택시 운행 중 상태변화에 대한 자율적 의사결정을 위한 퍼지 에이전트

임 춘 규† · 강 병 욱††

요 약

본 논문에서는 상태변화에 대한 자율적 의사결정을 하는 퍼지논리를 이용한 에이전트의 구현을 택시에 적용 하는 것을 연구의 목적으로 한다. 이를 위하여 인공 지능 이론을 기반으로 한 실시간 반응형 에이전트를 통하여 인공 지능적으로 운행하는 자동차에 대해서 실험을 하였다. 실시간 반응형 에이전트를 구성하기 위한 추론방식으로는 max-product 기법과 n개 퍼지 규칙들 또는 연관들 (A_1, B_1), ..., (A_n, B_n)을 가지는 상황을 고려하여 비퍼지화 작업을 수행하여 중심값을 추출하여 추론 작업을 실행하였다.

키워드 : 퍼지논리, 퍼지추론, 자율적 의사결정, 반응형 에이전트

A Fuzzy Agent System to Control the State Transition for an Autonomous Decision Making on Taxi Driving

Chun Kyu, Lim† · Byung Wook, Kang††

ABSTRACT

In this paper, we apply software agents, which use fuzzy logic and make autonomous decisions according to state transitions, to car driving environment. We carry out an experiment on artificial intelligent car driving in terms of real-time reactive agents. Inference techniques for constructing real-time reactive agents consider the settings with max-product inference, *n*-fuzzy rules, and *n*-associatives (A_1, B_1), ···, (A_n, B_n). Then we perform defuzzification processes, extract a central value, and work out inference processes.

Key Words : Fuzzy Logic, Fuzzy Inference, Autonomous Decision-making, Reactive Agent

1. 서 론

오늘날 교통량 증가로 인하여 차량의 통제가 어려운 문제점이 발생하고 있다. 이를 위하여 자율주행에 인공 지능을 부여하는 에이전트시스템의 활용이 부각되고 있으며, S-R 에이전트(Stimulus Response Agent)를 적용 및 응용한 화상 표현 행렬을 직접 행동으로 변환한 인공 신경망(artificial neural network)을 사용하기도 한다. 그 사례로 다수의 뉴런 결합에 의하여 복잡한 처리를 하는 자율주행차를 위한 ALVINN시스템이 있으며, 이것의 개발로 인하여 지능적 퍼지 에이전트 시스템의 성능이 향상되고 있다[3].

본 논문에서는 퍼지규칙 외부 환경의 상태변화에 즉각적인 응답을 하는 반응형 에이전트를 구현하고 실험한다. 에이전트를 구성한 추론 방식 max-pro duct 기법을 사용, n개 퍼지 규칙들 또는 연관들 A_i와 B_i 사이의 연관 또는 관계를 코드화하기 위해 n개 행렬 M_i, ... , M_n이 되므로 비퍼지화 수행, 중심값을 추출하여 추론 작업을 실행하였다[4].

† 정 회 원 : 학교법인 동경학원 이사장
†† 종신회원 : 영남대학교 전자정보공학부 교수
논문접수 : 2005년 4월 13일, 심사완료 : 2005년 6월 10일

2. 이론 및 기술 배경

2.1 퍼지를 이용한 반응형 에이전트 구성

2.1.1 지식 표현

에이전트가 문제해결을 위해 문제의 기술과, 그것에 이용할 지식을 컴퓨터에서 실행 가능한 형태로 나타내기 위해서는 지식의 표현이 많은 비중을 차지한다.

지식표현의 두 가지 측면을 살펴본다면, 사실이라는 관련된 세계에서의 진리로서, 우리가 표현하고자 하는 것이 있고 사실에 대한 표현으로 어떠한 사실을 특정한 구조로 기술한 것으로 인공지능 프로그램에서 처리할 대상이 있다. 우리가 표현하고자 하는 지식은 컴퓨터를 이용하여 처리되기 위해 심볼의 형태로 묘사된다. 이와 같이 하여 표현된 지식은 추론과정을 통하여 새로운 내부 표현 형태로 변환되

며, 이 결과를 이용하여 새로운 결론을 얻을 수 있다.

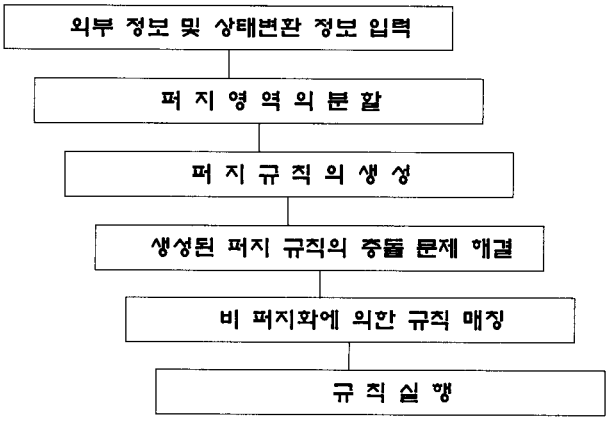
지식표현 중에서도 규칙을 이용한 표현이 가장 많이 사용되는 방법이다. 인공지능에서 규칙이란 일반적인 의미의 규칙보다 훨씬 좁은 의미의 용어이다. 여기서의 규칙이란 주어진 상황의 위한 권고, 지시, 전략을 나타내는 정형화된 표현방법이다. 규칙은 다음과 같이 조건(IF) - 결론(THEN) 문장으로 표현된다. 그리고 조건과 결론 사이의 규칙은 '→'기호로 나타내기도 한다. 즉 가정이 A이고 결론이 B일 때, (A → B)로 표시한다. 이와 같은 규칙을 이용하여 표현된 지식베이스를 규칙 기반 시스템이라 한다. 예를 들어 다음과 같은 규칙들을 생각해 보자.

① 조건 : 전방 100m전에 감지기가 있다.
결론 : 속도를 -5만큼 줄여라.
② 조건 : 직선 길에 주행 중에 장애물이 없다.
결론 : 엑셀레이션을 6만큼 올려라.
③ 조건 : 도착지점이 500m남았다.
결론 : 브레이크를 -4만큼 줄여라.

2.1.2 퍼지에 의한 추론 사이클모형

반응형 에이전트시스템에서 추론을 하기 위해서는 몇 단계를 거쳐야 한다. 퍼지 지식베이스에서 규칙이 생성되기 위해서는 우선적으로 외부로부터의 인지된 정보를 현재 시스템이 보유하고 있는 상태의 정보를 입력하여 퍼지영역의 분할 작업을 수행해야 한다.

일반적인 지식베이스 시스템에서는 규칙이 생성되면 생성된 여러 개의 충돌된 규칙 중에서 하나를 선택하는 알고리즘에 의해서 충돌문제를 해결한다. 일반적으로 FIFO방식이나 Round_Robin 방식 등으로 결정한다. 그러나 퍼지 지식베이스에서의 충돌된 규칙 중에서 2개 이상의 규칙이 충돌할 경우 조건절과 결론절의 멤버 쉽(product operation) 값을 곱하여 소속 함수 값이 큰 퍼지규칙을 선정하며, 다중규칙인 경우에는 비퍼지화에 의한 규칙 매칭 작업을 수행하여 출력 값을 만든다. 다음의 (그림 1)은 추론 사이클의 단계를 보인 그림이다.

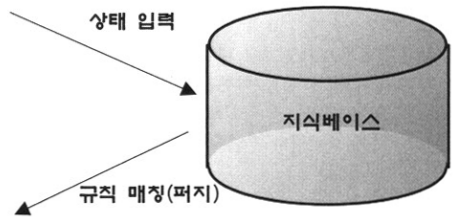


(그림 1) 추론 사이클의 단계

2.2 반응형 에이전트 알고리즘

2.2.1 상태변화에 따른 규칙결정 알고리즘

에이전트는 인지된 상태의 범위에 관한 내부 지식 을 계속적으로 기억하고 있는 에이전트를 말한다. 즉, 인지된 상태가 지식베이스와 조건부와 정확히 일치하는 것이 없더라도 인지된 상태의 범위로부터 유사한 결론 부를 찾아낼 수 있게 된다.



(그림 2) 입력정보와 내부 상태에 의하여 상황에 맞는 규칙을 결정과정

이런 에이전트는 내부 상태를 갖는 반응 에이전트로서, 입력정보와 내부 상태에 의하여 상황에 맞는 규칙을 결정한 후 행동한다.

다음은 상태를 갖는 반응형 에이전트 알고리즘의 개략을 보인 것이다. state-UPDATE_STATE(state, action)에 의해서 현재의 상태와 행동에 의해서 다음상태가 결정되고 반복적으로 처리됨을 볼 수 있다[8, 9].

```
Function Reflex-Agent-With-State(percept)
      return action
 static: state, a description of the current world
 state
     rules, a set of condition-action rules
  state-UPDATE_STATE(state, percept)
  rule-RULE_MATCH(state, rules)
  action-RULE[rule]
  state-UPDATE_STATE(state, action)
 return action
```

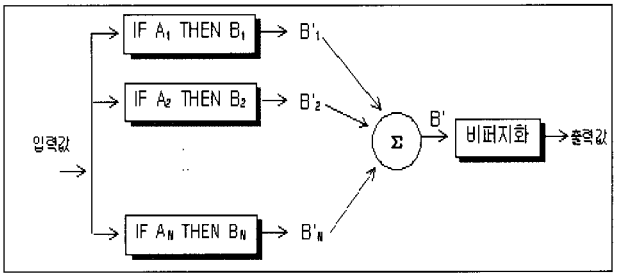
2.2.2 퍼지 규칙베이스 매칭 및 수행

n개 퍼지 규칙들 또는 연관들 (A_i, B_i), ... , (A_n, B_n)을 가지는 상황을 고려한다. 이 상황은 A_i와 B_i 사이의 연관 또는 관계를 코드화하기 위해 n개 행렬 M_i, ... , M_n이 된다.

퍼지 규칙들의 이 집합에서 우리의 관심은 단일 측정 A′가 주어진 B에서 전체 믿음 결과이다. 우리는 규칙들의 은행에 병렬적으로 A′를 적용하는 것, 각 규칙에 대해 유도된 퍼지 집합 B'_i를 생성한다. 그리고 나서 유도된 퍼지집합 B′의 결과 합성을 만들기 위해 B'_i의 모두를 합친다. 아래의 표준 집합 합집합 연산을 사용한다. 여기서 B는 도메인 X에서 정의한다.

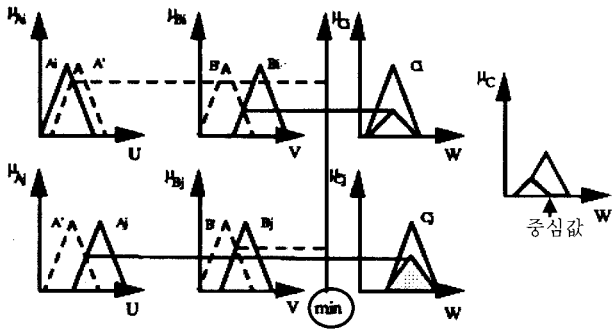
$$B' = B'_1 \cup B'_2 \cup \ldots B'_{n-1} \cup B'_n$$
$$= \max(B'_1(x), B'_2(x), \ldots, B'_{n-1}(x), B'_n(x)) \text{ for all } x \in X$$

이 합집합 연산 다음에, 우리는 중심 방법을 사용하여 결과 B′를 비퍼지화할 수 있다. 이 프로세스 는 보통 출력 값 y_i를 생산한다. 다음의 (그림 3)은 비퍼지화 작업을 예시하였다.



(그림 3) 퍼지 규칙 시스템 구현

다음의 (그림 4)는 max-product 추론에 의한 중심 값을 계산한 것이다.



(그림 4) 다중 규칙들을 위한 max-product 추론

3. 구 현

3.1 구현을 위한 조건

구현을 위한 전제조건은 다음과 같다.

(1) 주행구간 내 신호등은 2개(195m, 595m)가 있으며, 100 m 전에 인식한다고 가정한다.

(2) 갑자기 뛰어드는 사람과 앞의 차는 임의로 발생시키며, 차는 50%, 사람은 25%의 빈도로 발생한다.

(3) 퍼지화 단계로서 언어변수 정의는

① 입력 언어변수는 속도: 0 - 20m/s, 거리: 0 - 1000m 이고, 출력 언어변수는 엑셀레이션: -10 - 10m/s/s이다. 그리고 반응에 대한 언어 변수의 효율성을 측정하기 위해 동일한 차량에 언어변수의 소속 값의 범위를 달리하여 실험하 였다.

② 두 번째로 입력 언어변수, 거리, 출력 언어변수는 동일하게 적용하고 단, 언어변수의 범위를 달리하여 ①, ② 차량의 차이를 확인한다.

(4) 택시의 구동제어를 위한 퍼지 추론은 위와 같이 다중 입력, 단일 출력 방식을 선택한다. 본 논문에서는 전방에 신호등, 차, 사람에 관한 변수는 행렬(5×5)로 처리를 하였다.

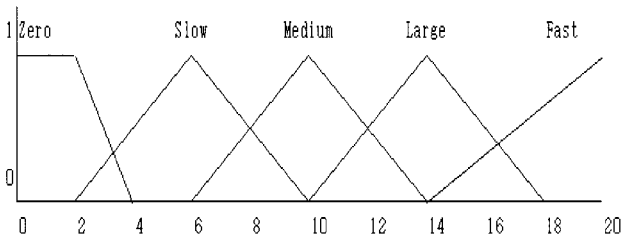
이 실험에서 ① 소속 값의 실험을 1번 실험이라 하고 ②번 소속 값의 실험을 2번 실험이라고 한다.

㉠ 퍼지 변수들의 용어

〈표 1〉 ①, ②언어변수들의 용어사전

속도	거리	엑셀레이션
Zero	Far	Hard-Brake
Slow	Medium	Slight-Brake
Large	Very-Close	Slight-Accel
Fast	Zero	Hard-Accel

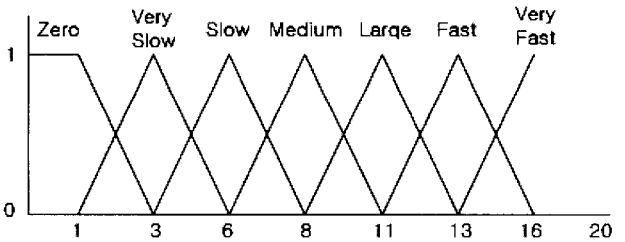
㉡ 속도에 대한 퍼지 집합



(그림 5) ①속도의 소속 함수

〈표 2〉 ① 속도의 소속 값들

Zero (ZS)		Slow (SS)		Medium (MS)		Slight-Fast(LS)		Fast (FS)	
X	Y	X	Y	X	Y	X	Y	X	Y
0	1	2	0	6	0	10	0	14	0
2	1	6	1	10	1	14	1	20	1
4	0	10	0	14	0	18	0		



(그림 6) ②속도의 소속 함수

〈표 3〉 ②속도의 소속 값들

Zero (ZS)		Very-Slow (VS)		Slow (SS)		Medium (MS)		Large (LR)		Fast (FS)		Very-Fast (VF)	
X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y
0	1	1	0	3	0	6	0	8	0	11	0	13	0
2	1	3	1	6	1	8	1	11	1	13	1	20	1
3	0	6	0	8	0	11	0	13	0	16	0		

㉢ 거리에 대한 퍼지 집합

〈표 4〉 ①거리의 소속 값들

Far (FD)		Medium (MD)		Close (CD)		Very-Close (VD)		Zero (ZD)	
X	Y	X	Y	X	Y	X	Y	X	Y
0	1	700	0	800	0	850	0	950	0
700	1	800	1	875	1	925	1	1000	1
800	0	900	0	950	0	100	0		

〈표 5〉 ②거리의 소속 값들

Very-Far (VF)		Far (FD)		Medium (MD)		More-Close (MC)		Close (CD)		Very-Close (VC)		Zero (ZD)	
X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y
0	1	400	0	500	0	600	0	700	0	800	0	900	0
400	1	500	1	600	1	700	1	800	1	900	1	1000	1
500	0	600	0	700	0	800	0	900	0	1000	0		

㉣ 엑셀에 대한 퍼지 집합

〈표 6〉 ①엑셀레이션의 소속 값들

Hard-Brake (HB)		Slight-Brake (SB)		Zero (ZA)		Slight-Accel (SA)		Hard-Accel (HA)	
X	Y	X	Y	X	Y	X	Y	X	Y
-10	1	-8	0	-4	0	0	0	4	0
-8	1	-4	1	0	1	4	1	8	1
-4	0	0	0	4	0	8	0	10	1

〈표 7〉 ②엑셀레이션의 소속 값들

Hard-Brake (HB)		Brake (BB)		Slight-Brake (SB)		Zero (ZA)		Slight-Accel (SA)		Accel (AA)		Hard-Accel (HA)	
X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y
-10	0	-9	0	-6	0	-3	0	0	0	3	0	6	0
-9	1	-6	1	-3	1	0	1	3	1	6	1	9	1
-6	0	-3	0	0	0	3	0	6	0	9	0	10	1

3.2 추론 엔진 구성을 위한 주요 알고리즘

3.2.1 소속 값 계산

거리의 경우 입력된 거리 값(xo)과 거리의 논의영역(행렬 X)과 퍼지 집합(행렬 A)을 통해 소속 값을 계산한다. 속도의 경우도 거리와 마찬가지로 입력된 속도 값과 속도의 논의영역과 퍼지 집합을 통해 소속 값을 계산한다.

```
float MemberShip(float xo, float X[], float A[])
{
 // 논의영역 밖이면 가장 가까운 소속 값 계산
 if (xo < X[0]) {
 return(A[0]);
 } else if (xo > X[N_ELEM -1]) {
 return(A[N_ELEM -1]);
 }

 // 소속 값 계산
 for (si = 0; si < N_ELEM-1; si++) {
 if ((xo >= X[si])&&(xo <= X[si+1])) {
 if (X[si] == X[si+1]) {
  소속 값은 가장 큰 값으로
 } else {
  소속 값은 선형 보간법으로
 }
 }
 }
 return(ao);
}
```

3.2.2 퍼지 추론

(1) max-product 추론

max-product 추론에 의해 엑셀레이션의 소속값을 계산한다. 예를 들어 「A_HA = max(S_SS*D_FD, A_H A)」의 경우는 「IF 속도 = SS AND 거리 = FD THEN 엑셀레이션 = HA」를 계산하는 것이다. A-HA에는 엑셀레이션의 소속 값이 저장된다.

```
A_SA = max(S_ZS, A_SA);
A_HA = max(S_SS*_FD, A_HA);
    . . . .    . . . .
A_SB = max(S_FS*D_VD, A_SB);
A_HB = max(S_FS*D_ZD, A_HB);
```

3.2.2 매칭되는 규칙들의 합집합

max-product 추론에서 계산된 엑셀레이션의 소 속 값과 엑셀레이션의 퍼지집합을 통해 엑셀레이션 의 퍼지집합을 추론할 수 있다. 예를 들면 「Infer Engine(A_HB, HB, R)」은 소속 값 A_HB와 퍼지 집합 HB의 곱을 통해 엑셀레이션의

```
InferEngine(A_HB, HB, R);
Maximum(R, ZERO, O);
InferEngine(A_SB, SB, R);
Maximum(R, O, O);
InferEngine(A_ZA, ZA, R);
Maximum(R, O, O);
InferEngine(A_SA, SA, R);
Maximum(R, O, O);
InferEngine(A_HA, HA, R);
Maximum(R, O, O);
```

집합(행렬 R)을 추론한다. 매칭되는 엑셀레이션 퍼지 집합은 「Maximum(R, O, O)」에 의해 행렬O에 더해진다.

3.2.3 비퍼지화

매칭되는 규칙들의 합집합에서 합해진 엑셀레이션의 퍼지 집합(행렬 B)과 엑셀레이션의 퍼지집합(행렬 Y)은 퍼지 중심값 공식에 의해 Crisp value(Yo)로 계산된다.

yB_sum은 $\sum_{j=1}^{p} y_j m_B(y_j)$, B_sum은 $\sum_{j=1}^{p} m_B(y_j)$을 나타낸다. yB_sum / B_sum을 통해 비퍼지화 값이 계산된다.

```
void DeFuzzyier(float Y[N_ELEM], float
  B[N_ELEM], float *Yo)
{
 yB_sum = 0.0; B_sum = 0.0;

 for (si = 0; si < (N_ELEM -1); si++) {
 yB_sum += ((Y[si] + Y[si+1])/2)*
   ((B[si] + B[si+1])/2);
 B_sum += (B[si] + B[si+1])/2;
 }

 if (B_sum != 0.0) {
 *Yo = yB_sum/B_sum;
 }
```

3.3 시스템의 모델링 및 설계

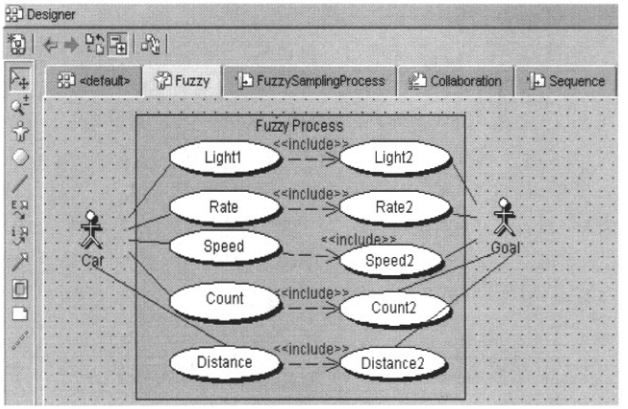
UML 다이어그램을 이용하여 어플리케이션을 모델링하고 이를 자동 변환하기 위하여 Together라는 객체 지향 CASE Tool을 사용 했다[7].

3.3.1 분석 및 설계

(1) 유스케이스 다이어그램

유스케이스 다이어그램은 엑터(Actor)와 유스케이스 간, 유스케이스와 유스케이스 간의 업무적인 관계를 나타낸다. 유스케이스 다이어그램을 작성하는 목적은 목표시스템의 대상 업무 영역을 분석하여 업무기능을 체계적으로 분할해 나간다.

(그림 7)은 무인자동차가 목표 지점에 도달하기 위해서 필요한 개략적인 내용을 도식화하였다.

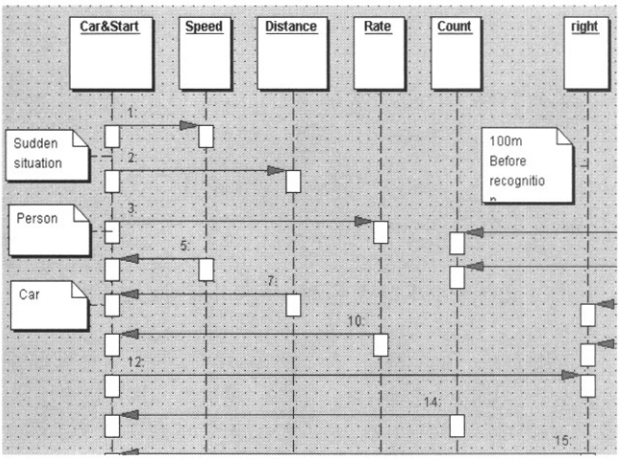


(그림 7) Usecase Diagram

(2) Sequence Diagram

순서도는 어떤 결과를 산출하기 위해 행위를 하는 분류자(Classifier)들 간의 상호작용을 시간적인 흐름으로 나타낸다. 그리고 객체 지향 프로그램인 어플리케이션 클래스의 프로그램 로직이 된다[1, 6].

아래의 그림은 신호등의 위치가 가까워짐에 따라 색깔에 따른 신호등의 시간을 받고 다음 새로운 신호등의 색깔과 그에 해당하는 신호의 입력을 기다리는 Sequence Diagram이다.



(그림 8) Sequence Diagram

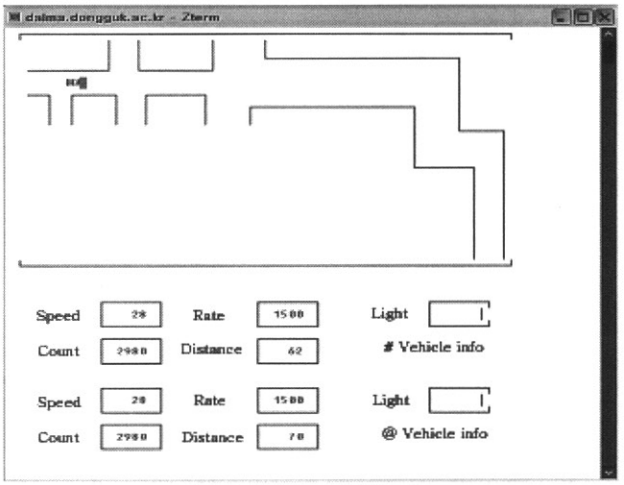
```
if(reason->Upside_time != 0)
        {
                reason->Upside_time = 1 ;
        } // if
                if( (strcmp(reason->Signal_color, Not) == 0) )
        {
                if((First_signal - Signal_distant) <=
                (int)reason->Total_distant &&
                (int)reason->Total_distant <= First_signal ) || ( (
                (Second_signal - Signal_distant) <=
                (int)reason->Total_distant) &&
                (int)reason->Total_distant <= Second_signal)
        {
                while(node->next != NULL)
                node = node->next ;
                if( (node->next == NULL) && (reason->Upside_time
                == 0) )
        {
                Signal_time2(reason) ;
        }
        }
        }
```

(그림 9) 신호등 확인 소스

4. 실험 결과

4.1 환경변수에 따른 입력이 다른 두 대 차량실험

(그림 10)은 입력 변수가 다른 두 대의 차량이 동시에 출발하는 상황이다. 두 대의 거리와 환경 변수의 조건은 동일하고 입력변수의 차이를 두어 두 대의 효율성을 측정하기 위한 방법이다.

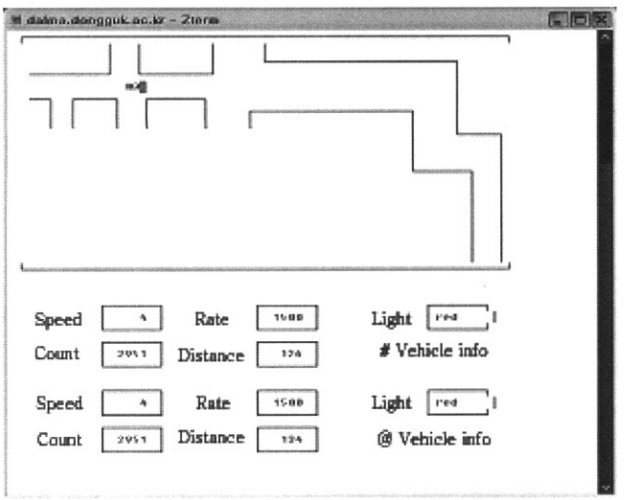


(그림 10) 입력 변수가 다른 두 대의 차량이 동시에 출발하는 상태

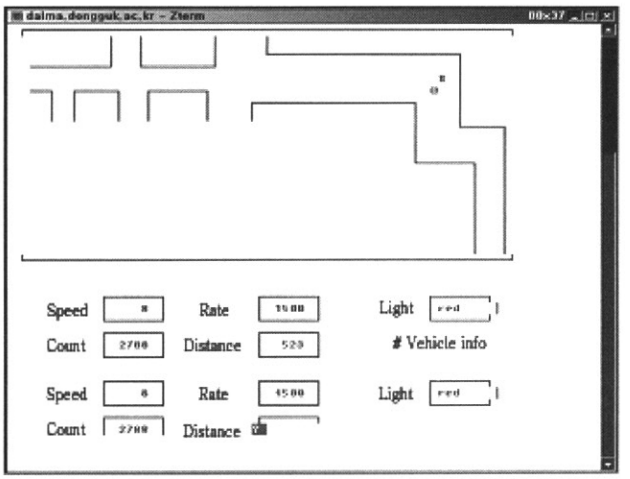
(그림 11)은 입력 변수가 다른 두 대의 차량이 주행 중에 신호등을 인지한 상태이다. 각각의 차량의 입력 변수가 다르기 때문에 똑같은 환경 변수에 조금씩 다르게 행동한다.

(그림 12)는 지금의 신호등의 상태는 red이고 일정시간이 지나면 다음 신호로 바뀐다. 실시간으로 신호에 따른 반응을 나타낸다.

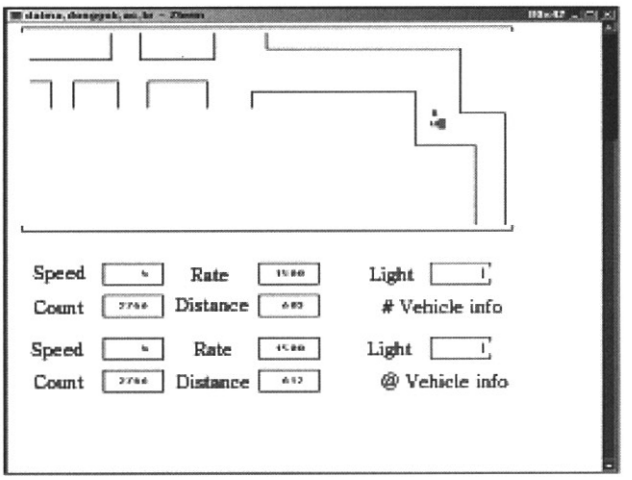
(그림 13)은 모든 신호를 지나 두 대의 차량이 목적지로 행하는 상태이다. 시작에서 운행 중에 많은 환경 변수로 다른 입력 변수에 의한 거리의 차이를 보인다.



(그림 11) 입력 변수가 다른 두 대의 차량이 주행 중에 신호등을 인지한 상태



(그림 12) 입력변수가 다른 두 대의 차량이 두 번째 신호등에 근접해 가는 상태



(그림 13) 입력변수가 다른 두 대의 차량이 목적지로향하는 상태

4.2 비교 분석

①, ②번 차량의 각기 다른 언어변수에 대한 효율성을 측정한다. 각기 차량의 언어변수에 따른 출력변수의 차이가 차량의 효율성을 확인 할 수 있는 것이라고 본다. 차이점을 언어변수의 측정에서는 두 차량의 입력변수에서 거리를 보면 1000m로 동일하다.

주행 중의 출력변수에서 나타나는 속력에 비례하는 거리를 효율성의 척도로 적용한다. 또한, 속력에 대해 시간도 비례한다. 그렇기 때문에 효율적인 출력변수가 적용되는 차량이 정해진 거리를 더욱더 빨리 도착한다는 이론이 나온다. 그것을 토대로 두 차량이 1000m앞의 도착지에 도착하기까지의 시간을 측정하여 두 차량의 효율성을 확인한다. 그리고 주행도중 차량에 제약을 주는 장애물이 없는 상황과 장애물이 있는 상황을 나누어서 효율성을 측정해야 한다.

4.2.1 입력 변수가 다른 차량의 환경 변수에 의한 반응 비교

입력이 다른 두 대의 차량에 적용되는 환경 변수에 따른 각각의 대처 능력을 알아본다. 두 대의 차량에 주어지는 환

경 변수는 신호등과 사람, 다른 차량으로 동일하고 반응 시 주어지는 조건을 동일하다는 가정 하에 비교해본다.

(1) 신호등을 인지한 경우

주행 중 신호등을 인지한 경우 차량은 신호등의 점멸색을 인지하고 차량의 감속을 실행해야한다. 두 대의 차량의 신호등인지를 동일하다고 가정하고 두 차량의 속력도 50km/h로 동일하게 주행한다는 전제하에 동일한 Slignt_Brake(SB)를 수행한다면 ①번 차량은 (-4)를 감속할 것이고 ②번 차량은 (-3)의 감속을 보인다.

(2) 전방에 사람을 인지한 경우

전방에 사람을 인지한 경우에는 두 부분으로 나뉜다. 전방에 사람이 인지되면 5m전방에서 무조건 정지 상태로 들어간다. 그리고 사람이 전방이 아닌 도로의 옆을 지나가고 있는 상황이 나올 수 있다. 그 경우 거리를 측정하여 속력을 조절한다. 거리가 동일하다고 가정할 경우 그것에 대한 속력 감소를 보인다. 거리에 따른 속도의 소속 함수가 Large(LS)라고 가정한다면 ①번 차량의 속도는 14km/h 이고 ②번 차량의 속도는 11km/h로 나타날 것이다. 이것에 대한 엑셀레이션은 이 속도에 대한 엑셀레이션의 작용이 상이해진다. 속도 14km/h에서의 엑셀레이션은 Slight_Brake(SB)가 작용하여 (-4)의 감속을 보이며 속도 11km/h에서의 엑셀레이션은 Zero(ZA)가 작용하여 (0)이 된다.

(3) 앞의 차를 인지한 경우

앞에 주행 중인 차량과의 거리는 주행 중에는 4m를 유지하고 정지 시에는 1m를 유지한다. 그러므로 속력에 따른 제한 사항을 주어야한다. 차와의 거리를 유지하기 위해 속도의 소속 함수 중 Slight_Accel(SA)를 적용한다면 ①차량의 경우 엑셀레이션의 작용이 (4)를 가속할 것이고, ②차량의 경우는 (3)를 가속한다.

(4) 주행거리 차이에 대한 비교

두 차량이 1000m의 목표지점에서 875m를 주행할 경우 ①차량의 거리에 대한 소속 함수는 Close(CD)가 되고 ②차량의 소속 함수는 Very_Close(VC)가 된다. 이와 같이 나타나는 거리의 소속 함수에 따른 엑셀레이션의 작용은 ①번 차량에서 Slight_Brake가 작용해 (-4)만큼 감속을 나타내고 ②번 차량에는 Brake가 작용하여 (-6)만큼 감속이 작용한다.

5. 결론 및 향후과제

본 논문에서는 실시간 반응형 에이전트를 통하여 인공 지능적으로 운행하는 자동차에 관하여 실험 하였다. 인공지능의 전체적인 분야가 아닌 반응형을 중점적으로 다룬 실험이다. 실제 환경에서 나타날 수 있는 몇 가지 환경 변수와 일정 거리를 기본사항으로 측정하였다. 실질적으로 운행을 위해서는 고해상도의 화상처리기와, 장면분석, 스테레오 시각과 깊이 정보를 위해 인공지능에서 사용되는 모든 분야를 총 동원하여야 한다.

이것을 구체화하기 위해서는 인간이 능력을 가장 잘 부합시킬 수 있는 인공지능 이론을 토대로 퍼지 지식을 학습하는 과정이 포함되는 지식베이스를 구성하여야 하며, 개선된 성능을 갖는 퍼지추론 엔진 구성을 구성하여야 한다[2, 5]. 자동차의 관점에서 봤을 때 모든 환경변수를 인식하고 즉각 반응을 하는 로봇이 생산된다면 몸이 불편한 사람이나 운전을 할 수 없는 사람들에게 많은 이득을 줄 것이다. 그리고 더 나아가 이런 환경 변화에 대한 즉각적인 반응을 보이는 에이전트가 구성된다면 자동차뿐만이 아니라 여러 산업 분야에 응용될 것이다.

차후의 연구 과제로는 영역별로 업무를 분담하는 반응형 에이전트와 같은 단일형 에이전트가 아닌 인공지능의 전반적 분야를 전체적으로 포괄한 멀티 에이전트에 대한 지속적인 연구가 요구된다.

참 고 문 헌

[1] Grady Booch. *Object-Oriented Analysis and Design, 2nd Edition*, Benjamin/Cummings, 1994.

[2] A. Gyenesei. *A Fuzzy Approach for Mining Quantitative Association Rules.* TUCS Technical Reports No.336, 2000.

[3] M. Jamshidi, R. Kelsey, and K. Bissel. *Traffic Fuzzy Control: Software and Hardware Implementations.* Proc. of the Fifth IFSA World Congress, 1993.

[4] J. Kim. *A Fuzzy Logic Control Simulator for Adaptive Traffic Management.* Proc. of IEEE Int'l Conf. on Fuzzy Sysetms, 1997.

[5] J. W. Kim. A Traffic Light Controlling FLC Adaptable to Various Traffic Volumes. *Journal of KISS, 24*(9), 1997.

[6] OMG, *OMG Unified Modeling Language Specification* Ver 1.3, 1999.

[7] Together, http://www.togethersoft.co.kr/together_web.

[8] J. M. Won and J. S. Choi. *Traffic Engineering.* Parkyoung Press, 1993.

[9] Y. T. Kim and Y. J. Lee. A Survey on the Fuzzy Control Systems with Learning/Adaptation Capability. *Journal of KFIS, 5*(3), 1995.

임 춘 규

e-mail : limchk@korea.com
1985년 경일대학교 전자계산학과(학사)
1993년 영남대학교 정보처리(공학석사)
2003년 영남대학교 컴퓨터공학과 박사수료
2005년 현재 학교법인 동경학원 이사장
관심분야 : UML, 인공지능, 에이전트 등

강 병 욱

e-mail : Bwkang@ynucc.yeungnam.ac.kr
현 재 영남대학교 전자정보공학부 교수
관심분야 : 소프트웨어공학, UML, 인공지능