2002년도 대한전자공학회/대한전기학회 시스템 및 제어분야 합동 추계학술대회 논문집(2002.11.30)

행동기반 제어방식을 위한 득점과 학습을 통한 행동선택기법

정석민, 오상록, 윤도영, 유범재, *정경주
한국과학기술연구원 지능제어센터,
*한양대학교 공과대학 전자전기컴퓨터 공학부
전화 : 02-958-5754 / 핸드폰 : 017-706-8407

Action Selection by Voting with Learning Capability for a Behavior-based Control Approach

Jeong, S.M., Oh, S.R., Yoon, D.Y., You, B.J., *Chung, C.C.
Intelligent System Control Research Center, KIST
* Division of Electrical and Computer Engineering, Hanyang University
E-mail : jsmaroma@kist.re.kr

Abstract

The voting algorithm for action selection performs self-improvement by Reinforcement learning algorithm in the dynamic environment. The proposed voting algorithm improves the navigation of the robot by adapting the eligibility of the behaviors and determining the Command Set Generator (CGS). The Navigator that using a proposed voting algorithm corresponds to the CGS for giving the weight values and taking the reward values. It is necessary to decide which Command Set control the mobile robot at given time and to select among the candidate actions. The Command Set was learnt online by means as Q-learning. Action Selector compares Q-values of Navigator with Heterogeneous behaviors. Finally, real-world experimentation was carried out. Results show the good performance for the selection on command set as well as the convergence of Q-value.

I. 서론

최근 들어 많은 연구 개발자들은 로봇에 많은 관심을 갖고 있다. 1960년대 이후 초기 로봇의 대표적인 구조는 Top-down방식의 중앙 집중적인 형태의 계층적인 구조가 제시되었으며 비교행동학에 근거한 행위기반 제어방식은 이동로봇에 대해서 시스템을 지능적으로 제어 할 수 있으며 로봇의 환경에 대해서 신속히 반응하는 제어 방식이고 현재 로보틱스에 많은 관심을 갖고 있는 분야이다.

행위기반에 의해 구성된 Behavior는 환경에 대해 오직 하나의 Behavior가 활성화되고 어떤 환경에 대해선 여러 개의 Behavior가 이동 로봇에 의해 활성화되는데 이 때 이동 로봇의 주행에 있어서 Behavior 간의 충돌이 발생하게 된다. 그러므로 행위기반 제어방식에 있어서 중요한 문제는 이렇게 충돌이 일어나는 경우 어떻게 이 문제를 해결하느냐는 것이다.

이 논문에서 우리는 환경의 변화에 대해 강인한 시스템을 만들기 위해 행위기반 제어방식에 학습을 통해서 행동을 선택하는 기법을 제안하였다.

II. 행동 선택 문제

행동선택문제는 유사한 목적을 갖는 행동을 선택하는 하위계층의 선택문제만을 고려하지 않고 서로 다른 목적을 갖는 행동들에 대해서 서로 독립적으로 선택하는 상위계층을 포괄적으로 고려하여 정의하였다. [1] 대표

적인 한 예로 이동중인 이동로봇이 장애물을 만났을 때 일어나는 행동 선택 문제를 그림 1에서 볼 수 있다.

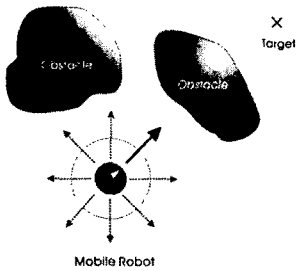

그림 1. 행동 선택이 필요한 경우

여기서 실선은 Target에 대한 "Move to Goal"이라는 행동패턴을 지칭하고 점선은 장애물에 대한 "Obstacle Avoidance"라는 행동패턴을 가리킨다.

이렇게 Action Selection Problem에 대해 많은 연구가 이루어졌으며 대표적인 구조로는 다음과 같다.

2.1 우선순위에 근거한 구조

분포된 구조에서 주어진 시간에 대해 적절한 행동을 선택하는 것은 이동로봇을 제어함에 있어서 꼭 필요로 하는 요소이다. 이러한 이유로 인해서 각각의 행동패턴들에 대해서 우선권을 부여하는 구조이다. 대표적인 구조로는 Subsumption Architecture [2] 를 들 수 있다. 이 구조는 그림 2와 같으며 Suppression/ Inhibition의 연결단자를 가지고 있다.

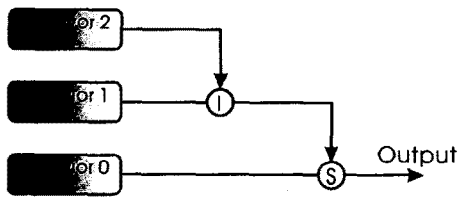

그림 2. Subsumption Architecture.

2.2 명령의 합성 (Command Fusion)

이 구조는 다양한 행동패턴들(Behaviors)로부터 명령을 합성하여 하나의 출력을 얻어내는 방식이며 이러한 구조의 대표적인 알고리즘으로는 Potential Field법 [3]과 Fuzzy Logic [4] 방식이다.

먼저 Potential Filed법은 수식(1)에 의거하여 크기와 방향을 결정한다.

$$V_{direction} = -180^{\circ}$$
$$V_{magnitude} = \begin{cases} \dfrac{(D-d)}{D} & \text{for } d \le D \\ 0 & \text{for } d > D \end{cases} \quad (1)$$

D : Maximum range of the field's effect

d : Sensor에 의해 얻은 거리 정보

다음으로 Fuzzy Logic은 수학적으로 소속 함수(membership function)를 정의하며 이를 통해서 행동패턴에 대한 입력 값을 얻으며 Rule-based를 통해 결과값을 얻어내는 방식이다.

2.3 DAMN

DAMN(A Distributed Architecture for Mobile Navigation) [5] 의 구조는 여러 개의 Arbiter를 가지고 있다. 즉 이동로봇의 주행에 필요한 속도와 방향을 중재하는 Arbiter를 가지고 있다. 만약 동시에 이동로봇의 행동들을 고려한다고 했을 때 DAMN은 이동로봇의 취할 수 있는 가능한 행동에 대해서 Voting을 함으로 ASP(Action Selection Problem)를 해결하였다.

2.3.1 Turn Arbiter

Turn Arbiter는 분리된 곡률로 이루어진 명령들 가운데서 중재하기 위해 만들었으며 곡률 공간 (Curvature Space)은 0인 점을 기준으로 오른쪽의 최대 곡률까지 N개로 선형적으로 분리되었으며 왼쪽도 동일하게 분리하였다. 이는 그림 3에서 볼 수 있다.

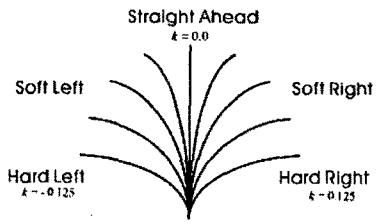

그림 3. 곡률에 근거한 turn command space

이로 인해서 Command Space에 대해서 Voting을 실시하여 -1부터 1 내에서의 값을 가지는 형태를 구성하고 있다.

2.3.2 Speed Arbiter

단순한 2개의 행동패턴(Behavior)에 대해서 DAMN에서 그림 4와 같이 조합을 통해서 출력값을 얻고 Voting 알고리즘에 의해서 최대의 값을 갖는 Node를 선택하는 방식이다. 그림 4에서 회색을 띄는 Node는 음수의 값을 가지며 Node의 원의 반경의 크기에 따라 값을 표현하였다. 따라서 아래 그림에선 S.L값이 선택된다. 그렇지만 이 구조의 가장 큰 단점은 Behavior의 각각의 노드의 값을 미리 정해놓아야 한다는 점이다. 이것은 Hard-wired 구조이므로 환경의 변화에 적응성이 떨어지게 된다.

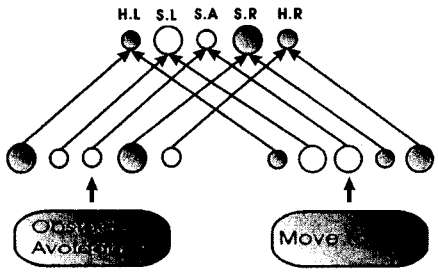

그림 4. 최대 turn vote에 비례한 속도 command

III. 강화 학습 (Reinforcement Learning)

강화학습은 주어진 환경에 대해서 행하는 행동 간의 대응이다. 이러한 Mapping은 가장 큰 Scalar Reward 값은 갖도록 하거나 강화신호를 얻도록 하는 방식이다. 대표적인 강화학습은 1989년 Watkins [6]에 의해서 제안된 Q-learning algorithm과 Sutton의 Temporal Difference(TD(λ)) algorithm을 들 수 있다.

3.1 Q-learning

주어진 환경과 행동 간의 대응에서 큰 보상 값을 얻도록 하는 목적을 갖고 Q-learning은 제안되었다. Q-learning은 불연속 상태공간과 미리 정해진 행동집합을 통해서 적절한 행동패턴을 얻고자 하는 학습 알고리즘이다. 이는 아무런 지식이 없이 학습할 수 있다는 Robustness를 보장하고 있으며 Episode가 증가할수록 보다 나은 값을 갖는 특성을 Watkins의 논문에서 증명하였다. [6]

Q-learning의 Optimal Value Function은 다음과 정의한다.

$$Q^*(x,a) = R_x(a) + \gamma \sum_y P_{xy}[\pi(x)] V^\pi(y) \quad (2)$$

여기서 γ 는 Discount Rate를 지칭하며 이 값은 $0<\gamma<1$ 을 갖는다.

이와 같은 특성을 갖는 강화학습은 주어진 환경에 대해 센서를 통해 현재 상태를 감지하고 이로 인해서 얻어진 센서 데이터를 통해 미리 정해진 행동을 선택하게 되고 이때 행한 행동에 대해서 보상치(Reward Value)를 생성하고 이를 이동로봇에게 되돌려 준다. 이러한 상황에 대해서 Q-learning을 통한 Q-value를 얻게 되고 Update를 행하게 된다.
이러한 순서를 Machine Learning에서는 Episode라 부르는데 Episode를 다시 순차적으로 표시하면 표 1과 같다.

표 1. episode의 순서

1	Observes its current state x_n
2	Selects and performs an action a_n
3	Observes the subsequent state y_n
4	Receives an immediate payoff r_n
5	Adjusts its Q_{n-1} values using a learning factor α_n

3.2 Update

주어진 환경이 같고 동일한 행동을 갖는다면 수식(3)과 같으며 그렇지 않은 경우에는 수식(4)의 값을 갖고 이를 통해서 Q-value를 Update한다.

$$Q_n(x,a) = (1-\alpha_n) Q_{n-1}(x,a) + \alpha_n [r_n + \gamma V_{n-1}(y_n)] \quad (3)$$

$$Q_n(x,a) = Q_{n-1}(x,a) \quad (4)$$

여기서 $V_{n-1}(y) \equiv Max_b Q_{n-1}(y,b)$ (5)

또한 Q-learning은 수식(6)에 의해 Episode에 대해서 수렴하는 성질을 가지고 있다.

$$Q_n(x,a) \rightarrow Q^*(x,a) \quad as \quad n \rightarrow \infty \quad (6)$$

IV. Action Selector

본 논문에서 제안한 제어구조는 반사적인 구조와 계층적인 구조를 조합한 복합적인(hybrid) 구조를 가지고 있다. 즉, Sensor-Action의 반사적인 구조의 특징인 센서에 의해 행동패턴(Behavior)의 입력 값과 Navigator를 통한 행동 간의 함수로 정의되어 있으며 CSG (Command Set Generator)를 통해서 학습하며 이를 통해 환경에 적합한 행동의 집합을 생성하는 과정을 통해 ASP(Action Selection Problem)을 해결하는 구조이다.

4.1 제안된 구조

본 구조는 그림 5와 같이 Sense - Act의 구조를 가지고 있으며 Behavior를 두 분류로 나누고 있다. 목적상 자율이동을 하기 위해 필요로 하는 Behavior와 그 외의 목적을 갖는 Behavior로 나누었다. 전자를 우리는 자율이동을 하기 위한 동일한 목적을 갖고 있으므로 이를 Homogeneous Behavior라 부르기로 하였다. Action Selection을 하기 위해서 우리는 Q-learning을 이용하여 보상치를 받으며 이 값으로 Command Set Generator의 각각의 노드 값을 결정하게 되며 이 값을 Navigator에 전달하고 이에 대해서 Behavior와 노드간의 값을 Voting을 통해 선택하는 구조이다.

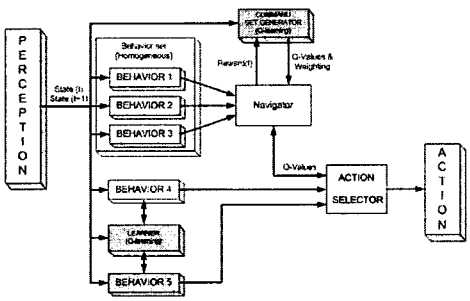

그림 5. ASP를 고려한 제어 구조

4.2 보상함수 (Reward Function)

Q-learning에 있어서 보상함수는 가장 중요한 요소이다. 왜냐하면 이 함수를 어떻게 정의하느냐에 따라서 보상치가 바뀌게 되며 결과적으로는 Q-value를 결정하는 함수이기 때문이다. 이러한 이유로 William D. Smart는 그의 논문에서 보상함수를 두 분류로 나누었다. [7]

4.2.1 Sparse Reward Function

목표점에 도달하면 보상값을 1을 주고 장애물과 충돌을 하게 되면 1을 준다. 위 경우를 제외한 모든 경우에는 0으로 하는 방법.

4.2.2 Dense Reward Function

: 센서를 통해 얻은 로봇과 장애물과의 거리의 총 합을 $\sum dist_{Obstacle}$ 이라 부르고 로봇과 목표점과의 거리를 $dist_{Target}$ 이라하고 이를 수식적으로 표현하면 다음과 같다.

$$R = \frac{\sum dist_{Obstacle}}{dist_{Target}} \tag{7}$$

하지만 이런 Sparse 보상함수는 최적의 정책을 찾기 위해 학습하기 위해서 꼭 필요하지만 Q-learning을 사용하기에 많은 문제점을 가지고 있다. 오직 "1"과 "-1"인 경우를 제외하고 모든 상황이 "0" 이므로 올바르게 선택이 불가능한 보상함수이다.

4.2.3 제안한 보상함수

이를 보완하기 위해서 Dense 보상함수를 이용하였다. 이는 대부분의 모든 상황이 "0" 이 아니므로 행동을 행한 후에 많은 자료를 주지만 Sparse 보상함수보다 실행하기가 어렵다는 단점을 안고 있다. 우리는 이 두 보상함수를 포괄하여 그림 6과 같이 보상 값을 정의하였다

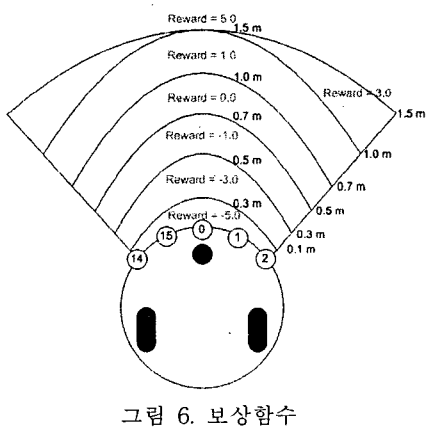

그림 6. 보상함수

4.3 Command Set Generator

CSG (Commend Set Generator)는 상태의 변화에 따라 Q-learning을 통해서 가중치를 부여하며 이로 인해서 Homogeneous Behavior의 Velocity Vector $V=[v,\omega]^T$ 구하게 된다. 구하고자 하는 Behavior의 출력은 그림 7과 같이 구할 수 있으며 이 출력은 CSG에 의해서 생성된 Node에 대해 Voting Algorithm을 이용하여 Vote를 하게 되며 이때 Homogeneous Behavior의 각각의 Behavior에 대해서 가중치를 부여하게 된다.

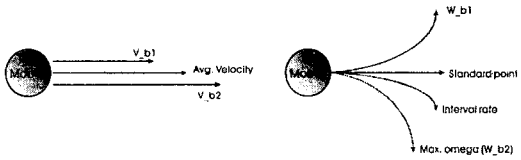

그림 7. Velocity Vector의 값을 얻기 위한 방법

위 그림에 의해서 CSG를 얻으며 이 값을 Navigator에 전달한다.

4.3 Navigator

이 모듈의 주 목적은 이동로봇이 주행을 하기 위해 필요로 하는 속도벡터를 얻는 것이다. Navigator는 CSG로부터 Command Set과 Behavior간의 Voting을 통해서 적절한 속도벡터를 취한다. 이러한 과정에서 다음과 같은 수식에 의해 ASP을 해결한다.

$$\arg\max_x [R_1(x), R_2(x), \cdots, R_n(x)] \quad (8)$$

subject to $x \in X$, $where\ x=(x_1, x_2, \cdots, x_n) \in R^n$

Voting을 함에 있어서 우리는 가중치를 부여하는 Weighted Consensus Voting을 사용하였다. 이로 인해서 식 (8)에 가중치를 부여하였으며 각각의 Behavior에 적절한 값을 가중치로 부여하여 적절한 Action Selection을 할 수 있었으며 여기에서 나오는 Q-value 값을 비교하여 Update하도록 하였다.

V. 모의실험

주어진 환경은 그림 8과 같은 가로 18m, 세로 13m의 가상공간이며 이 공간의 Resolution은 0.5m'이며 이동로봇은 16개의 초음파 센서를 가지고 있다. 또한 현재 단순한 형태의 Behavior 2개에 대해서만 실험을 하였다. 우선적으로 로봇을 제어하는 구조에 있어서 장애물과 회피하기 위해 "Obstacle Avoidance"와 주행 시 요구되는 목적지를 찾기 위한 "Move to Goal"이라는 Behavior를 정의하였다.

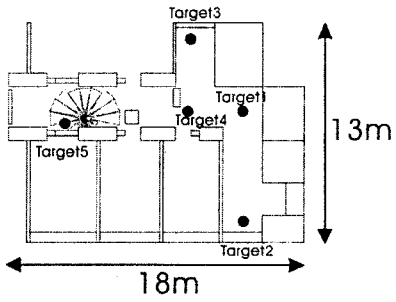

그림 8. 모의실험 환경

5.1 Action Selection

표 2. Command Set Generator의 출력값

State		Step	CSG_V	CSG_W	Diff_behavior
614		0	0	2	0.75
V_1	1.5	1	0.25	1	2
V_2	0	2	0.75	0	
W_1	-2	3	1.25	-1	
W_2	2	4	1.75	-2	
State		Step	CSG_V	CSG_W	Diff_behavior
694		0	0	2	1.224
V_1	1.447	1	0.526	1	2
V_2	1	2	1.224	0	
W_1	2	3	1.924	-1	
W_2	2	4	2	-2	
State		Step	CSG_V	CSG_W	Diff_behavior
655		0	0	2	1.25
V_1	1.5	1	0.55	1	2
V_2	1	2	1.25	0	
W_1	-1.89	3	1.95	-1	
W_2	-2	4	2	-2	

위의 표 2와 같이 CSG에선 독립적인 state에 대해 Velocity Vector값을 Navigator에 전달하고 그 값에 대해서 각각의 Behavior는 Voting을 행하게 된다. 이때 Voting 값은 수식 9에 의해서 얻을 수 있다.

$$Vote = \frac{1}{Dense\_reward} * Vote_{\text{밀}} + \frac{1}{Sparse\_reward} * Vote_{\text{희}} \quad (9)$$

이렇게 얻은 Vote에 대해서 $V=[v, \omega]^T$ 얻기 위해서 CSG의 최대 Vote를 갖는 Node를 찾아 Velocity Vector의 값을 얻는다.

선택된 V를 State에 따른 Q_value값을 Update함으로 최종적인 결과값을 갖게 되고 그 결과 그림 9와 그림 10의 결과를 얻게 되었다.

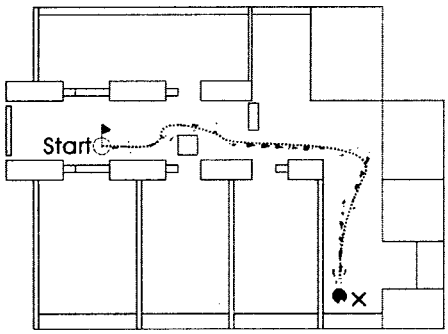

그림 9. Target0에서 Target2까지의 Trajectory

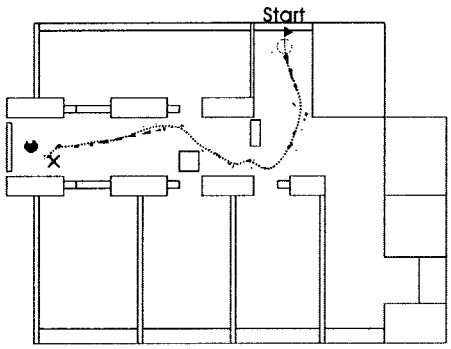

그림 10. Target3에서 Target5까지의 Trajectory

VI. 결론 및 향후계획

동적인 환경 아래에서의 이동로봇의 Velocity Vector V를 선택하기 위해서 기존의 ASP기법에 학습을 겸한 방법들이 있었지만 이들은 실행하기 복잡하다는 단점을 안고 있으며 각각의 Behavior에 대해서 정확히 설계해야 한다는 문제점을 안고 있었다. 이러한 면에서 이 논문은 수학적으로 명확한 Q-learning algorithm과 Voting을 통해 이러한 문제점을 보완하였으며 많은 Behavior를 가지고 이동하는 System에 보다 더 적절하게 적용할 수 있을 거라 기대된다. 차후에 더 많은 Behavior와 Real-world에서 실험을 하여야 할 것이며 학습하는데 걸리는 시간을 단축시켜야 할 것이다.

참고문헌 (Reference)

[1] Mark Humphrys, *Action Selection methods using Reinforcement learning*, (Ph.D. Dissertation), University of Cambridge, 1997.

[2] R.Brooks, *A Robust Layered Control System for a Mobie Robot*, IEEE Journal of Robotics and Automation, 1991.

[3] Khatib, O., *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots*, in proceedings of the International Conference on Robotics and Automation, 1990.

[4] Chin-Teng Lin and C.S. George Lee, *Neural Fuzzy Systems: a neuro-fuzzy synergism to intelligent systems*, Prentice-Hall, 1996.

[5] Julio K. Rosenblatt, *DAMN: A Distributed Architecture for Mobile Navigation, (PhD. Dissertation)*, Carnegie Mellon University Robotics Institute Technical Report CMU-RI-TR-97-01, 1997.

[6] C. Watkins, P. Dayan, *Technical Note: Q-learning*, Machine Learning, Vol.8, pp.279-292, 1992.

[7] William D. Smart, Leslie Pack Kaelbling, *Effective reinforcement learning for mobile robots*, IEEE International Conference on Robotics and Automation, Vol. 4, pp.3404-3410, 2002