A Novel Hierarchical Collision Free Path Planning Algorithm for Robots

행 정

This paper describes a novel hierarchical local search algorithm to effectively find out the collision-free path for mobile robots and robot manipulators in a known environment. Specifically, to avoid the computational complexity and the curse of dimensionality which are usually required by graph search methods, a hierarchical search algorithm using local information in configuration space is proposed. Firstly, a searching is performed to find out a tunnel including start cell and goal cell not with basic cells but with big cells which are made up of several basic cells. The tunnel is found by connecting the current big cell with neighboring big cells which not only minimize the ratio of number of obstacle cells to the size of big cell as well as the distance between current big cell and goal big cell, but also maximize the distance between current big cell and start big cell. Path is then found by connecting the basic cells in the tunnel with a similar method to the tunneling process. The dead end problem resulting from path planning methods using local information is defined and solved by retracking algorithm. In case that path is not found in tunnel, the tunneling process is repeated after the size of big cell is reduced. And if the size of big cell is reduced to a prespecified size usually given as twice of the basic cell, the connections of big cells in which the path cannot be found are recorded as failure connections and the tunneling process is repeated considering failure connections. To show the validity of our proposed method, several examples are illustrated for mobile robots and robot manipulators of SCARA type.

Key Words: Hierarchical search algorithm, Big Cell, Retracking, Tunnel

1. 서론

로봇 연구 분야에서 가장 궁극적인 목표는 로봇 스스로 계획하고 주위 환경에 대처하여 결정을 내리는 능력 로봇을 만들는데 있다. 이 목표를 위해서는 인식, 추론과 결정 그리고 실제 간 제어 등 많은 분야에서의 연구와 발전이 필요하다. 상기 분야중 추론과 결정 분야에서 가장 기본적인 문제중 하나는 복잡한 환경에서의 충돌회 피 경로 계획하는 문제이다. 더욱이 충돌 회피 경로 계획은 오프라인 프로그래밍(Off Line Programming)과 테스크 레벨 로봇 프로그래밍(Task-Level Robot Programming) [1]을 실현하기 위해 요구되는 기본적인 능력중 하나이다. 충돌 회피 경로 계획 문제는 최근까지 많은 연구가 되어져온바[2-19], 크게 인공 포텐셜 함수(Artificial Potential Function)를 이용한 방법[2, 3]과
그래프 탐색(Graph Search)을 이용한 탐색방법
[4~17]으로 대별된다. 전자는 장애물과 목표점
의 위치 관계를 근거로 작용환경의 모든 지역에
대하여 인공 포션형 탐색을 도입하여 위치에
게 설정하는 "경도(energy level)"를 포함하고, 어디가 작아
지는 방향으로 로봇이 추가하는 방법이다. 이
인공 포션형 탐색을 이용한 방법은 원리적으로
주요적인 경도에 의하여 경로가 결정되기 때문에 국
부 요소에 빠져 경로를 계획할 수 없는 경우가
발생할 수 있다[2]. 후자는 모양 공간(configuration-
space)이나 직교좌표 공간을 이진적화시켜 병
(Cell)로 분할한 후, 그것들의 접촉사항을 그래프
로 표시하여, 성정된 평가 함수를 최적으로 하는
경로를 그래프 상에서 탐색하는 계층 방비인 바
그래프 상에서 경로를 탐색하는 방법으로는 breadth-first search 방법과 A*알고리즘 등을 이용
한다[6, 8, 11, 12, 14, 18]. 최근 연구 되어진 그래
프 탐색 방법으로는 Lozano-Pérez [6]가 제안한
방법으로써 모양공간을 부정부의 정밀도를 높이
시킨슬라이스(slic)로서, 하나의 솔라라이스 내에서 장애
물이 아닌 지역을 그 그래프적으로 연결하고 초
발생지 있는 솔라라이스의 포착점은 있는 솔라라이스
를 그래프적으로 연결한 것에서 찾아내어 경로를 구
이 작업공간을 일정한 그리드(grid)로 나누고 로
보트가 장애물과 충돌하는 동작계를 구한 후 작업
공간 내에서 공격에 속하지 않는 그리드들 중에
서 충분히 포착점이 있는 경로를 탐색하려는 연결
경로중 최적경로, 즉 소요 시간과 가속도 제한
동의 조건을 만족하는 경로를 구하는 방법이 있
다. 하지만 성능의 범위는 대상으로 하는 환경이
크거나 로보트의 자유도가 많지 않으면 탐색하는 경
로와 계산량이 크기 증가적으로 높아지게 되는 문
제가 있다[8, 9]. 이러한 문제점을 보완하기 위
하여 별 분할(Cell decomposition)[10~17] 방법
이 제안 되었다. 이는 모양 공간(configuration-
space)을 분할하여 장애물이 있는 영역과 없는 영
역으로 나누어 각 영역을 노드(node)로 한 그래프
적인 연결 관계를 만들고 이 관계에서 경로를 탐
색하는 방법이다. 이 방법은 모양공간(configuration-
space)을 얼마나 정확히 장애물 영역과 장애
물이 없는 영역으로 구분하는가 하는 정도에 따라
정확한 별 분할(exact cell decomposition)이나 근사
적 별 분할(approximate cell decomposition)로
나누어 진다. 정확한 별 분할 방법[15~17]은 장애
물에 정확히 구분되어 지도를 그 형태가 충분히
단순해야만 가능하다[12]. 근사적 별 분할 방법
[11~14]은 모양 공간을 여러 영역으로 구분하
며, 그 영역은 구역화된 장애물의 포함여부에 따
라 지속적으로 경로가 결정되며, 즉, 영역이 장애물로 완
전히 차지되었다면 FULL, 일부가 차지되었으면
MIXED 그리고 영역에 장애물이 전혀 없으면
EMPTY로 구분한다. 그리고 MIXED영역의 경우
는 더 작은 영역으로 분할하여 전체적으로 모두
영역이 EMPTY와 FULL로서 표기와 되도록 하
는 것이다. 이러한 근사적 별 분할 방법은 계층적
구조가 가능하며, 일반적으로는 상기 방법과 같이
MIXED영역을 2"개의 영역으로 분할하는가, n=2
인 경우를 "quadtree"[14] 그리고 n=3인 경우를
"octree"[15]로 부른다. 상기 방법들의 문제점은
영역을 일정적으로 분할하여 영역을 분할하
는 수가 매우 많아지는 것이다[11]. 이러한 문
제점을 해결하는 방법으로 Latombe[11, 12]은 영
역 구분을 장애물을 정확히 영역과 장애물이 정확
한 영역 그리고 장애물이 없는 부분으로 구분하
였다. 하지만 그는 로보트를 한 차원 공간에서
회전과 이동이 가능한 다각형이라는 가정하에서
수행되었으며 로보트의 여러 링크가 가속의 제
한을 갖고 서로 연결된 경로를 최적 경로를 구하는
방법이 일체가 되며 모양 공간상에서의 가속학적
분할 역시 4차원 이상의 경우는 영역 구분이 매우
이해리하다[14]. 한편, 별 분할 방식과는 달리
탐색하는 별을 제한하는 방법으로 Kondo[8]가 제
안한 바와 같이 모양 공간상에서 해커스틱
(heuristic) 평가 함수를 최소화하는 별을 선택하
는 방법으로 두 충돌 별을 탐색하여 선택한 별로
모양공간의 그래프로의 연결 관계를 포함하고 그 그래프
상에서 A*알고리즘을 사용하여 경로를 탐색하는
방법이 있으며, 작업 공간 내에 복잡한 장애물이
많은 경우 영역내에 볼이 없어지면, 다른 탐색 별
의 수와 계산량의 크기 증가를 크게 계산하지
는 못한다.

이에 본 연구에서는 로보트의 동률회의 경로계
통을 위하여 계층적 별 분할 방식에 의한 국부 탐
색 방법을 제안한다. 이 방법은 그래프를 이용하
지 않고 모양공간 내에서 별과 별사이의 거리의
계산법을 별로 이루어진 별 내의 장애물 별의 경
로를 국소적인 정보를 이용함으로써 그래프의 기여
일반 없어져야 하는 별의 경로를 검색할 필요
가 없어 모양 공간에서 별의 경로에 따른 장애물은
전체 그래프에 대한 탐색을 하지 않으므로 계산량
만 줄어들어 별의 경로를 검색하므로 장애물에 따라 경로를 이루어지지 않고 필요 이상으로 우선하는 경로를 만드는 위험도

2. 모양 공간의 형성 및 경로 계획을 위한 정의

적교 좌표계에서의 작업공간 $\psi \subset R^n$, $m=2$ or 3라 하고 $\Theta$ 내에서의 경로를 $O_i$, $i=1,2,\ldots,M$이라 하자. $\Theta$ 내의 자유도를 갖는 로봇의 모양 공간은 $\Theta \subset R^n$로 표기하자. 그리고 로봇의 분할값은 $\theta=(\theta_1,\theta_2,\ldots,\theta_n)$라 하자. 로봇의 각 관절은 $\theta$에 의해서 그 자세와 위치가 변하는 체계로 볼 수 있기 때문에 $\theta$변환 맵을 가지는 $\Theta$에서의 엔딩을 $L(\theta)$로 나타낼 수 있다. 경로 지도를 $\Theta$로 표시하자. 그러면 $\Theta$는 $\Theta$내에 장애물 $O_j$와 로봇의 각 링크로드 축들 자세들이므로 다음과 같이 표현된다.

$\Theta = \bigcup_{i=1}^M \Theta_i \bigcup_{j=1}^M \Theta_j \bigcup_{k=1}^M \Theta_k$  \(1\)

여기서,

모양공간 $\psi$의 각각을 $\Delta_\psi$로 분해하자. 여기서 $\Delta_\psi$는 양의 상수이다. 그리고 $\Delta$에 의해 분해된 $\psi$와 $\theta$를 각각 $\psi_i$와 $\theta_i$라 하자. 그리고 공간 $\psi_i$와 $\theta_i$에서 분해된 범위 각 위치를 각각 $c_i$와 $c_i$로 표기하자. 그리고 $bc$를 범위의 이웃한 $c_i$의 모양으로 하고 "큰형"(Big Cell)이라 명명하자. 그러면 $bc\psi_i$에서는 $b$개의 $c_i$가 존재하는 것이 된다. 그리고 $bc\theta_i$에 의해 나누어진 결과 $\theta_i$를 $\theta_i$라 하자. 여기서 $bc\theta_i$는 $\psi_i$ 공간의 각각을 따라 $\Delta_\psi$로 분해되어 만드는 한 범위 $b$는 양의 정수이다. 이렇게 분리된 범위 혹은 $\theta_i$로 나타내고, 그리고 $bc\psi_i$와 $bc\theta_i$를 각각 $c_i$와 $c_i$로 포함하는 $\psi_i$를 정의하자.

본 논문에서는 경로를 계획하기 전에 $bc\psi_i$와 $bc\theta_i$를 포함한 $bc\psi_i$의 일부의 연계에 의해 터널을 만들고, 그리고 그 터널을 이루는 $bc\psi_i$에서 $c_i$ 등의 연결의 연계에 의해 경로를 찾아나간다. 이 경우 터널에서 경로를 계획하거나 터널을 만들기 위해서는 이웃한 세 개 사이의 연계를 위한 접촉 형태를 어떻게 결정하는가가 중요하게 된다.

이번 범 $A$과 그 범 $A$에 접해있는 범의 접촉형태는 다음과 같이 두 가지로 나누어질 수 있다: 1) 범 $A$에 대하여 그 범 $A$와 접한 범 $b$개 갖는 형태, 그것을 $Con(A)$라 명명하자. 그리고 2) 이번 범 $A$에 대하여 그 범 $A$를 두리진 $b-1$개의 범을 갖는 형태, 이를 $Sur(A)$라 명명하자. 그린 1과 2는 각각 2차원 공간에서 이러한 범 $A$에 대한 $Con(A)$와 $Sur(A)$를 나머지 주로 다룬다.

터널을 만들기 위해서는 각각의 $bc\psi_i$ 사이의 접촉 형태는 터널 내에서 경로를 발견할 가능성이 고려되어 결정되어야 한다. 즉, $bc\psi_i$ 사이에 접촉면이 많은 접촉 형태를 선택하게 되면 $bc\psi_i$ 내에서 경로를 발견할 가능성이 많기 때문에, 이러한 관계에서 $bc\psi_i$의 접촉 형태를 결정하여야 한다. 어떤 $bc\psi_i$와 그 $bc\psi_i$에 대한 $Con(bc\psi_i)$ 내의 $bc\psi_i$와의 접촉면은 두 이탈할 $bc\psi_i$ 사이에 접한 면과 같은 면, 어떤 $bc\psi_i$와 그 $bc\psi_i$에 대한 $Sur(bc\psi_i)$와의 사이의 접촉은 정점 접촉(point contact)이 생길 수가 있다. 따라서 접촉면이 많아야 한다는 것에서 보았을 때, 터널을 만들기 위한 $bc\psi_i$들의 접촉 형태는 $Con(bc\psi_i)$가 $Sur(bc\psi_i)$보다 더 나음이다. 또 경로를 계획함에 있어서, 범의 선택조건을 연속적이고 메모리 공간을 보장한다는 점을 기준으로 보면, $Con(c)$에서의 $c$의 선택으로 인한 경로의 생성은 모양
그림 1 2차원인 경우 Cell A의 2×2개의 Con(A)
Fig. 1 Con(A) in 2 Dimensional space.

<table>
<thead>
<tr>
<th>0</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>4 3 2</td>
</tr>
</tbody>
</table>

그림 2 2차원인 경우 Cell A의 3×1개의 Sur(A)
Fig. 2 Sur(A) in 2 Dimensional space.

공간 상에서 만족 캐시 등가하고 나머지는 정지해 있는 움직임을 유발하게 한다. 즉, 계단형 움직임(step motion)이 유발되는 반경 Sur(bc)에서 쓰를 선택하여 연결에 의한 움직임은 모든 충이 동시에 움직이는 움직임이 유발되어 부드러운 움직임이 유발되는 것이다. 따라서 경로를 계획하기 위한 c의 선택은 Con(c) 보다 Sur(c)에서 하는 것이 바람직하다.

3. 계층적 경로계획 알고리즘

3.1 세부 환경을 위한 경로 함수
고정 장애물이 존재하는 공간에서의 경로 계획을 위한 시점의 방법론에서 대역적인 방법은 먼저 모양 공간을 계층적으로 세부 분할(cell decomposition)한 후 계층적 나무구조(tree structure)를 그래프로 기록한다. 그리고 A* 알고리즘[13, 16, 18, 19] 등을 사용하여 경로를 탐색한다. 여기서 경로함수는 다음과 같이 사용한다.

\[ f(X) = g(X) + h(X) \]  (2)

방정식 Eq. (2)에서 X는 그래프의 노드, \( g(X) \)는 나무 구조(tree structure)의 그래프로부터 평가되

는 Clearance cost이다. 그리고 \( h(X) \)는 허리스틱(heuristic) 함수이다. 이러한 방법은 종종 그래프를 기록한 많은 기억용량을 요구한다. 예를 들어, 만일 한 로봇이 6 D.O.F을 갖고 있고, 모양공간의 각 축을 따라 셀의 수가 2^128개로 주어진다면, 그 그래프를 기억시킬 기억용량은 2^54개가 되는 것이다. 기계시킬 그래프의 크기를 줄기기 위하여 반복형 영 분할 방법[11, 12]도 제안되었는데, 이는 셀내에 포함되어 있는 장애물의 모양에 따라 분할된 셀의 크기를 정하는 방법이다. 하지만, 이 방법은 로봇이 단지 2차원에서 이동하는 다각형이 가정한 다각형이 가정에서 비해 절제울 절제될 그래프 상에서 계획되므로 그래프 7(c)와 7(d)e와 같이 장애물에 따라 필요한 경로의 긴 경로가 생성되며, 사용이 제한된다. 그러나 그래프를 다시 작성해야 하는 바, 그 계산량은 모양공간의 차원에 따라 기하급수적으로 증가하다.

본 논문에서는 이러한 문제의 해결하기 위하여 새로운 계층적 국소 탐색 알고리즘을 제안한다. 제안한 알고리즘은 그래프와 같은 대역적 정보를 사용하는 것이 아니라, 유클리디안 거리(Euclidian distance) 그리고 큰 범위의 정보를 작은 개수와 같은 국소적 경로를 이용한다. 따라서 계산된 알고리즘은 많은 용량의 메모리를 필요로 하지 않는다.

제안된 알고리즘에서는 경로를 계획하기에 앞서 터널을 만드는 바, 터널은 \( b_{Start} \)와 \( b_{Goal} \)를 포함하는 일련의 \( b_c \)들의 연결로 이루어진다. 경로는 터널을 구성하는 \( b_c \)내에서 장애물을 셀 \( c \)로 최적화하면서 현재 셀과 목표 셀간의 거리가 최소가 되게 하는 셀을 현재 셀과 이웃하는 셀 중에서 선택하여 연결해 나간다. \( b_c \)들을 연결할 때 \( b_c \)의 Con(bc) 중에서 다음 식을 최소로 하는 \( bc \)를 연결해 나간다.

\[ f(bc) = K_f b_c(k) + K_f b_c(j) + K_f b_c(h) \]  (3)

여기서 \( K_i \), \( K_j \)는 양의 상수이고, \( K_f \)는 음의 상수이다. 방정식 Eq. (3)에서 \( f(bc) \), \( f(bc) \) 그리고 \( f(bc) \)는 다음과 같이 주어진다.

\[ f(bc) = |bc_{Start} - bc| / |bc|_2 \]  (4)
\[ f(bc) = |bc_{Goal} - bc| / \|bc - bc\|_2 \]  (5)
그리고
\[ f_i(bc) = \delta(N_w(bc) - b^\theta) \quad \text{(6)} \]
*MAXIMUM VALUE + \[ N_w(bc) / b^\theta \]
여기서 \( N_w(bc) \)는 \( bc \)네의 \( c \)의 갯수이다. 그리고 \( \delta(X) \)는 (7)식으로 주어진다.
\[
\delta(X) = \begin{cases} 
1, & \text{if } X = 0, \\
0, & \text{otherwise.}
\end{cases} \quad \text{(7)}
\]
식 (4)와 식 (5)에서 \( bc\theta \)와 \( bc\nu \)는 자가 \( \theta = (0, 0, \ldots, 0) \)과 \( \theta = (N_{\text{max}}, N_{\text{max}}, \ldots, N_{\text{max}}) \)에서의 \( bc \)를 의미한다. 여기서 \( N_{\text{max}} \), \( i = 1, 2, \ldots, N \)는 \( bc \)공간에서 \( i \)번째 층 상의 \( bc \)들의 개수이며 MAXIMUM VALUE는 양의 큰 상수이다. (6)식을 최소화하는 \( bc \)를 선택한다는 것은 \( bc \)의 연결된 \( bc \)중에서 \( bc \)공간에 정해진 \( cd \)가 가장 작은 것을 선택하는 것을 의미한다. 단일 \( bc \)공간의 \( c \)가 모두 정해질 때이라면, (6)식은 그 \( bc \)에 특정한 양의 상수값 MAXIMUM VALUE를 부가하여 그 \( bc \)가 연결되지 못하도록 한다. 식 (5)의 \( f_i(bc) \)는 \( bc \)공간에서의 \( ||bc\theta - bc\nu||_\infty \)으로 주어진 최대거리에 대한 점점 \( bc \)의 \( bc\nu \)사이의 거리의 비이다. \( f_i(bc) \)와 (2)식의 \( g(X) \)의 차이는 \( g(X) \)는 그래프적 관계를 대역적 정보를 사용하지만, \( f_i(bc) \)는 국소적 정보를 사용한다는 것이다. \( i(bc) \)는 (2)식의 \( h(X) \)의 역할과 유사하다.

많은 국소적 방법 [3, 8, 9]과 마찬가지로 제안된 알고리즘은 경로찾기의 실제 또는 필요 이상의 큰 경로를 만드는 요인이 되는 위험감 현상을 수용하지는 않는다. 위험감 현상은 그림 3에서 보여진 바와 같다. 그림 4에서 1, 2 그리고 5는 그 머리 (head)의 꼬리 (tail)에 위치한 두 펜릴이 연결되어 있다. 하지만 3과 4는 그들 주변에 3개이상의 선과 연결되어 있다. 이것은 우리는 위험감 현상이나 부모기도 하지, 그러한 위험감 현상은 다음과 같이 쉽게 점차일 수 있다.

식 (3)을 최소화하는 \( bc\nu \)에서 \( \nu(bc) \)가 아닌 \( bc \)공간에서 \( \nu(bc) \)로 구성된 집합 \( B = \{b_1, b_2, \ldots\} \)
\( \subseteq bc\nu \)를 생성하자. 그리고 \( bc \)공간에 대하여 집합 \( B \subseteq \nu(bc) \)에 속하는 모든 \( bc \)를 \( bc \)라 표기하자.

**그림 3** 위험감 (dead end) 현상. 여기서 S는 start cell, G는 goal cell을 나타낸다.

**Fig. 3** The occurrence of dead end. Where S means start cell and G means goal cell.

**그림 4** 위험감 (dead end) 상태를 나타내는 연결.

**Fig. 4** The connections of cells which represent the dead end.
a) The case when the searching the path is failed with $b=3$.
b) Tunnel generated by forward tunneling process with $b=2$.

(그림 5) 성공 로보트에 대한 충돌회피 경로 계획.
(지형 예제 1)
Fig. 5 Collision free path for point robot (Map Example 1)

- ■ : Subgoal, □ : Big cell
- ○ : Basic cell, ■ : Obstacle cell
일 터널내에서 경로의 발견이 실패한다면 \( bc \)의 크기를 줄이고 터널 만들기 과정과 재추적 과정을 다시 한다. 만일 그대로 실패하면 그리고 \( bc \)의 크기가 정해진 크기의 크기를 연달아 실패한 \( bc \)의 경로를 가진 경로들로 가공한다. 그리고 다시 터널 만들기를 하여 이러한 경로들을 고려하여 경로를 재선정한다. 재선정된 재추적 경로에 경로와 터널은 다음 7개의 STEP들로 설명할 수 있다.

주어진 \( C_{start} \), \( C_{goal} \) 그리고 \( bc \)의 초기길이 \( b_{init} \), \( bc \)의 제한값 \( b_{limit} \)에 대하여, 

**STEP 1** (터널 만들기 과정)

\( bc \) 공간상에서 \( C_{start} \)에서부터 \( C_{goal} \)까지 연속적인 \( bc \)의 경로를 구하려 한다. 현재 \( bc \)의 \( Con(bc) \) 중에서 각 서로 묶어지게 하여서, \( T \)를 구성한다. \( T \)는 각 \( bc \)들의 경로를 가지며 \( b \)값으로 정해진다. 터널을 이루는 터널들의 \( bc \)들은 이러한 경로를 자원 자료로 한다. 이러한 \( bc \)들은 \( T \)의 각 \( bc \)들로 구성되어 있다. 터널을 이루는 \( bc \)들은 이러한 경로를 자원 자료로 한다. 이러한 \( bc \)들은 \( T \)의 각 \( bc \)들로 구성되어 있다. 이 경우에는 \( T \)의 각 \( bc \)들을 \( Con(bc) \) 내에서 (3)의 식을 고려하여 \( bc \)들을 연결해 나간다.

터널 만들기는 다음과 같이 자세히 설명된다.

\( T = \{ t_0, t_1, t_2, \ldots \} \)는 내부적으로 연결된 \( bc \)들의 위치의 집합이다.

**step 1.1** : \( p \)를 0으로 한다. 그리고 \( C_{start} \)의 위치를 \( mp_0 \)에 저장한다.

**step 1.2** : \( p \)를 \( p+1 \)로 한다. 그리고 현재 \( bc \)의 \( Con(bc) \) 중에서 각 서로 묶어지게 하는 다음 경로로 \( bc \)을 발견한다. 그리고 그것을 \( mp_p \)에 저장한다.

**step 1.3** : \( Con(mp_p) \)의 모든 \( bc \)들에 대하여 만약 (3)식의 \( f(bc) \)의 값이 MAXIMUM_VALUE가 아니라 현재까지 연결되어온 \( bc \)들의 길이 \( mp_p \)에 포함되어 있다면, 현재의 \( bc \)를 \( mp_{p-1} \)과 연속적으로 연결된 \( bc \)가 있는 경우가 되므로, 쌍은 \( mp_p \)로 연결되어온 \( bc \)들을 연속적으로 고려해 나간다. 만일 다른 방향으로 연속적으로 나간 \( bc \)들에 대한 \( mp_p \)의 모든 \( bc \)들을 고려하여 \( mp_p \)의 모든 \( bc \)들을 연결해 나간 \( bc \)들에서 start가 있다고 가정할 수 있다.

**step 1.4** : \( mp_p \)에서 \( bc_{goal} \)와 일치할 때까지 step 1.2)에서 step 1.3)까지 반복한다. \( mp_p \)가 \( bc_{goal} \)과 일치하였다면 단계 1.5)로 간다.

**step 1.5** : \( bc_{goal} \)과 \( C_{start} \)를 교환한다.
경로를 구한다. 만일 터널을 구성하는 $bc$들 빠져서 경로가 발견되지 않으면, $bc$의 크기를 나타내는 변수 $b$을 $b = b - 1$로 하여 $bc$의 크기를 줄인 후 STEP 1로 간다. 하지만 만일 $b$가 미리 정해진 $bc$의 크기의 최소값 $b_{\text{limit}}$과 같다면 다른 STEP으로 간다. 경로는 최종 터널 $F$를 이루는 $bc$들을 내에서 $c_{\text{start}}$에서 $F$ 내의 각 $bc$의 중심을 $c$를 목표로, $c$를 연결해 나가므로 생성된다. $c$들의 연결은 현재 $c$에 대한 $\Sigma(c)$ 내에서 (8)식을 최소화 하는 $c$를 선택하여 연결해 나간다.

$$f^*(c) = (K_d||c_{\text{mid}} - c||^2 + K_v||c_{\text{mid}} - c_{\text{start}} ||)$$

$$-c||c||_2$$

$$+\delta(N_0(c) - 1) \cdot \text{MAXIMUM}_V\_\text{VALUE}$$

여기서 $K_d$와 $K_v$은 각각 가중치 값의 성수이다. 그리고 $\Sigma(c)$는 $\Sigma(c_{\text{mid}}, c_{\text{start}}, c_{\text{end}})$의 성수이다. 여기서 $N_{\text{max}} = i, 1, 2, ... , N$이 공간 내에서 $i$번째 축을 따르는 $c$의 점사이며 $c_{\text{end}}$과 $c_{\text{start}}$는 $F$ 내의 $bc$들의 중심점을 나타내는 $c$들로, $c_{\text{mid}}$가 수락 가능한 $bc$는 $c_{\text{mid}}$가 포함된 $bc$의 탐색에 연결된 $bc$이다.

STEP 7 (실험의 수정과정)

로봇을 이루는 $bc$들 중에서, $bc$ 내에서 (8)식을 최소화 하는 $c$의 가속도가 존재하지 않는 $bc$는 경로 찾기에 실패한 $bc$들로 간주한다. 이러한 $bc$들은 가상 장애물로 기록한 후 STEP 1부터 반복한다.

상기 방법과 같이 $bc$의 크기를 작게한 후 가상 장애물로 간주하여 터널 만들기를 다시하고 그래도 경로가 만들어지지 않을 경우 실패한 $bc$들을 가상 장애물로 간주하여 터널 만들기를 다시하는 이유는 다음과 같다. 만일 큰 $b$ 값 갖는 $bc$가 만들어진 터널 내에서 경로 찾기가 실패하여 그 실패한 경로 부분을 가상 장애물로 할 경우 $bc$ 공간상의 상당 부분이 가상으로 간주되어 터널 만들기가 실패할 것이다. 따라서 실험의 수정에 있어서 $bc$의 크기의 축소는 특정 $b$의 크기의 $b_{\text{limit}}$까지하고 작아진 $bc$에 대하여 경로 찾기에 실패한 연결 부분을 가상 장애물로 기록한 후

4. 실험 및 결과

본 논문에서 제안한 알고리즘은 다음과 같이 이동 로봇과 스카라(SCARA)형의 로보트를 대상으로 한 예제를 통하여 모의실험 하였다. 사용된 컴퓨터는 IBM 486 컴퓨터이다. 예제는 적용 로봇에 따라 크게 2가지로 하였다. 특히 제1의 경우 로봇에 대한 지형 예제와 미로 찾기 문제 및 제2의 2림량을 갖는 로봇에 대한 실험은 $A^*$ 알고리즘을 이용한 시스템의 경로 계획 방법을 Kondo의 방법과 제안한 방법을 비교하였으며, 비교 결과는 경로 찾기경로에 터널에 개선된 경로와 터널 없는 경로로 두 가지 경로를 검증하고 적합한 경로를 선택하는데 Tzieh과 ‘L’형의 이동 로봇에 제안되는 경로 경로를 사용하였다 Zuh/Latombe [11] 그리고 Brooks/Lozano-Perez [13]가 제안한 방법들과 소요 시간의 관점에서 비교하였다. 예제 2는 장애물이 없는 환경에서 SCARA형의 2림량과 4림량을 갖는 로봇을 대상으로 각각 모의 실험하였다. 두 예제 모두 본 알고리즘에 적용되는 성수값인 $b_{\text{limit}}$은 2로, $b$의 초기값 $b_{\text{init}}$은 3으로 하였고 목적함수인 (3)식의 성수값 $K_d, K_v$ 그리고 $K_a$는 각각 80, 10, 20으로 하였으며 (8)식의 $K_d$와 $K_v$는 각각 80, 15로 하였다. 그리고 (8)식의 $\text{MAXIMUM}_V\_\text{VALUE}$와 터널 통로 길이의 허용치 성수인 $K_a$는 각각 100000과 2.0로 하였다. 예제 1에서 A는 로봇의 시작점, $B$는 목표점 을 의미한다. 분류된 예제들은 다음과 같다.

예제 1

적용대상 로봇은 정형과 박막형 그리고 ‘L’형의 이동 로봇이다. 정형 로봇을 대상으로한 예제는 그림 5(a)와 그림 6과 같은 복잡한 지형에 서의 경로 찾기와 그림 8과 같은 미로 찾기 문제도 한다. 두 경우 다 복잡 장애물을 존재한 지도의 $c$의 수는 92×70으로 하였다. 그림 5(a)부터 그림 5(f)까지는 제한된 알고리즘의 결과를 단계별로 보여주고 있다. 그림 5(a)는 $b = 3$으로 하여 구한 최종 경로에서 경로 찾기를 실패한 경로를 보여주고 있다. 표 1에서 보는 바와 같이 경로 경로는 0.8893초이다. 그림 5(b)의 그림 5(c)는 각각 $b = 2$로 최적의 경로를 찾기 실패한 경로를 보여주고 있다. 그림 5(d)의 그림 5(e)는 보여주고 있다. 그림 5(d)는 위험성 현상 없이 오른쪽 동작 경로의 $bc$들이 동작하기
표 1 예제 1에 대한 요소 시간
Table 1 Statistics of time of Example 1

<table>
<thead>
<tr>
<th>예 제 명</th>
<th>동로 찾기 소요 시간(sec)</th>
<th>정로 찾기 소요시간(sec)</th>
<th>전체 소요 시간(sec)</th>
<th>비 고</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>전 방향</td>
<td>후 방향</td>
<td></td>
<td></td>
</tr>
<tr>
<td>그림 5의 예</td>
<td>0.43956</td>
<td>0.43956</td>
<td>0.01</td>
<td>b=3</td>
</tr>
<tr>
<td></td>
<td>6.26376</td>
<td>2.632967</td>
<td>0.054945</td>
<td>9.445206</td>
</tr>
<tr>
<td></td>
<td>1.098901</td>
<td>1.813187</td>
<td>0.109890</td>
<td>b=3</td>
</tr>
<tr>
<td></td>
<td>3.954044</td>
<td>1.703297</td>
<td>0.109890</td>
<td>5.767231</td>
</tr>
<tr>
<td></td>
<td>3.846154</td>
<td>0.054945</td>
<td></td>
<td>b=3</td>
</tr>
<tr>
<td></td>
<td>9.615385</td>
<td>0.054945</td>
<td>13.571423</td>
<td>b=2</td>
</tr>
<tr>
<td>그림 11c의 예</td>
<td>0.274729</td>
<td>0.109890</td>
<td>0.384619</td>
<td>b=3</td>
</tr>
<tr>
<td>그림 11d의 예</td>
<td>1.813187</td>
<td>0.164835</td>
<td>0.054945</td>
<td>1.978022</td>
</tr>
</tbody>
</table>

표 2 이동로봇의 경로 계획에 대한 소요시간,
Table 2 Statistics of time for planning the collision free path of mobile robot.

<table>
<thead>
<tr>
<th>예 제 명</th>
<th>제안된 방법 (IBM 486, Boland C++)</th>
<th>Zuh의 방법 (Machintosh II, LISP)</th>
<th>Brooks/Lozano-pe'rez방법 (Machintosh II, LISP)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2.6 sec</td>
<td>5.5 min*</td>
<td>tens of mins*</td>
</tr>
<tr>
<td>그림 9의 예</td>
<td>2.7 sec</td>
<td>2.5 min*</td>
<td>tens of mins*</td>
</tr>
</tbody>
</table>


그림 6 지형 예제 2.
Fig. 6 Map example 2.

시작하는 곳에 나타나 있는 바, X형의 사각형이 그양 할당들이 중간 표시점이다, 그림 5(e)와 그림 5(f)는 각각 전방향 탐색 동로의 후방향 탐색 동로이며 후방향 탐색이 전방향 탐색보다 짧으므로 그림 5(g)에서는 후방향 탐색 동로 네에서 경로를 찾고 있음을 보이고 있다. 표 1에서 보는 바와 같이 그림 8의 경우는 경로들이 조밀하게 있어 b=3으로 한 경로 찾기를 실패하고 b=2로써 경로를 찾았으며 최종 경로를 찾기까지 약 14초 가량이 소요되었으나 이 경우 바탕으로 회전을 없애기 위하여 3회의 동로 만들기와 3회의 재추적 과정을 거쳤다. 그림 7(a)에서 그림 7(d)까지는 Kondo[8]에 제안한 방법을 지형 예제에 적용하여 경로와 경로를 구체적한 방법에 의한 두 측면 별 미클러사건으로 표시된 미클러는 경로의 형태를 나타내고 있다. 표 3은 제안된 방법과 Kondo가 제안한 방법에서 경로를 찾기위한 채로 탐색 결과를 나타내고 있다. 표 3에서도 알 수 있듯이 그림 5와 그림 6 예
표 3 소요 시간 및 무충돌 셀 수의 비교
Table 3 Statistics of the number of Free Cells and Time for planning the path.

<table>
<thead>
<tr>
<th>예 제 명</th>
<th>제안된 방법</th>
<th>Kondo의 방법</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bc의 탐색 개수</td>
<td>경로 찾기까지의 소요시간 (sec)</td>
</tr>
<tr>
<td>그림 5에 예</td>
<td>509</td>
<td>9.445206</td>
</tr>
<tr>
<td>그림 6에 예</td>
<td>449</td>
<td>5.767231</td>
</tr>
<tr>
<td>그림 8에 예</td>
<td>1033</td>
<td>11.571423</td>
</tr>
<tr>
<td>그림 11c 예</td>
<td>70</td>
<td>0.384619</td>
</tr>
<tr>
<td>그림 11d 예</td>
<td>409</td>
<td>1.978022</td>
</tr>
</tbody>
</table>

그림 7 Kondo방법에 의한 무충돌 cell과 경로.
Fig. 7 Free cells and collision free paths by Kondo's method.
■ : Obstacle Cell, ■ : Free Cell

는 본방법이 제안한 방법이 더 적은 셀을 탐색하고 있다. 더욱이 그림 5(g)와 그림 7(c) 그리고 그림 6과 그림 7(d)를 비교하여 보면 구체적으로 찾은 경로 그래프를 만들어 그 포인터를 추적함으로써 경로를 만든 Kondo방법의 경우가 본 방법에 비하여 오목한 (concave) 경로들이 있는 곳에서 많이 이상으로 범위하하거나 우회하는 경로가 만들어지고 경로 찾기에도 더 많은 시간이 소요되고 있
그림 8 미로 예제
Fig. 8 Maze Example.

그림 9 막대형 이동 로봇에 대한 무충돌 경로 계획 결과.
Fig. 9 Collision free motion for mobile robot of Bar type.

제시된 예제와 같은 것이다. 표 2에서 제안된 알고리즘 Zuh/Latombe 그리고 Brooks/Lozano-Pérez가 제안한 방법인 경로 계획의 효율 시간의 관점에서 비교하였다. Zuh/Latombe 그리고 Brooks/Lozano-Pérez가 제안한 방법인 썬을 사용 컴퓨터 machintosh II에서 프로그램 전파일로는 Allegro Common Lisp이며 본 알고리즘은 IBM 486에 Boland C++를 사용하였다. 두 Tool의 속도 비교는 본 알고리즘에 사용한 Tool이 Zuh/Latombe 그리고 Brooks/Lozano-Pérez등이 사용한 Tool보다 10배에 8배 정도 훨씬, (본 연구실에서의 자체 테스트 결과임). 이 로 고려하여 표 2를 고려하면 주어진 예제에 대하여 본 알고리즘의 효율성은 Zuh/Latombe나 Brooks/Lozano-Pérez가 제안한 방법보다 그림 9의 경우 약 15배, 그림 10의 경우 약 7배 정도 빠를 수 있다.

예제 2

차량 영역에 6개의 원시 (primitive) 형태로 모델링된 장애물이 있는 경우 스카라 형 로봇 방을 대상으로 종돌 회피 경로를 찾는 문제이다. 예제는 2링크를 갖는 로보트와 4링크를 갖는 여유 자유도 로보트에 대하여 각 각 모의 실험하였다. 2링크를 갖는 로보트의 경우 자 관절의 가동 범위는 각각 ±120°와 ±150°이며 조인트 공간에 투영된 장애물 지도의 문화는 톤먼지 관절은 2° 그리고 두면계 관절은 3°이다. 장애물 지도의 c의 수는 120×100이다. 일반적으로 조인트 최적 해에서의 장애물 지도를 만들기 위해서 적정 최적 로보트의 속도 경로 계획을 위한 새로운 계층적 알고리즘
제에 있는 작업 영역의 장애물은 근사화되어 모델링 되고 그것이 조인트 좌표계의 공간으로 투영되므로 상기 분해도 방법으로 \( b = 3 \) 이상으로 하여도 동로 만들기 가능함을 알 수 있다. 본 예제의 경우는 \( b \)의 초기 값을 \( b = 3 \)으로 초기지를 정하였다. 그림 11(a)는 조인트 공간상에서 \( b = 3 \)의 \( b_0 \)로 동로 간돌기름 한 결과이고 그림 11(b)는 이러한 동로 내에서 최종 경로를 찾은 결과이다. 그림 11(c)는 최적 자세가 Left Arm이고 목표 자세가 Right Arm인 경우이다. 그림 11(d)의 경우 로봇의 초기위치는 \( A \)이며 자세는 Right Arm이고 목표위치는 \( B \)이며 그때의 자세는 Left Arm이다. 그림 11(d)에서 볼 수 있듯이 로봇은 목표점으로 향하면서 \( A \)점 상단의 상하에
그림 13 4링크를 갖는 여유가유도 로보트에 대한 무충돌 경로 제어.

Fig. 13 Collision free motion for redundant robot with 4 links.

있는 장애물 사이의 공간간 둘레가 Right Arm으로 불어가 Right Arm으로 제어가 없게 되어 있음 그림 11(c)의 경우와 그림 11(d)의 경우 최적 경로를 찾아가는 것이된다 각각 1.978022, 0.384619 초인 마. 제한된 앵커점이 Off-Line Programming에 사용되어도 창조 하려다 사료된다. 그림 12(a)와 그림 12(b)는 Kondo방법을 스카라 형 로보트에 대한 무충돌 경로 계획에 적용한 결과이다. 그림 11(c), 그림 11(d), 그림 12(a)와 그림 12(b) 그리고 표 3에서 보는 바와 같이 Kondo방법에 의한 경로 계획 결과는 제한된 방법에 따라 경로 계획하여 탐색한 빠르고도 적합한 경로도 얻었다. 그림 13(a)와 그림 13(b)는 4링크를 갖는 여유가유도 로보트에 대하여 각각 모의 실질한 결과이다. 이 경우 각 장애물의 가용 범위는 $A \times B \times C \times D$가 $[-100^\circ, 100^\circ] \times [-120^\circ, 120^\circ] \times [-120^\circ, 120^\circ]$이며, 모양공간 $\mathbb{R}^4$의 차는 30 x 28 x 26 x 24로 하였다. 4 자유도 로봇의 경로는 모양 공간의 4차원이며, 이 경우에도 제한된 경로를 찾기에 성공하고 있음을 보여주고 있다.

상기 예제 1과 예제 2, 그리고 표 1, 표 2 그리고 표 3에서 알 수 있듯이 본 논문에서 제한된 앵커점은 로보트의 작업공간 내에 장애물이 있을 경우 충돌회피 경로를 비교적 빠른 시간내에 성공적으로 계획하여가능성이 있음을 보여주고 있다.

5. 결론

장애물들이 있는 작업 환경에서 로보트의 충돌회피 경로 계획 방법으로 새로운 계층적 방안 할 방법을 이용한 국부적 탐색 방법이 제안되었 다. 경로는 로보트의 모양공간(configuration space)에서 탐색되는 마. 모양공간을 임의한 크기의 간 단위로 분해하고 이 블록 중에서 로보트의 출발점을 나타내는 블록 목표점을 나타내는 블록 사이의 범위를 모양공간에서 국부적 탐색방법으로 연결하여 구하게 된다. 경로를 계획하기에 앞서 모양공간의 탐색 시간을 줄이고 많은 경로를 하여 탐색한 빠르게 제어하는 바, 탐색 시간과 기억 용량을 줄이기 위해 제한된 계층적 탐색방법을 사용하였다. 아동날 로봇공간에서 국부적 경로를 미로 경로를 탐색한 것으로서 발생하는 피딩크(leading end) 문제를 정의하고 그 해결 방안도 제시하였다. 제한된 앵커점은 여유가유도 로보트를 대상으로 한 미로 찾기 문제의 스카라 형(SCARA type)의 비 여유가유도(nonredundant) 그리고 여유가유도(redundant)로 로봇 활용 대상으로한 중등화된 경로 찾기 문제에 적용하여 그 유용성을 입증 하였으며 기존의 국부적 경로 찾기 알고리즘 및 계층적 알고리즘과 비교, 검토 되었다. 추후 피드백은 보다 효율적인 경로 탐색을 위하여 $b_0$의 크기의 초기치 $b_{init}$가 제한한 $b_{limit}$에 대한 분석적 결정 방법이 요구된다.

참고문헌


