퍼지 논리를 이용한 영어 소문자 오프라인 인식

박태환 · 한수환 · 우영운
동의대학교 멀티미디어공학과

Off-Line Recognition of English Lowercase Characters Using Fuzzy Logic

Tae-Hwan Park · Young Woon Woo · Soowhan Han
Dept. of Multimedia Eng., Dong-Eui University

E-mail : ywwoo@deu.ac.kr

요 약

영문자를 오프라인으로 인식하기 위한 다양한 기법들이 기존에 많이 제안되어 왔다. 그러나 이 논문에서는 기존의 방법들과는 달리 처리 속도에 장점을 갖는 형태학적인 특징 정보와 퍼지 함수만을 이용한 인식 기법을 제안하였다. 제안한 기법에서는 형태학적인 분석을 통해 미리 정의된 특징 정보들과, 입력되는 인식 대상 문자들과의 특징 정보의 일치 정도를 퍼지 함수를 이용하여 판단하는 방식을 활용하였다. 2가지 종류의 표본 문자와 4가지 종류의 테스트 문자를 이용하여 실험한 결과 폰트 형태에 따라서는 인식률이 저하되지 않았지만 폰트 굵기 차이에 의해서 인식률이 저하되는 현상이 나타남을 알 수 있었다.

키워드

영어 소문자, 오프라인 인식, 퍼지 논리

I. 서 론

문자 인식은 카메라(camera)나 스캐너와 같은 입력 장치를 사용하여 입력을 받아들인 다음 전, 후처리를 통하여 인식하는 (off-line)방식과 일상적으로 사용되는 필기구인 펜이나 종이의 특성과 유사한 스타일러스(stylus), 타블렛(tablet), 라이트 펜(lingt pen), LCD판, 마우스(mouse)와 같은 입력 장치를 사용하여 입력을 받아들임과 동시에 인식이 되는 온라인(on-line)방식으로 크게 분류 할수 있다.

오프라인 문자인식은 고품질의 단일 인쇄체, 다중 활자체를 거쳐 필기체 인식에 이른다. 일반적인 오프라인 문자 인식 시스템의 구조는 입력 영상의 전처리 과정, 영역분할과정, 특징 추출과정, 인식 과정, 후처리 과정을 거쳐 인식되어진 문자들의 집합을 결과로 얻게 되는 전체의 과정을 의미한다.

최근 임베디드 시스템, 유비쿼터스 등으로 하드웨어가 초소형화 되어감에 따라 빠른 알고리즘의 필요성이 부각되고 있다. 이에 영상인식의 분야 또한 빠른 알고리즘의 필요성이 커져가고 있다. 본 논문에서는 영상인식에서 FFT, DCT등 복잡한 수식을 필요로 하는 알고리즘을 제외한 형태학적 특징와 간단한 퍼지 논리만으로 처리 속도 면에서 빠른 영상인식 알고리즘을 제안하였다.

본 논문에서 제안한 오프라인 문자인식을 처리 흐름도는 그림 1과 같다.

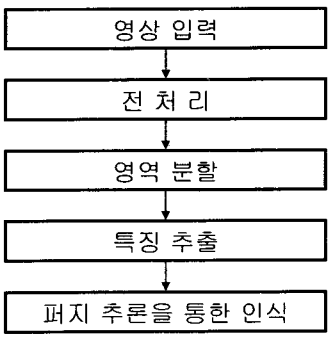

그림 1. 처리 흐름도

먼저 영상이 입력되면 입력된 영상의 그레이 변환, 이진 변환, 문자영역 추출, 크기 정규화 등

의 전처리 과정을 거친 후 본 논문에서 제안한 블록화를 이용하여 영상의 특징을 추출한다. 이후 퍼지 논리를 이용하여 영상을 인식한다.

II. 전처리

전처리 과정은 그림 2와 같이 컬러 문자 영상이 입력되면 영상을 그레이 레벨로 변환한다. 전체 영상의 평균값을 임계값으로 설정한 뒤 반복적으로 추정 값을 향상시키는 방법인 반복 이진화를 적용한다[1].

그림 2. 전처리 과정

반복 이진화 알고리즘은 다음과 같다.

1. 영상의 명암도 평균을 구한 후 이를 임계값 T로 정한다.
2. T를 이용하여 영상을 두 영역 R1과 R2로 분할한다.
3. R1과 R2의 평균 명암도 u1과 u2를 구한다.
4. 구해진 u1과 u2의 평균값이 T값과 같은지 비교후 같다면 이진화 임계값을 T값으로 정하고 알고리즘을 끝낸다. 다르다면 2번으로 돌아간다.

문자 영상의 분할은 먼저 가로축으로 명암도 변화가 없는 부분을 찾아내어 가로축으로 영상을 분할하고 분할되어진 영상을 같은 방법으로 세로축으로 분할하여 문자 영역을 찾는다. 이때 영상의 분할 등을 위해선 이진 영상을 사용하지만 잘려진 영상의 복사는 이진 영상이 아닌 그레이 영상으로 복사 하도록 한다. 이는 영상의 확대나 축소를 위해선 이진 영상보다 그레이 영상을 사용하는 것이 좋기 때문이며 앞에서 소개된 반복 이진화 알고리즘을 이용하여 다시 이진 영상으로 변환한다.

이후 잘려진 문자영역의 영상을 m×m크기의 영상으로 정규화하여야 하는데, 정규화 과정 중 영상의 가로, 세로의 비율이 변하지 않도록 영상의 가로, 세로값 모두가 m값보다 크다면 0.99를 계속 곱하여 가로, 세로값이 m값보다 작아질 때까지 반복한다. 만약 가로나 세로값이 m보다 작다면 1.01을 계속 곱하여 가로 또는 세로가 m값이 되도록 한다. 이후 양선형 보간법을 이용하여 영상을 m×m의 크기로 정규화시킨다[2].

영상의 크기 정규화 및 이진영상으로의 변화에 따라 영상에 잡음 및 왜곡 현상이 많이 발생하는데 이를 줄이기 위해 Dilation 필터를 이용하여 영상을 확장한다[3].

III. 블록화를 이용한 영상 세그멘테이션

3.1 영상 블록화

각 문자영상의 블록화는 M×M크기의 영상을 p×p크기의 영상으로 분할한다. 그러면 원영상은 m = [M/p]이라 할 때 m×m개의 블록으로 분할되어 진다. 이때 본 논문에서는 m의 값이 정수값이 나오도록 M값과 p의 값을 맞춰준다[4].

다음 식으로부터 분할 되어진 블록(i, j)의 명암도 I_{ij}를 산출한다.

$$I_{i,j} = \sum_{i=0}^{i-1}\sum_{j=0}^{j-1} Hist(h,v) \qquad (1)$$

식 (1)로부터 얻어진 I_{ij}의 값은 분할된 블록의 대표 명암도로 정해지며 대표 명암도 값이 어떠한 임계값 x보다 크다면 우리는 이를 목표블록이라 정하고 대표명암도가 임계값 x보다 작다면 비목표 블록으로 정하여 영상의 특징추출을 위해 사용한다.

3.2 영상의 특징추출

영상의 특징추출은 분할된 블록의 인접성을 검사하게 되는데 여기서 인접성이란 분할된 블록의 주변 8방향 좌상, 상, 우상, 좌, 우, 좌하, 하, 우하의 이웃 위치를 말한다. 여기서 우리는 분할된 블록의 인접된 블록 중에 목표 블록이 있는지 찾는다. 이후 전체 블록에 대해 인접된 목표블록의 수를 세어 각 블록의 특징값으로 사용하며, 이 값이 퍼지 논리의 특징값이 되도록 한다.

IV. 퍼지 추론을 통한 인식

영상의 특징값을 퍼지 추론을 이용하여 인식하게 되며 퍼지 제어 구조는 그림 3과 같다[5].

그림 3. 퍼지 제어 구조

입력 변수는 대상블록의 인접블록의 수가 된다. 입력값은 0부터 8까지이며, 입력값이 0값이거나 표본값이 0이면 주위에 인접블록이 없는 것으로 인지하고 이는 특징값으로 포함하지 않고, 퍼지 논리에서 제외한다.

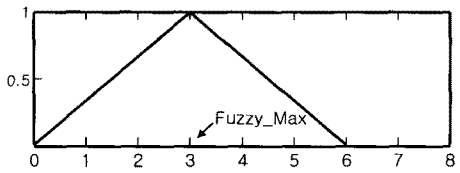

그림 4. 퍼지 멤버십 함수

그림 4는 표본 영상 a(0, 1)의 특징값(3)에 대한 퍼지 함수이다. 입력 영상의 (0, 1)을 위의 퍼지함수 입력값으로 두고 식 (2)를 이용해 멤버십 값을 얻는다.

$$\begin{aligned} &\text{if}(input \leq Fuzzy\_Max) \\ &\quad output = \frac{1}{3}(input - Fuzzy\_Max + 3) \\ &else \\ &\quad output = -\frac{1}{3}(input - Fuzzy\_Max - 3) \end{aligned} \tag{2}$$

$$Output_i = \sum_{j=0}^{j<m} \sum_{k=0}^{k<m} output_{j,k} (i = a,..,z) \tag{3}$$

$$Max(Output_i)\ (i = a,...,z) \tag{4}$$

입력 영상의 분할된 m×m블록과 표본 영상의 m×m블록에 대해서 각각 위 식 (2)를 이용해 멤버십 값을 얻는다.

이후 식 (3)을 이용하여 각 문자에 대한 전체 멤버십 같을 얻는다. 이 과정을 표본 영상 a~z까지 반복하여 수행하도록 한다.

위 연산을 이용하여 얻은 값들 중 식 (4)를 이용하여 최대값을 구하면 그 값이 최종 인식 결과가 된다.

V. 실험 및 결과 분석

실험 환경은 Intel Pentium-IV 15GHz CPU와 512MB RAM이 장착된 PC 상에서 VC++ 6.0으로 구현하였다. 표본 문자는 그림 5와 같은 '돋움', '바탕'의 2가지를 사용했으며 폰트의 크기는 32 point를 사용하였다. 테스트를 위한 입력 문자로는 '한컴돋움', '한컴바탕', '신명조', '맑은고딕'을 사용했으며 1,633자를 대상으로 실험을 수행하였다.

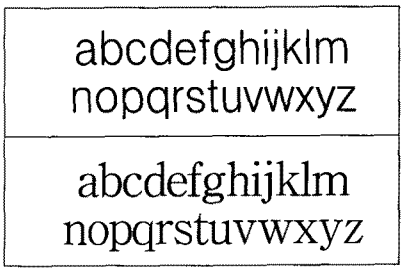

그림 5. 표본 문자(돋움, 바탕)

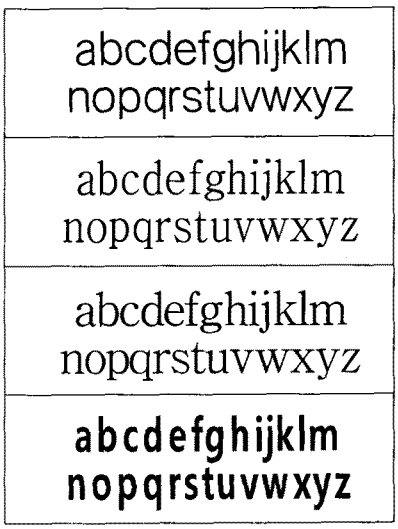

그림 6. 테스트 문자(한컴돋움, 한컴바탕, 신명조, 맑은고딕)

그리고 글자의 폰트의 경우 상대적으로 높은 인식률을 가지는 신명조에 대해서 표본문자보다 작은 폰트크기 20, 24와 표본 문자보다 큰 폰트크기 36과 48에 대하여 성능 평가에 적용하였다.

본 논문에서 제안한 소문자 오프라인 인식방법을 사용하여 폰트 크기가 같을 경우 표 1과 같은 인식 결과를 얻을 수 있었다.

입력 문자의 폰트 굵기가 영상의 인식에 많은 영향을 끼친다. 표 1에서 보면 알 수 있듯이 영상의 폰트 굵기가 비슷한 한컴돋움, 한컴바탕, 신명조 같은 경우는 높은 인식률을 얻을 수 있었지만 폰트의 굵기가 다른 맑은고딕 같은 경우는 낮은 인식률을 보이는걸 알 수 있다. 이와 같은 경우 영상을 블록으로 나눌 때 잡았던 p의 값을 크게 잡는 다면 맑은 고딕의 인식률을 높일 순 있지만 그럴 경우 특징값의 범위가 넓어져 전체 인식률은 낮아진다. 이와 같은 문제점을 해결하기 위해선 영상의 굵기를 일정하게 해줄 수 있는 세선화(thinning) 기법이 필요할 것이다.

그림 7은 굵기 차이에 의해 잘못된 인식의 예를 보여 준다.

표 1. 소문자 인식 결과

대상문자	인식률
한컴돋움	92.30%
한컴바탕	92.12%
신명조	98.70%
맑은고딕	84.61%

그림 7. 굵기 차이에 따른 잘못된 인식

폰트 크기에 따른 인식 결과는 표 2와 같다. 표 2를 보면 알 수 있듯이 영상의 폰트 크기가 표본 영상보다 상대적으로 많이 작아지게 되면 인식률이 많이 저하 되는걸 볼 수 있으며 표본 영상보다 폰트크기가 비슷하거나 커도 문자 인식률엔 거의 영향을 미치지 않는다는 걸 알 수 있다.

이는 본 논문에서 제안된 특징값에 의한 오인식이라기 보단 영상의 전처리 과정에서 작은 영상을 크게 확대하면서 생긴 왜곡현상 때문으로 보여진다.

그림 8은 j로 인식되어진 l의 영상이다. 이처럼 작은 문자의 경우 영상의 확대에서 영상의 왜곡이 크다는 걸 알 수 있다.

표 2. 폰트 크기에 따른 인식 결과

폰트 크기	인식률
20	73.00%
24	98.29%
36	97.56%
48	100%

그림 8. j로 인식된 l

VI. 결 론

본 논문에서는 영상의 형태학적 특징과 퍼지 논리를 이용하여 문자를 인식하는 방법을 제안하였다. 스캔하여 입력 받은 영상을 그레이 레벨로 변환한 후, Determine Threshold 알고리즘을 이용하여 영상을 이진화하였다. 이후 영상에서 각 문자 영역을 추출하여 분할되어진 영상을 블록화하여 이를 퍼지 논리의 특징값으로 사용하였다.

제안된 블록화를 이용한 인식 방법은 입력 문자와 폰트 굵기가 비슷한 경우 높은 인식률을 보이지만 영상과 폰트 굵기가 다른 경우 상대적으로 인식률이 많이 낮아지는 걸 볼 수 있었다. 또한 영상의 크기가 기준 영상보다 작을 경우 영상의 확대에서 왜곡 현상이 일어나 인식률이 떨어지는 걸 볼 수 있었다.

향후 연구 과제로는 입력된 영상 폰트의 굵기에 따라 인식률이 저하되지 않게 전처리 과정에서 영상의 굵기를 일정하게 유지하는 방법이 연구되어야 할 것이며, 영상의 확대나 축소에 따라 왜곡 현상을 줄일 수 있는 방법이 연구되어야 할 것이다.

참고 문헌

[1] 김광백, 조재현, 안상호, "콘크리트 표면의 영상 처리 기법," 한국해양정보통신학회논문지, 제9권 7호, pp.1575-1582, 2005.

[2] 정성환, 이문호 Visual C++ 디지털 영상처리, 홍릉과학 출판사, pp.550-553, 2006.

[3] 이임건, 박성대, 한수환 Matlab을 이용한 영상처리, 미래컴, pp.154-155, 2006.

[4] 윤영근, 이석룡, 박호현, 정진완, "그레이 영상에서의 영역 확장에 기반한 영상 세그멘테이션 기법," 데이터베이스연구 제21권, 제1호, pp.1-12, 2004.

[5] Timothy J. Ross, Fuzzy Logic with Engineering Applications(2nd ed.), Wiley, 2004.