전역경로계획을 위한 단경로 스트링에서 당기기와 밀어내기 SOFM을 이용한 방법의 비교

The Comparison of Pulled- and Pushed-SOFM in Single String for Global Path Planning

차 영 엽*, 김 곤 우
(Young-Youp Cha and Gon-Woo Kim)

Abstract : This paper provides a comparison of global path planning method in single string by using pulled and pushed SOFM (Self-Organizing Feature Map) which is a method among a number of neural network. The self-organizing feature map 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 SOFM method in this research uses a predetermined initial weight vectors of the one 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 or reverse the input vector, by using a pulled- or a pushed-SOFM. According to simulation results one can conclude that the modified neural networks in single string are useful tool for the global path planning problem of a mobile robot. In comparison of the number of iteration for converging to the solution, the pushed-SOFM is more useful than the pulled-SOFM in global path planning for mobile robot.

Keywords : mobile robot, self-organizig feature maps, neural network

I. 서론

이동로봇이 목표점에 도착하기 위하여 그 사이의 경로를 여러 개의 기본 운동형태로 나누는 것을 경로계획이라고 한다. 이와 같은 경로계획은 크게 전역 경로계획과 지역 경로계획으로 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[1-5]. 전역 경로 계획은 형상공간 방법(configuration space method), 포텐셜 방법(potential approach), 그리고 퍼지, 신경회로망, 유전자 알고리즘에 기초한 인공지능 알고리즘이 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 모델링 하는 효과적인 방법을 제안 하였다. 그러나 이러한 quadtree는 주위환경의 근사적인 표현일 뿐만 아니라 설정된 좌표계에 의존한다. 또한, 이러한 형상공간 방법은 계산시간을 많이 요구한다.

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

인공지능 알고리즘을 로봇의 전역 경로계획에 사용하려고 하는 노력이 최근에 있었다. 퍼지, 신경회로망, 유전자 알고리즘 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 이용하였고, Bourbakis[9]는 skeletonization과 신경회로망을 이용하였으며, Chaiyaratana [10]는 유전자 알고리즘을 이용하여 경로계획 문제를 해결하였다. 그러나 이러한 노력에도 불구하고 아직도 지역 또는 전역 경로계획문제는 계산시간, 이동물체 회피, 간단성 등의 문제점이 있다.

본 연구는 이동로봇[11]의 자율주행을 위하여 1차원 단경로 스트링(string)을 신경회로망 중에서 SOFM (Self-Organizing Feature Map)을 수정하여 적용한 당기는 SOFM과 밀어내는 SOFM을 이용한 전역 경로계획 알고리듬의 비

* 책임저자(Corresponding Author)
논문접수 : 2009. 1. 30., 채택확정 : 2009. 2. 25.
차영엽 : 원광대학교 기계자동차공학부(ggypcha@wku.ac.kr)
김곤우 : 원광대학교 전자및제어공학부(kgw0510@wku.ac.kr)
※ 본 논문은 2007년도 원광대학교의 교비지원에 의하여 연구하였음.

교에 대한 것이다. 기존의 SOFM[12]은 초기에 가중치벡터를 작은 랜덤(random)값으로 초기화하고, 학습계수와 이웃관계 함수를 초기화한다. 그리고 한 개의 입력이 랜덤하게 가해진다. 가해진 입력에 가장 가까운 가중치벡터를 갖는 승자뉴런(winning neuron)을 찾아서, 승자뉴런 근방의 뉴런을 입력벡터 방향으로 움직이게 함으로서 가중치벡터를 재계산한다. 이와 같은 과정을 반복하여 뉴런의 위치를 재배치함으로서, 주어진 입력에 따른 특징을 구현할 수 있다.

Cha[13,14]는 이동로봇의 전역경로계획에 2차원 메쉬를 사용하여 잡아당기는 SOFM과 밀어내는 SOFM을 적용하기 위하여 기존의 SOFM 알고리듬을 수정하였다. 즉, 초기의 가중치 벡터를 이미 결정된 값으로 초기화하고, 입력을 장애물 내부와 외부에 랜덤하게 가하는 대신에 외부 또는 내부에 규칙적으로 가한다. 가해지는 입력에 가장 가까운 가중치벡터를 갖는 승자뉴런을 찾아서, 승자뉴런 근방의 뉴런을 입력벡터 방향과 반대방향으로 각각 움직이게 함으로써 가중치벡터를 재 계산한다. 이와 같은 과정을 반복하여 뉴런의 위치를 재배치함으로서 주어진 입력에 따른 이동로봇의 경로를 생성할 수 있었다. 본 연구에서는 1차원 단경로 스트링을 제안하고 잡아당기는 SOFM과 밀어내는 SOFM을 이용한 전역 경로계획 알고리즘의 비교와 효율성을 입증하기 위하여 장애물이 있는 환경에서 1차원 단경로 스트링을 가중치로 가진 신경회로망의 모의실험을 통하여 결과가 보여진다.

II. SOFM

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

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

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

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 i 번째 입력 뉴런과 j 번째 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. 첫 번째는

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

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

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

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 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 연결강도를 달리하는 방법으로 연결강도는 거리에 반비례한다. 이웃관계 함수를 이용한 학습법칙은 다음과 같고, 여기서 $\eta(n)$은 n시간에서의 학습율이다.

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

이동로봇의 전역경로계획에 밀어내는 SOFM을 적용하기 위하여 앞에서 거론한 원래의 잡아당기는 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 재배치함으로서 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.

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

본 연구에서는 이러한 SOFM을 전역 경로계획에 적용하기 위하여 가중치 벡터를 1차원 스트링으로 수정한다. 즉, 초기의 가중치 벡터를 랜덤값 대신 이미 결정된 값으로 초

기화하고, 입력을 장애물 외부 또는 내부에 규칙적으로 가하여, 가해지는 입력에 가장 가까운 가중치벡터를 갖는 승자뉴런을 찾아서, 승자뉴런 근방의 뉴런을 입력벡터 방향과 반대방향으로 각각 움직이게 함으로서 가중치벡터를 재 계산한다. 이와 같은 과정을 반복하여 뉴런의 위치를 재 배치함으로서 주어진 입력에 따른 이동로봇의 경로를 생성할 수 있다.

이와 같이 수정된 SOFM은 다음과 같은 단계로 표현된다.

Step 1: 초기화

초기 가중치 벡터, $W_j(0)$을, 기존의 SOFM에서는 아주 작은 랜덤값으로 초기화 시키는데 반하여, 이동로봇의 작업영역 내에 출발점과 목표점을 일직선으로 하고 이를 분할하여 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: 신경회로망의 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: 학습

식 (6) 또는 (7)과 같이 activity bubble 내의 뉴런들을 입력벡터 방향 또는 반대방향으로 각각 가중치 벡터를 학습시킨다.

Step 3: 학습율, $\eta(n)$의 갱신

학습율의 선형감축은 만족할만한 결과를 얻도록 해야 한다.

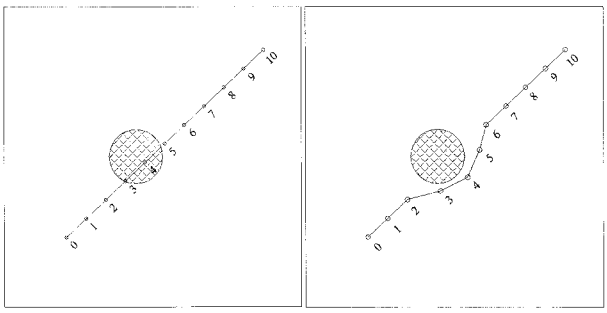
Step 4: 이웃관계 함수, $\Lambda_{i(X)}(n)$의 감축

Step 5: 정지 조건의 확인

Feature maps에 식별할 수 있는 변화가 일어나지 않는 경우에 반복계산을 정지하고, 그렇지 않으면 Step 2로 간다.

III. 스트링에서 경로계획

이동로봇의 전역경로계획에서 특정시간에 출발점과 목표점이 각각 하나라면 작업환경 전체에 대한 학습이 필요 없이 출발점과 목표점을 잇는 단경로 스트링을 사용하면 보다 계산시간을 절감할 수 있을 것이다. 이를 위하여 그림 1은 단경로 스트링에서 SOFM을 사용한 학습전과 학습후의 전역경로계획의 예를 보여주고 있다. 초기에 그림 1(a)와 같이 이동로봇의 작업영역에 출발점과 목표점의 위치가 결정되면 간격이 일정한 곳에 가중치 벡터를 갖는 초기 스트링을 설정한다. 여기서도 일반 이동로봇의 작업영역이 2차원 공간인 점을 감안하여 이동로봇의 작업영역은 사각형이라고 가정하고, 빗금 친 원은 장애물을 나타내고 있다. 주어진 스트링에서 출발점은 0, 목표점을 n으로 표시한다. 스트링을 일정간격으로 n등분하면, 분할된 각 점은 1, 2, ---, n-2, n-1로 나타낼 수 있다. 여기서는 n=10 이고, 각 점은 0



(a) (b)

그림 1. 단경로 스트링에서 SOFM을 이용한 (a) 학습 전과 (b) 학습 후 전역경로계획의 예.

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

부터 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, 앞에서 기술한 밖에서 잡아당기거나 안에서 밀어내는 수정된 SOFM을 이용하여 네트워크를 학습시키면, 즉, 장애물 외부 또는 내부에 규칙적으로 입력이 주어지고, activity bubble 내의 뉴런은 가중치 벡터가 입력벡터 방향 또는 반대방향으로 주어지므로, 그 내부에 있는 점들은 점차 장애물 바깥쪽으로 빠져나가게 된다. 그 결과가 그림 1(b)와 같다고 하면, 실제 이동로봇을 위한 전역 경로는 학습시키기 전의 분할된 점 번호를 따라서, 학습 후의 점 번호에 따른 좌표순서대로 지정하면 된다. 그리고 학습 전에 해당 경로위에 장애물이 없었다면, 학습 후에도 그 경로는 거의 변화가 없을 것이다. 즉, 작업영역 위의 출발점에서 목표점으로 갈 수 있는 최단 경로점의 위치는 학습 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 없을 것이다. 이렇게 함으로써 이미 주어진 장애물 지도를 기본으로, 출발점에서 목표점까지 장애물과 충돌을 피하면서 가장 빠르게 갈 수 있는 스트링을 이용한 최단경로를 찾는 것이 가능하다.

IV. Experimental Results

그림 2(a)는 1차원 단경로 스트링으로 구성된 당기는 SOFM을 전역 경로계획에 적용한 순차적인 결과를 보여주고 있다. 스트링의 뉴런 개수는 16개로 하고, 이동로봇의 작업영역에 있는 장애물은 2개를 배치하였다. 기존의 SOFM에서는 초기에 가중치 벡터를 작은 랜덤값으로 주는 대신에, 여기서는 이동로봇의 작업영역에서 출발점과 도착점을 스트링 양끝으로 설정하여 일정한 간격으로 배치한 것이 iteration 0에 나와 있다. 이때 스트링의 양쪽 끝에 있

는 뉴런 2개는 입력벡터의 영향을 무시하도록 하여, 작업영역 내부의 출발점과 목표점으로 주어져서 이동하는 것을 제한한다. 점차적으로 입력벡터를 가하는 횟수를 증가시키면, 장애물인 원 내부에 있는 스트링의 부분이 원 바깥으로 끌려 나가는 결과가 얻어진다. iteration 100은 100회 반복의 결과로 장애물 2개에 모두 스트링이 걸쳐져 있었다. iteration 200은 200회 반복 결과로 우측보다는 좌측 장애물에 스트링이 더 많이 걸쳐져 있었다. iteration 400은 400회 반복의 결과로 좌측과 우측 장애물 모두에서 스트링이 걸쳐진 부분이 없었다.

밀어내는 SOFM을 적용한 순차적인 결과가 그림 2(b)에서 보여주고 있다. 점차적으로 입력벡터를 가하는 횟수를 증가시키면, 장애물인 원 내부에 있는 스트링이 원 바깥으로 밀려나가는 결과가 얻어진다. 100회 반복의 경우에 당기는 SOFM을 이용한 그림 2(a)의 경우에는 스트링이 아직 장애물 안에 있었지만, 밀어내는 SOFM을 이용한 그림 2(b)의 경우에는 스트링이 장애물 바깥으로 거의 빠져나온 것을 알 수 있었다. 잡아당기는 SOFM의 경우에 그림 2(a)와 같이 약 400회에서 스트링이 장애물 바깥으로 빠져나오지만, 밀어내는 SOFM의 경우에 그림 2(b)와 같이 약 200회에서 스트링이 장애물 바깥으로 완전히 빠져나왔다. 결과적으로 같은 반복회수에서 당기는 SOFM보다는 밀어내는 SOFM의 결과가 얻으려고 하는 해에 더 빨리 수렴함을 알 수 있었다.

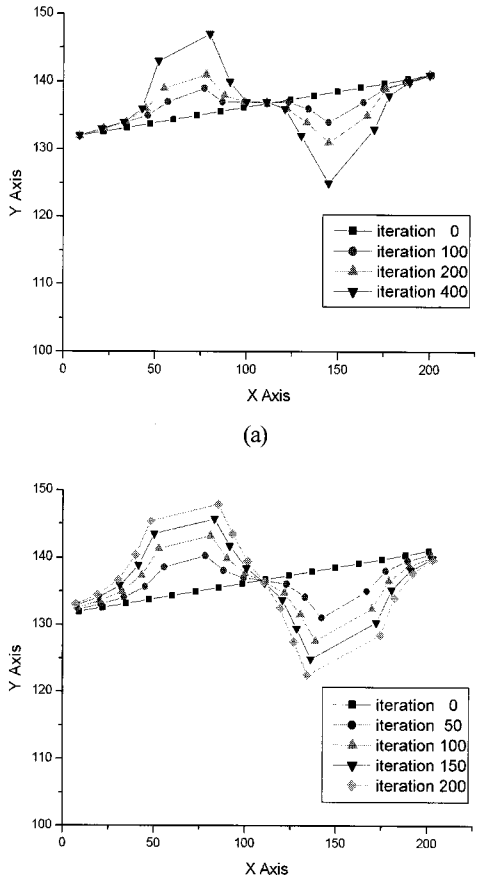


그림 2. (a) 당기는 SOFM과 (b) 밀어내는 SOFM을 이용한 단경로 스트링의 순차적 위치변화.

Fig. 2. Sequential position change of single string using (a) pulled- and (b) pushed-SOFM.

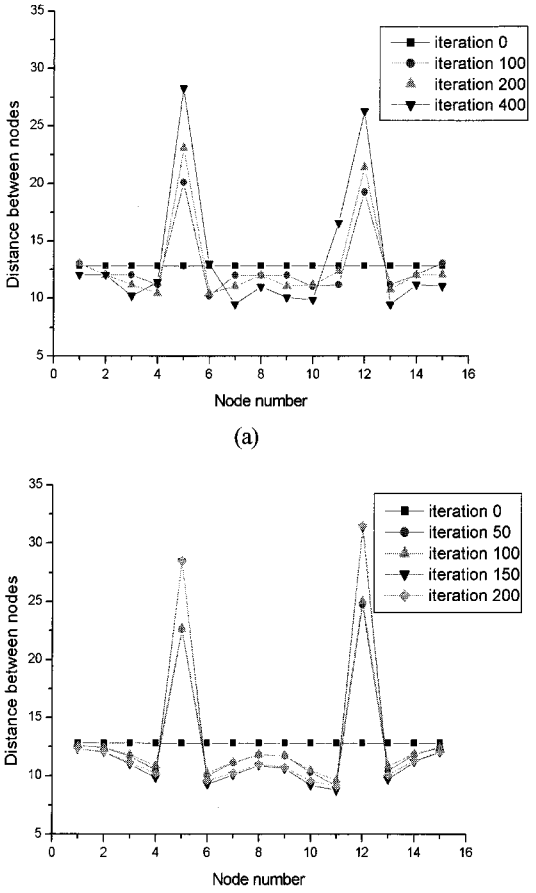


그림 3. (a) 당기는 SOFM과 (b) 밀어내는 SOFM을 이용한 노드 사이의 거리.

Fig. 3. Distance between nodes by using (a) pulled- and (b) pushed-SOFM with string.

그림 3은 단경로 스트링에서 반복회수 증가에 따른 노드 사이의 거리를 나타내고 있다. 당기는 SOFM을 이용한 그림 3(a)의 경우에는 반복회수에 따라서 노드사이의 거리가 불규칙적으로 변화하는 것을 볼 수 있다. 이에 비하여 밀어내는 SOFM을 이용한 그림 3(b)의 경우에는 반복회수 증가에 따라 노드사이의 거리가 규칙적으로 변화하고 지속적으로 감소나 증가함을 알 수 있다. 이는 그림 2의 결과와 같은 의미를 가지며 밀어내는 SOFM이 당기는 SOFM보다 장애물에서 노드들이 빨리 벗어나게 해주는 것을 알 수 있었다. 그림 4는 반복회수에 따른 노드사이 거리의 총합을 보여준다. 여기서 보면 밀어내는 SOFM의 결과가 당기는 SOFM의 결과보다 같은 반복회수에서 노드사이 거리의 총합이 더 큰 값을 가지는 것을 알 수 있다. 이는 당기는 SOFM 보다는 밀어내는 SOFM 방법이 노드들의 이동방향을 유지하고 거리를 지속적으로 증가시켜 장애물에서 빨리 벗어나게 해주는 것으로 생각할 수 있다.

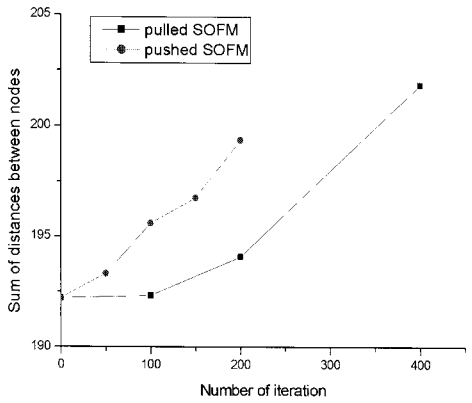


그림 4. 스트링에서 당기는 SOFM과 밀어내는 SOFM을 이용한 반복회수에 따른 노드 사이 거리 총합.

Fig. 4. Sum of distances between nodes according to the number of iteration by using pulled- and pushed-SOFM with string.

V. 결론

본 연구에서는 이동로봇의 전역 경로계획에 적용하기 위하여 1차원 단경로 스트링으로 가중치를 이용하여 Kohonen의 SOFM을 수정한 당기는 SOFM과 밀어내는 SOFM 방법을 비교하였다. 즉, 초기의 가중 벡터를 이미 결정된 값으로 초기화하고, 입력을 장애물 외부 또는 내부에 가하여, 가해지는 입력에 가장 가까운 가중벡터를 갖는 승자뉴런을 찾아서, 승자뉴런 근방의 뉴런을 입력벡터 방향과 반대방향으로 움직이게 함으로써 가중치벡터를 재 계산하였다. 이와 같은 과정을 반복하여 뉴런의 위치를 재 배치 함으로써 주어진 입력에 따른 이동로봇의 경로를 생성할 수 있었다. 결과적으로 1차원 단경로 스트링에서 당기는 SOFM 보다는 밀어내는 SOFM 방법이 노드들의 이동방향과 거리를 지속적으로 증가시켜 장애물에서 빨리 벗어나게 해주어 얻으려고 하는 해에 더 빨리 수렴하는 것을 알 수 있었다.

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 and 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, "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.

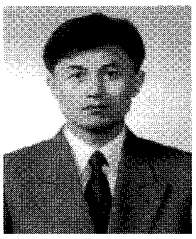
[10] N. Chaiyaratana and A. M. S. Zalzala, "Time-optimal path planning and control using neural networks and a genetic algorithm," *International Journal of Computational Intelligence and Applications*, vol. 2, no. 2, pp. 153-172, 2002.

[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] T. Kohonen, "The self-organizing map," *Proceedings of the Institute of Electrical and Electronics Engineers*, vol. 78, no. 9, pp. 1464-1480, 1990.

[13] Y. Y. Cha and H. G. Kang, "A global path planning of mobile robot by using self-organizing feature map," *Journal of Control, Automation and Systems Engineering*, vol. 11, no. 2, pp. 137-143, 2005.

[14] Y. Y. Cha, D. W. Yu, and S. M. Jeong, "A global path planning of mobile robot using modified SOFM," *Journal of Control, Automation and Systems Engineering*, vol. 12, no. 5, pp. 473-479, 2005.



차 영 엽

1984년 부산대 기계공학과 졸업. 1987년 한국과학기술원 생산공학과 석사. 1995년 한국과학기술원 정밀공학과 박사. 1995년~현재 원광대학교 기계자동차공학부 교수. 관심분야는 이동로봇, 영상처리.

김 곤 우

제어 · 자동화 · 시스템공학 논문지 제14권 제8호 참조.