수정된 SOFM을 이용한 이동로봇의 전역 경로계획

A Global Path Planning of Mobile Robot Using Modified SOFM

차 영 엽*, 유 대 원, 정 세 미
(Young-Youp Cha, Dae-Won Yu, and Se-Mi Jeong)

Abstract : A global path planning algorithm using modified self-organizing feature map(SOFM) which is a method among a number of neural network is presented. The SOFM 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 move toward the input vector. On the other hand, the modified method in this research uses a predetermined initial weight vectors of the 2-dimensional mesh, 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 modified neural network is useful tool for the global path planning problem of a mobile robot.

Keywords : global path planning, self-organizing feature map(SOFM), neural network, mobile robot

I. 서론

경로계획은 이동로봇이 목표점에 도달하기 위해서 그 사이의 경로를 여러 개의 기본 운동형태(basic motion)로 나누는 것이다[1-5]. 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.

여기서 전역 경로계획은 형상공간 방법(configuration space method), 포텐셜 방법(potential approach), 그리고 퍼지, 신경회로망, 그리고 유전자 알고리듬에 기초한 인공지능 알고리듬이 이동로봇의 경로계획에 적용되었다. 형상 공간 방법의 경우, Lozano-Perez[1]는 V-그래프(Visibility graph)에 의한 다각형들로 이루어진 환경에서 이동로봇을 한점으로 간주한 경로문제를 처음으로 다루었다. 그러나 이러한 V-그래프는 이동로봇의 주위환경뿐만 아니라 그 크기에도 영향을 받는다. 이러한 단점을 해결하기 위하여 Noborio[2]는 환경을 quadtree로 모델링하는 효과적인 방법을 제안하였다. 그러나 이러한 quadtree는 주위환경의 근사적인 표현일 뿐만 아니라 설정된 좌표계에 의존한다. 또한, 이러한 형상 공간 방법은 계산시간을 많이 요구한다.

포텐셜 방법의 경우, Brooks[3]과 Adams[4]는 반력을 장애물과 상사(identify)시키고, 여기에 목표점 쪽으로 인력을 첨가하여 경로생성을 하였다. 따라서 이동로봇은 그들의 합력벡터 방향으로 운동을 한다. Borenstein[5]은 반력과 인력의 합력벡터 뿐만 아니라, 환경을 사각형으로 분할하고 각 사각형을 장애물로 판단될 확률로 결합하였다. 이 확률에 기초하여, 인공적인 힘을 계산하였다. 그러나 이러한 인공적인 힘의 방향과 크기는 실시간으로 구현하였지만, 실재로 인간이 장애물을 피해서 목표점에 도달하는 경험적 방식과는 큰 차이가 있다.

다른 한편으로 인공지능 알고리듬을 로봇의 전역 경로계획에 사용하려고 하는 노력이 최근에 있었다. 퍼지, 신경회로망, 유전자 알고리듬 또는 그들을 함께 사용하는 것이 그것인데, Qunjie[6]는 퍼지 알고지즘을 사용하여 지역 경로계획 문제[7]를 다루었고, Zhu[8]는 신경회로망과 함께 cost 함수로 충돌 에너지함수를 이용하였다. Bourbakis[9]는 세선화와 신경회로망을 이용하였으며, Chaiyaratana[10]는 유전자 알고리듬을 이용하여 경로계획 문제를 해결하였다. 그러나 이러한 노력에도 불구하고 아직도 지역 또는 전역 경로계획문제는 완전히 해결이 되지 않은 실정이다.

이동로봇[11]의 자율주행을 위하여 신경회로망 중에서 SOFM을 이용한 전역 경로계획 알고리듬을 차[12]가 최초로 제안한바 있다. 기존의 SOFM 알고리듬[13]은 초기 가중치 벡터를 아주 작은 랜덤값으로 주고 모든 가중치 벡터를 학습시키며, 입력벡터를 작업영역 전체에 랜덤하게 분포시킨다. 이에 반하여, 차[12]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

* 책임저자(Corresponding Author)
논문접수 : 2005. 7. 8., 채택확정 : 2006. 1. 22.
차영엽 : 원광대학교 기계자동차공학부(ggypcha@wonkwang.ac.kr)
유대원, 정세미 : 원광대학교 대학원
(ekqh@wonkwang.ac.kr/oriolehot@nate.com)
※ 이 논문은 2005년도 원광대학교의 교비 지원에 의해서 수행됨.

재 계산한다. 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.

이 연구에서는 차[12]의 결과에서 발생하는 경로 왜곡 문제를 해결하기 위하여 수정 알고리듬을 제안한다. 즉, 차[12]는 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 재 계산한다. 이와 같은 과정을 반복하여 뉴런의 위치를 재배치함으로서 주어진 입력에 따른 이동로봇의 경로를 왜곡 없이 생성할 수 있음을 보인다. 제안된 전역 경로계획 알고리듬의 효율성을 입증하기 위하여, 차[12]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, 경로왜곡 해소뿐만 아니라 계산회수도 훨씬 더 적게 요구되는 등 후자가 더 효과적임을 보여준다.

II. SOFM

본 연구는 이동로봇의 자율주행을 위하여 신경회로망 중에서 SOFM을 이용한 전역 경로계획 알고리듬에 대한 것이다. 그 중에서도 Kohonen[13]에 의해 개발된 SOFM을 기본으로 하였다. SOFM 회로망에서는 기대된 출력이 없다. 대신에 신경 회로망은 자기 조직화 특성에 의해 임의의 추상적인 관계를 추론할 수 있으며, 더 많은 입력이 인가 될수록 회로망은 그 학습을 개선하고 변화된 입력들에 적응하여 출력을 내보낸다. 이러한 구조의 한 이점으로는 변화하는 상태와 입력에 대해 대처할 수 있다는 것이다. 그러므로 이 회로망은 입력을 다른 카테고리로 분류할 때와 음성 인식, 로봇 제어 등에 사용된다. 결과적으로 SOFM의 목적은 N-차원의 입력 공간을 의미 있는 지형학적인 순서로 1차원 또는 2차원의 출력 공간 즉 출력 뉴런에 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[13].

이러한 SOFM 알고리듬은 다음과 같다. 입력 벡터 X와 j 번째 출력층 뉴런과 상응하는 가중치 벡터 W_j 를

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

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

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

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

여기서 η 는 학습율(learning rate)을 나타내고, i 는 1에서 p 까지 입력 뉴런(neuron)의 수를 나타내며, j 는 1에서 N 까지 출력 뉴런의 수이다. 그리고 w_{ji} 는 i 번째 입력 뉴런과 j 번째 출력 뉴런을 연결하는 가중치를 나타낸다.

입력 벡터의 분류를 위한 출력 층의 승자 뉴런 결정은 입력 X와 가장 비슷한 가중치 W_j 를 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_j 를 선택하는 것이고, 두 번째는 입력 벡터와 최소의 Euclidean norm을 갖는 가중치를 선택하는 것이다. 즉, $i(X)$ 가 승자 뉴런이라 한다면 이 식은 다음과 같다.

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

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(normalization) 시키는데, 그 이유는 학습규칙이 입력벡터로부터 가중치 벡터를 뺀 값을 사용하기 때문이다.

그리고, 좀더 효율적인 패턴 분류를 위하여 측면 제어를 이용한다. 측면 제어의 대표적인 방법은 이웃관계 함수(neighborhood function), $\Lambda_{i(X)}(n)$의 사용으로서, 승자 뉴런이 선택되면 승자 뉴런의 이웃하는 거리에 따라 연결강도를 달리하는 방법으로 연결 강도는 거리에 반비례한다. 이웃관계 함수를 이용한 학습 법칙은 다음과 같다.

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

여기서 $\eta(n)$ 은 n 시간에서의 학습율이다.

그러나 본 연구에서는 입력을 장애물 내부에 가하여, 가해지는 입력에 가장 가까운 가중치벡터를 갖는 승자뉴런을 찾는다. 그리고 승자뉴런 근방의 뉴런들을 입력벡터 반대방향으로 움직이게 하도록 가중치벡터를 재 계산하기 위하여 (6)의 학습법칙을 다음과 같이 수정한다.

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

III. 전역 경로계획

1. 수정된 SOFM

본 연구에서는 1장과 2장에서 거론한 것처럼 기존의 SOFM 알고리듬을 전역 경로계획에 적용하기 위하여 수정한다. 즉, 초기의 가중치 벡터를 그물망처럼 이미 결정된

값으로 초기화하고, 입력을 장애물 내부에 규칙적으로 가하여, 가해지는 입력에 가장 가까운 가중치벡터를 갖는 승자뉴런을 찾아서, 승자뉴런 근방의 뉴런들을 입력벡터 반대방향으로 움직이게 함으로서 가중치벡터를 재 계산한다. 이와 같은 과정을 반복하여 뉴런의 위치를 재배치함으로써 주어진 입력에 따른 이동로봇의 전역경로를 생성할 수 있다.

이와 같이 수정된 SOFM 알고리듬은 다음과 같은 단계로 표현된다.

Step 1: 초기화-초기 가중치 벡터, $W_j(0)$을, 기존의 SOFM에서는 아주 작은 랜덤값으로 초기화 시키는데 반하여, 이동로봇의 작업영역에 격자모양으로 배치시킨다(그림 1). 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: 입력-회로망의 입력층에 입력벡터, X를 위치시킨다. 이때 입력은 장애물 내부에 규칙적으로 가한다.

Step 2b: Similarity matching-입력벡터, X에 가장 근접한 가중치 벡터를 갖는 뉴런을 선택하여 승자뉴런으로 한다. 이는 (5)를 사용하여 구할 수 있다.

Step 2c: 학습-(7)과 같이 activity bubble 내의 뉴런들을 입력벡터 반대방향으로 가중치 벡터를 학습시킨다. 단, 가장자리의 가중치 벡터는 학습에 관계없이 고정되도록 한다.

Step 3: 학습율, $\eta(n)$ 의 갱신-학습율의 선형감축은 만족할만한 결과를 얻도록 해야 한다.

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

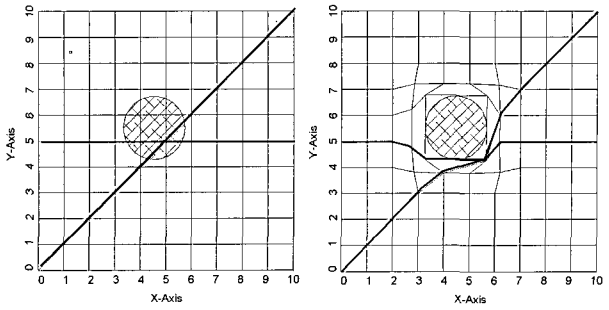
Step 5: 반복정지 조건의 확인-전역 경로계획을 위한 회로망에 식별할 수 있는 변화가 일어나지 않는 경우에 반복계산을 정지하고, 그렇지 않으면 Step 2로 간다.

2. 전역 경로계획 알고리듬

본 연구에서는 전역 경로계획에 앞에서 거론한 수정된 SOFM을 사용한다. 이를 위하여 초기에 Fig. 1(a)와 같이 이동로봇의 작업영역에 위치가 결정된 초기 그물망을 설정한다. 여기서 이동로봇의 작업영역은 사각형이라고 가정하고, 원은 장애물을 나타내고 있다. 작업영역의 수평축을 X축, 그리고 수직축을 Y축이라 하면, 그물망 위의 한 점은 (X, Y)로 나타낼 수 있다.

만약 원으로 나타낸 장애물이 없다고 가정하면, 작업영역 위의 한점 (0, 5)에서 다른 한점 (10, 5)로 갈 수 있는 최단 경로는 그물망 위에 굵은 선으로 표시되어 있다. 즉, (0, 5), (1, 5), --, (10, 5)를 지나는 선이 될 것이다. 같은 방법으로, 작업영역 위의 한점 (0, 0)에서 다른 한점 (10, 10)으로 갈 수 있는 최단 경로는 그물망 위에 굵은 선으로 표시되어 있다. 즉, (0, 0), (1, 1), --, (10,10)을 지나는 선이 될 것이다.

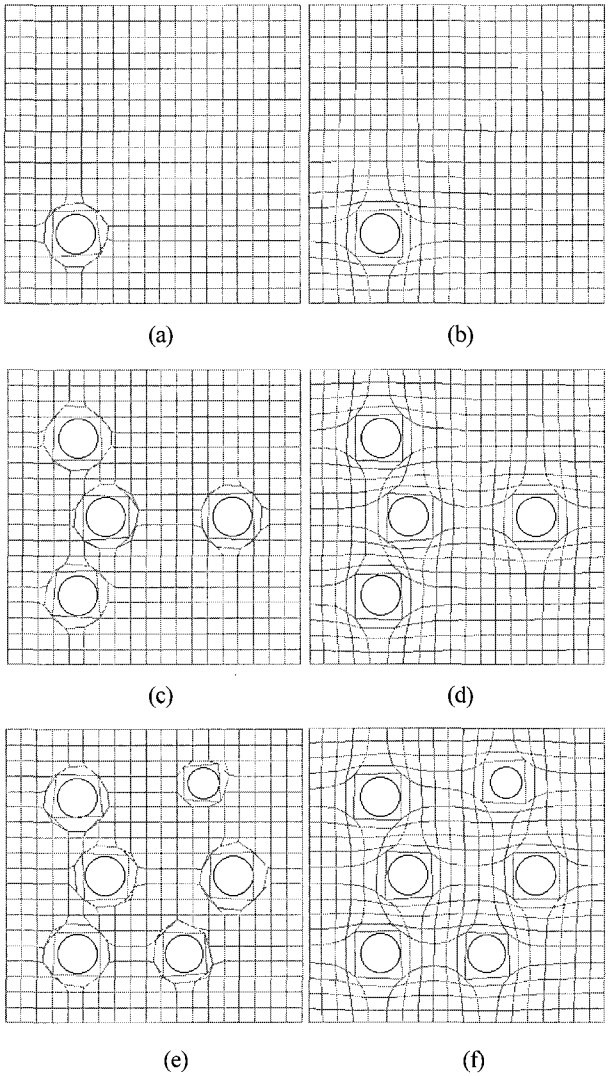
그리고 앞의 3.1절에서 기술한 수정된 SOFM을 이용하여 회로망을 학습시키기 위하여, 장애물 내부에 규칙적으로 입력이 주어지고, activity bubble 내의 뉴런은 가중치 벡터 증분이 입력벡터 반대방향으로 주어지므로, 그 내부에 있는 그물망들은 점차 장애물 바깥쪽으로 빠져나가게 된다. 그 결과가 Fig. 1(b)와 같다고 하면, 실제 이동로봇을 위한 전



(a) (b)

그림 1. SOFM을 이용한 (a) 학습 전과 (b) 후의 예.

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



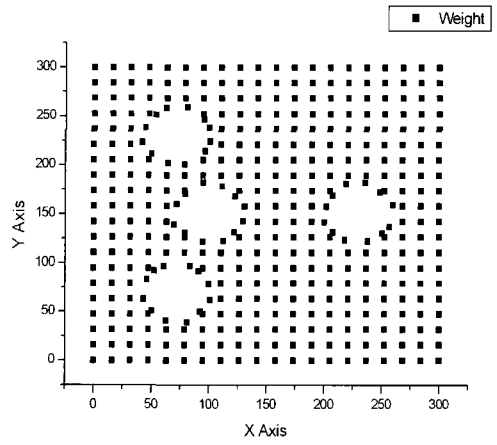
(a) (b)

(c) (d)

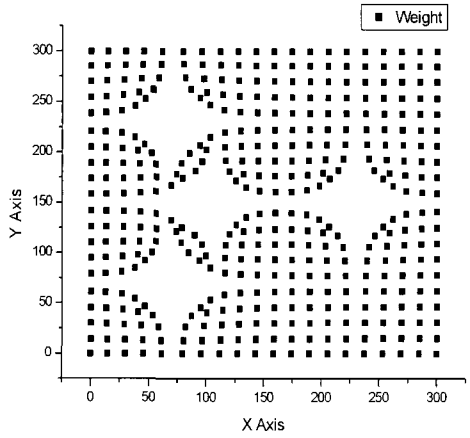
(e) (f)

그림 2. 장애물이 1개, 4개, 6개인 경우에 (a)(c)(e) 당기는 SOFM과, (b)(d)(f) 밀어내는 SOFM을 적용한 후의 전역경로계획 결과.

Fig. 2. Results of global path planning by using (a)(c)(e) pulled SOFM and (b)(d)(f) pushed SOFM in environment with 1, 4, and 6 obstacles.



(a)



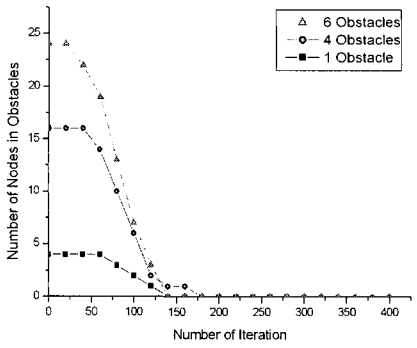
(b)

그림 3. 장애물이 4개인 경우에 (a) 당기는 SOFM과 (b)밀어내는 SOFM을 적용한 후의 가중치 위치.

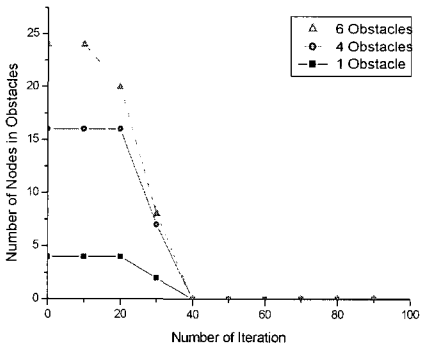
Fig. 3. The location of weights after global path planning by using (a) pulled-SOFM and (b) pushed-SOFM in environment with 4 obstacles.

역 경로는 학습 하기전의 좌표 순서를 따라서, 굵은 선으로 표시되어 있는 것처럼 학습 후의 좌표순서대로 따라가면 된다. 그리고 학습 전에 해당 경로위에 장애물이 없었다면, 학습 후에도 그 경로는 거의 변화가 없음을 알 수 있다. 즉, 작업영역 위의 한점 (0, 1)에서 다른 한점 (10, 1)로 갈 수 있는 최단 경로 점의 위치는 학습 전이나 후에도 거의 변화가 없음을 알 수 있다.

또한 작업영역 위의 한점 (0, 5)에서 다른 한점 (10, 5)로 갈 수 있는 경로를 보면, 출발과 도착부근에서는 그물망의 위치변화가 거의 없지만, 장애물이 있는 중간부의 위치변화는 상당한 것으로 보여 진다. 즉 출발부의 (0, 5), (1, 5), (2, 5)와 도착부의 (7, 5), (8, 5), (9, 5), (10, 5)의 위치는 학습 전과 후에 거의 변화가 없지만, 장애물이 있는 중간부, 즉 (3, 5), (4, 5), (5, 5), (6, 5)는 그물망의 위치변화가 상당히 나타난다. 이렇게 함으로서 이미 주어진 장애물 지도를 기본으로 출발점에서 목표점까지 장애물과 충돌을 피하면서 빠르게 갈 수 있는 최단경로를 찾는 것이 가능하다.



(a)



(b)

그림 4. 장애물이 1개, 4개, 6개인 경우에 (a) 당기는 SOFM과 (b) 밀어내는 SOFM의 적용에 따른 반복 회수와 장애물 내에 있는 노드의 총 개수.

Fig. 4. The number of remained nodes in obstacles according to the number of iteration in environment with 1, 4, and 6 obstacles by using (a) pulled and (b) pushed SOFM algorithm.

IV. 실험결과

Fig. 2는 앞에서 설명한 SOFM을 수정하여 전역 경로계획에 적용한 결과를 비교하여 보여주고 있다. 이동로봇의 작업영역을 사각형으로 가정하고 원형 장애물을 1개, 4개, 6개를 배치하였다. 가로와 세로의 뉴런 개수가 각각 20개인 2차원 회로망을 구성하였다. 입력벡터 방향으로 당기는 SOFM[12]을 적용한 결과는 (a), (c), (e)에서 보여 지고, 본 연구에서 제안한 입력벡터 반대방향으로 밀어내는 SOFM을 적용한 결과는 (b), (d), (f)에 나와 있다. 이들 결과는 원래의 SOFM과 비교하여, 초기에 가중치 벡터를 작은 랜덤값으로 주는 대신에 그물망처럼 전체 작업영역에 일정한 간격으로 배치하여 얻은 것이다.

당기는 SOFM에서는 각 물체의 바깥에 입력벡터를 원주방향으로 일정한 간격만큼 가하고, 승자뉴런 주위의 가중치 벡터를 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. 이와 반대로 본 연구에서 제안하는 밀어내는 SOFM에서는 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. 장애물이 한 개인 경우에 Fig. 2(a)의 당기는 SOFM과 Fig. 2(b)의 밀어내는 SOFM을 적용한 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. 장애물을 4개 그리고 6개로 늘린 경우에 당기는 SOFM과 밀어내는 SOFM을 적용한 결과가 각각의 Fig. 2(c)와 (d), 그리고 Fig. 2(e)와 (f)에 나와 있고 비슷한 결과를 보여준다.

Fig. 3은 장애물이 4개인 Fig. 2(c)와 (d)의 경우에 가중치 위치를 보여주고 있다. 앞에서 거론한 것처럼 Fig. 3(a)의 당기는 SOFM을 적용한 결과 보다는 Fig. 3(b)의 밀어내는 SOFM을 적용한 결과에서 훨씬 더 가중치의 위치가 규칙적이고 유연하게 분포되어 있는 것을 알 수 있다.

Fig. 4는 Fig. 2와 같이 장애물이 1개, 4개, 6개인 경우에 수정된 SOFM의 적용에 따른 반복 회수와 장애물 내에 있는 노드의 총 개수를 보여주고 있다. Fig. 4(a)의 당기는 SOFM의 적용에 따른 결과에서 장애물의 개수가 많을수록 장애물 내에 있는 노드의 총 개수는 더 큰 값을 갖고, 150 반복 이후에 세 경우 모두 0으로 근접되는 것을 알 수 있다. 150 반복 근처에서 장애물이 6개인 경우가 4개인 경우보다 더 빨리 수렴하는 것을 보여주는데, 이는 주변 장애물에서의 입력이 근처에 있는 다른 장애물의 노드에 순방향 영향을 끼친 결과로서, 장애물의 상대위치에 따른 제한적인 결과로 보여 진다. Fig. 4(b)의 밀어내는 SOFM의 적용에 따른 결과에서, Fig. 4(a)보다 훨씬 빠른 40 반복 이후에 세 경우 모두 0으로 근접되는 것을 알 수 있다.

Fig. 5는 Fig. 2의 장애물이 1개, 4개, 6개인 경우에 당기는 SOFM과 밀어내는 SOFM의 적용에 따른 반복 회수와 전체 노드의 이동거리 총합을 보여주고 있다. Fig. 5(a)의 당기는 SOFM의 경우에 장애물이 많을수록 이동거리의 총합은 커지고, 장애물이 1개, 4개, 6개인 경우 모두 50 반복까지는 이동거리의 총합이 급격히 변하다가, 그 이후 300 반복까지는 점차적으로 감소하고, 300 반복 이후에는 이동거리의 총합이 세 경우 모두 0으로 근접되는 것을 볼 수 있다. Fig. 5(b)의 밀어내는 SOFM의 경우에 장애물이 1개, 4개, 6개인 경우 모두 20 반복 까지는 이동거리의 총합이 급격히 변하다가, 그 이후부터는 점차적으로 감소하는 것을 볼 수 있다. 여기서 90 반복 까지만 계산한 것은 이미 Fig. 4(b)에서 거론 한 것처럼, 40 반복에서 장애물 내에 노드가

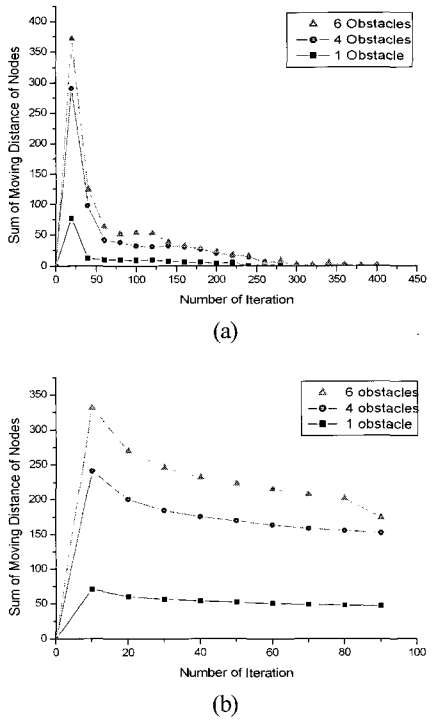


그림 5. 장애물이 1개, 4개, 그리고 6개인 경우에 (a) 당기는 SOFM과 (b) 밀어내는 SOFM의 적용에 따른 반복 회수와 노드의 이동거리 총합.

Fig. 5. Sum of moving distance according to the number of iteration in environment with 1, 4, and 6 obstacles by using (a) pulled and (b) pushed SOFM algorithm.

하나도 남아있지 않기 때문이다. 결과적으로 잡아당기는 SOFM 알고리듬을 적용한 결과[12] 보다는 본 연구에서 제안한 밀어내는 SOFM을 적용한 결과가 훨씬 더 빨리 수렴하는 것을 알 수 있다.

Fig. 6은 Fig. 2(d)와 같이 장애물이 4개인 경우에 밀어내는 SOFM 알고리듬을 적용한 후, 원점에서 각 노드들까지의 총 이동거리를 3차원으로 보여주고 있다. 이 그림에서 음영의 변화는 전체 이동거리를 등고선 형태로 보여주고 있다. 장애물에 영향을 받지 않는 좌, 우측의 경우는 SOFM을 적용하기 전과 같이 아무런 변화가 없다. 그러나 장애물 바로 앞의 노드까지 거리는 그물망의 변형에 따라서 최초의 상태보다 총 이동거리가 줄어들고, 장애물 뒤에서는 늘어나는 것을 알 수 있다.

Fig. 7은 Fig. 2(d)와 같이 장애물이 4개인 경우에 밀어내는 SOFM 알고리듬을 적용한 후와 반복하기 전의 원점에서 각 노드들까지의 총 이동거리 차를 3차원으로 보여준다.

Fig. 6에서 거론한 것처럼 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 알고리듬[12]보다 이동로봇의 전역 경로계획에 더욱 효과적으로 사용될 수 있음을 알 수 있다.

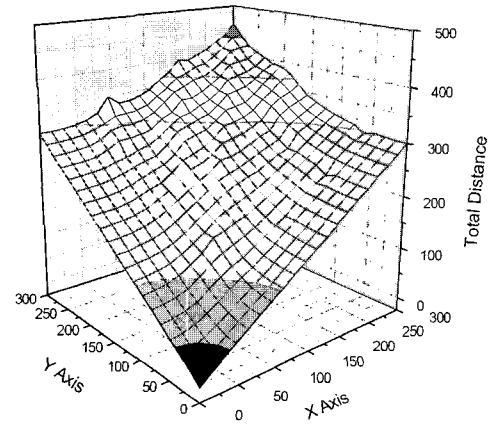


그림 6. 장애물이 4개인 경우에 밀어내는 SOFM을 적용한 후, 원점에서 각 노드들까지의 총 거리.

Fig. 6. Total distance from origin point to each node by using pushed SOFM in environment with 4 obstacles.

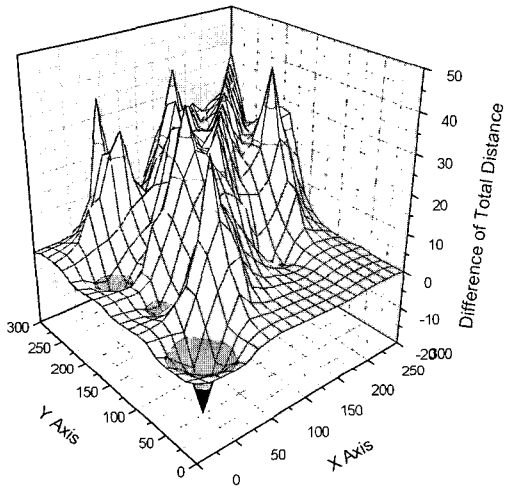


그림 7. 장애물이 4개인 경우에 밀어내는 SOFM을 적용한 후와 적용 전의 원점에서 각 노드들까지의 거리 차.

Fig. 7. Difference of total distance from origin point between applied to pushed SOFM and before with 4 obstacles.

V. 결론

본 연구에서는 신경회로망의 하나인 Kohonen의 SOFM을 전역 경로계획에 적용하기 위하여 수정하였다. 즉, 초기의 가중 벡터를 이미 결정된 값으로 초기화하고, 입력을 장애물 내부에 가하여, 가해지는 입력에 가장 가까운 가중벡터를 갖는 승자뉴런을 찾아서, 승자뉴런 근방의 뉴런들을 입력벡터 반대방향으로 움직이게 함으로서 가중치벡터를 재계산하였다. 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 SOFM[12]과 본 연구에서 제안한 밀어내는 SOFM 알고리듬을 이용한 결과를 비교분석하였다. 결론적으로 당기는 SOFM을 이용한 방법에서 나타나는 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.

참고문헌

[1] T. Lozano-Perez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedrial obstacles," *Commun. ACM*, pp. 560-570, 1979.

[2] H. Noborio, T. Naniwa, and S. Arimoto, "A fast path planning algorithm by synchronizing modification and search of its path-graph," *Proc. IEEE Intern. Workshop on Artificial intelligent for Industrial Application*, pp. 351-357, 1988.

[3] R. Brooks, "Solving the find path problems by good representation of free space," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-13, no. 3, pp. 190-197, 1983.

[4] M. D. Adams and P. J. Probert, "Towards a real-time navigation strategy for a mobile robot," *Proc. of the IEEE Intern Workshop on Intelligent Robots and Systems*, pp. 743-748, 1990.

[5] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Trans. on Robotics and Automation*, no. 3, pp. 278-298, 1991.

[6] D. Qunjie, Z. Mingjun, "Local path planning method for AUV based on fuzzy-neural network," *SHIP ENGINEERING*, vol. 1, pp. 54-58, 2001.

[7] Y. Y. Cha, "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.

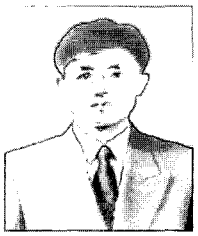
[8] Y. Zhu, J. Chang, and S. Wang, "A new path-planning algorithm for mobile robot based on neural network," *TENCOM '02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, vol. 3, pp. 1570-1573, 2002.

[9] N. G. Bourbakis, D. Goldman, R. Fematt, I. Vlachavas, and L. H. Tsoukalas, "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.

[10] N. Chaiyaratana and A. M. S. Zalzala, "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.

[11] Y. Y. Cha and D. G. Gweon, "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.
[12] 차영엽, 강현규, "Self-organizing feature map를 이용한 이동로봇의 전역 경로계획," 제어·자동화·시스템 공학회지, 제11권, 제2호, pp. 137-143, 2005.
[13] T. Kohonen, "The self-organizing map," *Proc. of the IEEE*, vol. 78, no. 9, pp. 1464-1480, 1990.



차 영 엽

1961년 11월 18일생. 1984년 부산대 기계공학과 졸업. 1987년 한국과학기술원 생산공학과(석사). 1995년 한국과학기술원 정밀공학과 박사. 1995년~현재 원광대학교 기계공학부 부교수. 관심분야는 이동로봇, 영상처리.

정 세 미

1983년 1월 11일생. 2005년 원광대 기계공학과 졸업. 현재 원광대 생산설계 석사과정. 관심분야는 나노 포지션닝.

유 대 원

1980년 6월 1일생. 2005년 원광대 기계공학과 졸업. 현재 원광대 생산설계 석사과정. 관심분야는 이족보행로봇.