論文96-33B-10-9

모멘트 균형점의 효율적 탐색을 갖는 비제산기 COA 비퍼지화기

(A Dividerless COA Defuzzifier with An Efficient Searching of Momentum Equilibrium Point)

金 大 鎭 *, 趙 仁 顯 *

(Daijin Kim and Inhyun Cho)

요 약

본 논문은 보다 정확하고 신속한 비퍼지화 연산을 수행하는 새로운 COA(Center of Area) 비퍼지화기를 제안한다. 제안한 COA 비퍼지화기는 비퍼지화 연산시 소속값 뿐 아니라 소속 함수의 폭을 고려하여 보다 정확한 비퍼지화값을 얻는다. 나아가, 출력 변수상에서 좌·우측 모멘트가 균형을 이루는 모멘트 균형점을 찾는 것으로 비퍼지화 연산을 대신해 연산 비용과 시간이 많이 드는 나눗셈 연산을 피한다. 모멘트 균형점은 Coarse 탐색시 모멘트 계산점이 퍼지항 단위로 움직여 인접하는 두 항까지 도달시킨 다음 Fine 탐색시 모멘트 계산점을 단위 구간씩 움직이는 Coarse-to-Fine 탐색에 의해 Ruitz[1] 등에 의해 제안된 동일 간격 탐색보다 퍼지항의 개수가 M일 때 최대 O(M)배의 탐색 속도 향상을 얻을 수 있다. 제안한 비퍼지화기의 정확성을 확인하기 위해, 모멘트 균형점 탐색에 의해 얻어진 비퍼지화값과 산술 연산 결과 얻어진 정확한 비퍼지화값을 비교해 본다. 제안한 COA 비퍼지화기를 트럭 후진 주차 문제에 적용하여 기존의 COA 비퍼지화기와 제어 성능 개선 정도를 평균 주행 거리면에서 비교해 본다.

Abstract

This paper proposes a new COA (Center of Area) defuzzifier that is working in the accurate and fast manner. The proposed COA defuzzifier involves both membership values and the spans of membership functions in calculating a crisp value. In addition, it avoid division by replacing the COA calculation with the searching of the momentum equilibrium point. The moment equilibrium point is searched in the coarse-to-fine manner such that the moment computing points during the coarse searching are moved in the interval of fuzzy terms until they are reached at two adjacent fuzzy terms and those points during the fine searching are moved in the interval of one unit. This coarse-to-fine searching method accerlates the finding of the moment equilibrium point by O(M) maximally when compared with the equal interval searching method of Ruitz[1]. In order to verify the accuracy of the proposed COA defuzzifier, the crisp values obtained from the proposed coarse-to-fine searching are compared with the precise crisp values from the arithmetic calculation. Application to the truck backer-upper control problem of the proposed COA defuzzifier is presented. The control performance is compared with that of the conventional COA defuzzifier in terms of the average tracing distance.

* 正會員, 東亞大學校 컴퓨터工學科

(Dept. of Computer Eng., DongA University)

※ 이 논문은 95년도 정보통신부 과학 기초 연구지원 사업 연구비에 의하여 연구되었음

接受日字:1996年5月21日, 수정완료일:1996年9月16日

I. 서 론

퍼지 논리는 기존 논리 체계보다 인간의 사고나 자연어의 특성과 많은 유사성을 가지고 있어서, 실세계의 근사적이고 불확실한 현상을 기술하는데 효과적으로 이용될 수가 있다.[2] 이러한 퍼지 논리에 기초를 두고

있는 퍼지 제어기는 인간의 판단과 같은 애매성을 포함하는 제어 규칙을 언어적 형태인 IF-THEN 형식으로 표현하고 이들로부터 퍼지 연관 관계에 근거한 퍼지 추론을 행한 다음 얻어진 퍼지 출력을 비퍼지화 과정에 의해 확정된(Crisp) 값으로 바꾸어 제어를 실행하는 제어 시스템을 말한다. 퍼지 제어기의 장점은 시스템의 특성이 복잡하여 기존의 정량적인 방법으로는 해석할 수 없거나, 얻어지는 정보가 정성적이며, 부정확하고 불확실한 경우에 있어서 기존의 제어기보다 우수한 제어결과를 나타낸다는 점이다.

그림 1은 퍼지 제어기의 일반적 구성도를 나타낸 것으로 크게 4가지 구성 요소 - 퍼지화부, 추론 엔진부, 퍼지 규칙 베이스부, 그리고 비퍼지화부로 나뉜다. 각 부분의 동작 설명은 다음과 같다.

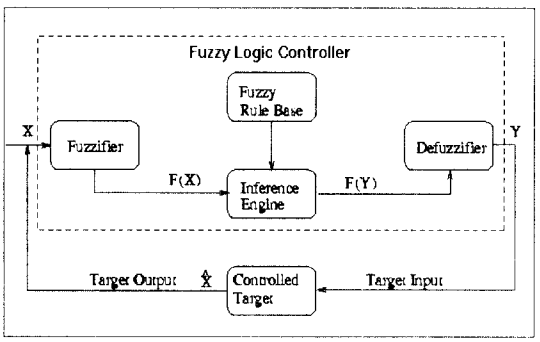

그림 1. 퍼지 제어기의 일반적 구조
Fig. 1. A typical organization of fuzzy logic controller.

퍼지화부에서는 입력 변수의 값을 측정하고, 입력 변수의 영역을 전체 집합범위에 맞게끔 크기 변환한 뒤 입력값을 적절한 언어적인 값으로 변환시키고, 추론부에서는 퍼지 관계와 퍼지 논리의 추론 규칙을 사용하여 퍼지 제어 출력을 결정하며, 퍼지 규칙 베이스부에서는 퍼지 논리 제어에서의 퍼지 자료를 조작하고 언어적 제어 규칙을 정의하는데 필요한 사항들을 정의한 데이타 베이스와 제어 전문가가 수행하는 일련의 제어 과정을 언어적 제어 규칙들로 나타낸 제어 규칙부로 구성되고, 그리고 비퍼지화부에서는 퍼지 출력값을 실제 제어 입력에 맞게끔 변환시켜 실제 제어 입력으로 사용할 수 있는 확정된 출력값으로 변환시켜 준다.

기존의 퍼지 제어기가 가지는 비퍼지화 방법에는 최대값 방법(Max), 최대 평균법(Mean Of Maximum: MOM), 그리고 무게 중심법(Center Of Area:COA)이 있는데 일반적으로 무게 중심법이 가장 좋은 성능을 보인다.[3] COA 비퍼지화기를 하드웨어로 구성하는 경우 흔히 하드웨어의 복잡도를 줄이기 위해 출력 변수의 소속 함수 중심에서의 단일 출력값(Singleton)을 사용하여 비퍼지화를 행하는데, 이는 비퍼지화 출력값에 많은 오차를 유발하여 정확한 제어를 어렵게 만들 수 있다. 나아가 COA 방법은 비퍼지화 연산시 나눗셈기를 사용하는데, 이는 퍼지 제어기를 하드웨어로 구성할 경우 구현 비용과 연산 시간을 증가시키는 요인이 된다.

따라서, 기존의 COA 비퍼지화기가 갖는 부정확한 제어 결과를 개선하고자 출력 변수가 갖는 소속 함수 중심에서의 단일 출력값뿐 아니라 소속 함수가 갖는 면적도 동시에 고려하여 비퍼지화값을 연산할 것을 제안한다. 그러나 퍼지 제어기의 하드웨어 구현시 소속 함수가 갖는 면적을 직접 계산하려면 적분기가 요구되는데 이는 연산 시간을 매우 증가시키는 단점이 있다. 이를 해결하고자, 적분기를 사용할 경우 얻어지는 연산의 정확도에 비해 크게 뒤떨어지지 않으면서 비퍼지화 연산시 하드웨어 복잡도를 크게 증가시키지 않는 방안으로 소속 함수의 면적을 소속 함수가 갖는 폭으로 근사화시키는 방안을 제시한다.

나아가, 기존의 COA 비퍼지화기는 비퍼지화값 연산시 나눗셈을 수행해야 하는데 나눗셈기를 하드웨어로 구현하는 경우 구현 비용 및 연산 시간이 크게 문제가 된다. Ruitz 등은 COA 비퍼지화기의 비퍼지화값을 퍼지 제어기의 출력 변수상의 좌·우측 모멘트의 균형점으로 보고 이 균형점을 찾는 것으로 비퍼지화기를 대신함으로서 나눗셈 없이 비퍼지화 연산을 할 수 있음을 보였다. 그러나 이들이 제안한 모멘트 균형점 탐색은 매 탐색마다 출력 변수상에 매우 작은 크기의 동일 간격만큼 좌측 또는 우측으로 탐색점을 이동시킴으로서 탐색 시간이 많이 걸리는 단점이 있다. 본 논문은 이러한 단점을 극복하고자 Coarse-to-Fine 탐색 방법을 제안한다. 이 방법은 Coarse 탐색시 출력 변수상 탐색점의 이동을 퍼지항 단위로 수행하며, 인접하는 두 퍼지항에 도달한 다음 Ruitz 등이 사용한 Fine 탐색을 수행함으로서 탐색 시간을 크게 줄일 수 있다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 기존의 COA 비퍼지화 방법이 갖는 문제점을 고찰해보고, Ⅲ장에서는 이를 극복하고자 비퍼지화 출력 계산시 소속 함수 중심값과 함께 소속 함수의 면적을 고려하는 새

로운 비퍼지화기를 제안한다. 아울러 비퍼지화 연산을 Coarse-to-Fine 탐색법에 의해 모멘트 균형점을 효율적으로 탐색하는 것으로 대신하여 나눗셈기 없이 비퍼지화 연산을 할 수 있음을 보인다. IV장에서는 제안한 비퍼지화기의 타당성을 검증하고자 컴퓨터 시뮬레이션을 통해 모멘트 균형점 탐색 결과 얻어진 비퍼지화 출력값의 정확성과 소속함수 폭을 고려함으로서 얻어지는 제어 성능의 개선 정도를 보이고, 나아가 비퍼지화기의 모멘트 균형점 탐색에 걸리는 시간을 Ruitz[1]에 의한 균등 분할 탐색과 비교한다. 마지막으로, V장은 결론과 앞으로의 연구 방향을 언급한다.

Ⅱ. COA 비퍼지화기

1. COA 비퍼지화기의 동작 원리

COA 비퍼지화 방법은 비퍼지화값을 Max-Min 추론 엔진에 의해 얻어진 절단된(Clipped) 출력 변수가 갖는 소속 함수값의 무게 중심으로 얻는 방법으로 아래 식과 같이 나타내어진다.

$$y_c = \frac{\int_y y \cdot \mu_Y(y)\, dy}{\int_y \mu_Y(y)\, dy} \tag{1}$$

여기서 y는 추론된 소속함수의 지지값, $\mu_y(y_i)$는 주어진 지지값에서 추론된 소속함수의 최대값을 나타낸다.

COA 비퍼지화기에서 추론된 소속 함수를 최대 소속값에 대응하는 단일값(Singleton)으로 바꿔어도 어느 정도 만족할 만한 제어결과를 보이는데[4] 이 경우 식 (1)을 이산적으로 표현하면 아래 식과 같이 나타내어진다.

$$y_c = \frac{\sum_{i=1}^{n} y_i \cdot \mu_Y(y_i)}{\sum_{i=1}^{n} \mu_Y(y_i)} \tag{2}$$

여기서 n은 이산 퍼지항의 갯수, y_i는 i번째 퍼지 항의 단일 지지값, $\mu_Y(y_i)$는 단일 지지값 y_i에서의 소속함수값을 나타낸다.

2. 기존 COA 비퍼지화 방법의 문제점

식 (1)과 식 (2)는 각 퍼지 항의 소속함수가 대칭이고, 같은 폭을 가지고, 서로 같은 간격으로 배치될 때에만 동일한 비퍼지화값을 나타낸다. 그런데 아래 그림 2에서와 같이 위의 조건이 만족하지 않는 경우 식 (1)과 식 (2) 사이에 큰 오차가 초래된다. 아래 그림 2에서 a) 와 b)의 경우 서로 동일한 두 개의 단일 지지값을 갖지만, 소속함수들의 폭을 보면 a)의 경우는 서로 같고, b)의 경우는 서로 다름을 알 수 있다. 기존의 COA 비퍼지화기는 단일 지지값의 소속값만을 고려하기 때문에 a)와 b)는 서로 같은 비퍼지화값을 나타낸다($y_c^a = y_c^b$). 그러나 이 결과는 면적이 더 넓은 곳 쪽으로 무게 중심이 이동해야 하는 사실과는 거리가 멀다.

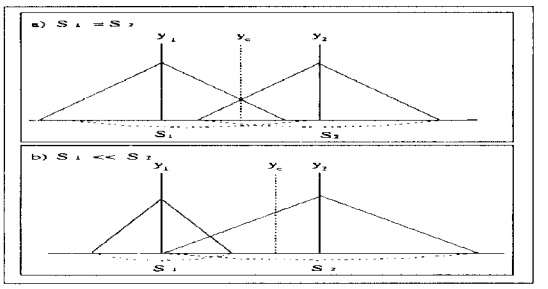

그림 2. 기존 COA 비퍼지화기의 오류의 예시
Fig. 2. An illustration of misusing the conventional COA defuzzifier.

Ⅲ. 새로 제안한 COA 비퍼지화기

1. 소속 함수 폭을 고려

식 (1)의 분자와 분모는 y의 모든 출력 범위에서의 $y \cdot \mu_Y(y)$와 $\mu_Y(y)$의 영역을 나타내는데 비해 식 (2)는 특정한 소속값이나 단일 지지값만을 고려하기 때문에 식 (1)과 식 (2)로부터 얻어지는 비퍼지화값 사이에는 뚜렷한 차이가 나타난다. 이를 해결하기 위해, 본 논문은 비퍼지화값을 다음과 같이 계산할 것을 제안한다.

$$y_c = \frac{\sum_{i=1}^{n} A_Y(y_i) \cdot y_i}{\sum_{i=1}^{n} A_Y(y_i)} \tag{3}$$

여기서 $A_Y(y)$는 i번째 추론된 소속함수의 절단된 소속 함수값의 아래 영역, n은 퍼지항의 갯수, y_i는 i번째 퍼지항의 단일 지지값을 나타낸다.

본 논문에서는 삼각형 모양의 소속 함수 $\mu_Y(y)$를 일반화시킨 일반화된 삼각형 소속 함수 $\mu_Y^n(y)$를 고려하였다[5]. 여기서 n은 소속 함수의 모양을 결정하는 모양

상수로 n = 1일 경우는 보통의 삼각형 모양, n < 1일 경우는 그림 3의 모양을 갖는 팽창된 (Dilated) 삼각형, n > 1의 경우는 압축된 (Concentrated) 삼각형 모양을 갖는다. 여기서는 n < 1 경우에 대해서만 설명하기로 한다. 그림 3에서 절단된 $\mu_Y^n(y)$의 아래 영역 $A_Y(y_i)$는 직사각형 면적 A_r과 소속함수 아래의 양 Tail 부분의 면적 A_i의 합으로 다음과 같은 식에 의해 얻어진다. (식 (4)의 증명은 부록 1에 나타나 있다.)

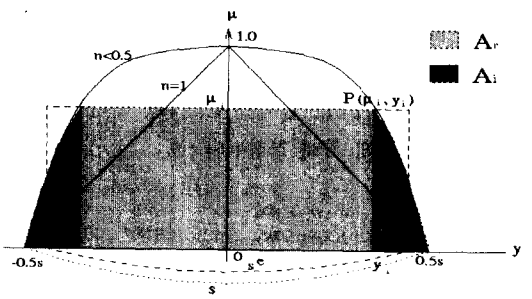

그림 3. 일반화된 소속함수 $\mu_Y^n(y_i)$

Fig. 3. A generalized membership function $\mu_Y^n(y_i)$.

$$A_Y(y_i) = \mu_Y(y_i)\cdot s_i - \frac{n}{n+1}\mu_Y^{\frac{n+1}{n}}(y_i)\cdot s_i = \mu_Y(y_i)\cdot s_i\cdot\left(1-\frac{n}{n+1}\mu_Y^{\frac{1}{n}}(y_i)\right) \tag{4}$$

여기서, n, $\mu_Y(y_i)$와 s_i는 각각 모양 상수, i번째 퍼지항이 갖는 소속함수와 소속함수가 갖는 폭(Span)을 의미한다. 위식으로부터, $A_Y(y_i)$ 는 $n\to 0$ 이면 $\mu_Y(y_i)\cdot s_i$로 접근하고, $n\to\infty$이면 0으로 접근함을 알 수 있다. 식 (4)의 $A_Y(y_i)$는 s_i^e를 $s_i\cdot\left(1-\frac{n}{n+1}\mu_Y^{\frac{1}{n}}(y_i)\right)$라고 정의하면, 아래식과 같이 높이가 $\mu_Y(y_i)$이고 폭이 s_i^e인 사각형의 면적으로 볼 수 있다.

$$A_Y^e(y_i) = \mu_Y(y_i)\cdot s_i^e \tag{5}$$

이 경우에 비퍼지화값 y_c^e는 아래와 같이 구해진다.

$$y_c^e = \frac{\sum_{i=0}^{n} A_Y^e(y_i)\cdot y_i}{\sum_{i=0}^{n} A_Y^e(y_i)} = \frac{\sum_{i=1}^{n}\mu_Y(y_i)\cdot s_i^e\cdot y_i}{\sum_{i=1}^{n}\mu_Y(y_i)\cdot s_i^e} = \frac{\sum_{i=0}^{n}\mu_Y(y_i)\cdot s_i\cdot\left(1-\frac{n}{n+1}\cdot\mu_Y^{\frac{1}{n}}(y_i)\right)\cdot y_i}{\sum_{i=0}^{n}\mu_Y(y_i)\cdot s_i\cdot\left(1-\frac{n}{n+1}\cdot\mu_Y^{\frac{1}{n}}(y_i)\right)} \tag{6}$$

소속 함수값 $\mu_Y(y_i)$가 [0, 1] 사이에 균일하게 분포되어 균일 분포 함수 $U(0,1)$를 갖는다고 가정하면 $\mu_Y^{\frac{1}{n}}(y_i)$의 기대값 $E[\mu_Y^{\frac{1}{n}}(y_i)]$ 은 $\frac{n}{n+1}$이 되고, $n\le 2$ 일 때 $\frac{n}{n+1}\cdot E[\mu_Y^{\frac{1}{n}}(y_i)] = (\frac{n}{n+1})^2 \ll 1$이므로 식 (6)에 있는 두번째 항 $\frac{n}{n+1}\cdot\mu_Y^{\frac{1}{n}}(y_i)$은 무시할 수 있다. 이 경우 앞서의 사각형 면적은 근사적으로 $A_Y^a(y_i) = \mu_Y(y_i)\cdot s_i^a = \mu_Y(y_i)\cdot s_i$ 이 된다. 따라서 근사적 비퍼지화값 y_c^a는 다음과 같이 구해진다.

$$y_c^a = \frac{\sum_{i=0}^{n} A_Y^a(y_i)\cdot y_i}{\sum_{i=0}^{n} A_Y^a(y_i)} = \frac{\sum_{i=0}^{n}\mu_Y(y_i)\cdot s_i^a\cdot y_i}{\sum_{i=0}^{n}\mu_Y(y_i)\cdot s_i^a} = \frac{\sum_{i=0}^{n}\mu_Y(y_i)\cdot s_i\cdot y_i}{\sum_{i=0}^{n}\mu_Y(y_i)\cdot s_i} \tag{7}$$

기존의 COA 비퍼지화기와 비교해보면, 근사적 비퍼지화값 y_c^a를 계산하기 위해서는 부가적인 곱셈기를 요구한다. 그러나, 뒷장의 시뮬레이션 결과에서 볼 수 있듯이, 이러한 추가적인 하드웨어의 요구는 제어 성능의 향상에 의해서 보상되어진다.

2. 모멘트 균형점의 효율적 탐색

제안한 COA 비퍼지화기는 비퍼지화값을 계산하기 위하여 소속값과 소속함수 폭의 값을 동시에 곱셈 처리해야 하므로 추가적인 곱셈기를 요구한다. 본 논문에서 직접 언급하지는 않았지만 곱셈기는 Stochastic 컴퓨팅[6]에 기초한 간단한 AND 연산으로 대치함으로써 추가적인 곱셈기 요구에 따른 하드웨어 복잡도 증가 문제를 해결할 수 있다. 나아가 비퍼지화값 계산은 나눗셈을 요구하는데 이는 하드웨어 복잡도를 높이는 또 다른 요인이기 때문에 본 논문에서는 나눗셈을 하지 않고서도 모멘트 균형점을 효과적으로 찾음으로써 비퍼지화값을 얻는 방안을 제안한다.

먼저, 출력 변수의 양끝에서 부터 마주 보는 방향으로 출발하여 $y=y_c$까지의 왼쪽과 오른쪽의 모멘트 M_l과 M_r는 각각 아래식과 같이 정의된다.

$$M_l = \sum_{i=1}^{c}\mu_Y(y_i)\cdot s_i\cdot y_i \qquad M_r = \sum_{i=c}^{M}\mu_Y(y_i)\cdot s_i\cdot y_i \tag{8}$$

여기서 $\mu_Y(y_i), s_i, y_i$와 M은 각각 i번째 소속 함수의 절단된 소속값, i번째 소속 함수의 폭, i번째 소속 함수의 단일 지지값과 출력 변수 y가 갖는 퍼지항의 갯수를 나타낸다. 위의 왼쪽 또는 오른쪽 모멘트 M_l과 M_r는 아래식과 같은 반복식에 의해 쉽게 계산된다. (식 (9)의 증명은 부록 2에 나타나 있다.)

$$\begin{aligned} M_l(n) &= M_l(n-1)+A_l(n-1)\cdot(y_n-y_{n-1}) \\ A_l(n) &= A_l(n-1)+\mu_n\cdot s_n \end{aligned} \tag{9}$$

여기서 μ_n, s_n 과 y_n은 각각 n번째 반복에서 사용되는 소속함수의 절단된 소속값, 소속 함수의 폭, 그리고 소속함수의 단일 지지값을 나타내며, $A_l(n)$과 $M_l(n)$은 각각 n번째 반복에서 얻어지는 왼쪽 면적과 왼쪽 모멘트를 나타낸다. 비슷하게 오른쪽 면적 $A_r(n)$과 오른쪽 모멘트 $M_r(n)$의 반복식은 아래와 같이 표현된다.

$$\begin{aligned} M_r(n) &= M_r(n-1)+A_r(n-1)\cdot(y_{M-n}-y_{M-n-1}) \\ A_r(n) &= A_r(n-1)+\mu_{M-n-1}\cdot s_{M-n-1} \end{aligned} \tag{10}$$

여기서 M은 출력 변수가 갖는 퍼지항의 갯수이다.

Ruitz[11] 등은 m_l과 m_r을 계산하는데 출력 변수가 가지는 전체 정의역을 256등분으로 균등 분할 하여 탐색을 수행하므로 탐색을 수행하는데 걸리는 반복 횟수가 최소 128에서 최대 256회를 수행해야 하는 문제점을 갖고 있다. 이 문제를 해결하기 위해 본 논문에서는 Coarse-to-Fine 탐색 방법을 제안한다. (여기서 대문자로 나타낸 양들은 Coarse 탐색과 관련된 것이고, 소문자로 나타낸 양들은 Fine 탐색과 관련된 것이다.)

Coarse-to-Fine 탐색 방법은 다음과 같이 수행된다. 탐색을 수행하기에 앞서 A_l과 A_r 그리고 M_l과 M_r을 0으로, $I_l=0$, $I_r=M+1$ (단, M은 퍼지항의 갯수)으로 초기화 시킨다. Coarse 탐색은 앞의 식 (9)과 (10)을 이용하여 M_l과 M_r을 구한 뒤 이들을 서로 비교하여 작은 값을 갖는 쪽이 한 퍼지항 단위로 이동하는데 (즉, M_l이 작으면 $I_l=I_l+1$이고 M_r이 작으면 $I_r=I_r-1$) 이 과정을 $|I_r-I_l|\le 1$이 될 때까지 반복한다. $M_l=M_r$인 경우에는 양쪽 모두 이동(즉, $I_l=I_l+1$ 과 $I_r=I_r-1$) 한다.

Coarse 탐색의 종료 조건을 만족하는 순간의 I_l과 I_r을 각각 I_l^c, I_r^c라고 하면, 왼쪽 및 오른쪽 모멘트의 값은 각각 $M_l(I_l^c)$과 $M_r(I_r^c)$가 된다. 이들 값을 Fine 탐색을 위한 초기값으로 사용하기 위해 $i_l=I_l^c$, $i_r=I_r^c$ 그리고 $m_l(i_l)=M_l(I_l^c)$, $m_r(i_r)=M_r(I_r^c)$, $a_l=A_l(I_l^c)$, $a_r=A_r(I_r^c)$로 둔다. Fine 탐색 과정에서 왼쪽 및 오른쪽 모멘트의 반복식은 아래와 같다.

$$\begin{aligned} m_l(n+1) &= m_l(n)+a_l \quad n=i_l, i_l+1,\cdots \\ m_r(n+1) &= m_r(n)+a_r \quad n=i_r, i_r-1,\cdots \end{aligned} \tag{11}$$

위 식을 이용하여 계산된 m_l과 m_r을 서로 비교하여 작은 값의 모멘트를 갖는 쪽을 한 단위만큼 (여기서 한 단위는 $\frac{\text{출력 변수 범위}}{256}$ 의 크기를 갖는다.) 이동하는데, (즉, m_l이 작으면 $i_l=i_l+1$이고 m_r이 작으면 $i_r=i_r-1$) 이 과정을 $|i_R-i_l|\le 1$이 될 때까지 반복한다. $m_l=m_r$인 경우에는 양쪽 모두 이동 (즉, $i_l=i_l+1$ 과 $i_r=i_r-1$) 한다. Fine 탐색이 끝난 뒤, 비퍼지화값 y_c는 다음 식 (12)에 의해 결정된다.

$$y_c=\begin{cases} i_l & \text{if } m_l\ge m_r \\ i_r & \text{if } m_l< m_r \end{cases} \tag{12}$$

아래 그림 4는 위에서 설명한 Coarse-to-Fine 탐색 알고리즘을 하드웨어 기술언어인 VHDL[7]을 사용하여 나타낸 것이다.

```
Library          IEEE;
Use              IEEE.Std_logic_1164.All;
Use              Work.FLC_Type.All;

ENTITY  Defuzzy IS
PORT(M: IN       Std_Word_Vector;
     Y:OUT       Std_Word);
END Defuzzy;

ARCHITECTURE Behaviour OF Defuzzy IS
BEGIN
  PROCESS
        Variable   ml, mr: Integer;
        Variable   Al, Ar: Integer;
        Variable   Il, Ir: Integer;
        Variable   Jl, Jr: Integer;
  BEGIN
        l := 0;    mr :- 0;
        Il := 0;   Ir := M'Length + 1;
        Al := 0;   Ar :- 0;
-- Coarse Searching
        WHILE ( |Ir-Il| > 1)          LOOP
          IF (ml < mr) THEN
                Il := Il + 1;
                ml := ml +
                    *s(Al, (yc(Il)-yc(Il-1)), 255);
                Al := Al + *s(s(Il), M(Il), 255);
--*s는 stochastic 곱셈을 의미
          END IF;
```

```
            IF (ml > mr) THEN
                Ir := Ir - 1;
                mr := mr +
                  *s(Ar, (yc(Ir+1)-yc(Ir)), 255);
                Ar := Ar + *s(s(Ir), M(Ir), 255);
            END IF;
            IF (ml = mr) THEN
              IF (al <= ar) THEN
                Il := Il + 1;
              ml := ml +
                  *s(Al, (yc(Il)-yc(Il-1)), 255);
                Al := Al + *s(s(Il), M(Il), 255);
              END IF;
              IF (al >= ar) THEN
                Ir := Ir - 1;
                mr := mr -
                  *s(Ar, (yc(Ir+1)-yc(Ir)), 255);
                Ar :=Ar + *s(s(Ir), M(Ir), 255);
              END IF;
            END IF;
          END LOOP;
-- Fine Searching
          Jl := yc(Il);   Jr := yc(Ir);
          WHILE ( | Jr-Jl | > 1)          LOOP
            IF (ml ≤ mr) THEN
              Jl := Jl + 1;
              ml := ml + Al;
            END IF;
            IF (ml ≥ mr) THEN
              Jr := Jr - 1;
              mr := mr + Ar;
            END IF;
          END LOOP;
-- 비퍼지화값 결정
        IF (ml ≥ mr) THEN
            Y <= Jl;
          ELSE
            Y <= Jr;
          END IF;
    END PROCESS;
End Behaviour;
```

그림 4. Coarse-to-Fine 탐색 알고리즘의 VHDL 코드

Fig. 4. A VHDL code for coarse-to-fine searching algorithm.

Ⅳ. 실험 결과 및 분석

제안한 비퍼지화기를 트럭 후진 주차 문제에 적용하여 제어 성능을 기존의 비퍼지화기와 비교하였다. 트럭 주차 문제의 목표는 가능한 빨리, 그리고 정확하게 트럭을 주차시키는 것이며, 이 문제는 기존 제어 기술로는 풀기 힘든 전형적인 비선형 제어 문제이다. 그림 5는 트럭 주차 제어 문제에서 사용된 트럭과 주차대의 위치를 보여준다. 트럭의 위치는 (x, y, ϕ)에 의해 결정된다. 단, 여기에서 ϕ는 트럭 진행 방향과 x축간의 각도이며, 트럭의 후진 주행 제어는 ϕ와 핸들의 축 간의 각도인 θ에 의해 결정된다. 트럭이 움직이는 운동 방정식은 아래와 같이 나타내어진다.[8]

$$\begin{aligned} x(t+1) &= x(t) + \cos[\phi(t) + \theta(t)] \\ &\quad + \sin[\theta(t) \cdot \sin[\phi(t)] \\ y(t+1) &= y(t) + \sin[\phi(t) + \theta(t)] \\ &\quad - \sin[\theta(t) \cdot \sin[\phi(t)] \\ \phi(t+1) &= \phi(t) - \sin^{-1}[2\sin(\frac{\theta(t)}{b})] \end{aligned} \tag{13}$$

여기서 b는 트럭의 길이이며, 본 논문에서는 b = 4로 하였다.

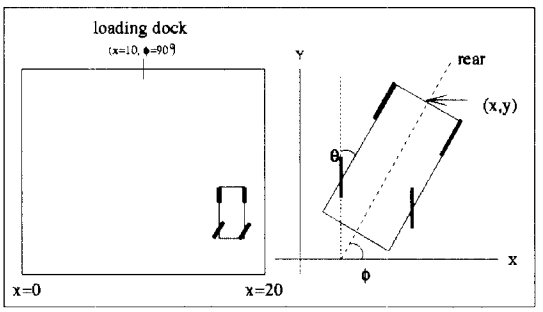

그림 5. 모형 트럭과 주차대 위치

Fig. 5. A model truck and loading dock.

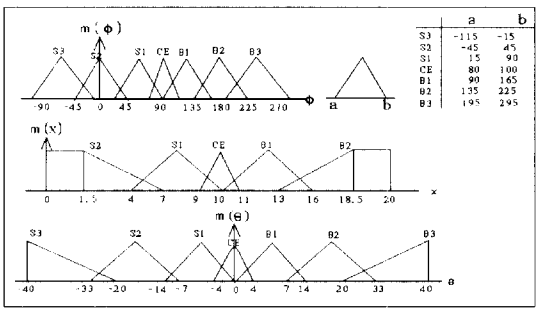

그림 6. 트럭 후진 주차 문제에 사용된 소속 함수

Fig. 6. Membership functions for the truck backer-upper control problem.

만약 트럭과 주차대까지의 거리가 충분하다면 트럭이 $x=10$, $\phi=90^{\circ}$ 가까이 오면 트럭을 곧장 후진하기만 하면 되기 때문에 변수 y를 퍼지 입력 변수 (x, y, ϕ)에서 뺄 수 있다. 그러므로 트럭 주차 제어 문제는 주어진 공간내 $\{0 \le x \le 20,\ -90^{\circ} \le \phi \le 270^{\circ}\}$ 임의의 초기 위치 (x_0, ϕ_0)에서 가능하면 신속·정확하게 주차

대 ($x=10$, $\phi=90°$) 쪽으로 후진하도록 바퀴 각도 $\theta(-40\le\theta\le40)$ 를 제어하는 것이 요구된다. 그림 6과 7은 각각 Wang과 Mendel[9]이 사용한 퍼지 제어기의 입·출력 변수의 소속함수와 퍼지 제어를 위한 퍼지 규칙 베이스를 나타낸 것이다.

φ \ x	S2	S1	CE	B1	B2
S3	S2	S3			
S2	S2	S3	S3	S3	
S1	B1	S1	S2	S3	S2
CE	B2	B2	CE	S2	S2
B1	B2	B3	B2	B1	S1
B2		B3	B3	B3	B2
B3				B3	B2

그림 7. 트럭 후진 주차 문제에 사용된 퍼지 규칙 베이스

Fig. 7. Fuzzy rule base for the truck backer-upper control problem.

표 1은 서로 다른 비퍼지화기를 사용하는 퍼지 제어기의 제어 성능을 비교한 것으로 임의로 발생시킨 1,000개의 초기 위치에 대한 주행 거리의 평균을 사용하여 비교하였다. 여기서 n은 소속함수의 모양을 나타내는 변수이고, 기존 COA, Effective COA 및 근사 COA는 각각 식 (2), (6), 그리고 (7)을 이용하여 비퍼지화값을 얻는다. 이 표로부터 Effective COA 비퍼지화기를 사용하는 퍼지 제어기는 기존의 COA 비퍼지화기를 사용하는 퍼지 제어기보다 평균 주행거리가 4.37 ~ 4.79 step 작아졌으며, 소속 함수 모양이 Concentration 될수록 (즉 n이 커질수록) 성능 개선 효과가 더욱 증가됨을 알 수 있다. Effective COA 비퍼지화기를 근사화시킨 근사 COA 비퍼지화기를 사용하는 퍼지 제어기는 평균 주행거리가 기존 COA 비퍼지화기를 사용하는 퍼지 제어기보다 4.07 step 정도 작아졌는데 이는 성능 향상 정도가 Effective COA 비퍼지화기를 사용하는 퍼지제어기에 비해 크게 떨어지지 않음을 보여준다. 나아가, 근사 COA 비퍼지화기는 하드웨어 복잡도가 Effective COA 비퍼지화기에 비해 훨씬 간단하므로 성능 향상 정도 면에서 뿐만 아니라 하드웨어 복잡도 면에서 다른 COA 비퍼지화기보다 훨씬 유리함을 알 수 있다.

아래 그림 8은 임의의 두점 (왼쪽 그림 시작점 : (19.1, 0.0, 206°), 오른쪽 그림 시작점 (2.3, 0.0, -34.6°))으로부터 트럭의 주행 과정을 나타낸 것이다.

표 1. 여러 가지 COA 비퍼지화기를 사용한 퍼지제어기의 평균 주행거리 비교

Table 1. Comparison of average tracing distance of various COA defuzzifier.

Shape Constant	기존 COA	Effective COA	근사 COA
n = 0.5	25.48	21.11	21.41
n = 1.0	25.48	20.96	21.41
n = 2.0	25.48	20.69	21.41

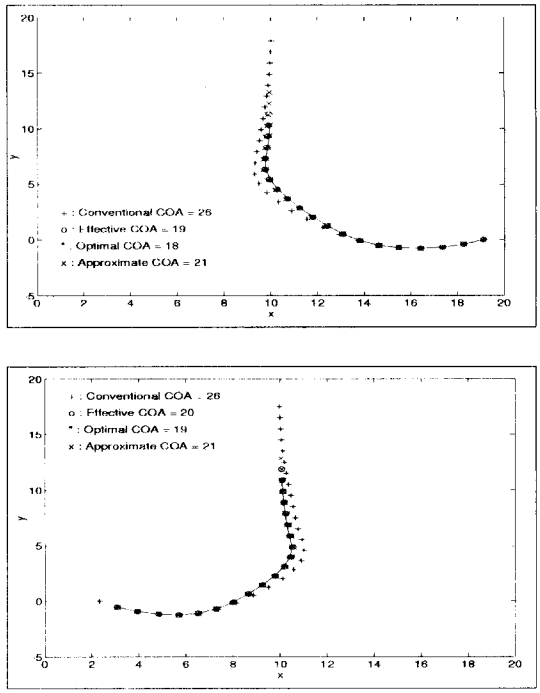

그림 8. 임의의 두점 {(19.1,0.0,206o)과 (2.3, 0.0, -34.6o)}으로부터 트럭의 주행곡선

Fig. 8. Trajectories of a model car fromtwo random positions. {(19.1,0.0,206o) and (2.3, 0.0, -34.6o)}

왼쪽 그림의 경우, 기존의 COA 비퍼지화기를 사용한 퍼지제어기의 경우는 주차대에 도달하는데 26 step이 걸렸는 데 반해, Effective COA 및 근사 비퍼지화기를 사용한 퍼지제어기의 경우는 각각 19 및 21 step이 걸렸음을 보인다. 오른쪽 그림의 경우, 기존의 COA 비퍼지화기를 사용한 퍼지제어기의 경우는 주차대에 도달하는데 26 step이 걸렸는 데 반해, Effective COA 및 근사 비퍼지화기를 사용한 퍼지제어기의 경우

는 각각 20 및 21 step이 걸렸음을 보인다. 이들 그림들로 부터, 주행 거리가 짧아지는 것은 주로 진행 방향이 급격히 변화하는 근처에서 주행 각도 θ의 정확한 제어에 의해 얻어짐을 알 수 있다. 즉 그림 6의 θ에 대한 소속함수를 보면 θ값이 양쪽으로 커질수록 소속함수의 폭이 커지는 데 기존의 COA 비퍼지화기는 비퍼지화값 계산시 이를 효과적으로 반영하지 못하므로 진행 방향이 급격히 변화하는 근처에서(즉 θ값이 큰 영역) 정확한 제어를 하지 못하여 전체적으로 주행거리가 증가시키는 결과를 초래한다.

다음 표 2는 제안한 Coarse-to-Fine 탐색법과 Ruitz[1]등이 제안한 균등 간격 탐색법을 탐색에 걸리는 탐색 반복 횟수면에서 비교한 것이다. 이 표로 부터 $M < 16$인 범위 내에서 대략 $O(M)$배의 속도 향상을 얻을 수 있으며, M이 증가할 수록 속도 개선 효과가 증가함을 알 수 있다.

표 2. 두 알고리즘의 연산 반복 횟수의 비교
Table 2. Comparison of the number of searchings between two algorithms.

	최 소	최 대
Ruitz 알고리즘	256/2	256
제안한 알고리즘	M/2+256/2M	M+256/M
Coarse 탐색	M/2	M
Fine 탐색	256/2M	256/M

제안한 Coarse-to-Fine 탐색 알고리즘이 어느 정도 정확하게 연산을 수행하는지 알아보기 위해 임의의 퍼지 출력에 대해, 한편으로는 컴퓨터 내에서 제공하는 곱셈과 나눗셈을 이용하여 얻어진 정확한 비퍼지화값 ($y_c(i)$)을 구하고, 다른 한편으로는 전체 범위를 256등분으로 양자화 시킨 출력 변수에 대해 Coarse-to-Fine 탐색법에 의해 비퍼지화값 ($y_c^m(i)$)을 구한 뒤 이들 사이의 오차 $\varepsilon(i) = |y_c(i) - y_c^m(i)|$ 를 계산해 보았다. 이상의 과정을 임의의 서로 다른 1,000개의 퍼지 출력에 대해 반복 수행하여 얻은 오차의 평균 ($E_{avg} = \frac{1}{1000}\sum_{i=1}^{1000}\varepsilon(i)$)이 약 0.87 등분(256등분의 0.34 %) 정도였다. 나아가, Coarse-to-Fine 탐색시 요구하는 곱셈을 stochastic 컴퓨팅에 의한 AND 연산으로 대치한 경우, 얻어진 비퍼지화값을 $y_c^s(i)$라고 할 때 앞서 구한 정확한 비퍼지화값 ($y_c(i)$)과의 오차의 평균 ($E_{avg} = \frac{1}{1000}\sum_{i=1}^{1000}|y_c(i) - y_c^s(i)|$)을 구해보니 약 7.45 등분(256등분의 2.9 %) 정도였다. 이 결과로부터 제안한 Coarse-to-Fine 탐색법의 단점인 추가적 곱셈기를 Stochastic 컴퓨팅에 기초한 AND 게이트로 대신하여 구현하더라도 전체 출력 변수를 256등분 했을 때 얻어진 비퍼지화값과 실제 계산치와의 오차를 3 %이내로 줄일 수 있음을 의미한다.

V. 결 론

본 논문은 기존의 COA 비퍼지화기가 비퍼지화값 연산시 소속 함수의 폭이 다름에도 불구하고 같은 비퍼지화값을 나타내는 잘못을 바로잡기 위해, 비퍼지화값을 계산하는데 있어 소속 함수의 중심 출력값과 소속함수의 폭을 동시에 고려하였다. 나아가, 비퍼지화값 연산시 나눗셈을 필요로 하는데, 나눗셈기를 사용하지 않고서 모멘트 균형점을 찾음으로써 비퍼지화값을 대신할 수 있음을 제안하였다. 효과적인 모멘트 균형점 탐색을 위해 Coarse-to-Fine 탐색법을 제시하였는데 Coarse 탐색시 모멘트 계산점의 이동은 퍼지항 단위로 이동시킴으로써 전체적으로 탐색 시간을 크게 줄일 수 있었다.

본 논문의 가장 중요한 결과는 비퍼지화값 연산시 소속함수의 폭을 고려함으로서 퍼지 제어기의 제어 성능을 개선시켰다는 점과 (예를 들면, Truck 후진 제어 문제의 경우 목적지에 도달하는 데 걸리는 평균 주행 거리를 20% 이상 줄일 수 있었다.) 모멘트 균형점의 탐색시 Coarse-to-Fine 탐색 방법을 사용함으로써 기존의 균일 간격 탐색법과 비교해 볼 때 최대 O(M)배의 속도 개선을 할 수 있었다는 점이다.

현재 제안한 COA 비퍼지화기를 갖는 퍼지 제어기를 SYNOPSYS사의 FPGA 합성 장비와 XILINX사의 XACT5.0을 사용하여 FPGA상에 구현하여 제안한 COA 비퍼지화기의 제어 성능 향상과 제산기를 사용하지 않고서도 COA 비퍼지화기가 구현이 가능함을 확인하고 궁극적으로 이를 자동차의 주행 제어에 직접 적용하여 볼 계획이다.

참 고 문 헌

[1] A. Ruitz, J. Gutiérrez, and J. Fernández, "A

Fuzzy Controller with an Optimized Defuzzification Algorithm," *IEEE Micro*, pp. 1-10, Dec. 1995.

[2] E. H. Mandani, "Application of fuzzy algorithms for control of simple dynamic plant," *IEEE Proc. Control & Science*, Vol. 121, No. 12, pp. 1585-1588, Dec. 1974.

[3] C. C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, No. 2, pp. 404-435, Feb. 1990.

[4] T. Miki, H. Matsumoto, K. Ohto, and T. Yamakama, "Silicon Implementation for a Novel High Speed Inference Engine: Mega-FLIPS Analog Fuzzy Processor," *Journal of Intelligence and Fuzzy System*, vol. 1, no. 1, pp. 27-42, 1993.

[5] T. A. Runkler and M. Glesner, "A set of Axioms for Defuzzification Strategies Towards a Theory of Rational Defuzzification Operators," *Second IEEE International Conference on Fuzzy Systems*, vol. 2, pp. 1161-1166, 1993.

[6] Y. Kondo and Y. Sawada, "Functional Abilities of a Stochastic Logic Neural Network," *IEEE Transactions on Nerual Networks*, vol. 3, No. 3, pp. 434-443, May 1992.

[7] S. Mazor and P. Langstraat, A Guide to VHDL, Kluwer Academic Publishers, 1993.

[8] Li-Xin Wang And Jerry M. Mendel, "Generating Fuzzy Rules from Numerical Data with Applications," *USC-SIPI Report*, no. 169, 1991.

[9] Li-Xin Wang and Jerry M. Mendel, "Generating Fuzzy Rules by Learning from Examples," *IEEE Transactions on System, Man, and Cybernetics*, vol. 22, no. 6, pp. 1414-1427, Nov. 1992.

부록 1 : 식 (4)의 유도

그림 3에서, 절단된 소속함수 아래의 영역 $A_Y(y_i)$은 사각형 영역 A_r과 적분 영역 A_i의 합이다. 그리고 소속 함수가 갖는 포물선 방정식 $\mu_Y(y_i)$는 $\mu_Y(y_i)=m\cdot(y_i-\frac{s_i}{2})^n$로 나타내어진다. 상수 m은 조건 $\mu_Y(0)=1$을 이용하면 $m\cdot(0-\frac{s_i}{2})^n=1$로 부터 $m=(-\frac{s_i}{2})^{-n}$이다. 따라서, 포물선 방정식 $\mu_Y(y_i)$는 $\mu_Y(y_i)=(-\frac{s_i}{2})^{-n}\cdot(y_i-\frac{s_i}{2})^n=(1-\frac{2y_i}{s})^n$로 쓸 수 있다. 교차점 $P(y_i, u_i)$에서 $\mu_Y(y_i)=\mu_i$이므로, $(1-\frac{2y_i}{s_i})^n=\mu_i$로 부터 y_i가 $\frac{s_i}{2}\cdot(1-\mu_Y(y_i)^{\frac{1}{n}})$임을 알 수 있다.

사각형 영역 A_r는

$$\begin{aligned}A_r &= 2\cdot\mu_i\cdot y_i\\ &= 2\cdot\mu_Y(y_i)\cdot y_i\\ &= s_i\cdot(1-\mu_Y(y_i)^{\frac{1}{n}})\cdot\mu_Y(y_i)\end{aligned}$$

이 되고, 적분 영역 A_i는

$$\begin{aligned}A_i &= 2\cdot\int_{y_i}^{\frac{s_i}{2}}(1-\frac{2y}{s_i})^n dy\\ &= 2[-\frac{si}{2}\cdot\frac{1}{n+1}\cdot(1-\frac{2y}{s_i})^{n+1}]_{y_i}^{\frac{s_i}{2}}\\ &= \frac{s_i}{n+1}\mu_Y^{\frac{n+1}{n}}(y_i)\end{aligned}$$

이 된다. 그러므로 전체 영역 $A_Y(y_i)$는

$$\begin{aligned}A_Y(y_i) &= A_r+A_i\\ &= s_i\cdot(1-\mu_Y^{\frac{1}{n}}(y_i))\cdot\mu_Y(y_i)\\ &+\frac{s_i}{n+1}\cdot\mu_Y^{\frac{n+1}{n}}(y_i)\\ &= s_i\cdot\mu_Y(y_i)-s_i\cdot\frac{n}{n+1}\cdot\mu_Y^{\frac{n+1}{n}}(y_i)\\ &= s_i\cdot\mu_Y(y_i)\cdot(1-\frac{n}{n+1}\cdot\mu_Y^{\frac{1}{n}}(y_i))\end{aligned}$$

이 된다.

$n>1$ 인 경우에도 같은 방법으로 증명이 된다. 이로써 식 (4)의 증명을 끝낸다.

부록 2 : 식 (8)의 유도

먼저 n=1일 때 (즉 제일 왼쪽에 있는 퍼지항에 대해) 왼쪽 모멘트는 0이고 면적은 소속 함수값과 소속함수 폭의 곱으로 나타내어지므로 다음 관계식을 얻는다.

$$\begin{aligned}m_l(1) &= 0\\ a_l(1) &= \mu_1\cdot s_1\end{aligned}$$

n=2일 때 왼쪽 모멘트는 n=1일 때 면적에다가 두 퍼지항 사이의 거리 y_2-y_1를 곱한 것과 같고 면적은 그 때까지의 모든 절단된 소속함수의 합으로 정의되므로 $\mu_1\cdot s_1+\mu_2\cdot s_2$이다.

$$m_f(2) = \mu_1 \cdot s_1 \cdot (y_2 - y_1) = m_f(1) + a_f(1) \cdot (y_2 - y_1)$$
$$a_f(2) = \mu_1 \cdot s_1 + \mu_2 \cdot s_2 = a_f(1) + \mu_2 \cdot s_2$$

n=3일 때 왼쪽 모멘트는 n=1일 때 면적에다가 두 퍼지항 사이의 거리 $y_3 - y_1$를 곱한 것과 n=2일 때 면적에다가 두 퍼지항 사이의 거리 $y_3 - y_2$를 곱한 것의 합과 같고 면적은 그때까지의 모든 절단된 소속함수의 합으로 정의되므로 $\mu_1 \cdot s_1 + \mu_2 \cdot s_2 + \mu_3 \cdot s_3$이다.

$$\begin{aligned} m_f(3) &= \mu_1 \cdot s_1 \cdot (y_3 - y_1) + \mu_2 \cdot s_2 \cdot (y_3 - y_2) \\ &= \mu_1 \cdot s_1 \cdot (y_3 - y_2) + \mu_1 \cdot s_1 \cdot (y_2 - y_1) + \mu_2 \cdot s_2 (y_3 - y_2) \\ &= \mu_1 \cdot s_1 \cdot (y_2 - y_1) + (\mu_1 \cdot s_1 + \mu_2 \cdot s_2) \cdot (y_3 - y_2) \\ &= m_f(2) + a_f(2) \cdot (y_3 - y_2) \end{aligned}$$

이를 일반화 시키면 다음과 같은 반복식을 얻는다.

$$m_f(n) = m_f(n-1) + a_f(n-1) \cdot (y_n - y_{n-1})$$
$$a_f(n) = a_f(n-1) + \mu_n \cdot s_n$$

이로써 증명을 끝낸다.

저 자 소 개

金 大 鎭(正會員) 第 33卷 B編 第 5號 參照
현재 동아대학교 컴퓨터공학과 조교수

趙 仁 顯(正會員)
1996년 2월 동아대학교 컴퓨터공학과(공학사), 현재동아대학교 컴퓨터공학과 석사과정, 주관심분야는 소프트웨어 공학, 소프트 컴퓨팅, VLSI/ASIC 설계등