関節空間における障害物回避パス探索の一方[†]

丁 全 鋼*・湯 浅 秀 男*・伊 藤 正 美*

An Approach to Collision-Free Motion Planning of Robot Manipulator

Quan Gang DING*, Hideo YUASA* and Masami ITO*

In this paper, an approach to plan collision-free motions for general robot manipulators based on the artificial potential function in joint space is proposed. The potential function is defined as the intersection of the obstacles and the path which is described by some sine functions in the joint space. The collision-free path is obtained by minimizing the value of this potential function using the steepest descent method. Since the obtained path is constructed by the sums of some sine functions, it is ensured to be smooth. The proposed algorithm is applied to the movement of the manipulators with two and three links and its effectiveness is shown by computer simulation.

Key Words: manipulator, potential function, collision-free motion planning, steepest descent method, joint space

1. はじめに

マニピュレータに動作を教示するには、ティーチングブレーバック方式を用いる場合が多い。この方法による動作の教示はオンラインで行う必要があるので、ティーチングするとき、生産ラインを止めなければならない。また、教示者は作業現場に入らなければならない。さらに、ロボットが突然暴走したりすると、とても危険である。そこで、ロボット言語を使うことによって、教示者が現場を離れて動作を教示する方法が開発されている。しかし、ロボットの近くに障害物があるとき、ロボットが障害物と衝突を起こさない動作を計画しなければならない。コンピュータシミュレーションによる対話的な教示法もあるが、多くは専用のロボットに

対象が限定されており、まだ多くの時間と労力を費やす。ロボットが大量に使われ、製品のモデルチェンジが頻繁に行われる現在、最小限の情報によって、複雑な作業環境下で自動的にロボットの動作を生成する一般的な方法が強く望まれている。

障害物の存在する環境でロボットの動作を自動的に生成することは動作計画、障害物回避パス（経路）計画などと呼ばれている。本論文では以後、パス計画と呼ぶ。パス計画はつぎのように定義できる。『マニピュレータの初期位置と姿勢、目標位置と姿勢、障害物とマニピュレータの幾何学的な情報がわかったとき、マニピュレータが障害物と衝突せずに、初期位置から目標位置まで動く経路を求める』。この問題に関しては、これまで多数の研究が行われているが、どれも一長一短で、決定的な方法はまだない。それらの方法は、主に空間分割法とポテンシャル法の2種類に分類できる。

空間分割法にはスライス法¹⁾、オクトツリー法²⁾、迷路法³⁾、自由空間分類表現法⁴⁾などがある。これらの方法はいずれもマニピュレータの動きうる空間（自由空間）を陽に求め、その空間でグラフ探索の手法を利用して、経路探索を行うものである。この方法にはさらに、作業空間を分割する方法と関節空間を分割する方法の2種類がある。作業空間を分割するとき、一般に分割された各キューブの属性（free, full, mixed）を決めるためにきめ細かな計算をしなければならない。経路探索するとき、各キューブにマニピュレータの一部が入っているかどうかチェックするのもかなり時間がかかる。これらの問題に関しては、ロボットや障害物を表わす幾何モデルを直方体や球、円柱など、なるべく単純なものに限定し、計算を簡単にすることにより解決がはかられている。また、関節空間を分割するときは各関節角の可動範囲を求めて、その範囲を適当な刻みで分ければよい。しかし、マニピュレータの自由度が増えるにつれ、メモリー量と計算量が指数的に

[†] 日本ロボット学会第5回学術講演会で発表（1987・11）

* 名古屋大学工学部 名古屋市千種区不老町

* Faculty of Engineering, Nagoya University, Nagoya

(Received March 9, 1988)

(Revised August 1, 1989)

増加してしまう。この問題の対応策はまだない。

一方、ポテンシャル法は、マニピュレータと障害物との距離に応じた反発力と目標位置までの距離に応じた引力との合力を生ずるような仮想的なポテンシャルを定義し、それにしたがって経路探索を行う方法である。この場合も作業空間で行う方法と関節空間で行う方法がある。作業空間は最大 3 次元であるが、ポテンシャル関数の定義と計算が煩雑である^{5), 6)}。それに対し、関節空間では点にポテンシャルを与えるだけで簡単に定義できる⁷⁾。しかしそのためには、作業空間にある障害物を関節空間に変換する必要がある。そして、それを考慮した上で、できる限り一般的な幾何モデルを用いて障害物、マニピュレータを表現しなければならない。この問題についても、まだ決定的な方法は存在していない。

本論文では、一般的で、簡単な障害物回避の一方法を提案する。関節空間でのポテンシャル法を用いるが、ポテンシャル関数は、マニピュレータの一つの位置姿勢ではなく、一つの経路に対して定義される。そして、一つの初期経路を与えた場合、ポテンシャルを小さくすることによって、障害物回避経路が求められる。干渉チェックは文献 8) で提案された方法を利用して逐次行うので、非常に少ないメモリですむ。

2. システムの記述

2.1 モデルのデータ構造

パス計画するためには、まずモデリングシステムによってマニピュレータと環境（障害物）の幾何モデルを記述しなければならない。パス計画のためのモデリングシステムには簡潔なモデルのデータ構造と記述の簡易さが要求される。パス計画のための干渉チェックの時間は幾何モデルのデータ構造の複雑さに大きく依存する。良いデータ構造を用いると経路探索の効率も向上する。幾何モデルを簡単に記述できなければ、作業環境が変化したとき、素早く対応できない。障害物とマニピュレータのリンクの形状を記述するには、既存のモデリングシステムを用いることが多いが、本論文では干渉チェック、コンピュータシミュレーションを行うために必要な最小限の情報だけを保持する。一般の専用のモデリングシステムはこのような基本的な情報をもっているの、簡単にインプリメントできる。

本論文では、障害物とマニピュレータの各リンクは凸多面体で近似する。凸でないときは、複数の凸多面体に分割して処理する。多面体の頂点を齊次座標で表現すると

$$\mathbf{v} = (x, y, z, 1)^t$$

となる。ここで、 x, y, z はデカルト座標値を、 t は転置を表す。多面体の面を齊次座標で表現すると

$$\mathbf{f} = (a, b, c, d)$$

$$\text{ただし、} a^2 + b^2 + c^2 = 1$$

となる。ここで (a, b, c) はその平面の外向き法線ベクトルであり、 d は、絶対値が原点とその平面の距離、符号が原点からその平面に下ろした垂線ベクトルと外向き法線ベクトルとの内積の反対の符号である。多面体の稜はその稜の二つの端点、または稜を挟む二つの面で表わせる。実際のプログラムでは、頂点と面の配列と、稜を定義するためのリンケージ配列だけをもっている。

2.2 座標変換

マニピュレータは n リンクからなっているとし、各リンクに Denavit らの表記法⁹⁾ にしたがって座標系を定義する。第 k リンク座標系からワールド座標系への変換行列は

$${}^kT = T_1(\theta_1)T_2(\theta_2) \cdots T_k(\theta_k) \quad (1)$$

$${}^kT^{-1} = T_k^{-1}(\theta_k) \cdots T_2^{-1}(\theta_2)T_1^{-1}(\theta_1) \quad (2)$$

となる。ここに、 $\theta_1, \theta_2, \dots, \theta_k$ は関節変数、 $T_1, T_2, \dots, T_k$ は各リンク座標系間の変換行列である。第 k リンク座標系で定義された任意の点と面を $\mathbf{v}_k$ と $\mathbf{f}_k$ とすると、その点と面はワールド座標ではつぎのように表わされる。

$$\mathbf{v}_0(\theta_1, \theta_2, \dots, \theta_k) = {}^kT \mathbf{v}_k \quad (3)$$

$$\mathbf{f}_0(\theta_1, \theta_2, \dots, \theta_k) = \mathbf{f}_k {}^kT^{-1} \quad (4)$$

ここで、 $\mathbf{v}_0, \mathbf{f}_0$ はそれぞれワールド座標系で表わされた点と面である。

2.3 作業空間と関節空間

作業空間とは障害物とマニピュレータが実際に存在する空間で、ワールド座標系がこの空間で定義される。そして、障害物の形状と位置はこの座標系で定義される。各リンクの形状はリンクごとに定義された座標系で定義される。関節空間とはマニピュレータの位置姿勢を一意に決める変数を成分とした空間のことで、マニピュレータはこの空間で形状と大きさ、関節の型によらず、点で表わされる。したがって、マニピュレータの動作計画が点の経路計画になり、問題はかなり簡単になる。しかし、関節空間での障害物は作業空間での障害物の位置と形だけではなく、マニピュレータの形、大きさにも関係がある。関節空間での障害物を領域として求めるには、マニピュレータの侵入禁止領域を求めればよい。可動領域（自由空間）を利用して探索を行うならば、その補空間を求めればよい。本論文で提案する方法は経路に対して陽な微分を利用

するので、障害物を関数で表わさなければならない。
この関数の求め方は、後で述べる。

関節空間での点は

$$\theta = (\theta_1, \theta_2, \dots, \theta_n) \quad (5)$$

で表わす。ここで $\theta_1, \theta_2, \dots, \theta_n$ は関節変数である。この点 θ は作業空間でのマニピュレータの一つの位置姿勢と対応している。マニピュレータと障害物が接触するときの点 $(\theta_1, \theta_2, \dots, \theta_n)$ は関節空間での障害物を囲む一つの符号付きの超曲面の上にある。この超曲面はマニピュレータと障害物との接触状態によって切り換わる。マニピュレータと障害物が凸多面体で表わされているので、接触状態は頂点と面、稜と稜の2種類に大きくまとめることができる。したがって、超曲面関数 $F(\theta)$ は具体的につぎのような2種類の関数からなる。

頂点と面の接触の場合：点 v が平面 f 上にあるとき、 $f \cdot v = 0$ が成立することを利用すれば第 k リンクの頂点 v_k が障害物の面 f_0 の上にあるとき、超曲面関数は

$$F(\theta) = f_0^k T(\theta) v_k = 0 \quad (6)$$

となり、リンクの面 f_k が障害物の頂点 v_0 をとおるとき、超曲面関数は

$$F(\theta) = f_k^k T^{-1}(\theta) v_0 = 0 \quad (7)$$

となる。

稜と稜の接触の場合：リンクの稜を v_{k1}, v_{k2} 、障害物の稜を f_{01}, f_{02} で表わすと、リンクの稜と障害物の稜が交わるとき、超曲面関数は

$$\begin{aligned} F(\theta) = & (f_{01}^k T(\theta) v_{k1}) \times (f_{02}^k T(\theta) v_{k2}) \\ & - (f_{01}^k T(\theta) v_{k2}) \times (f_{02}^k T(\theta) v_{k1}) \\ = & 0 \end{aligned} \quad (8)$$

となる。

障害物とリンクの凸性より、関節空間での障害物は、これらの超曲面に囲まれた領域となる。したがって、第 i 番目の障害物の j 番目の超曲面を F_{ij} で表わすと、関節空間での障害物がつぎの式で表わせる。

$$\left\{ \theta : \bigcup_{i=1}^b \bigcap_{j=1}^{m_i} F_{ij}(\theta) \leq 0 \right\} \quad (9)$$

ここに、 b は障害物の個数で、 m_i は i 番目の障害物を囲む超曲面の個数である。

3. 干渉チェック

パス探索するには、マニピュレータと障害物との干渉チェックをしなければならない。探索法に合せた干渉チェック法もあるが、独立の問題としてもいろいろな干渉チェック法が提案されている^{10)~13)}。本論文では文献8)で提案した方法を用いる。この方法は同じ

座標空間にある二つの任意の凸多面体が干渉、接触、または離れているかをチェックすることができる。とくに、二つの凸多面体が接触するとき、接触関係つまり、どの頂点とどの面が接触しているか、どの稜とどの稜が接触しているかがわかる。これらの情報は陽な微分を行うために必要なので、本方法に対しては不可欠なものである。今までに用いてきた凸多面体を表わすデータ（頂点、面、つながりぐあい）をまとめて p と記す。すると、二つの凸多面体 p_1, p_2 が干渉しない必要十分条件はつぎのように与えられる⁸⁾

$$q(p_1, p_2) > 0 \quad (10)$$

ここで、 q はいつも (6), (7), (8) 式で定義される $F(\theta)$ のうちのどれかである。(10) 式が0になるとき、 p_1, p_2 が接触することになる。干渉チェックを行うとともにその時点の面、稜、頂点のナンバーを保存すれば、接触するとき、どの面とどの頂点、またはどの稜とどの稜が干渉するかがわかる。そして、(10) 式を0にすることによって、そのときのリンクの拘束条件がわかる。すべてのリンクと障害物に対して、このようなことを行うと、現在の位置のマニピュレータが障害物とどんな位置関係で接触するかがわかる。

4. 経路の定義

関節空間での経路をつぎのように定義する

$$\begin{aligned} \theta(t) = & S(1-t) + Et + \sum_{i=1}^m A_i \sin i\pi t \\ \theta, E, S, A_1, \dots, A_m \in & R^n \\ 0 \leq & t \leq 1 \end{aligned} \quad (11)$$

ここで、 S は初期点、 E は目標点、 $A_1, \dots, A_m$ が経路を決定する変数で、 $\theta(t)$ は経路を表わす関数である。このように定義した理由は $\sin i\pi t$ が任意回微分可能

Fig. 1 Obstacles in joint space and paths determined by sine functions

であるので、微分することによって角速度、角加速度が簡単に求められるからである。\$m\$ が大きいほど複雑な経路を記述することができるが、\$m=3\$ 程度で通常の経路を表わすのには十分であり、むやみに未知数 \$(\mathbf{A}_1, \dots, \mathbf{A}_m)\$ を多くする必要はない。\$\mathbf{A}_1, \dots, \mathbf{A}_m\$ を変えることによって、いろいろな経路が得られる。経路を決めるには、\$\sin\$ 関数にこだわる必要はないが、\$C^2\$ 級連続な関数が望ましい。経路を表わす関数を微分することが必要なので、少なくとも \$C^1\$ 級連続でなければならない。Fig. 1 は 2 リンクのときに関節空間へ変換された障害物と、リンクを表わす点がたどる経路を示した例である。

5. ポテンシャル関数の定義

Fig. 1 に示すように、ある \$\mathbf{A}_1, \dots, \mathbf{A}_m\$ によって一つの経路が決まる。マニピュレータが初期位置と目標位置にあるとき、障害物とは干渉していないことが前提なので、マニピュレータと障害物が干渉すると、必ず接近の接触点と脱離の接触点対になって存在する。接触点は経路上にあるから、\$t\$ の値で表わせる。\$2m\$ 個の接触点 \$t_1 < t_2 < \dots < t_{2m-1} < t_{2m}\$ があるとすると、経路の \$t_{2k-1}\$ から \$t_{2k}\$ (\$0 < k < m\$) までの部分が障害物のなかに入っている。そこで、ポテンシャル関数 \$L\$ をつぎのように定める。

$$L = \sum_{k=1}^m (t_{2k} - t_{2k-1}) \geq 0 \quad (12)$$

この関数の値が大きいと、障害物に入っている部分が多く、値が小さいと障害物に入っている部分が少ないことを意味する。だから、この \$L\$ を 0 にすることによって、障害物を回避するパスが見つけれられる。

6. 経路探索アルゴリズム

与えられた一組の \$\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3\$ によって決まった経路に対して、一定の刻み \$\Delta t\$ ごとに \$t\$ の値を取って (10) 式の \$q\$ の値を調べる、\$q\$ の符号が変わるところに接触点がある。\$q\$ の符号が変わる時点での頂点、面または稜のナンバーを記憶しておけば、勾配を求めるために必要な超曲面関数 \$F(\theta)\$ がどれかがわかる。求められた \$k\$ 番目の接触点 \$t_k\$ はつぎの式を満たす。

$$\theta(t_k) = \mathbf{S}(1 - t_k) + \mathbf{E}t_k + \sum_{i=1}^m \mathbf{A}_i \sin i\pi t_k \quad (13)$$

$$F(\theta(t_k)) = 0 \quad (14)$$

ここで、\$t_k\$ も変数と考えて (13)、(14) 式を微分するとつぎのようになる

$$\delta\theta = \sum_{i=1}^m \frac{\partial\theta}{\partial\mathbf{A}_i} \cdot \delta\mathbf{A}_i + \frac{\partial\theta}{\partial t_k} \cdot \delta t_k = f(\delta\mathbf{A}_1, \dots, \delta\mathbf{A}_m, \delta t_k) \quad (15)$$

$$\frac{\partial F(\theta)}{\partial\theta} \delta\theta = 0 \quad (16)$$

(15)、(16) 式より

$$\frac{\partial F(\theta)}{\partial\theta} f(\delta\mathbf{A}_1, \dots, \delta\mathbf{A}_m, \delta t_k) = 0 \quad (17)$$

この式により、\$t_k\$ の \$\mathbf{A}_1, \dots, \mathbf{A}_m\$ による偏微分係数が求められる。

(12) 式を微分すると

$$\delta L = \sum_{k=1}^m (\delta t_{2k} - \delta t_{2k-1}) \quad (18)$$

であり、(17)、(18) 式により、\$\partial L / \partial \mathbf{A}_1, \dots, \partial L / \partial \mathbf{A}_m\$ が求められる。

つぎの経路を決める変数 \$(\mathbf{A}_1, \dots, \mathbf{A}_m)\$ はつぎの式により求まる

$$\mathbf{A}_j^{(n+1)} = \mathbf{A}_j^{(n)} + \alpha \frac{\partial L}{\partial \mathbf{A}_j^{(n)}} \quad (19)$$

$$j = 1, \dots, m$$

(a) Joint space

(b) Work space

Fig. 2 A collision-free path in case of 2 link manipulator

以上のステップを干渉しなくなるまで繰り返すと、障害物回避パスが求められる。

以上に提案したアルゴリズムを実際のコンピュータにインプリメントし、シミュレーションを行った。以下では、すべて $m=3$ としている。Fig. 2 は 2 リンクの場合で、障害物とマニピュレータは多角形で近似されている。Fig. 2(a) では得られた障害物回避パスを示しており、Fig. 2(b) ではマニピュレータが(a)で得られた経路に沿っていくことを示してある。

Fig. 3 は 500 型 Puma ロボットのハンドを除いた部分の構造を参照して作ったモデルを用いて行った例である。関節の型は三つとも回転関節である。マニピュレータの初期位置 $S=(0.30, -0.40, 0.40)^t$ 、目標位置 $E=(-1.50, -1.50, 0.40)^t$ 、初期経路 $A_1=(0.00, -0.30, 1.20)^t$ 、 $A_2=(0.00, 0.00, 0.00)^t$ 、 $A_3=(0.00, -0.15, 0.50)^t$ 、 $\Delta t=0.001$ 、 $\alpha=0.20$ 。繰り返した回数は 11 回。費やした CPU 計算時間は 31.68 秒だった。使用した言語は Fortran、計算機は名古屋大学大

型計算機センター FACOM・M-280 大型計算機である。

Δt 、 α を決めるにはいくつかの要素を考慮しなければならない。 Δt は計算の精度に関わり、 α は繰返し回数、つまり、経路探索時間に関係がある。また経路と障害物との交点を求めるにはもっと効率のよい方法がある^{14), 15)}。これまでのシミュレーション結果からは、障害物が少ない場合、両方とも大きめに取ると、障害物が多い場合、 Δt と α を小さめにすれば有効な結果が得られている。 $A_1, \dots, A_m$ ははじめに 0 にして、経路が見つからなければ、実空間のマニピュレータの動きを参考にして $A_1, \dots, A_m$ を適当に決め、経路探索をやり直せばよい。

7. おわりに

本論文では、関節空間でのポテンシャル法を用いて、障害物回避パス探索を行う一方法を提案した。任意回微分可能な $\sin$ 関数を用いて経路を定義している

ので、滑らかな経路が求められる。そして、経路を表わす関数を直接微分することによって、各リンクの角速度、角加速度を求めることができる。経路を表わす関数からもわかるように、本論文のアルゴリズムは一般の多自由度マニピュレータにも容易に拡張できる。もちろん、微分を導くには複雑になり、探索時間も長くなる。干渉チェックは各ステップごとに行われているので、ほかの方法に比べ必要とするメモリー量が非常に少ない。モデルを記述するためのデータ構造が簡単のため、探索時間はそれほど長くなかった。しかし、本論文で提案した方法はポテンシャルを利用しているので、多峰性の問題が生じる。つまり不適当な初期値を与えると、ポテンシャル L が 0 ではない極小点に入り込んでしまい障害物回避経路が得られない。この問題の解決には、良い初期値を与えればよいのだが、実際の問題においては障害物を回避するマニピュレータのほしい動きは予想できる場合がほとんどであるので、その情報を用いれば良い初期値を得ることはそんなに難しいことではないと思われる。また障害物が非常に複雑に入り組んでいて、狭い迷路を通り抜けるような場合にはポテンシャル L が 0 となる経路をみつけれないこともある。この場合は、(11)式における経路を定義するパラメータの数 (m) を増やすことにより、より細か

Fig. 3 Simulation of a 3-link manipulator

い経路も発見できるようになる。この m の値は、空間分割法による探索の場合の、空間分割の細かさに対応するものである。またははじめから障害物回避経路が存在しない場合は、当然 m をいくら増やしてもポテンシャル L が 0 となる最小値は存在しない。しかし通常のロボットのパス探索では、このような極端に困難な場合は考えなくてもよいであろう。普通、 $m=3$ 程度でほとんどの場合が解決できると思われる。

最後に、貴重なご助言をいただいた細江教授、菅沼助教授に感謝の意を表する。

参 考 文 献

- 1) T. Lozano-Perez: A Simple Motion-Planning Algorithm for General Robot Manipulators, IEEE Journal of Robotics and Automation, **3**-3, 224/238 (1987)
- 2) 登尾, ほか: オクトツリーを用いた高速干渉チェック法, 日本ロボット学会誌, **5**-3, 189/198 (1987)
- 3) 近藤, 木村: 迷路法に基づくマニピュレータの障害物回避法, 日本ロボット学会誌, **5**-4, 263/272 (1987)
- 4) 長谷川 勉: 自由区間分類表現法によるマニピュレータの衝突回避動作の計画, 計測自動制御学会論文集, **22**-6, 616/622 (1986)
- 5) O. Khatib: Real-Time Obstacle Avoidance for Manipulators and Mobile Robots, Int. J. Robotics Research, **5**-1, 90/98 (1986)
- 6) 比留川, 北村: 安全第一アルゴリズムとポテンシャル関数に基づくマニピュレータの障害物回避法, 日本ロボット学会誌, **5**-3, 171/179 (1987)
- 7) 奥富, 森: ポテンシャル場を用いたロボットの動作決定, 日本ロボット学会誌, **1**-3, 226/232 (1983)
- 8) 湯浅, 伊藤: マニピュレータの障害物回避動作計画を考慮した凸多面体の干渉チェック法, 日本ロボット学会誌, **5**-4, 291/296 (1987)
- 9) Richard P. Paul 著, 吉川恒夫訳: ロボット・マニピュレータ, コロナ社 (1984)
- 10) J.W. Boyse: Interference Detection Among Solids and Surfaces, Communication of the ACM, **22**-1, 3/9 (1979)
- 11) 重末, ほか: 3-D 形状モデル間干渉チェック問題の一解法, 精密機械, **49**-11, 89/94 (1983)
- 12) 水垣, ほか: 幾何モデルによるロボットの簡易干渉チェック, 精密機械, **50**-6, 30/36 (1984)
- 13) 小沢, ほか: オフラインロボット教示における高速干渉チェックの一方式, 日本ロボット学会誌, **4**-2, 79/88 (1986)
- 14) B. Jones, et al.: Root Isolation for Transcendental Equations, The Computer Journal, **27**-2, 184/187 (1984)
- 15) D.B. Popovski: A Note on King's Method for Finding a Bracketed Root, Computing, **29**-4, 355/359 (1982)