OFM

Path Planning of Mo ile Ro ot using tring and Pushed OFM

*# 1

*#Y. Y. Cha(ggypcha@wonkwang.ac.kr)[1]

Key words : Path Planning, SOFM, String, Mobile Robot

1.

이동로봇[1]의 자율주행을 위하여 신경회로망 중에서 SOFM을 이용한 전역 경로계획 알고리듬을 차[2]가 제안한바 있다. 기존의 SOFM 알고리듬[3]은 초기 가중치 벡터를 아주 작은 랜덤값으로 주고 모든 가중치 벡터를 학습시키며, 입력벡터를 작업영역 전체에 랜덤하게 분포시킨다. 이에 반하여, 차[2]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 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.

본 연구에서는 차[2]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 재 계산한다. 두 번째로 계산시간 단축을 위하여 가중치 벡터로 2차원의 그물망 대신에 1차원의 스트링(string)을 사용한다.

2. OFM

이동로봇의 전역경로계획에 SOFM을 적용하기 위하여 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 학습법칙을 다음과 같이 수정한다.

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

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.

$$W_j^{new} = W_j^{old}+\eta(X_i-W_j^{old}) \qquad (2)$$

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

Step 2a. network의 입력층에 입력벡터, X 를 위치시킨다.

Step 2b. Similarity matching

입력벡터, X 에 가장 근접한 가중치 벡터를 갖는 뉴런을 선택하여 승자뉴런으로 한다. 이는 다음 식을 사용하여 구할 수 있다.

$$i(X)=k,\ \ where\ \|W_k-X\|<\|W_j-X\| \qquad (3)$$

Step 2c. Training

식 (1)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 않는 경우에 반복계산을 정지하고, 그렇지 않으면 Step 2로 간다.

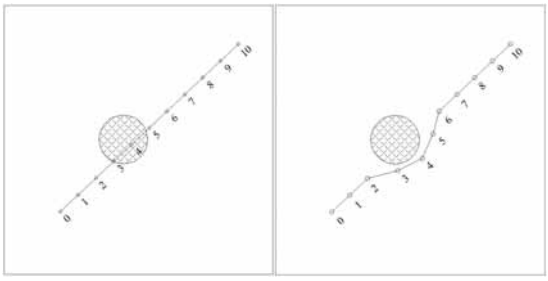
3. lo al Path Planning

이와 같은 전역 경로계획에 기중치 벡터로서 1차원의 스트링과 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, 빗금친 원은 장애물을 나타내고 있다. 이 스트링에서 출발점은 0, 목표점을 n으로 표시한다. 이 스트링을 일정간격으로 n등분하면, 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, 밖으로 밀어내는 수정된 SOFM을 이용하여 네트워크를 훈련하면, 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

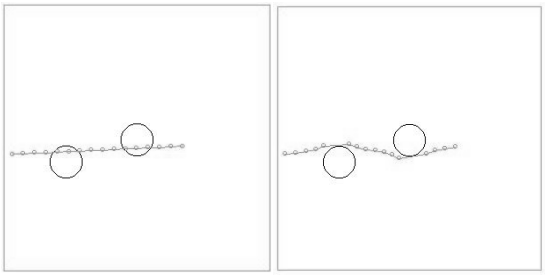
좌표순서대로 지정하면 된다. 그리고 훈련 전에 해당 경로위에 장애물이 없었다면, 훈련 후에도 그 경로는 거의 변화가 없을 것이다. 즉, 작업영역 위의 출발점에서 목표점으로 갈 수 있는 최단 경로 점의 위치는 훈련 전이나 후에도 거의 변화가 없을 것이다.

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.

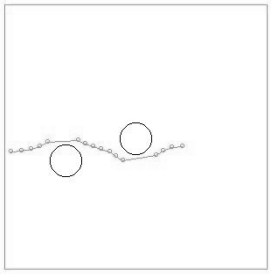


(a) (b)

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



(a) iteration 0 (b) iteration 100



(c) iteration 200

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

4.

본 연구에서 제안한 밀어내는(pushed) SOFM을 적용한 순차적인 결과가 Fig. 2에서 보여주고 있다. 스트링의 뉴런 개수는 16개로 하고, 이동로봇의 작업영역에 있는 장애물은 2개를 배치하였다. 초기에 가중치 벡터를 작은 랜덤값으로 주는 대신에 작업영역에서 이동로봇의 출발점과 도착점을 스트링 양끝으로 설정하여 일정한 간격으로 배치한 것이 Fig. 2(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 반복의 결과가 차례로 보여진다. 100회 반복의 경우에 밀어내는 SOFM을 이용한 Fig. 2(b)의 경우에는 스트링이 장애물 바깥으로 거의 빠져나온 것을 보여 준다. 밀어내는 SOFM의 경우에 Fig. 2(c)와 같이 약 200회에서 스트링이 장애물 바깥으로 완전히 빠져나오는 것을 알 수 있다.

·

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. 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.

[1] 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.

[2] 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.

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