Journal of the Korean Institute
of Industrial Engineers
Vol.22, No.4, December, 1996

다관절 로보트를 위한 충돌 회피 경로 계획

Collision-Free Path Planning for Articulated Robots

최진섭* · 김동원**

Jin-Seob Choi* · Dong-Won Kim**

Abstract

The purpose of this paper is to develop a method of Collision-Free Path Planning (CFPP) for an articulated robot. First, the configuration of the robot is built by a set of robot joint angles derived from robot inverse kinematics. The joint space, that is made of the joint angle set, forms a Configuration space (Cspace). Obstacles in the robot workcell are also transformed into the Cobstacles using slice projection method. Actually the Cobstacles means the configurations of the robot causing collision with obstacles. Secondly, a connected graph, a kind of roadmap, is constructed by the free configurations in the Cspace, where the free configurations are randomly sampled from a free Cspace immune from the collision. Thirdly, robot paths are optimally determinant in the connected graph. A path searching algorithm based on A* is employed in determining the paths. Finally, the whole procedures for the CFPP method are shown for a proper articulated robot as an illustrative example.

1. 서 론

로보트의 작업 데이터의 생성을 위해서 산업현장에서는 재생 방법(teach and play method)이 주로 이용되고 있으나, 최근 컴퓨터를 이용하여 작업 데이터를 미리 생성하는 오프-라인 프로그래밍(Off-Line Programming : OLP)이 활발히 연구되고 있다. OLP에 의

* 동신대학교 산업공학과
** 전북대학교 산업공학과

한 데이터의 생성은 현장 작업에 영향을 미치지 않고 사전에 데이터를 준비하고 검증할 수 있다. 또한 복잡한 계산을 간편화하고 로보트 기종에 관계없이 데이터의 생성이 가능하고 기존의 CAD/CAM 시스템과의 연결이 가능한 장점이 있다. 그러나 OLP를 이용한 원활한 데이터의 산출을 위해서는 몇 가지 해결해야 할 난제들이 있으며 그 중의 하나가 장애물과의 충돌이 없는 안전한 작업 경로를 생성하는 문제(Collision-Free Path Planning : CFPP)이다. 이중 다관절 로보트(articulated robot)는 현업에서 많이 이용되고 있는 형태의 로보트로서 OLP를 위한 CFPP가 더욱 요구된다고 할 수 있다. 따라서 본 연구에서는 다관절 로보트를 위한 효과적인 CFPP를 다루고자 한다.

관절이 없이 움직이는 로보트(moving robot)를 위한 CFPP는 로보트를 점으로 가정하고, 대신 장애물의 크기를 확장시키는 형상공간(configuration space : Cspace)을 많이 이용한다. 이 방법은 경로계획시 로보트의 부피를 고려하지 않아도 되는 장점이 있다. 한편 다관절 로보트의 경우에는 직교좌표 공간을 형상공간으로 이용할 수 없으므로 일반적으로 관절공간(joint space)을 형상공간으로 이용한다[1]. 이 경우 로보트는 한 점으로 표현되고, 또한 가능한 관절 범위(allowable joint range) 내에서만 표현되기 때문에 역기구학(inverse kinematics)해의 존재여부를 고려하지 않아도 되는 장점이 있다[1, 2]. 이때 장애물들은 관절공간에서 새로운 형태의 영역(Cobstacle)으로 표현된다. 따라서 CFPP는 첫째, 관절공간에서 장애물들을 찾는 과정과 둘째, 장애물들을 제외한 자유공간(free space)에서 로보트의 경로를 찾는 과정으로 요약 될 수 있다[1, 3].

관절 공간에서의 장애물들을 찾는 방법에는 슬라이스 투영(slice projection)방법[4]과 역기구학 방법[5]이 있다. 슬라이스 투영 방법이란 n차원 형상공간을 n-1차원 슬라이스 투영들의 집합으로 표현하는 방법이다. 이 방법은 알고리즘이 간단하고 관절이 많은 로보트의 해석에 적합하나 많은 계산시간과 기억용량을 필요로 하는 단점이있다. 역기구학 방법이란 장애물과 로보트가 접촉이 일어날 수 있는 모든 가능성을 로보트의 역기구학 식을 이용하여 파악하는 방법이다. 역기구학 방법은 슬라이스 투영 방법에 비해 계산시간이 단축될 수 있으나 관절 수가 증가함에 따라 적용이 어려운 단점이 있다.

한편 자유공간에서 로보트의 경로를 찾는 방법에는 기존의 Cell Decomposition, Potential Field, Visibility Graph, Supervisory Control등의 방법이 있다. 그러나 본 연구에서는 자유공간에서 임의로 추출한 형상(configuration)들을 연결하여 그래프를 구성하고, 그 그래프상에서 A* 알고리즘[3]을 적용하여 충돌회피 경로를 구하고자 한다. 이 방법은 기존의

Table 1. Comparison of CFPP algorithm

Cspace Representation	Slice Projection	[4][본연구]
	Inverse Kinematics	[5]
Path Planning in Free Space	Network Search	[6][본연구]
	Cell Decomposition	[3]
	Visibility Graph	[7]
	Supervisory Control	[8]
	Potential Field	[5]

방법들에 비해 관절이 많은 로보트의 해석에 적합한 장점이 있다. 본 연구와 기존의 방법들과의 비교가 Table 1에 있다.

따라서 본 연구에서는 다관절 로보트의 CFPP를 3단계로 나누어 2장에서는 슬라이스 투영방법에 의하여 관절공간에서 장애물을 생성하는 과정을, 3장에서는 장애물을 제외한 자유공간에서 탐색 그래프를 생성하는 과정을, 4장에서는 그래프 상에서 최적경로를 생성하는 과정을 설명하였다. 그리고 5장에서는 적절한 예를 택하여 시뮬레이션을 수행하였다.

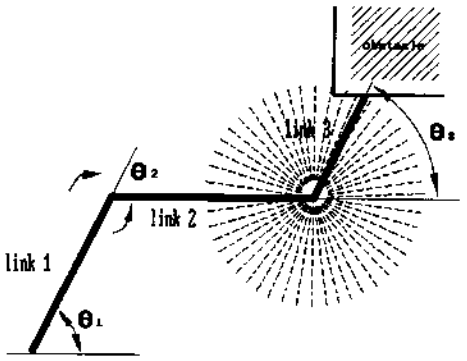


Figure 1. Slice projection method

2. 슬라이스 투영 방법을 이용한 관절 공간에서의 장애물 생성

N개의 관절을 갖는 로보트의 슬라이스 투영에 의한 관절 공간에서의 장애물 생성 방법은 다음과 같다. 우선 첫번째 관절(θ_1)의 허용범위내에서 일정한 간격으로 움직이면서 링크 1과 장애물이 충돌하는가를 체크한다. 충돌하는 경우는 더 이상의 탐색을 할 필요가 없고, 충돌하지 않는 경우에는 두번째 관절(θ_2)에 대하여 일정한 간격으로 움직이면서 링크 2와 장애물의 충돌여부를 추적한다. 위와 같은 과정을 θn에 이를때까지 반복한다. Figure 1에 특정한θ_1 및 θ_2에 대하여 θ_3를 변화시키면서 충돌여부를 판단하는 과정이 나타나 있다.

로보트와 장애물의 정확한 충돌 검출을 위해서는 로보트의 기구학적 구조(kinematics structure)뿐만 아니라 로보트의 기하학적 형상(geometric shape)도 고려되어야 한다. 이를 위하여 많은 연구들이 로보트 링크및 장애물을 다면체로 가정하고 다면체들간의 충돌을 검출하였다. 다면체들간의 충돌 형태는 크게 장애물의 꼭지점과 링크 면간의충돌(type A), 링크의 꼭지점과 장애물의 면간의 충돌(type B), 장애물의 모서리(edge)와 링크의 모서리간의 충돌(type C)의 3가지 형태가 있다[1, 4]. 이와같은 다면체간의 충돌 검색은 많은 계산을 필요로 하며 별도의 충돌검색 모듈로 개발되는 것이 바람직하다. 본 연구에서는 충돌 검색을 간략화하기 위하여 로보트를 두께가 없는 구조로 가정하고 대신 장애물을 확대시키는 방법을 이용하였다. 이때 장애물의 오프셋(offset)량은 로보트 링크의 기구학적 기준 위치선으로부터 가장 먼 거리에 있는 링크상의 점까지의 거리이다.

이와같은 슬라이스 투영 방법을 이용하여 관절공간에서 장애물을 생성하는 알고리즘은 다음과 같다. 각 관절 및 링크에 대하여 알고리즘이 같으므로 함수를 재귀적으로 호출하여 이용하였다.

```
PROCEDURE Generate_Cobstacles
                ( θ[i], obstacles[i], link[i] ;
                Var Cobstacles)
BEGIN
 FOR θ := θ[i].min TO θ[i].max STEP
       θ[i].interval DO
  IF (Collision_detection (link[i], obstacles[i] ;
     Var obstacles[i+1] )=='YES')
    RETURN
  ELSE
    Generate_Cobstacles( θ[i+1], obstacles[i+
    1], link[i+1] ; Var Cobstacles);
  ENDIF
END
```

Cobstacle을 저장하기 위한 자료구조로 배열을 이용할 경우 구조가간단한 장점이 있으나 관절의 개수가 많고 충돌 체크를 위한 관절간격이 좁아질 경우 많은 기억용량을 필요로 한다. 따라서 Figure 2와 같은 링크드 리스트(linked list)를 이용하였으며 이와같은 노드의 표현을 위하여 다음과 같은 자기참조 구조체(self-referential structure)를 이용하였다.

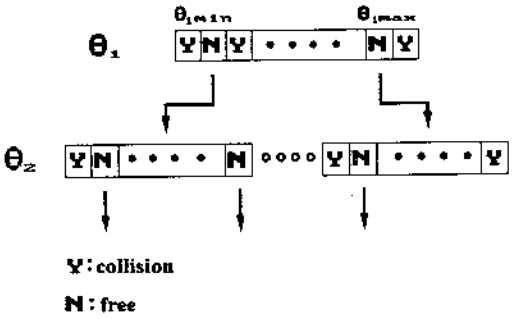


Figure 2. Data structure for Cobstacles

```
structure joint_node{
  char collision ; /*collision detection index*/
  structure joint_node *sub_joint_node[number_
  of_section];
}
```

이와같은 알고리즘을 3축 로보트에 적용하여 관절공간에서 장애물들을 표현한 예가 Figure 3에 나타나 있다. 그림에서 검게 형성된 부분이 장애물이고 그 외의 지역이 자유공간이다.

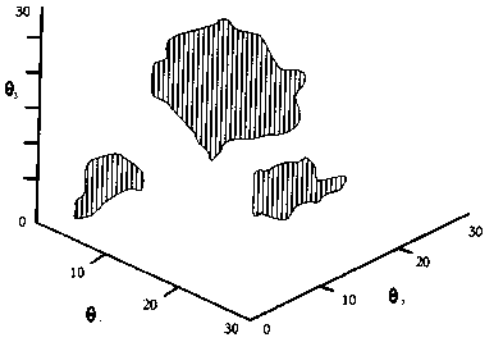


Figure 3. Example of Cobstacles

3. 자유공간에서의 탐색 그래프 생성

관절공간에서 장애물들이 표현되면 장애물들을 배제한 자유공간에서 로보트의 임의의 형상들을 표현하는 노드(node)들을 추출한다. 그리고 추출된 노드들과 로보트의 초기자세를 표현하는 시작노드와 목표자세를 표현하는 목표노드를 포함하여 그래프를 생성한다.

먼저 자유공간에서 노드를 추출하는 방법은 다음과 같다. 관절영역에서 임의로 노드를 추출하여 장애물과의 비교를 통해서 충돌

검색을 한다. 추출노드가 장애물이 점유한 지역에 포함되는 경우는 이 노드를 제외시킨다. 추출 노드의 개수는 안전한 경로 생성 보장과 탐색시간의 측면을 동시에 고려하여 결정한다[6]. Figure 4에 3차원인 경우 자유공간에서 추출된 임의의 노드들의 예가 나타나 있다.

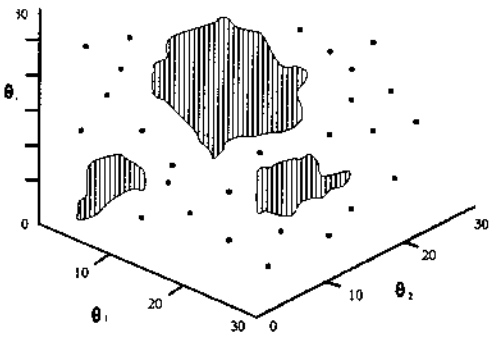


Figure 4. Example of random sampling in free space

자유공간에서 노드들이 추출되었으면 이들을 연결하여 그래프를 형성한다. 그래프를 형성하기 위하여 각각의 노드들에 대하여 근접 노드들과 연결을 시도한다. 만일 연결된 edge가 장애물을 통과하는 경우는 그 edge를 제외시킨다. 추출된 노드의 개수 및 연결된 근접노드의 개수의 선택에 따라서, 연결되지 않는 그래프(unconnected graph)가 형성될 수도 있으나 본 연구에서는 고려하지 않기로 한다. 이상의 자유공간에서의 노드의 추출 및 그래프의 생성을 위한 알고리즘은 다음과 같다.

```
PROCEDURE Generate_graph(Cobstacles ;
                         Var network)
BEGIN
 REPEAT
   temporary_node:=Random_sampling( );
   IF (Detect_collision
      (temporary_node, Cobstacles)='NO')
      nodes[i]:=temporary_node;
      i=i+1;
 UNTIL(i=number_of_node)
 FOR i:=1 TO number_of_node DO
  FOR j:=1 TO number_of_node DO
    IF (i<>j) distance[j]:=Calculate_distance
       (nodes[i], nodes[j]);
  Select_neighbor_nodes
    (distance, Var neighbors);
  Connect_edge
    (nodes[i], neighbors ; Var network);
END
```

3차원인 경우 자유공간에서 형성된 그래프의 예가 Figure 5에 나타나 있다.

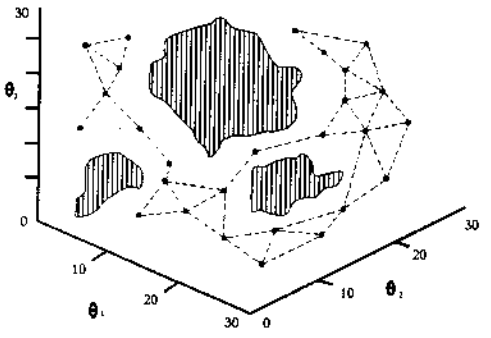


Figure 5. Example of graph in free space

4. 최적 경로 생성

자유공간에서 그래프가 생성되면 그래프상의 초기노드에서 목표노드까지 최적 경로

를 추적한다. 이를 위해 A* 알고리즘을 이용하였다. A* 알고리즘은 해의 생성이 확실하며, 평가 함수를 이용하여 목표점으로 수렴이 빠른 장점이 있다. 그러나 탐색점의 수가 증가함에 따라 탐색 시간과 저장 공간 측면에서 비효율적인 단점도 있다[3].

그래프상에서 최적 경로의 생성을 위하여 임의의 노드 n마다 근접 노드들과의 평가치를 부여하였다. 근접노드 n'의 평가치 f(n')는 다음과 같이 구한다.

$$f(n') = g(n') + h(n')$$

여기에서 g(n')은 노드 n에서 노드 n'까지의 평가치이며 h(n')는 노드 n'으로부터 목표 노드까지의 평가치이다. 본 논문에서는 평가치의 기준으로서 노드간의 직선거리를 이용하였다. 초기노드에서부터 f(n')값이 가장 작은 근접노드로 이동하여 목표노드가 발견될 때까지 경로를 추적하는 알고리즘을 요약하면 다음과 같다.

```
PROCEDURE Search_optimal_path(network
                ; Var path)
BEGIN
 current_node:=start_node;
 REPEAT
  FOR i:=1 TO number_of_neighbor DO
    f(i):=Evaluate_cost (current_node,i);
  min_neighbor:=Find_min_neighbor (f);
  current_node:=min_neighbor;
 UNTIL current_node=goal_node
END
```

Figure 6에 그래프상에서 구한 충돌회피 경로의 예가 나타나 있다. 이 관절공간상의 노드들에 대하여 정기구학식(direct kinematics)을 이용하여 직교좌표 공간에서의 충돌회피 경로를 얻을 수 있다.

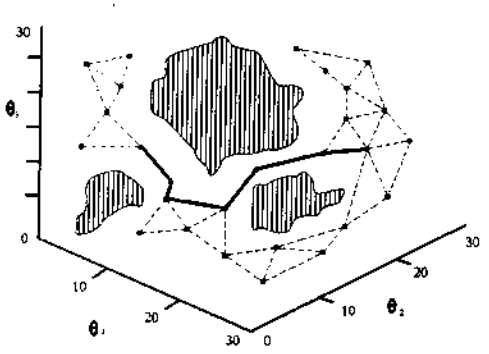


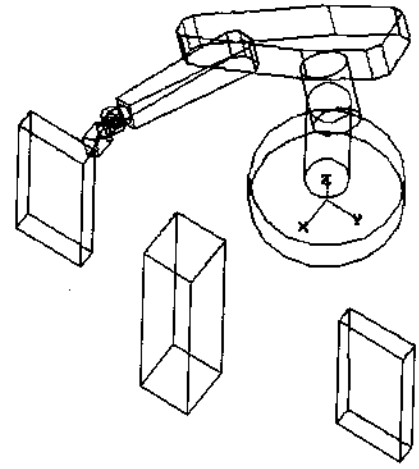
Figure 6. Example of collision_free path

이상의 알고리즘들을 모두 포함하여, 다관절 로보트의 충돌회피를 위한 전체적인 알고리즘을 서술하면 다음과 같다.

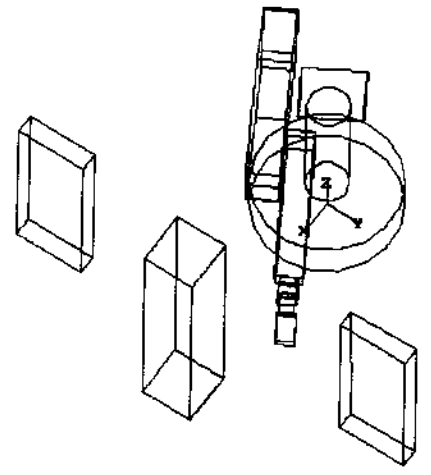
```
PROCEDURE CFPP_for_articulated_robot
                (obstacles, robot kinematics
                structure ; Var cartesian_path)
BEGIN
  Generate_Cobstacles
    ( θ , obstacles, link ; Var Cobstacles)
  Generate_graph(Cobstacles ; Var network);
  Search_optimal_path
    (network ; Var joint_path);
  Direct_kinematics
    (joint_path ; Var cartesian_path);
END
```

5. 적용 예

본 연구에서 제시한 알고리즘의 타당성을 검증하기 위하여 6축 PUMA로보트를 이용한 시뮬레이션이 수행되었다. 슬라이스 투영방법을 이용하여 6차원의 관절공간을 생성하고 충돌회피 경로를 구하는 작업은 많은 계산시간을 필요로 한다. 한편 대부분의 로보트 작업은 초기위치 및 최종위치에서만 정교한 손목운동을 필요로 하고 중간 경로에서는 정교한 손목운동을 요구하지 않는다[4]. 따라서 시뮬레이션의 간략화를 위하여 로보트 손목부분의 관절은 각각 θ_4=0°, θ_5=-30°, θ_6=0°로 고정시키고 $\theta_1 \sim \theta_3$만을 변수로 하였다. 그러나 엔드이펙터의 부피는 고려되었다. 충돌회피 경로를 위한 계산은 PC에서 C-언어를 사용하여 구현되었다. Figure 7은 시뮬레이션을 위한 로보트 작업장을 나타내고 있다. 로보트 작업장은 6축 PUMA 로보트와 3개의 장애물로 구성되어 있다. Figure 7의 (a)는 작업 초기위치를, (b)는 작업 목표위치를 나타내고 있다.



(a) start posture



(b) goal posture

Figure 7. Robot workcell for simulation

슬라이스 투영 방법을 이용하여 $\theta_1 \sim \theta_3$의 3차원 관절영역에서 형성된 장애물이 Figure 8에 초기위치 및 목표위치와 함께 나타나 있다. 3차원 관절영역에서의 그래픽 표현은 Mathematica 패키지를 이용하였다. 그림에서 q1은 waist(θ_1)을, q2는 shoulder(θ_2)를, q3는 elbow(θ_3)를 각각 표현한다. 각 관절의 슬라이스 간격은 6° 를 이용하였다. 즉 Figure 8에서 장애물의 단위요소를 표현하는 입방체의 각변의 길이는 6° 이다. 자유공간에서의 노드의 추출 개수는 500개를 사용하였다. Figure 9에 장애물과 함께 표현된 자유 형상을 표현하는 노드들이 시뮬레이션되어 있다.

자유 형상노드들의 그래프 형성을 위하여 연결된 근접 노드의 개수는 5개를 이용하였다. 두 노드를 연결하는 edge가 장애물에 충

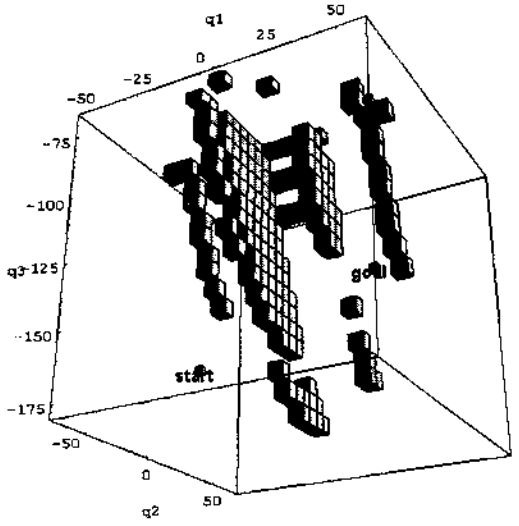


Figure 8. Simulation of Cobstacles

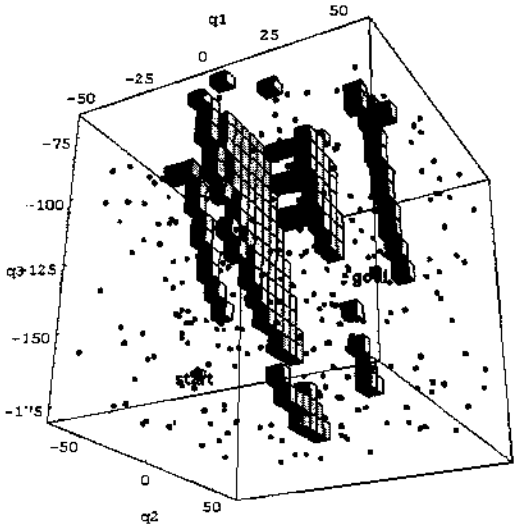


Figure 9. Simulation of free configuration

듦하는 지의 여부는 다음과 같이 판별하였다. 장애물을 표현하는 각각의 입방체의 중심으로부터 그 edge까지의 최단거리가 입방체의 중심으로부터 입방체 꼭지점까지의 거리인 $(\sqrt{3}/2)\times 6°$ 보다 작으면 그 단위입체에 충돌한다고 가정하였다. A*알고리즘을 이용하여 관절공간에서 최종적으로 생성된 경로의 시뮬레이션이 Figure 10에 나타나 있다.

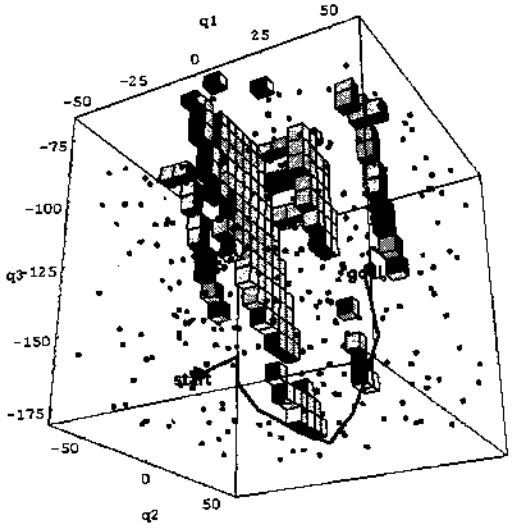


Figure 10. Simulation of collision-free path in joint space

이와같이 관절공간에서 생성된 충돌회피 경로는 PUMA 로보트의 정기구학식을 이용하여 작업공간에서의 충돌회피 경로로 변환되었다. 작업공간에서 수행된 충돌회피 경로의 그래픽 시뮬레이션이 Figure 11에 나타나 있다. Figure 11에서 충돌회피 경로는 초기위치 (a)에서 시작하여 중간점들인 (b), (c), (d)를 지나 목표점인 (e)에 도달한다. (a)~(e)에서 각각 왼쪽그림은 측면도를, 오른쪽 그림은 평면도를 보여주고 있다.

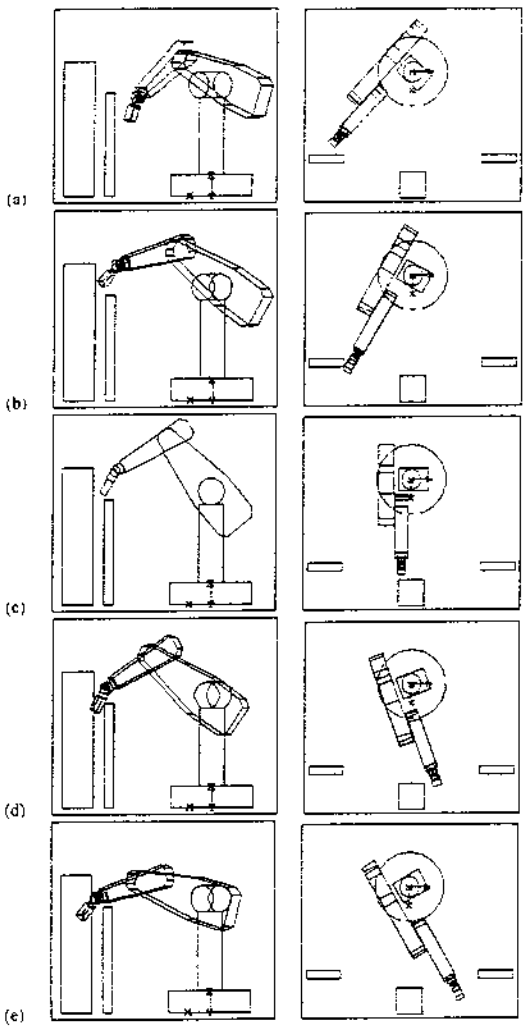


Figure 11. Simulation of collision-free path in Cartesian space

6. 결 론

본 연구에서는 다관절 로보트의 충돌회피 경로계획을 위하여 관절공간에서 장애물을 형성시키고 장애물을 제외한 자유공간에서 탐색 그래프를 이용하였다. 관절공간에서의 장애물 형성을 위하여는 알고리즘이 간단하고 다관절에 적합한 슬라이스 투영 방법을 이용하였으며, 자유공간상의 그래프에서는 A* 알고리즘을 이용하여 안전하면서도 빠른 충돌회피 경로를 구하였다. 그리고 적절한 예제에 대하여 그래픽 시뮬레이션을 수행하였다.

본 연구에서는 충돌검색시 로보트 링크의 형상을 고려하기 위하여 장애물의 크기를 오프셋시키는 방법을 이용하였다. 보다 정확한 충돌회피 경로의 계산을 위해서는 로보트 링크의 기하학적 형상이 정확히 고려되어야 한다. 또한 자유공간상의 그래프 생성시 연결되지 않는 그래프가 발생될 경우에 대한 고려가 요구된다. 또한 초기위치 및 목표위치 근처에서의 3차원 이상의 관절공간에서의 경로계획이 요구된다.

참 고 문 헌

[1] J. C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, 1993.

[2] 김정훈, 최진섭, 강희용, 김동원, 양성모, "형상공간을 이용한 다관절 로보트의 충돌 회피 경로 계획," 한국자동차공학회 논문집 제 2권 제 6호, pp. 57~65, 1994.

[3] C. W. Warren, "Fast Path Planning using Modified A* Method," Proceeding IEEE International Conference and Automation, pp.668~673, 1993.

[4] T. Lozano-Perez, "A Simple Motion-Planning Algorithm for General Robot Manipulators," IEEE Journal of Robotics and Automation, Vol. RA-3, no.3, june 1987.

[5] C. W. Warren, J. C. Danos and B. W. Mooring, "An Approach to Manipulator Path Planning," The International Journal of Robotics Research, Vol.8, no.5, pp. 87-95, 1989.

[6] Horsch, T. and Schwarz, F., "Motion Planning with Many Degree of Freedom - Random Reflections at C-space Obstacles," Proceeding IEEE International Conference on Robotics and Automation, Vol.3, pp. 3318-3323, 1994.

[7] N.S.V.Rao, S.S.Iyengar and G. de Saussure, "The Visit Problem : Visibility Graph-Based Solution", Proceeding IEEE International Conference on Robotics and Automation, pp.1650-1655, 1988.

[8] 박종현, "감독자 제어를 이용한 로봇의 장애물 피해가기," 대한기계학회논문집 제 17권, 제 11호, pp.2782-2789,1993.

96년 2월 최초 접수, 96년 10월 최종 수정