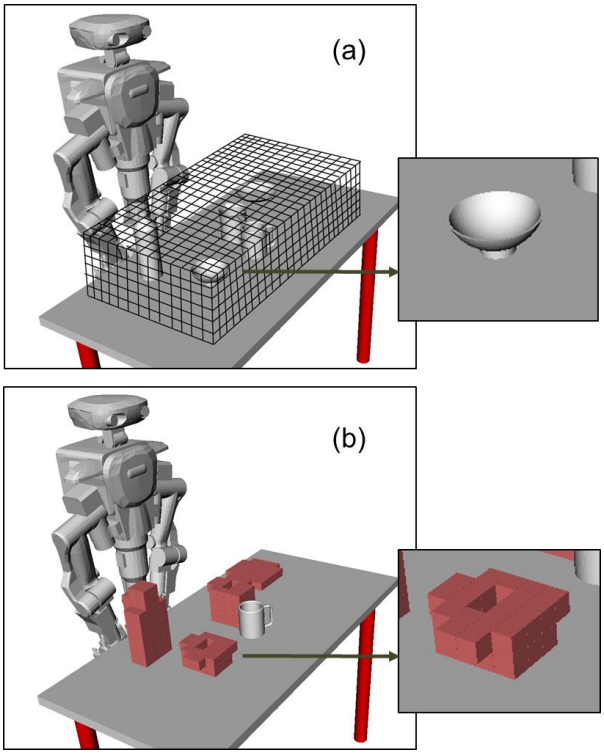


Fig.2 Generating object map

この後，行動データの物体配置マップと現在配置マップとの比較をおこなうため，マップの”近さ”の評価がよりばらつきやすくなるための処理をする．”近さ”の評価は，ボクセルに代入された値を使った類似度計算により行うが，0 と 1 のみでおこなう場合，類似度が同じになるマップが多数現れると考えられる．そこで，現在配置マップにマスクをかけてマップをぼかした表現に変更する．図 3 にマスクのイメージを示す．マスクは，マップと同じ大きさの 3×3×3 個のボクセルで構成される．中心の格子に近いほど大きな値を代入し，それぞれの値を$f_{xyz}(-1 \le x \le 1, -1 \le y \le 1, -1 \le z \le 1)$と表現する．

現在配置マップがマスクによってぼかされていく過程を，XY平面から見たイメージで図 4 に示す．適用前の現在配置マップのボクセルに，縦，横，高さ方向に対してX, Y, Zと番号を割り振った時，それぞれの値をR_{XYZ}とする．また，現在配置マップとボクセルの量，配置が同じである新たなマップを用意し，格子の値をR'_{XYZ}とする．初期状態では全ての格子において，$R'_{XYZ}|_{t=0} = 0$である．適用前のマップのある任意の格子に対し，$R_{XYZ} = 1$の時，式(1)を適用する．

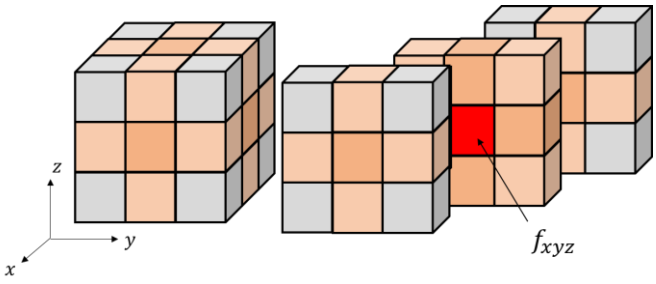


Fig.3 Smoothing mask

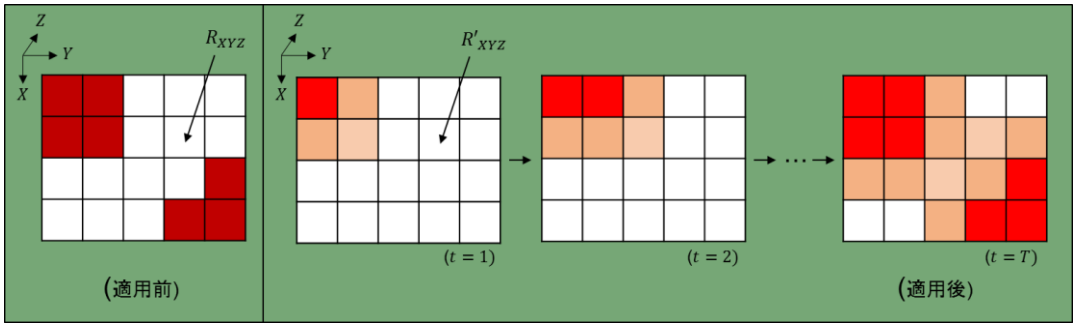


Fig.4 Smoothingprocedure

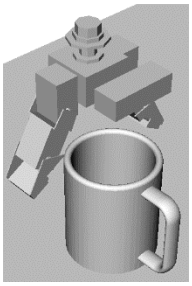


Fig.5 Method of grasping

$$R'_{opq}|_{t+1} = R'_{opq}|_t + f_{xyz} \quad (1)$$

ただし，o,p,qは式(2)~(4)で表せる番号である．o,p,qがとり得る全ての番号の組み合わせに対して式(1)を適用する．また，新たに用意したマップのその時の状態をtとした時，1 つの格子に関する適用後の状態を，式(1)では$t+1$と表現している．

$$o = X + x \quad (0 \le o \le X_{max}) \quad (2)$$
$$p = Y + y \quad (0 \le p \le Y_{max}) \quad (3)$$
$$q = Z + z \quad (0 \le q \le Z_{max}) \quad (4)$$

この処理を$R_{XYZ} = 1$である全ての格子の分だけ行い，最終的に図 4 の$t = T$のようなマップを得る．2 種類の肌色の格子が先に述べたぼかした箇所であり，赤のボクセルと比べて小さな値が代入されている．今後はこのマップを現在配置マップとし，検索型動作計画の処理で用いる．

4.2 類似クラスの選択

まず，クラス $\boldsymbol{C}$ の設定方法について説明する．テーブル天板に定義した平面を小さな格子に等分割し，それぞれの格子の中心座標を求める．ここで，行動データの関節角度列はアームの最終関節角度及び最終手先姿勢の情報を含んでいるので，それらより最終手先座標が求められる．最終手先座標とそれぞれの格子の中心座標の距離を計算し，最終手先座標が含まれる格子を求める．この処理を全ての行動データの関節角度列に対して行い，各格子に収まる最終手先座標をひとつのクラスに所属させる．関節角度列と物体配置マップは，その最終手先姿勢に紐づけておく．

類似クラスを選択する処理では，把持対象物から得た目標手先座標を得たのちに，検索対象とする行動蓄積データのクラス $\boldsymbol{C}_i$ を探す．クラスの決定は，クラス作成時同様，目標手先座標がどの格子に含まれるかを調べる事で行う．目標手先座標が含まれる格子のクラスを使用するクラスとし，それを類似クラスとする．

4.3 行動データの絞り込み

4.2 節で得た類似クラス内で更に絞り込みを行い，使用できる行動データを複数選出する．この処理では，手先座標だけでなく手先の向きにも着目して絞り込みを行う．行動データの関節角度列から順に最終手先姿勢を引用し，目標手先姿勢との相対距離と相対角度を算出する．あらかじめしきい値を設定しておき，算出された距離と角度の違いが所定の範囲内ならば，その最終手先姿勢に紐づけされた関節角度列を得る．その後，それらの関節角度列に関する行動データを再構築し，保存しておく．これを再構築行動データと呼ぶ．

4.4 マップの比較

4.3 節で再構成された全ての行動データについて物体配置マップの比較を行い，最も現在配置マップに類似したマップを選び出す．

再構築行動データから，物体配置マップを順に引用する．現在配置マップ同様，物体配置マップのボクセルにX, Y, Zと番号を割り振った時，それぞれのボクセルの値をD_{XYZ}と表すものとする．このとき，引用したマップに関する類似度Sを式(5)のように定義する．

$$S = \sum_{X,Y,Z} R_{XYZ} D_{XYZ} \quad (5)$$

この計算では，2 つのマップで同じ位置関係にあるボクセルが共に正の値を持てば，それが加算される．この類似度を求める処理を，すべての再構築行動データの物体配置マップに対して行う．そして，求まった類似度の集合$\boldsymbol{S} = \{S_1, S_2, \cdots, S_N\}$のうち最も値が大きくなった$S_k$のマップを，最も類似したマップとする．

4.5 動作

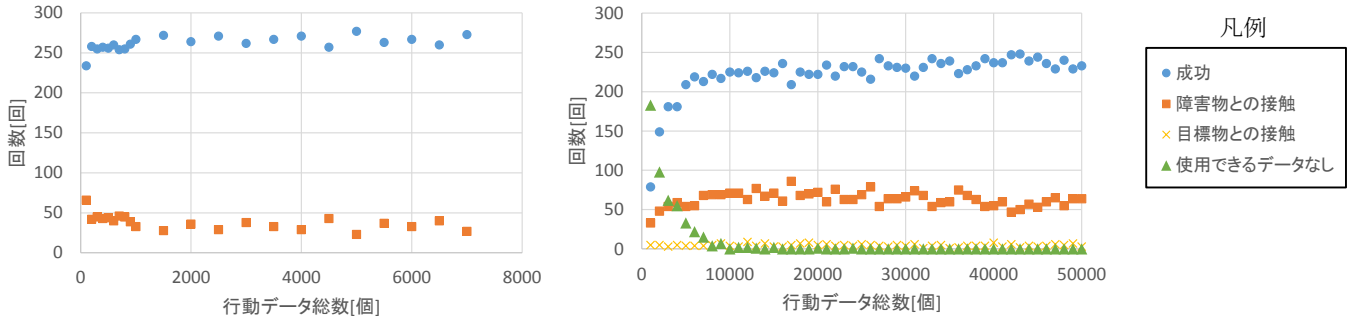
4.4 節で得られたマップが属している行動データを引用する．そのデータの最終手先座標が目標手先座標に最も近い関節角度列を呼び出す．その関節角度列中の関節角度を，アームが順に行う事で動作する．

5. 手法の検証

5.1 検証条件

シミュレーションによる検証をおこなった．多関節アームを有するロボットとして川田工業株式会社製の HIRO のモデルを用いた．左腕の 6 関節と腰軸の 1 関節を動作計画の対象とした．テーブル上に置く物体として，実在するマグカップ，ペットボトル，茶碗，ビーカー，皿を選定し，それらのＣＡＤモデルを作成した．

把持目標物に対する手先姿勢の選出と，BiRRT による軌道計画には，Openrave[4]を利用した．目標物をマグカップとして， Openrave の手先姿勢探索機能により図 5 のような目標手先姿勢を選出した．その他の物体はテーブル上にランダムに



①位置と方位固定　　②Y軸方向の位置のみ固定

Fig.6 Each Times of referring planning with two type limits

配置し，障害物として扱うこととした．

物体配置マップの生成では，ボクセル空間の対象領域の大きさを，テーブル天板中心を基準とし前後に 570mm，左右に 1080mm，上方向に 240mm とした．ボクセルの 1 辺の長さは 30mm とした．4.3 節で記述したしきい値は，相対距離が 3[mm]，相対角度が5[deg]とした．

シミュレーションは，①目標物の位置と方位を固定，②目標物の横方向（Y 軸方向）の位置のみを固定，の 2 種類の条件でおこなった．①に関しては，取得した行動データの関節角度列は図 5 の把持方法の 1 種類に対するもののみとし，②に関しては，現在配置で簡便な把持可否判定を行い，アームが図 5 の把持をできないと判定した場合，物体を置き直すこととした．

5.2 検証結果

図 6 は，検索型動作計画による動作を 300 回試行した時の成功回数・失敗回数と，行動データの数との関係を示したものである．検索結果として得られた動作で，目標物・障害物との干渉が無ければ成功とした．アームがそれらと干渉した場合や，4.3 節に述べた絞り込みによりデータ数が 0 となった場合は失敗とした．

条件①では，最大行動データ数 7000 個で検証を行った．目標物との接触と，使用できるデータ数が 0 となる事による失敗は無く，データ数が 100 個の時点ですでに成功率が 78%に達し，最終的に約 92%が最も高い成功率となった．

条件②では，50000 個の最大行動データ数で検証を行い，最大成功率が約 83%となった．行動データ数が少ない場合においては，使用できる行動データが 0 と判断される失敗の回数が多いが，総数が 25000 個以降は完全に 0 回となった．また，全ての総数での試行において，目標物と接触する失敗が数回見られた．50000 個のデータ総数では，1 回の成功ケースの関節角度列を得るために平均で約 0.652 秒の時間がかかったが，これは BiRRT 法の約 10 倍前後の早さである．

5.3 考察

①，②の両検証において一定の成功率が得られたが，まだ障害物との接触によって失敗する場合が見られる．その中には，図 5 の把持方法では明らかに実現不能な現在配置での失敗も含まれていた．よって，成功率向上のためには，複数の把持候補を与えておき，その現在配置に適した手先姿勢を評価してから，検索型動作計画を行うなどの方法が考えられる．

また，②の検証では目標物と接触してしまうケースがあった．これは，検索した関節角度列の最終手先姿勢が目標物と干渉しなくても，アームの移動中に接触してしまう場合があるためと考えられる．例えば，行動データの生成中，目標物に辛うじて干渉しない関節角度列を取得したとする．取得時の物体配置よりわずかに目標物がずれた現在配置が現れた時，その関節角度列で動作しようとすると，目標物と干渉する可能性が出てくる．この問題に関しても，上記の複数の把持候補を与える方法で解決できると考える．

6. 結言

動作計画の計算量を軽減する新たな手法を提案した．過去の物体配置と動作を紐付けしたデータを蓄積し，その中から現在の状況に近い配置を検索して動作する方法を述べた．また，その手法を用いた検証実験を行って性能を評価し，問題点を指摘して解決策を示した．

今後の課題として，5.3 節で述べたような方法を実装し動作の成功率を上げると共に，物体の個数を増加させた場合や，実空間で動作した場合の問題の発見，解決を行う必要がある．

文　献

[1] T. Sekiguchi, Y. Kobayashi, A. Shimizu and T. Kaneko, “Online learning of optimal robot behavior for object manipulation using mode switching”, Proc. of IEEE Int. Symposium on Robotic and Sensors Environments, pp61-66, Nov. 2012, Magdeburg, Germany

[2] 加藤央昌，” ロボットモーションプランニングの自動化に向けてのロボットモーション実行基盤の開発”，人工知能学会研究会資料，pp13～18

[3] J.J. Kuffner and S.M. LaValle, “RRT-connect: An efficient approach to single-query path planning”, Proc. of IEEE Int’l Conf. on Robotics and Automation, pp995-1001, April 2000, San Francisco, CA

[4] 出杏光 魯仙，”海外の動向 : ROS・OpenRAVE の新オープンソース開発環境が活かす知的マニピュレーション”，日本ロボット学会誌 Vol.28 No.5，pp585~588，2010

[5] 太田順，”知能ロボット入門”，コロナ社，pp34～39，2001