한국정밀공학회지 제 25 권 제 4 호 (2008 년 4 월)
Journal of the Korean Society for Precision Engineering Vol. 25, No. 4, April 2008

스트링과 수정된 SOFM을 이용한 이동로봇의 전역 경로계획

차영엽#

Global Path Planning of Mobile Robot Using String and Modified SOFM

Young-Youp Cha#

ABSTRACT

The self-organizing feature map(SOFM) among a number of neural network uses a randomized small valued initial weight vectors, selects the neuron whose weight vector best matches input as the winning neuron, and trains the weight vectors such that neurons within the activity bubble are moved toward the input vector. On the other hand, the modified method in this research uses a predetermined initial weight vectors of the 1-dimensional string, gives the systematic input vector whose position best matches obstacles, and trains the weight vectors such that neurons within the activity bubble are move toward the opposite direction of input vector. According to simulation results one can conclude that the method using string and the modified neural network is useful tool to mobile robot for the global path planning.

Key Words : Global path planning (전역경로계획), Mobile robot (이동로봇)

1. 서론

이동로봇의 경로계획은 이동로봇이 목표점에 도착하기 위하여 초기위치와 목표점 사이의 경로를 여러 개의 기본 운동형태로 나누는 것이다. 이와 같은 경로계획은 크게 전역 경로계획(global path planning)과 지역 경로계획(local path planning)으로 나눌 수 있다. 전역 경로계획은 이미 주어진 장애물 지도를 기본으로 출발점에서 목표점까지 장애물과 충돌을 피하면서 가장 빠르게 갈 수 있는 경로를 찾는 것이다. 이에 반하여, 지역 경로계획은 지도가 없는 미지의 환경을 이동하거나, 이미 작성된 지도를 이용한 전역 경로계획에 따라서 이동로봇이 이동할 때 지도에 나와 있지 않은 장애물이나 이동 장애물을 피하기 위하여 실시간 센서 정보를 이용하여 국부적으로 경로를 재 생성하는 것이다. 전역 경로 계획은 형상공간 방법(configuration space method), 포텐셜 방법(potential approach), 그리고 퍼지, 신경회로망, 유전자 알고리즘에 기초한 인공지능 알고리즘이 이동로봇의 경로계획에 적용되었다.

☞ 접수일: 2007년 8월 1일; 게재승인일: 2008년 1월 25일
교신저자: 원광대학교 공과대학 기계자동차공학부
E-mail: ggypcha@wonkwang.ac.kr Tel. (063)850-6693

이들 중에서 인공지능 알고리즘을 이동로봇의 전역 경로계획에 사용하려고 하는 노력이 최근에 있었다. Qunjie[1]는 퍼지 알고지즘을 사용하여 지역 경로계획 문제[2]를 다루었고, Zhu[3]는 신경회로망과 함께 cost 함수로 충돌 에너지 함수를 이용하였고, Bourbakis[4]는 골격화(skeletonization)와 신경회로망을 이용하였으며, Chaiyaratana[5]는 유전자 알고리즘을 이용하여 경로계획 문제를 해결하였다. 이러한 노력에도 불구하고 아직도 지역 또는 전역 경로계획 문제는 계산시간, 이동물체 회피, 간단성 등의 문제점이 있다.

이동로봇[6]의 자율주행을 위하여 신경회로망 중에서 SOFM을 이용한 전역 경로계획 알고리듬을 차[7]가 제안한바 있다. 기존의 SOFM 알고리듬[8]은 초기 가중치 벡터를 아주 작은 랜덤값으로 주고 모든 가중치 벡터를 학습시키며, 입력벡터를 작업영역 전체에 랜덤하게 분포시킨다. 이에 반하여, 차[7]는 초기 가중치 벡터를 그물망(mesh)처럼 이미 결정된 값으로 주고, 가장자리의 가중치 벡터는 학습에 관계없이 고정되도록 하였으며, 입력벡터를 장애물 주위에 전략적으로 분포하도록 하였다. 그 이외에는 기존의 SOFM 알고리듬을 따르도록 하였다. 즉 학습계수와 이웃관계 함수를 초기화한 후, 장애물 주위에 한 개의 입력이 가하고, 가해진 입력에 가장 가까운 가중치벡터를 갖는 승자뉴런(winning neuron)을 찾아서, 승자뉴런 근방의 뉴런들을 입력벡터 방향으로 움직이게 함으로서 가중치벡터를 재계산한다. 이와 같은 과정을 반복하여, 뉴런의 위치를 재배치함으로서, 주어진 입력에 따른 특징을 구현하여 이 결과를 전역 경로계획에 이용하였다. 그러나 이 방법에서는 장애물 주위에 가해지는 입력벡터에 의해서 가중치가 재 계산되어 배치되는 과정에서 가중치들끼리 위치가 일정하게 펼쳐지지 않아서 생성된 경로가 왜곡되는 문제가 발생한다.

본 연구에서는 차[7]의 결과에서 발생하는 경로 왜곡문제와 계산시간 단축을 위하여 수정 알고리듬을 제안한다. 첫 번째로 왜곡문제 해결을 위하여, 입력을 장애물 외부에 가하여, 가해지는 입력에 가장 가까운 가중치벡터를 갖는 승자뉴런을 찾아서, 승자뉴런 근방의 뉴런들을 입력벡터 방향으로 움직이는 대신에, 여기서는 입력을 장애물 내부에 가하여, 가해지는 입력에 가장 가까운 가중치벡터를 갖는 승자뉴런을 찾아서, 승자뉴런 근방의 뉴런들을 입력벡터 반대방향으로 움직이게 함으로서 가중치벡터를 재계산한다. 두 번째로 계산시간 단축을 위하여 가중치 벡터로 2차원의 그물망 대신에 1차원의 스트링(string)을 사용한다.

제안된 스트링을 이용한 전역 경로계획 알고리듬의 효율성을 입증하기 위하여, 차[7]가 제안한 장애물 외부에 입력벡터를 가하고 가중치 벡터가 입력벡터 방향으로 움직이는 결과와 본 연구에서 제안한 장애물 내부에 입력벡터를 가하고 가중치 벡터가 입력벡터 반대방향으로 움직이는 결과를 모의실험을 통하여 비교한다. 결과적으로 경로왜곡 해소뿐만 아니라 계산회수도 더 적게 요구되는 등 후자가 더 효과적임을 보여준다.

2. SOFM

2.1 원래의 SOFM

본 연구에서 이용하려고 하는 SOFM 네트워크의 신경 회로망은 자기 조직화 특성에 의해 임의의 추상적인 관계를 추론할 수 있으며, 더 많은 입력이 인가 될수록 네트워크는 그 학습을 개선하고 변화된 입력들에 적응하여 출력을 내보낸다. SOFM의 목적은 N-차원의 입력 공간을 의미 있는 지형학적인 순서로 1차원 또는 2차원의 출력 공간(출력 뉴런)에 맵핑할 수 있게 하는 것이다. 이러한 목적을 성취하기 위해 경쟁학습(competitive learning)에 의한 승자독점(winner-take-all)원리와 측면 제어(lateral inhibition)가 이용된다.[8] 이러한 구조의 한 이점으로는 변화하는 상태와 입력에 대해 대처할 수 있다는 것이다. 그러므로 이 네트워크는 입력을 다른 카테고리로 분류할 때와 음성 인식, 로봇 모터 제어 등에 사용된다.

SOFM 알고리듬은 입력 벡터 X와 j번째 출력층 뉴런과 상응하는 가중치 벡터 W_j를

$$X = [x_1, x_2, \ldots x_p]^T \tag{1}$$

$$W_j = [\omega_{j1}, \omega_{j2}, \ldots, \omega_{jn}]^T,\ j=1,2..,N \tag{2}$$

로 표시한다면, 입력 벡터 X와 가중치 벡터 W_j를 이용한 학습 법칙은 다음과 같다.

$$W_j^{new} = W_j^{old} + \eta(X_i - W_j^{old}) \tag{3}$$

여기서 η는 학습율을 나타내고, i 는 1에서 p 까지 입력 뉴런의 수를 나타내며, j 는 1에서 N 까지 출력 뉴런의 수이다. 그리고 w_{ji}는 i번째 입력 뉴런과 j번째 출력 뉴런을 연결하는 가중치를 나타낸다.

입력 벡터의 분류를 위한 출력 층의 승자 뉴런의 결정은 입력값 X와 가장 비슷한 가중치 W_j를 갖는 출력 뉴런을 선택하는 것과 같다. 이러한 출력 뉴런을 선택하는 방법은 두 가지가 있다. 첫째는

$$I_{j\,\max} = \sum_{i=1}^{P} \omega_{ji}\, x_i \tag{4}$$

과 같은 I_j를 선택하는 것이고, 두 번째는 입력 벡터와 최소의 Euclidean norm을 갖는 가중치를 선택하는 것이다. 즉, i(X) 가 승자 뉴런이라 한다면 이 식은 다음과 같다.

$$i(\mathrm{X}) = k, \quad where \ \|W_k - \mathrm{X}\| < \|W_j - \mathrm{X}\| \tag{5}$$

위와 같이 출력 뉴런을 선택하여 승자가 된 뉴런만이 "1"이 되고 나머지는 "0"이 되는데 이러한 방법이 경쟁학습에 의한 승자독점 원리이다. 보통 가중치 벡터와 입력 벡터는 정규화 시키는데, 그 이유는 학습규칙이 입력 벡터로부터 가중치 벡터를 뺀 값을 사용하기 때문이다.

그리고, 좀더 효율적인 패턴 분류를 위하여 측면 제어를 이용한다. 측면 제어의 대표적인 방법은 이웃관계 함수, $\Lambda_{i(X)}(n)$의 사용으로, 승자 뉴런이 선택되면 승자 뉴런의 이웃하는 거리에 따라 연결 강도를 달리하는 방법으로 연결 강도는 거리에 반비례한다. 이웃관계 함수를 이용한 학습 법칙은 다음과 같고, 여기서 $\eta(n)$ 은 n 시간에서의 학습율이다.

$$W_j(n+1) = \begin{cases} W_j(n) + \eta(n)\,[\mathrm{X} - W_j(n)], & j \in \Lambda_{i(X)}(n) \\ W_j(n)\ , & otherwise \end{cases} \tag{6}$$

2.2 밖으로 밀어내는 수정된 SOFM

이동로봇의 전역경로계획에 SOFM을 적용하기 위하여 앞 절에서 거론한 원래의 SOFM 알고리듬을 수정한다. 즉, 초기의 가중치 벡터를 이미 결정된 값으로 초기화하고, 입력을 장애물 외부에 가하는 대신에 내부에 규칙적으로 가한다. 가해지는 입력에 가장 가까운 가중치벡터를 갖는 승자뉴런을 찾아서, 승자뉴런 근방의 뉴런을 입력벡터 방향으로 움직이게 하는 대신에 반대방향으로 움직이게 함으로써 가중치벡터를 재 계산한다. 이와 같은 과정을 반복하여 뉴런의 위치를 재배치 함 으로서 주어진 입력에 따른 이동로봇의 경로를 생성할 수 있다. 입력을 장애물 내부에 가하여, 가해지는 입력에 가장 가까운 가중치벡터를 갖는 승자뉴런을 찾고, 승자뉴런 근방의 뉴런들을 입력벡터 반대방향으로 움직이게 하도록 가중치벡터를 재 계산하기 위하여 식 (6)의 학습법칙을 다음과 같이 수정한다.

$$W_j(n+1) = \begin{cases} W_j(n) - \eta(n)\,[\mathrm{X} - W_j(n)], & j \in \Lambda_{i(X)}(n) \\ W_j(n)\ , & otherwise \end{cases} \tag{7}$$

수정된 Kohonen의 SOFM은 다음과 같은 단계로 학습된다.

Step 1. 초기화

초기 가중치 벡터, $W_j(0)$ 을, 기존의 SOFM에서는 아주 작은 랜덤값으로 초기화 시키는데 반하여, 이동로봇의 작업영역에 출발점과 목표점을 잇는 스트링 모양으로 배치시킨다. 그리고 학습계수, $\eta(0)$ 와 이웃관계 함수 $\Lambda_{i(X)}(0)$를 초기화한다. 두 값 모두 초기에는 큰 값을 부여한다.

Step 2. 각 장애물들에서 입력 벡터, X의 경우에 다음의 Step 2a, 2b, 2c를 수행한다.

Step 2a. network의 입력층에 입력벡터, X를 위치시킨다.

Step 2b. Similarity matching

입력벡터, X에 가장 근접한 가중치 벡터를 갖는 뉴런을 선택하여 승자뉴런으로 한다. 이는 식 (5)를 사용하여 구할 수 있다.

Step 2c. Training

식 (7)과 같이 activity bubble 내의 뉴런들을 입력벡터 반대방향으로 가중치 벡터를 훈련시킨다.

Step 3. 학습율, $\eta(n)$ 의 갱신

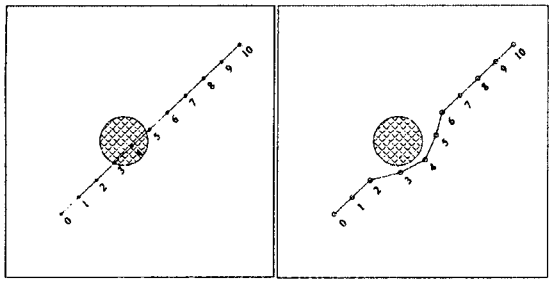
학습율의 선형감축은 만족할만한 결과를 얻도록 해야 한다.

Step 4. 이웃관계 함수, $\Lambda_{i(X)}(n)$의 감축

Step 5. 정지 조건의 확인

feature map에 식별할 수 있는 변화가 일어나지 않

는 경우에 반복계산을 정지하고, 그렇지 않으면 Step 2로 간다.



(a) (b)

Fig. 1 Example of global path planning using the string SOFM (a) before and (b) after training

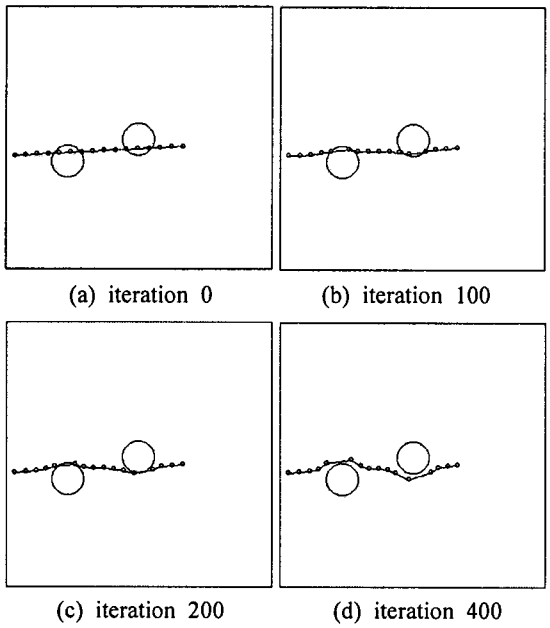
3. Global Path Planning

이동로봇의 전역 경로계획은 이미 주어진 장애물 지도를 기본으로 출발점에서 목표점까지 장애물과의 충돌을 피하면서 가장 빠르게 갈 수 있는 경로를 찾는 것이다. 즉 이동로봇의 작업 영역에서의 장애물에 대한 정보를 미리 가지고 있으므로 이를 이용하여 이동로봇의 최단경로를 찾는 것이다.

이와 같은 전역 경로계획에 기중치 벡터로서 1차원의 스트링과 2.2절에서 거론한 밖으로 밀어내는 수정된 SOFM 알고리듬을 사용한다. 이를 위하여 초기에 Fig. 1(a)와 같이 이동로봇의 작업영역에 출발점과 목표점의 위치가 결정되면 간격이 일정한 초기 스트링을 설정한다. 여기서 이동로봇의 작업영역은 사각형이라고 가정하고, 빗금 친 원은 장애물을 나타내고 있다. 이 스트링에서 출발점은 0, 목표점을 n으로 표시한다. 이 스트링을 일정간격으로 n등분 하면, 분할된 각 점은 1, 2, ---, n-2, n-1로 나타낼 수 있다. 여기서는 n=10 이고, 각 점은 0부터 10까지 표시된다. 만약 원으로 나타낸 장애물이 없다고 가정하면, 작업영역 위의 출발점 0 에서 목표점 10으로 갈 수 있는 최단 경로는 스트링 위의 점 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 그리고 10을 지나는 선이 될 것이다.

그러나 이동로봇의 경로에는 장애물이 존재하고, 앞의 2.2절에서 기술한 밖으로 밀어내는 수정된 SOFM을 이용하여 네트워크를 훈련 하면, 즉, 장애물 내부에 규칙적으로 입력이 주어지고, activity bubble 내의 뉴런은 가중치 벡터가 입력벡터 반대방향으로 주어지므로, 그 내부에 있는 점들은 점차 장애물 바깥쪽으로 빠져나가게 된다. 그 결과가 Fig. 1(b)와 같다고 하면, 실제 이동로봇을 위한 전역 경로는 훈련 하기전의 점 번호를 따라서, 훈련 후의 점 번호에 따른 좌표순서대로 지정하면 된다. 그리고 훈련 전에 해당 경로위에 장애물이 없었다면, 훈련 후에도 그 경로는 거의 변화가 없을 것이다. 즉, 작업영역 위의 출발점에서 목표점으로 갈 수 있는 최단 경로 점의 위치는 훈련 전이나 후에도 거의 변화가 없을 것이다.

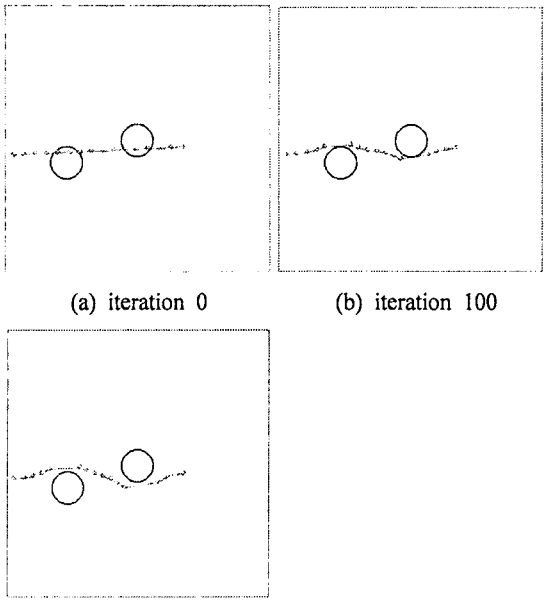
출발점 0에서 시작하여 점 1과 2는 장애물이 없기 때문에 초기와 비교하여 변화가 거의 없고, 점 3, 4, 5는 장애물을 피하는 새로운 경로점으로 위치가 변경되고, 점 6부터 10까지는 장애물이 없기 때문에 초기와 비교하여 변화가 거의 없다. 이렇게 함으로써 이미 주어진 장애물 지도를 기본으로, 출발점에서 목표점까지 장애물과 충돌을 피하면서 가장 빠르게 갈 수 있는 최단경로를 찾는 것이 가능하다.



(a) iteration 0 (b) iteration 100

(c) iteration 200 (d) iteration 400

Fig. 2 Sequential results of global path planning by using pulled SOFM with string



(a) iteration 0 (b) iteration 100

(c) iteration 200

Fig. 3 Sequential results of global path planning using pushed SOFM with string

4. 모의실험

차[7]가 제안한 당기는(pulled) SOFM을 1차원인스트링 가중치의 전역 경로계획에 적용한 순차적인 결과가 Fig. 2에서 보여주고, 본 연구에서 제안한 밀어내는(pushed) SOFM을 적용한 순차적인 결과가 Fig. 3에서 보여주고 있다. 비교를 위하여 이들 두 경우 모두 출발점과 목표점의 위치와 장애물의 위치와 크기를 동일한 조건으로 하였다. 스트링의 뉴런 개수는 둘 다 16개로 하고, 이동로봇의 작업영역에 있는 장애물은 2개를 배치하였다. 두 경우 모두 기존의 SOFM과 비교하여, 초기에 가중치 벡터를 작은 랜덤값으로 주는 대신에 작업영역에서 이동로봇의 출발점과 도착점을 스트링 양끝으로 설정하여 일정한 간격으로 배치한 것이 Fig. 2(a)와 Fig. 3(a)에 나와 있다. 이때 스트링의 양 끝에 있는 뉴런 2개는 입력벡터의 영향을 무시하도록 하고, 작업영역의 경계를 나타내는 사각형 바깥으로 스트링이 나가는 것과 안쪽으로 이동하는 것을 제한한다. 점차적으로 입력벡터를 가하는 횟수를 증가시키면, 장애물인 원 내부에 있는 스트링이 원 바깥으로 밀려나가는 결과가 얻어지는데 100회, 200회, 400회 반복의 결과가 차례로 보여진다. 100회 반복의 경우에 당기는 SOFM을 이용한 Fig. 2(b)의 경우에는 스트링이 아직 장애물 안에 있지만, 밀어내는 SOFM을 이용한 Fig. 3(c)의 경우에는 스트링이 장애물 바깥으로 거의 빠져나온 것을 보여 준다. 잡아당기는 SOFM의 경우에 Fig. 2(d)와 같이 약 400회에서 스트링이 장애물 바깥으로 빠져나오지만, 밀어내는 SOFM의 경우에 Fig. 3(c)와 같이 약 200회에서 스트링이 장애물 바깥으로 완전히 빠져나오는 것을 알 수 있다.

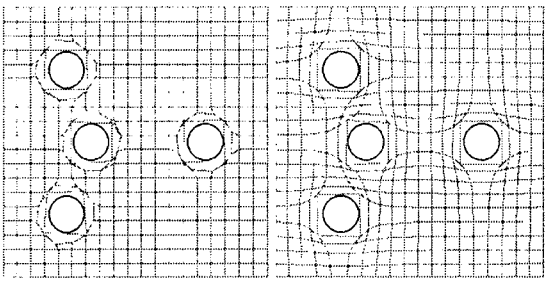
Fig. 4는 앞에서 설명한 당기는 SOFM과 밀어내는 SOFM을 2차원 격자의 전역 경로계획에 적용한 결과를 비교하여 보여주고 있다. 이동로봇의 작업영역을 사각형으로 가정하고 원형 장애물을 4개를 배치하였다. 가로와 세로의 뉴런 개수가 각각 20개인 2차원 회로망을 구성하였다. 입력벡터 방향으로 당기는 SOFM을 적용한 결과는 Fig. 4(a)에서 보여지고, 본 연구에서 제안한 입력벡터 반대방향으로 밀어내는 SOFM을 적용한 결과는 Fig. 4(b)에 나와 있다.

Fig. 4(a)의 당기는 SOFM에서는 각 물체의 바깥에 입력벡터를 원주방향으로 일정한 간격만큼 가하고, 승자뉴런 주위의 가중치벡터를 입력벡터 방향으로 끌어당겨 점차적으로 그물망을 장애물 바깥에 위치하도록 하였다. 이때 작업영역의 맨 가장자리에 있는 뉴런은 입력벡터의 영향을 무시하도록 하여, 작업영역의 경계를 나타내는 사각형 바깥으로 그물망이 나가는 것과 안쪽으로 이동하는 것을 제한한다.

Fig. 4(b)는 이와 반대로 본 연구에서 제안하는 밀어내는 SOFM을 적용한 결과로, 입력벡터를 각 물체의 내부에 원주방향으로 일정한 간격만큼 가하고, 승자뉴런 주위의 가중치벡터를 입력벡터 반대방향으로 밀어 점차적으로 그물망을 장애물 바깥으로 위치하도록 하였다. 여기서 입력벡터를 가하는 회수를 반경방향으로 증가시키면, 장애물인 원 내부에 있는 그물망이 원 바깥으로 밀려나가는 결과가 얻어진다.

이들 결과를 비교하면, 당기는 SOFM에서는 장애물 주위의 가중치 벡터가 서로 겹치고 그물망이 왜곡되어 있는 것을 알 수 있다. 반면에 본 연구에서 제안한 밀어내는 SOFM에서는 장애물 주위의 가중치 벡터가 서로 겹치지 않고 그물망이 규칙적으로 퍼져 있는 것을 알 수 있다. 특히 장애물에서

떨어진 곳에서도 그물망이 유연하게 펼쳐져 있다. 이동로봇의 전역 경로계획에서는 그물망이 유연하게 펼쳐있는 것이 이동속도의 연속적인 구현이라는 관점에서 에너지소모를 줄일 수 있기 때문에 유리하다.



(a) (b)

Fig. 4 Results of global path planning by using (a) pulled SOFM and (b) pushed SOFM in environment with 4 obstacles

Fig. 5는 Fig. 4의 각각의 경우에 좌 하단을 원점으로 가정하여, 원점에서 각 node들까지의 총 이동거리를 3차원으로 보여주고 있다. 이 그림에서 음영의 변화는 전체 이동거리를 등고선 형태로 보여주는 것이다. 두 경우 모두 장애물에 영향을 받지 않는 좌, 우측의 경우는 SOFM을 적용하기 전과 같이 아무런 변화가 없다. 그러나 장애물 바로 앞의 node까지 거리는 mesh의 변형에 따라서 최초의 상태보다 총 이동거리가 감소하고, 장애물 뒤에서는 증가하는 것을 알 수 있다.

Fig. 6은 Fig. 4의 경우 각각에 SOFM을 적용한 후와 적용 전의 원점에서 각 노드들까지의 총 이동거리 차를 3차원으로 보여주고 있다. 장애물 앞의 노드까지 총 이동거리 차는 음수를 갖고, 장애물 뒤는 양수를 갖는 것을 알 수 있다. 또한 장애물 좌, 우의 노드에서는 그 차가 0이 되는 것을 알 수 있다. 결과적으로 본 연구에서 제안한 밀어내는 SOFM 알고리듬은 당기는 SOFM 알고리듬보다 이동로봇의 전역 경로계획에 더욱 효과적으로 사용될 수 있음을 알 수 있다.

Fig. 7은 Fig. 4와 같이 장애물이 4개인 경우에 당기는 SOFM과 미는 SOFM의 적용에 따른 반복회수와 장애물 내에 있는 노드의 총 개수를 보여주고 있다. 당기는 SOFM의 적용에 따른 결과에서 170 반복 이후에 0으로 근접되는 것을 알 수 있다. 반면에 밀어내는 SOFM의 적용에 따른 결과는 당기는 SOFM의 적용에 따른 결과보다 훨씬 빠른 40 반복 이후에 0으로 근접되는 것을 알 수 있다. 결과적으로 밀어내는 SOFM이 당기는 SOFM보다 경로 왜곡 해소뿐만 아니라, 계산회수도 더 적게 요구되어 더 효과적임을 보여준다.

Fig. 8은 밀어내는 SOFM을 사각형 장애물이 있는 경우에 적용한 결과를 보여주고 있다. 여기서도 Fig. 4의 원형장애물과 같이 그물망이 유연하게 펼쳐지는 것을 알 수 있다.

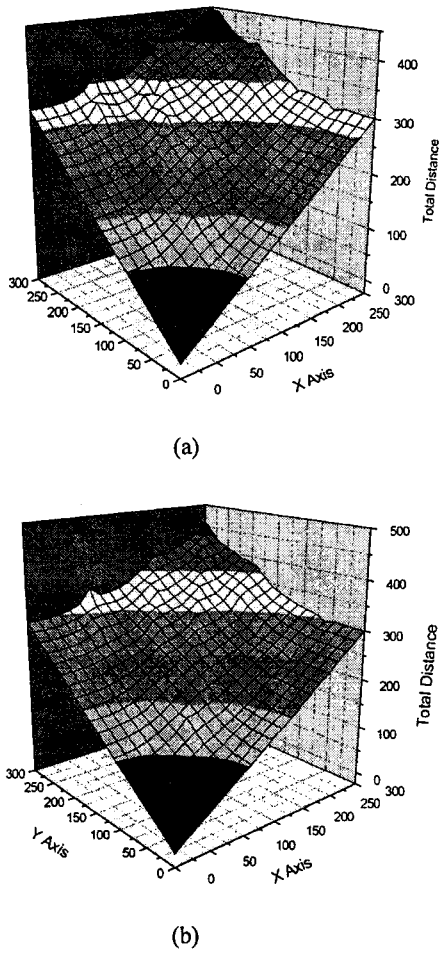
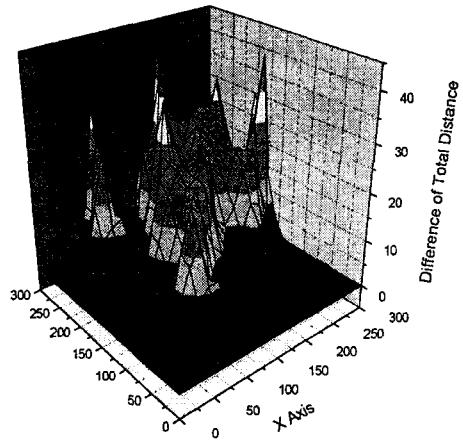
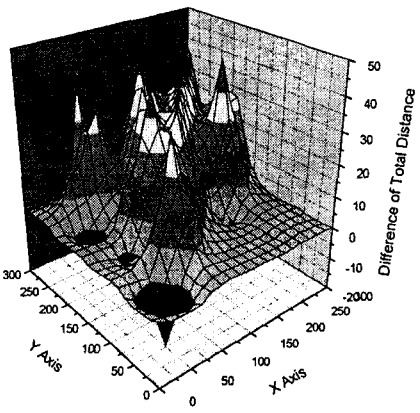


Fig. 5 Total distance from origin point to each node using (a) pulled SOFM and (b) pushed SOFM in environment with 4 obstacles



(a)



(b)

Fig. 6 Differences of total distance between origin and deformed points using (a) pulled SOFM and (b) pushed SOFM in environment with 4 obstacles

5. 결론

본 연구에서는 이동로봇의 전역 경로계획을 위하여 신경회로망의 하나인 Kohonen의 SOFM을 수정하고, 스트링을 사용하였다. 초기의 가중치 벡터를 이미 결정된 값으로 초기화하고, 입력을 장애물 내부에 가하여, 가해지는 입력에 가장 가까운 가중치 벡터를 갖는 승자뉴런을 찾아서, 승자뉴런 근방의 뉴런들을 입력벡터 반대방향으로 움직이게 함으로서 가중치벡터를 재 계산하였다. 이와 같은 과정을 반복하여 뉴런의 위치를 재배치함으로서 주어진 입력에 따른 이동로봇의 경로를 생성할 수 있었다. 제안된 전역 경로계획 알고리듬의 효율성을 입증하기 위하여 장애물이 있는 환경에서 모의실험을 통하여 이미 발표된 당기는 SOFM과 본 연구에서 제안한 밀어내는 SOFM 알고리듬을 이용한 결과를 비교분석하였다. 결론적으로 당기는 SOFM을 이용한 방법에서 가중치들끼리 위치가 일정하게 펼쳐지지 않아서 생기는 경로왜곡 문제를 본 연구에서 제안한 밀어내는 SOFM을 이용한 방법에서 해결하였고, 계산회수도 훨씬 더 적게 요구되어 보다 더 효과적임을 보였다.

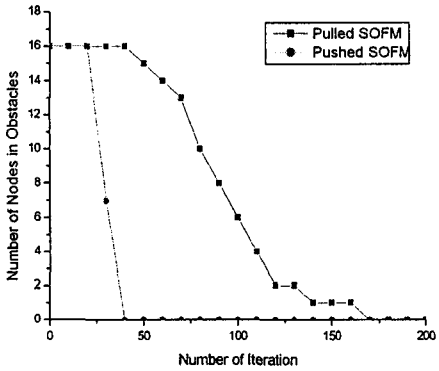


Fig. 7 The number of remained nodes in obstacles according to the number of iteration in environment with 4 obstacles using pulled and pushed SOFM algorithm

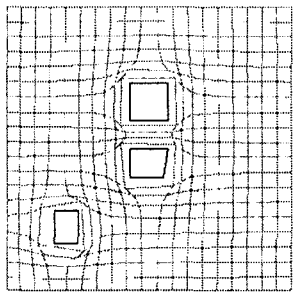


Fig. 8 Result of global path planning by using pushed SOFM in rectangle obstacles

후 기

이 논문은 2006년도 원광대학교의 교비 지원에 의해서 수행됨

참고문헌

1. Qunjie, D. and Mingjun, Z., "Local Path Planning Method for AUV Based on Fuzzy-neural Network," SHIP ENGINEERING, Vol. 1, pp. 54-58, 2001.
2. Cha, Y. Y., "Navigation of a free ranging mobile robot using heuristic local path planning algorithm," Robotics and Computer Integrated Manufacturing, Vol. 13, No. 2, pp. 145-156, 1997.
3. Zhu, Y., Chang, J. and Wang, S., "A new path-planning algorithm for mobile robot based on neural network," TENCOM Proceedings, IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering, Vol. 3, pp. 1570-1573, 2002.
4. Bourbakis, N. G., "Path Planning in a 2-D Known Space Using Neural Networks and Skeletonization," Conference proceedings : IEEE International Conference on Systems, Man and Cybernetics, Vol. 3, pp. 2001-2005, 1997.
5. Chaiyaratana, N. and Zalzala, A. M. S., "Time-Optimal Path Planning and Control using Neural Networks and a Genetic Algorithm," International Journal of Computational Intelligence and Applications, Vol. 2, No. 2, pp. 153-172, 2002.
6. Cha, Y. Y. and Gweon, D. G., "The development of a free ranging mobile robot equipped with a structured light range sensor," Intelligent Automation and Soft Computing, Vol. 4, No. 4, pp. 289-312, 1998.
7. Cha, Y. Y. and Jeong, S. M., "Self-organizing Feature Map for Global Path Planning of Mobile Robot," Journal of the Korean Society for Precision Engineering, Vol. 23, No. 3, pp. 94-101, 2006.
8. Kohonen, T., "The self-organizing map," Proceedings of the Institute of Electrical and Electronics Engineers, Vol. 78, No. 9, pp. 1464-1480, 1990.