オクト・ツリーによるパス・プランニング

4W-1

藤坂正昭[1]　藤代一成[2]　國井利泰[3]

1 日立ＳＫＳ（株）　2 筑波大学電子情報工学系　3 東京大学理学部情報科学科

1．はじめに

現在の商品の生産形態は、大量生産から多品種小量生産へと移ってきている。現在、主に使われている工業用ロボットは、教示による動作指定の必要なプレイバック方式のロボットであるため製品が変わるたびに繰り返し教示しなければならない。このようなことから、教示時間の短縮が要求されている。

現在の教示方式では、ロボットアームの先端を作業空間のある地点からある地点まで移動させる際に、その運動の軌道上に障害物がある場合には、教示者がそれを避けて通るように教示を行わなければならない。そこで、計算機上でパス・プラニングを自動的に行い、この軌道を求めることができれば、教示時間の短縮が可能になる。

本論文では、作業空間上の障害物をオクト・ツリーにより表現し、障害物を含まない八隅子の中心座標をスタートからゴールまで結んだパスの中から最短距離のパスを発見するパス・プラニングの方法を提案する。

オクト・ツリーを用いることにより、周辺に障害物の無い部分空間は大きな八隅子で、また、近くに障害物の存在する部分空間は小さな八隅子で表現される。よって、障害物を含まない八隅子の中心を結んで生成されたパスでは、周辺を障害物に囲まれた危険な箇所はパスの点の間隔の細かいパスになり、それ以外の箇所はパスの点の間隔の広いパスとなる。すなわち、オクト・ツリーを用いることにより、空間格子法による間隔の均一なパスと比較して、パスの通る八隅子の大きさによりパスの間隔の変化する、適応性のあるパスを作ることが可能になる。

2．作業空間の表現方法

作業空間の表現方法として、まず空間格子法[1]を取り上げてみる。空間格子法は作業空間に仮想的な座標を設定し、各軸に平行にかつ等間隔に空間を区切り、同じ大きさの立方体に分割するものである。この立方体の中から障害物を含む立方体を選び出して障害物を表現する。しかし、この方法により作業空間を分割した場合、近似値を上げようとすると大量の記憶容量が要求される。そこで、作業空間を効率的に表現する手法が必要である。本論文では、この要求を満たすためにオクト・ツリー[1]の概念を導入した。

オクト・ツリーは、作業空間の入る立方体を考え、それで障害物の形状の近似が十分でなければ、各軸方向にその立方体を２等分するプロセスを再帰的に行うものである。したがって、小さくなった立方体が障害物の中にすべて含まれている部分、あるいはその立方体が完全に障害物の外にある部分については再分割を必要としないので、冗長な表現を避けられ、空間格子法に比較して記憶容量の大幅な増大を防ぐことができる。

図1.1、図1.2に説明の簡単化のため、オクト・ツリーを２次元化したクァード・ツリーの例を示した。斜線の部分が障害物である。

Path Planning by Oct-tree
Masaaki FUJISAKA[1],Issei FUJISHIRO[2],Tosiyasu L. KUNII[3]
1 HITACHI SKS Co.Ltd ,2 The University of Tsukuba
3 The University of Tokyo

3．問題の設定

本論文では以下の条件を仮定する。

仮定１．
作業空間を移動するものは、ロボットのアーム全体ではなく、ロボットアームの先端のみとする。したがって、作業空間を移動するものは動点とみなせる。

仮定２．
作業空間は立方体であると仮定する。もし、立方体でない場合はその作業空間を含むような立方体を考え、作業空間以外の場所は障害物として扱う。

仮定３．
障害物の位置の入力は、ロボットに画像認識装置を付加し、画像処理の技術を応用して自動的に検出する方法も開発されているが、ここでは問題を簡単化し、障害物を直方体に限り障害物の位置の仮想的な座標の座標値を入力する方法をとる。

仮定４．
スタートとゴールの座標を含むノードをそれぞれスタート・ノード、ゴール・ノードと呼ぶ。ここで、スタート・ノードとゴール・ノードは一致してはならない。もし、２点間を往復する場合は、往路と復路に分割して考えることにより、一般性は失わなくてすむ。

4．３次元パス・プラニング

障害物を２節で述べたようにオクト・ツリーにより表現する。オクト・ツリーの各ノードは、作業空間を分割した八隅子に相当する。各八隅子の中心座標をそのノードの座標とする。ある八隅子の中心座標と隣の八隅子の中心座標を結んだパスを「プリミテイブ・パス・セグメント」と呼ぶ。動点はこのプリミテイブ・パス・セグメント上を移動するものとする。障害物のない八隅子の中心を結んでいったプリミテイブ・パス・セグメントから構成される連続したパスが、障害物に衝突しないパスである。このパスの中から一番長さの短いものを最短パスと定義する。

最短パスを捜し出すために、スタート・ノードを根とする探索木を作成する。探索木の各ノードは障害物を表現したオクト・ツリーの八隅子に相当する。探索木のノードからノードへの枝はプリミテイブ・パス・セグメントである。スタート・ノードを探索木の根に入れ、スタート・ノードの八隅子の隣の八隅子をスタート・ノードの子供として探索木に入れ、再び、その子供の各々について、隣の八隅子を求め、そのノードの子供とする。こうした木の拡張をゴール・ノードが現れるまで行なう。一番最初に現れたゴール・ノードから探索木を逆方向にスタート・ノードまでたどっていったパスが最短パスである。以下にパス探索を行なうために必要な「探索木の探索アルゴリズム」[2]と「隣接ノード発見アルゴリズム」[3]について述べる。

4．1　探索木の探索アルゴリズム

木の探索方法はいくつか考えられている。代表的なものに幅優先探索、深さ優先探索があるが、幅優先探索や深さ優先

探索のように、ゴールの位置に関係なく探索を行なうと、探索木全体を探索することになり、探索時間や記憶容量が膨大になる可能性がある。そこで本論文ではヒューリスティック探索を用いた。ヒューリスティック探索は、探索木の各ノードに評価値を与え、拡張の候補のノードの中で評価値の小さいノードから探索木を拡張していくものである。評価値は拡張の候補ノードのうち、どのノードがゴール・ノードへの最適のパス上にあるかを決めるものである。例えば、あるノードnの評価関数f(n)は探索木におけるノードnの深さg(n)とノードnとゴール・ノードとの状態差w(n)の和で表現される。:　　f(n) = g(n) + w(n).

本論文では、スタート・ノードから現在のノードに至るまでの距離とゴールまでの直線距離の和を評価値とした。ここで、距離はノードの座標から簡単に求められる。

4．2　隣接ノード発見アルゴリズム

本論文では隣接ノードが障害物を含むノードであるかどうかにより、その方向のパスが障害物に衝突しないパスであるかを判定するため、八偶子の各面に接している6方向の八偶子を隣接した八偶子と考える。この6方向の隣接した八隅子を求めるには、障害物を表現したオクト・ツリーのノードの6方向の隣接ノードを捜し出す必要がある。

そこで、本論文では2次元のクァード・ツリーのための隣接ノード発見アルゴリズム[3]をオクト・ツリーに拡張したものを用いた。

この隣接ノード発見アルゴリズムは、ノードのある方向の隣接ノードを求め、そのノードのアドレスを返すものである。その方向に八隅子が存在しなければNULLを返す。

いま、あるノードnのd方向に隣接するノードを求めたいとする。このアルゴリズムでは、まずノードnを始点として、nとそのd方向の隣接ノードの共通の先祖にたどり着くまでツリーをさかのぼる。次に、そのノードから今度はツリーを下にたどっていくのであるが、その際、先にたどったパスと、ノードnのd方向の面に対して対称なパスをたどることにする。このようにしてたどり着いたノードが求める隣接ノードである。

5．パス探索アルゴリズム

以下に、4節で述べた2つのアルゴリズムを統合した「パス探索アルゴリズム」を示す:

入力:　スタートとゴールの座標、障害物の座標
出力:　最短パスの点の座標
アルゴリズム:
ステップ1：スタート・ノードをリストopenに入れる。
ステップ2：リストopenの先頭のノードを取り出し探索木に入れる。このノードをPとする。
ステップ3：if（Pがゴール）
　　then　Pから探索木をスタート・ノードまでたどり、そのパスのノードの座標を順に出力する。
ステップ4：for each 方向(N,E,S,W,U,D)
　　begin
　　4-1：隣接ノード発見アルゴリズムにより隣接ノードを捜す。
　　4-2：if（隣接ノードがnullでない　or　隣接ノードが障害物を含むノードでない）
　　then　隣接ノードをリストopenに証価値の小さい順に入れる。
　　end
ステップ5：go to　ステップ2

図2に図1にパス探索アルゴリズムを適応して得られた探索木を示す。図2の探索木の各ノードの左上の数字は拡張の順番を示し、斜字体の数値は評価値である。

7．まとめと今後の課題

作業空間をオクト・ツリーを用いて表現することにより、作業空間内でパス・プラニングを行うアルゴルズムを提案した。本アルゴリズムの特長は次のとおりである。:

（1）八隅子の中心を結ぶ線分列によってパスを定義するので、パスの点の間隔は作業空間の広い部分では広く、障害物に近く狭い空間では細かくなる。このため、周辺の空間の広さに応じてパスの点の間隔が変化するような、効率的なパス・プラニングが可能になる。

（2）パスの点の間隔の長さに対応してロボットの動作速度を決定することにより、危険度に応じたロボットの動作速度を設定することができる。

（3）オクト・ツリーで表現可能な範囲内においては完全な3次元のパス・プラニングを行うことができる。

（4）空間格子法を用いるよりも記憶容量が大幅に節約できる。

今後の課題として、本論文では空間を移動する対象を点に限定しているが、実際のロボットに応用するためには、移動する対象をロボットのアーム全体とする必要がある。すなわち、移動する対象をいくつかの自由度を持つ直方体としてパス・プラニングを行う必要がある。

参考文献

[1]　國井、藤代、ソフトウエア工学ハンドブック、第15章、オーム社、1985.

[2]　N.J.Nilsson、「人工知能　問題解決のシステム論」、コロナ社、pp.46-88,1973.

[3]　Hanan Samet,"Neighbor Finding Techniques for Images Represented by Quadtrees",Computer Graphics and Image processing ,vol.18,pp.37-57,1982.

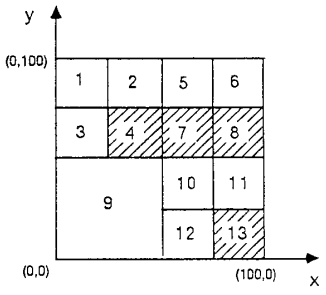

図1.1 二次元作業空間

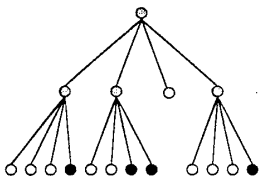

図1.2 クァドツリー表現

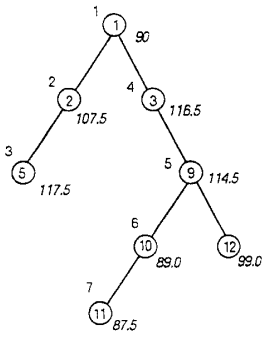

図2 探索木