制約論理プログラミングによる衝突回避動作計画*

1N－4

溝口文雄‡　大和田勇人‡　長谷部正信†

東京理科大学 理工学部‡

1 はじめに

動作計画 (motion planning) は,物体を初期位置から目標位置まで移動させることを目的として行なわれる. その際に,物体の初期位置・初期姿勢・作業空間内の障害物が与えられる. 従来の研究では,把握物体の回転移動の問題は,Avnaim[1] や Lazano-Pérez[2] によって議論されたが,その物体の姿勢が限られているので,部分的にしか解決されていない. そのため,物体の大きさはハンドよりも小さいか,ハンドと同じ大きさに限られているものが多い.

現実的な問題において,三次元空間を対象にしてハンドよりも大きい把握物体も扱う必要がある. そこで本稿では,このような点を考慮した動作計画システムを提案する. 本システムは,経路導出モジュールと動作計画モジュールから実現されている. 前者は,障害物回避経路を生成する. 後者は,その経路に沿って把握物体を移動させる動作プランを生成する. これによって,ハンドよりも大きい把握物体も扱うことが可能である.

システムの実現には,5自由度垂直多関節型マニピュレータを対象にし,実数領域上の制約を扱え,バックトラック機能をもった制約論理型言語 CLP($\Re$) を用いる.

2 把握物体を考慮した動作計画

2.1 基本的な考え方

把握物体を考慮した動作計画を行なうために,その物体の形状と姿勢の変換を考えて,障害物回避経路を生成することが必要である. その両方を一緒に考えた障害物回避計画問題は,複雑である. そこで本研究では,初めに把握物体を考慮しない障害物回避経路を導出し,次にその経路に沿って,物体を移動させることを考える. もし,現在の姿勢で移動できない場合には,物体の姿勢を変えて移動させる. さらに,その経路で物体を移動することができない場合,別の経路を求めて,そこで再び物体を移動させることを考える. これを繰り返すことにより,移動できる動作を計画する. また,障害物回避経路を導出する際には,三次元空間における経路を容易に求めるために,作業環境をセル (立方体) で区切りモデル化して考える. 物体を移動させる際には,平行移動と回転移動を組み合わせて移動させる. 本システムは,経路導出モジュールと動作計画モジュールから実現する. 前者は,把握物体を考慮しない障害物回避経路を導出する. また後者は,障害物に衝突しないように物体を移動させるプランを生成する. 本システムの構成を図1に示す.

また本システムは,二つのモジュールを用いて,物体を移動させるプランを生成するだけでなく,それぞれのモジュールを独立したシステムとして扱うことができる. さらに,経路の入力から,その経路対する障害物の存在可能領域を求めることもできる. これらは,制約論理型言語の特徴のモジュール性と入出力の双方向性から実現されている.

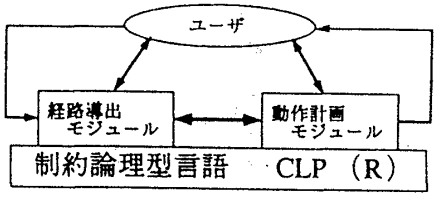

図 1: システム構成

従来は把握物体を考慮するために,その大きさにしたがって障害物が拡大され,把握物体の姿勢は固定されていた. そのため図2に示すような場合,自由空間 (経路になりえる空間) が障害物の拡大により障害物領域になり,実際には存在する回避経路が失われて,物体を移動することができなかった. しかし,本システムでは図2に示すように,物体の姿勢を変えることにより移動させることができる.

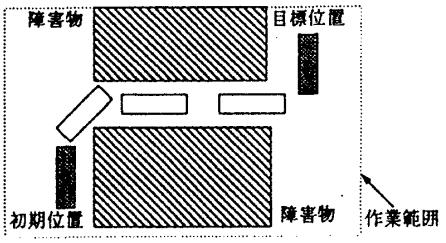

図 2: 問題例と移動の様子

2.2 経路導出プロセス

経路は,セルの重心座標を組み合わせて途切れることなくつないだものであると考える. また障害物との衝突は,経路のセルと障害物のセルの重心座標が一致した時に起こると考える. ゆえに,このプロセスでは以下表1のような制約を充足させることにより,障害物との衝突を回避した経路を生成する.

表 1 : 衝突回避のための幾何制約

	制約	意味
経路の導出	*obs_non_overlap*	経路のセルが障害物のセルと一致しない
	path_connect	経路のセルが途切れることなく結合している
	path_non_overlap	経路のセルが重複しない
ロボットの姿勢の表現	*link_connect*	リンクとリンクが決められた場所で連結している
	link_non_overlap	リンクとリンクが重ならない

制約は,以下のように制約論理プログラムで記述されている.

```
obs_non_overlap(_,[]).
obs_non_overlap(Obs,[Path|Ps]) :-
    non_overlap(Obs,Path)
    obs_non_overlap2(Obs,Ps).

non_overlap([],_).
non_overlap([p(Xo,Yo,Zo)|Obss],p(X,Y,Z)) :-
    abs(Xo-X)+abs(Yo-Y)+abs(Zo-Z) > 0, %幾何制約
    non_overlap(Obss,p(X,Y,Z)).
```

*The Collision Avoidance System using Constraint Logic Programing
†Fumio MIZOGUCHI, Hayato OHWADA,Masanobu HASEBE
‡Faculty of Sci. and Tech. Science University of Tokyo

その他の制約も同様な幾何制約で表現されている．ゆえに，以下に示すように問い合わせをすると，

```
?- first_conect(p(-3,6,0),Path),  % 初期位置
   last_conect(p(2,5,0),Path),    % 目標位置
   path_conect(Path),
   path_non_overlap(Path),
   obs(Obs),                      % 障害物の位置
   obs_non_overlap(Obs,Path),
   path_domain(Path).
```

次のような，経路のセルの重心座標のリストが導出される．

```
Path=[p(-3,6,0),p(-3,5,0),p(-2,5,0),
      p(-1,5,0),p(0,5,0),p(1,5,0),p(2,5,0)]
```

このように経路をつなぎ合わせて，障害物回避経路が求まる．

また，二つ以上のセルでモデル化される移動物体 (多角柱，円柱,etc) を扱えるようにするために中間ゴールを設定する．さらに，制約充足だけで解くと非効率的なのでヒューリスティックスなどを付け加えて探索を効率的にする．

2.3 動作計画プロセス

このプロセスでは，実際の作業空間を考える．把握物体の移動は，平行移動と回転移動である．基本的には，経路に沿って平行移動させる．もし，平行移動できなかった場合には，回転移動することにより物体の姿勢を変える．もしこれでもできない場合は，物体を障害物から離れる方向に平行移動しこの操作を繰り返す．このようなアルゴリズムで，これら二つの動作を組み合わせて，経路に沿って物体を移動させるプランを生成する．以下に示すようなプランが生成される．

```
Plan=[...,pa(p(50,250,100)),ro(0.31,p(50,250,100)),
   pa(p(62.5,250,100)),ro(0.18,p(62.5,250,100)),...]
                        pa(A)  …… A まで平行移動
                        ro(R,B) …… B で R だけ回転移動
```

障害物と物体の衝突のチェックは，それぞれを領域で表現し，共通領域の有無で判断する．もし共通領域が存在すれば，衝突を起こしていることになる．物体の領域を線形式で表現することができ，以下のように記述される．

```
space(X,Y,Z,List) :- make_space(X,Y,Z,0,0,0,0,List).

make_space(X,Y,Z,X,Y,Z,L,[]) :-
        L=1.
make_space(X,Y,Z,Xp,Yp,Zp,Lp,[p(X1,Y1,Z1)|Ps]) :-
        Xn=Xp+L*X1,
        Yn=Yp+L*Y1,
        Zn=Zp+L*Z1,
        L>=0,
        1>=L,
        Ln=L+Lp,
        make_space(X,Y,Z,Xn,Yn,Zn,Ln,Ps).
```

以下のように問い合わせると衝突の有無がわかる．

```
?-space(X,Y,Z,Obs),       % 障害物領域
  space(X,Y,Z,Obj).       % 物体領域
                  % Obs と Obj は頂点座標のリスト
yes : 衝突する or no :衝突しない
```

同時に，障害物・把握物体とマニピュレータの衝突チェックが行なわれる．

三次元空間においてマニピュレータと障害物との衝突をチェックするために，マニピュレータの姿勢を表現する必要がある．マニピュレータの各リンクを直方体にモデル化することにより，マニピュレータはこれらの直方体が連結したものと考える．ゆえに，表 1 のような制約を充足することによってマニピュレータの姿勢を表現する．以下に示すように，制約を充足させてリンクモデルの重心座標と回転ベクトルと頂点座標が求まる．

```
?-mp([0,160,371,-90,0],Rva,Rvb,Rvc), %リンク角の制約
  link_connect(Lista,Listb,Listc),
  link_non_overlap(Lista,Listb,Listc),
  link_conf(a,Lista,Ea,Rva),   %ショルダーの形状モデル
  link_conf(b,Listb,Eb,Rvb),   %エルボの形状モデル
  link_conf(c,Listc,Ec,Rvc),   %リストの形状モデル
  link_model(a0,List0).

Ea=p(0,0,425)
Rva=[1.57,0,0]
  ..........
List=[p(0,0,425),p(50,-50,550),p(50,50,550),...]
  ..........
```

この頂点座標から上述した衝突チェックを行なう．もしすべての経路点で障害物と把握物体に衝突しなかったら，その経路においてマニピュレータは動作可能と考える．

3 実験例

本システムを，SUN SPARC 10 の CLR (※) 上で実行させた．対象としたロボットは，5 自由度垂直多関節型マニピュレータの MOVEMASTER-EX である．図 3に示すような作業環境において，図 3中に示している直方体を目標位置，姿勢に移動させる．

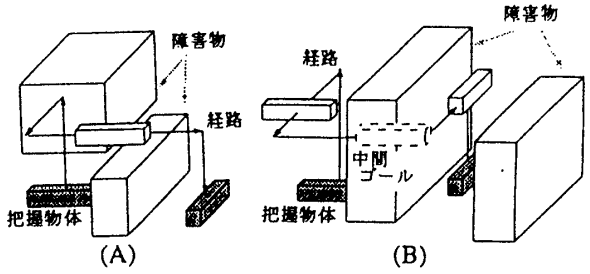

図 3: 問題の作業環境と経路

図 3の (A) のように障害物の上方を通ったり，図 3の (B) のように障害物の手前を通る動作経路が計画され，物体を実際に移動することができた．

4 まとめ

本稿は，ハンドより大きい把握物体を含めた物体の回転移動を考慮した動作計画システムについて述べた．これにより，このような物体も移動可能であり，従来求められなかった回避経路についても導出可能である．

制約論理プログラミングは，従来の論理プログラミングを拡張したものであるので，STRIPS などのＡＩプランナを扱うことができる．これを用いて，部品の形状や接続関係等の幾何学的制約処理を含むＡＩプランナが実現できる [3]．本システムは，制約プログラミングの枠組で実現されているので，このようなプランナとの融合が容易である．

参考文献

[1] F.Avnaim,J.D.Boissonnat,and B,Faverjon: A practical exact motion planning algorithm for objects amidst polygonal obstacles, *IEEE* Int.Conf.Robotics Automat.pp.1656-1661,1988

[2] Tomas Lozano-Perez: A simple motion planning algorithm for general robot manipulators, *AAAI'86*,pp.626-631,1986

[3] 谷口，大和田，溝口: 制約論理プログラミングによる作業レベルプランニング，日本ソフトウェア科学会第 10 回大会,C1-2,1993