Proceedings of KFIS Spring Conference 2006 Vol. 16, No. 1.

CMACRRT를 이용한 로봇 매뉴플레이터 경로계획

A Path Planning for Robot Manipulator using CMACRRT

오경세[1], 김은태[1]

[1] 서울시 서대문구 연세대학교 전기전자공학부
E-mail: etkim@yonsei.ac.kr

요 약

매니퓰레이션 기술 중에서 경로 계획은 중요한 문제 중의 하나이다. RRT는 경로 계획 알고리즘으로 최근에 제안되었다. RRT는 기존 알고리즘보다 빠르게 장애물을 회피하여 경로를 계획할 수 있다. 기존의 경로 계획 알고리즘은 그 상황에 따라 반복적으로 경로 계획을 하였다. 이러한 점을 개선하기위해 RRT와 인간의 소뇌구조를 모방한 CMAC을 결합한 CMACRRT를 제안한다. CMAC은 RRT가 만들어낸 경로와 그 상황을 기억하여 유사한 상황에서 경로를 다시 사용할 수 있게 해준다. 이렇게해서서 CMAC을 통해 학습된 상황에서 RRT를 사용하지 않고 기존의 경로를 사용할 수 있게 된다.

Key Words : CMAC, RRT, Path Planning, Robot manipulation

1. 서 론

21세기에는 기술융합 가속화에 따라 새로운 제품과 산업이 등장하고 있고 자율 로봇 시스템은 첨단 요소기술의 발전을 견인하면서 타 분야로의 기술 확산을 통해 전자, 자동화 등 연관 산업의 국제경쟁력 강화에 직접 기여하고 있다.

이러한 자율로봇의 가장 기본적인 기능 중 하나가 매니퓰레이션 기술이다. 매니퓰레이션 기술의 향상 및 지능화는 임의의 환경에서 다양한 기능을 수행해야 하는 자율로봇을 위한 필수 불가결한 요소이고 이러한 로봇 매니퓰레이션에 대하여 많은 연구가 진행되어 왔다. 특히, 장애물의 인식 및 인식된 장애물에 따라 경로를 변환하는 기능에 관련된 연구가 활발하다. 이러한 기능은 산업용 로봇의 세대에서뿐만 아니라 지능형 로봇의 세대에서도 작업의 중추에 놓이는 가장 대표적인 로봇의 작업기능이다.

매니퓰레이션 기술은 이제 서비스 로봇분야에도 응용이 되고 있다. 서비스 로봇은 유사한 환경에서 사용자자에게 도움을 주는 역할을 수행한다. 이러한 상황에서 기존의 경로 계획은 비효율적으로 운용되는 면이 있다.

본 연구에서는 CMAC과 RRT 방법을 결합한 CMACRRT를 서비스 로봇의 경로 계획 알고리즘으로 제안하고자 한다.

제안하는 CMACRRT의 특징은 유사한 환경에서 RRT로 기 생성된 경로를 재사용함으로써 경로 계획에 따른 연산량을 줄여 시간을 단축할 수 있는 장점이 있다.

본 논문의 구성은 2장에서는 CMAC에 대한 설명을 하고 3장에서는 RRT에 대한 간단한 기술과 실제 컴퓨터 시뮬레이션 한 결과를 보여준다. 4장에서는 CMAC과 RRT를 결합 CMACRRT 알고리즘에 대한 설명을 한다. 마지막으로 5장에서 결론을 맺는다.

2. Cerebella Model Arithmetic Controller

Albus에 의해서 제안된 CMAC 신경망은 빠른 학습 속도와 하드웨어 구현의 편리성 등의 특성을 지니고 있다. 이런 특성 때문에 학자들의 관심이 점차 높아지고 있는 신경망 구조이다[1][2][3][4]. CMAC은 인간의 소뇌 구조를 모방한 신경망으로 매우 간단한 구조를 가지고 있다. 그림 1은 Albus가 제안한 CMAC의 구조를 나타내고 있다.

CMAC은 다음과 같은 3가지 단계로 이루어져 있다.

1. 주 제어부 혹은 센서부의 정보는 양자화되고 가상의 메모리(Conceptual Memory)에 대한 주소로 사용된다.

2. CMAC에서 상태에 대한 정보는 메모리에 분산되어 저장되고 주소의 키(Key)로써 상태변수를 이용하여 검색하게 된다.

3. 데이터 검색을 위해 상태변수들은 중간변수들로 사상된 후 출력 값을 검색하기 위해 물리적 메모리 주소로 사상된다.

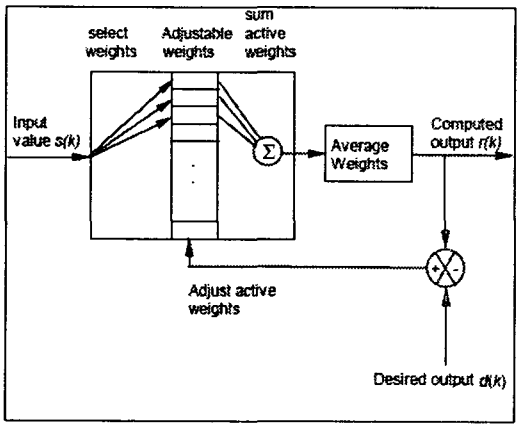


그림 1. Albus의 CMAC 모델

3. Rapidly-Exploring Random Trees

처음에 이 알고리즘은 단일 트리구조가 형성되는 Kinodynamic motion planning에 적용하기 위해 개발되었다. RRT 알고리즘은 확률적으로 완벽하다는 것이 밝혀져 지금은 그 응용범위가 더 넓어졌다. 또한,. RRT는 nonconvex한 고차원의 영역에서 효과적으로 찾을 수 있게 만들어진 데이터 구조이자 알고리즘이다[5][6].

3.1 RRT 트리 구조의 생성

RRT의 알고리즘은 그림 3과 같다. 단순한 반복을 통해서 임의로 선택을 통해 바이어스된 새로운 Vertex를 증가시켜 RRT를 EXTEND 함수를 이용하여 확장시켜나는 것이다. 이 함수는 RRT에서 주어진 x에 가장 가까이에 있는 Vertex를 선택한다. 여기서 NEW_CONFIG는 정해진 ε의 간격으로 장애물과의 충돌여부를 판단하면서 x를 향해가도록 한다. 이 과정은 증가의 거리 계산을 사용하여 빠르게 수행한다.

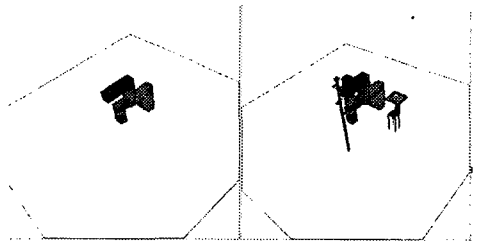
```
- - - - - - - - - - - - - - - - - - - - - - - -
BUILD _ RRT (x_init)
1. T_init (x_init);
2. for k = 1 to K do
3.    x_rand <- RANDOM _ CONFIG ();
4.    EXTEND (T, x_rand);
5. Return T
- - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - -
EXTEND (T, x)
1. x_near <- NEAREST_N EIGHBOR (T, x);
2. if NEW_CONFIG (x, x_near, x_new) then
3.    τ_add_vertex (x_new);
4.    τ_add_edge (x_near, x_new, u);
5.    if x_new = x then
6.          Return Re ached ;
7.    else
8.          Return Advanced ;
9. Return Trapped ;
- - - - - - - - - - - - - - - - - - - - - - - -
```

그림 3. RRT 알고리즘

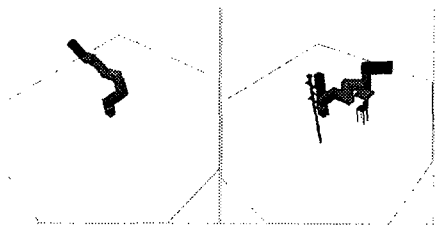
그 결과로 세 가지의 상황이 나온다. 먼저 Reached 상황에서는 ε 거리 이내에 x가 존재하여 Vertex가 x에 생성되는 경우이다. 그리고 Advanced 상황은 새로 형성된 Vertex가 $x_{new} \neq x$인 경우이다. 마지막으로 Trapped 상황은 새로 형성된 Vertex가 장애물과의 충돌이 일어난 상황이다.

3.2 RRT 시뮬레이션

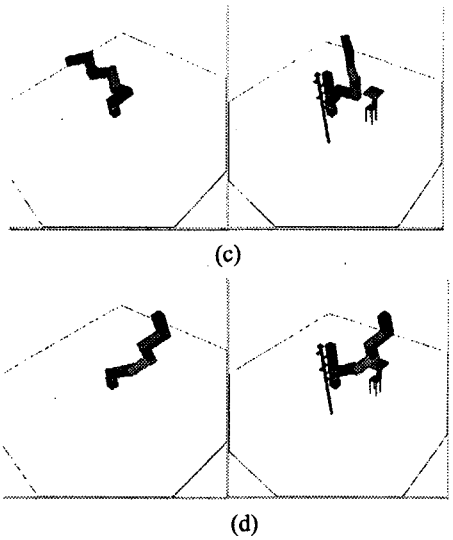
본 연구에서 RRT에서 생성된 경로를 테스트하기 위해 그림4 와 같은 컴퓨터 시뮬레이션 환경을 구성하였다. RRT는 Motion Strategy Library(MSL)을 이용하여 프로그램을 하였다. 그림 4는 장애물이 있는 환경과 그렇지 않은 환경에서 동일한 목적지에 도달하는 과정을 나타내고 있다.



(a)



(b)



(c)

(d)

그림 4. MSL을 이용한 RRT 시뮬레이션

3.3 RecurDyn을 이용한 Co-Simulation

본 연구에서 구성한 시뮬레이션 환경은 Matlab™과 RecurDyn™ [7]을 연동하여 KUKA™ [8]모델을 제어하도록 구성하였다. 그러기위해 MatLab™의 KUKA™ 모델과 RecurDyn™의 KUKA™모델의 파라미터를 일치시키는 과정이 필요하다. Co-Simulation 환경을 그림 5과 같이 나타낼 수 있다. KUKA™는 6축 매니퓰레이터를 모델링 한 것이고 RRT는 Motion Strategy Library[9]를 기반으로 하여 프로그램을 하였다. RRT에서 생성된 경로를 제어부의 입력으로 하여 KUKA 모델이 원하는 위치로 이동하게 Co-Simulation 하였다 [10].

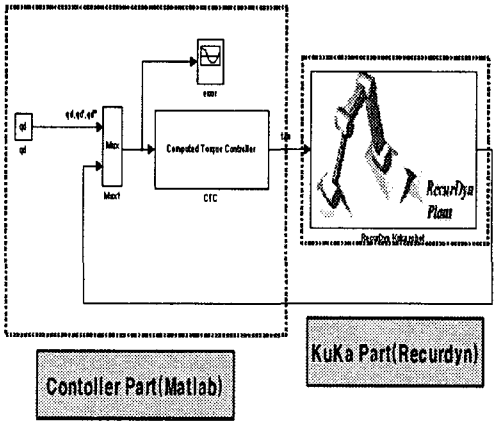


그림 5. RRT Co-Simulation 환경

4. CMACRRT

본 논문에서는 RRT를 이용하여 생성된 경로를 유사한 상황에서 재사용하여 새로운 경로 계획에 따른 연산량을 줄여 시간을 단축하는데 목적이 있다. 여기서 유사한 상황에 대한 인식은 CMAC을 이용 한다.

이러한 CMACRRT의 알고리즘은 그림 6에 나타나있다. 매니퓰레이터가 처음 상황에 접했을 때는 기존 RRT를 사용하여 경로를 계획하게 된다. 이렇게 해서 하나의 검증된 경로가 생성된다. CMAC은 처음 형성된 경로는 PATH1으로 기억하게 된다. 다음의 또 다른 상황에서 CMAC은 현재의 상황이 PATH1의 상황과의 유사성을 확인한다. 유사성이 있으면 PATH1의 경로를 사용하고 그렇지 않으면 RRT를 이용하여 새로운 경로 PATH2를 생성한다. 이런 식으로 과거의 경로를 기억해두었다가 현재의 상황과의 유사성을 판단하는 알고리즘이다.

그림 7은 구현된 CMACRRT 기반으로 매니퓰레이터가 냉장고에서 컵을 꺼내는 시나리오를 수행하는 시뮬레이션 환경을 나타내고 있다.

```
-------------------------
BUILD_BHT-OAA(DB)
1. Divide two group within DB using SVM;
2. Store parameter of SVM;
-------------------------
-------------------------
FIND_PATH
2. BHT-OAA(Input);
3. If Collision then
4        Path_new = RRT(Input);
5.       Build DB_new(Path_new ,Input);
6.       BUILD_BHT-OAA(DB_new);
7.       Go to STEP 1
8. else
9.       Build DB_new(Input);
10.      BUILD_BHT-OAA(DB_new);
11.      Return SUCCESS;
-------------------------
```

그림 6. CMACRRT 알고리즘

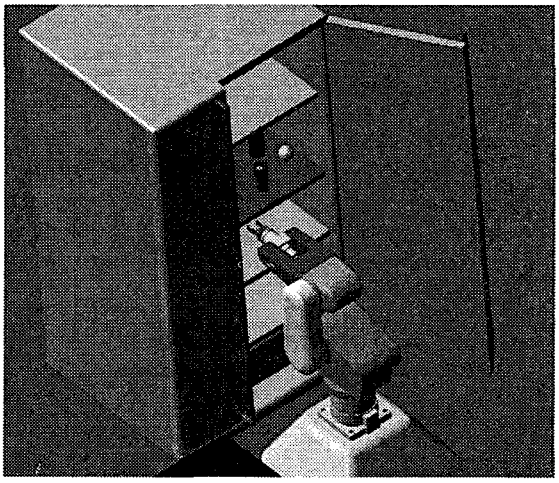


그림 7. CMACRRT 시뮬레이션 환경

5. 결 론

본 논문에서는 CMAC을 통한 상황의 유사성 판단과 RRT를 이용한 경로 계획을 통합한 CMACRRT를 제안하였다. 서비스 로봇의 경우 유사한 실내 환경에서 동작을 하게 된다. 이런 환경에서 RRT 알고리즘만을 사용하게 되면 비슷한 경로를 계속 계획하여야 하므로 필요 없는 시간을 사용하게 된다. 이를 보완하여 상황의 유사성을 CMAC 구조로 학습하여 RRT의 불필요한 연산량을 줄여주어 경로 계획 시간을 단축할 수 있는 효과를 얻을 수 있었다.

본 연구에서는 정지된 장애물이 있는 환경에서만 시뮬레이션을 했다. 추후에 CMACRRT를 다양한 장애물이 있는 환경에 적용하는 연구가 필요하고 실제 서비스 로봇에도 적용해보야 할 것이다.

감사의 글

본 논문은 21C프론티어 사업(인간기능 생활지원 지능로봇 기술개발 사업단)의 지원을 받았습니다.

참 고 문 헌

[1] 김형석,최종수," CMAC 신경회로망 기반 로봇 제어 기술," 전자공학회지 vol. 23, no. 12, pp. 136-144, Dec 1996.

[2] F.H. Glanz, W.T. Miller and L.G. Kraft, "An Overview of the CMAC Neural Network," in Proceeding of IEEE Conference on Neural Networks for Ocean Engineering, pp. 301-308, 1991.

[3] J.S.Albus, "Data storage in the Cerebella Model Articulation Controller (CMAC)," In Journal of Dynamic Systems, Measurement, and Control, vol 97(3), pp. 228-233, 1975.

[4] W.T. Miller, F.H. Glanz and L.G. Kraft, "Application of a General Learning Algorithm to the Control of Robotic Manipulators," Int. J. Robotics Research, vol 6, No. 2, pp. 84-98,Summer 1987.

[5] J. J. Kuffner, S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," In Proc. IEEE Int'l Conf. on Robotics and Automation, pp. 995-1001, 2000.

[6] S. M. LaValle, J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," In B. R. Donald, K. M. Lynch,and D. Rus, editors, Algorithmic and Computational Robotics: New Directions, pp. 293-308. A K Peters, Wellesley, MA, 2001.

[7] FuntionBay,Inc.
http://www.functionbay.co.kr/

[8] KUKAmanipuator,
http://www.kuka.com/en/

[9] The Motion Strategy Library(MSL) at University o fIllinois,
http://msl.cs.uiuc.edu/~lavalle/

[10] S. Park, K. Oh , E. Kim , S. Lim and S. Lee, "Cosimulation of the control 6-DOF Kuka manipulator by Simulink and Recurdyn," International Technical Conference on Circuits/Systems, Computers and Communications 2005, Vol. 1, pp 271-272, Jul.,2005.