한국산학기술학회논문지
Vol. 4, No. 3, pp. 248-251, 2003

측면 윤곽 패턴을 이용한 접합 문자 분할법

정 민 철*

Character Segmentation Using Side Profile Pattern

Min Chul Jung*

요 약 본 연구에서는 접합 문자를 분할하는 알고리듬을 소개한다. 문자 인식기는 문자를 인식하기 위해 문자 분할을 전처리 단계에 필요로 하는데, 문자 분할은 높은 수행력을 위해 문자 인식 결과를 필요로 한다. 이 딜레마를 해결하기 위해서는 문자 분할과 문자 인식, 이 두 문제를 동시에 해결하는 방법이 필요하다. 이를 위해 본 논문에서는 문자 분할 전에 접합 문자 내에 있는 소속 문자를 인식하고 문자를 분할하는 새로운 문자 분할 방법을 제시한다. 본 연구에서 제시한 문자 분할 알고리즘은 접합 문자 내에 있는 소속 문자를 문자 분할 전에 인식하기 위해 측면 윤곽을 정의하고, 그 히스토그램을 구해 프로토타입에 있는 단일 문자의 측면 윤곽 히스토그램과 비교 계산하여 가장 적은 거리차를 가지는 단일 문자를 분할 문자의 일차 후보로 내정하여, 분할 비용을 가지고 접합 문자를 분할한다.

Abstract In this paper, a new segmentation method of machine printed character string with arbitrary length is proposed. Character recognition requires character segmentation as a previous step. However character segmentation itself requires a character recognition capability for less error segmentation. It is necessary to attack both these problem simultaneously. It is proposed that a new recognition-based segmentation method, which recognizes a character in touching characters with help of defined side-profiles. The match of 'side-profiles of touching characters' with 'side-profiles of prototypes' gives single character candidates in touching characters. It segments touching characters according to cutting costs.

Key Words : Character Segmentation, OCR, Cutting Cost

1. 서 론

문서를 인쇄할 때나 또는 스캐닝할 때 각 문자는 이웃하는 문자와 접합(touching)되어 원래 개개의 문자 패턴과는 전혀 다른 새로운 패턴을 형성한다. 이러한 접합 문자(touching characters)를 원래 개개의 문자로 분리하는 것을 문자 분할(character Segmentation)이라 하며, 문서 자동 처리와 인식 분야에서는 현재까지도 완결되지 않은 문제이다. OCR(Optical Character Recognition) 시스템이 문자를 자동으로 인식할 때 접합 문자는 문자 인식률을 크게 저하시키고 에러율은 높게 한다. 그 이유는 OCR 시스템은 단일 문자를 입력으로 받아 특징 벡터를 추출하여 그 문자를 분류하는 데, 두 문자 이상이 접합된 문자열은 단일 문자의 특징 벡터 추출을 매우 어렵게 하기 때문이다. 부정확한 문자 분할은 분류 오류의 가장 큰 원인들 중 하나이다. 사실, 문자 인식을 연구하면서 관찰한 결과 인식 오류의 반 이상은 접합 문자들이 원인이 되었다. 문자가 정확히 분할되는 한, 각 문자 이미지의 질 저하는 OCR 시스템의 전체적인 인식률에 크게 영향을 미치지 않는다. 만약 인식 결과를 신뢰할 수 없다면 그 이유는 바로 접합 문자들을 제대로 분할할 수 없기 때문이다[1]. 소속 문자가 먼저 인식되지 않으면 접합 문자가 분할될 수 없다. 그러나 OCR 시스템은 문자 인식을 위해 먼저 분할된, 즉 독립된 개개의 문자를 필요로 한다. 즉 문자 인식기는 문자 분할을 전 처리 단계에 필요로 하는데, 문자 분할은 문자 인식 결과를 필요로 한다[2]. 위의 문장은 '닭이 먼저냐? 달걀이 먼저냐?'하는 문제와 흡사하다. 이 문제를 해결하기 위해서는 문자 분할과 문자 인식, 이 두 문제를 동시에 해결하는 방법이 필요하다. 이를 위해 본 논문에서는 문자 분할 전에 접합 문자 내에 있는 소속 문자를 인식하고 문자를 분할하는 새로운 문자 분할 방법을 제시한다.

*상명대학교 컴퓨터정보통신공학부

2. 문자 분할의 접근법

본 연구에서 문자 분할을 위해 사용한 접근법은 문자

인식에 기초한 문자 분할이다. 문서에서 분리되어 있던 문자들은 스캐너에 의해 스캔된 후, 이미지 이진화 과정을 거치며 임계값에 따라 문자가 접합된다. 이 때, 접합 문자에서 단일 문자의 접합부는 단일 문자와는 전혀 다른 새로운 패턴을 형성한다. 즉 접합 문자는 단일 문자와는 전혀 다른 새로운 패턴이 되는 것이다. 그러나 접합 문자의 맨 왼쪽 측면 패턴과 맨 오른쪽 측면 패턴은 단일 문자일 때와 동일하며 이는 문자 접합에 영향을 받지 않는다. 이러한 측면 윤곽 패턴을 분석하여 접합 문자 내에 있는 문자를 문자 분할 전에 인식할 수 있다. 영문자 및 숫자는 그림 1에서 알 수 있듯이 독특한 네 방향의 측면 윤곽 패턴을 가진다(한글도 독특한 측면 윤곽패턴을 가지나 본 연구에서는 한글은 제외한다). 대부분의 문자들은 이러한 네 방향의 독특한 측면 윤곽 패턴 중 한 방향의 측면 윤곽 패턴만을 가지고도 인식될 수 있다. 그러나 그림 1에서 보듯이 문자 'd'와 문자 'l'의 오른쪽 측면 윤곽 패턴은 동일함으로, 이 경우 오른쪽 측면 윤곽 패턴 하나만으로는 문자 'd'와 문자 'l'을 구분하여 인식 할 수는 없다. 이러한 경우에는 오른쪽 측면 윤곽 패턴 분석은 인식될 문자의 두 후보 문자 'd'와 'l'을 제공하고, 나머지 방향들의 측면 윤곽 분석을 통해 문자를 결정한다.

3. 문자 분할의 전처리 단계

3.1 측면 윤곽의 정의

위쪽 방향의 측면 윤곽, 아래쪽 방향의 측면 윤곽, 오른쪽 방향의 측면 윤곽, 왼쪽 방향의 측면 윤곽들은 다음과 같이 정의되어진다. y_0, y_2, x_0, x_2는 테두리 박스 (bounded box) 상의 좌표를 y_1, y_3, x_1, x_3는 수직 또는 수평으로 스캔할 때 최초로 문자를 형성하는 픽셀을 만나는 좌표를 말한다(그림 2 참조).

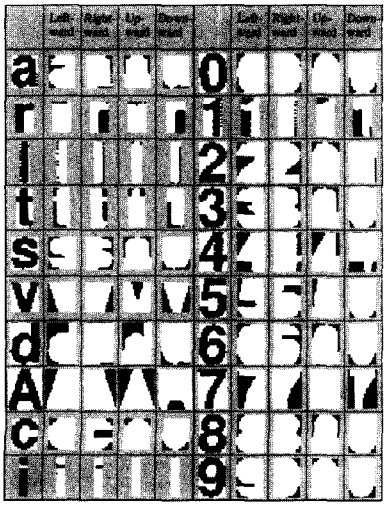

그림 1. 측면 윤곽 패턴의 예

- 위쪽 방향의 측면윤곽: 이미지 $I(x, y)$를 위쪽(y_0)에서 아래쪽으로 수직으로 스캔하면서 블랙 픽셀을 만날 때까지(y_1)의 화이트 픽셀 갯수의 합

$$P_{up}(y) = \sum_{y=y_0}^{y_1} I(x, y) \tag{1}$$

- 아래쪽 방향의 측면윤곽: 이미지 $I(x, y)$를 아래쪽(y_2)에서 위쪽으로 수직으로 스캔하면서 블랙 픽셀을 만날 때까지(y_3)의 화이트 픽셀 갯수의 합

$$P_{down}(y) = \sum_{y=y_2}^{y_3} I(x, y) \tag{2}$$

- 왼쪽 방향의 측면윤곽: 이미지 $I(x, y)$를 왼쪽(x_0)에서 오른쪽으로 수평으로 스캔하면서 블랙 픽셀을 만날 때까지(x_1)의 화이트 픽셀 갯수의 합

$$P_{left}(x) = \sum_{x=x_0}^{x_1} I(x, y) \tag{3}$$

- 오른쪽 방향의 측면윤곽: 이미지 $I(x, y)$를 오른쪽(x_2)에서 왼쪽으로 수평으로 스캔하면서 블랙 픽셀을 만날 때까지(x_3)의 화이트 픽셀 갯수의 합

$$P_{right}(x) = \sum_{x=x_2}^{x_3} I(x, y) \tag{4}$$

3.2 접합 문자의 측면 윤곽 패턴 분석

그림 2는 접합 문자들의 측면 윤곽 패턴을 나타낸다. 그림에서 보듯이 각 접합 문자들의 맨 왼쪽 측면 윤곽 패턴과 맨 오른쪽 측면 윤곽 패턴은 문자들의 접합에 영향을 받지 않음을 알 수 있다. 이러한 접합문자의 맨 오른쪽 측면 윤곽으로부터 히스토그램을 구한다. 여기서 히스토그램의 x축은 픽셀의 개수를 y축은 문자 높이를 나타낸다. 접합 문자의 맨 오른쪽 측면 윤곽 히스토그램은 단일 문자들의 오른쪽 측면 윤곽 히스토그램

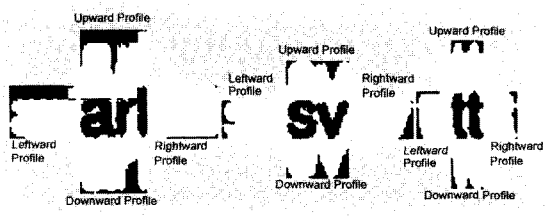

그림 2. 접합 문자의 측면 윤곽 패턴의 예

들과 비교된다(단일 문자들의 네 방향의 측면 윤곽 히스토그램은 미리 구하여 프로토타입으로 저장하여 놓는다). 접합 문자의 맨 오른쪽 측면 윤곽 히스토그램(q)와 프로토타입에 저장된 히스토그램(P)사의 거리 차는 다음과 같이 구해진다.

$$d(q,P) = \sum_{i=1}^{N} |x_{iq} - x_{iP}| \quad (5)$$

위 식 (5)에서 N은 정규화된 문자 높이이고, x_{iq}는 접합 문자 히스토그램의 y축 값이고, x_{iP}는 프로토타입으로서 단일문자 히스토 그램의 y축 값이다. 가장 적은 거리차를 가지는 프로토타입의 단일 문자가 접합 문자의 맨 오른쪽 문자의 후보가 된다. 단일 문자의 문자 폭(정규화된 단일 문자 폭을 프로토타입으로 가지고 있다)으로 접합 문자는 분할하고, 분할된 패턴으로부터 왼쪽 방향의 측면 윤곽, 위쪽 방향의 측면 윤곽, 아래쪽 방향의 측면 윤곽의 히스토그램을 구해 후보가 된 단일 문자의 히스토그램들과 각각 비교한다.

4. 문자 분할

4.1 분할 비용

비록 정규화를 하더라도 접합 문자에 속에 있는 단일 문자의 폭은 잡음, 스캔 해상도 등등에 의해 프로토타입의 문자 폭과는 약간의 차이가 존재한다. 따라서 프로토타입의 맨 마지막 문자 폭은 분할 패스를 제공하는 것이 아니라 분할할 가능성이 있는 영역을 제공한다. 분할 패스는 이 분할 영역에서 분할 비용을 고려하여 결정된다. 본 연구에서 분할 비용은 "접합 문자를 분할하기 위해 얼마나 많은 갯수의 블랙 픽셀을 나누어야 하는가?"이다. 그림 3에서 라인 ①의 분할 비용은 22, 라인 ②의 분할 비용은 21, 라인 ③의 분할 비용은 3, 라인 ④의 분할 비용은 라인 ③과 같은 3이다.

4.2 분할 비용에 따른 문자 분할

그림 3의 접합 문자 'ix'에서 오른쪽 측면 윤곽 히스토그램과 프로토타입들의 히스토그램의 거리차를 식 (5) 로부터 계산하면 접합 문자의 맨 오른쪽 문자 후보는 x로 나타나진다. 그러나 x의 문자 폭 16으로 오른쪽에서 왼쪽으로 16 픽셀만큼(라인 ②) 접합 문자를 분할할 경우 부정확한 문자 분할이 된다. 이를 해결하기 위한 방법으로, 프로토타입에 있는 문자 'x'의 문자 폭 16은 단지 참조 라인 ②를 찾는다. 이 참조 라인 전후로 2 픽셀 라인을 추가하여 분할할 가능성이 있는 분할 영역(neighborhood)을 구한 후, 앞에서 언급한 분할 비용을 구하면 라인 ③과 라인 ④가 최소 분할 비용 3을 갖는다.

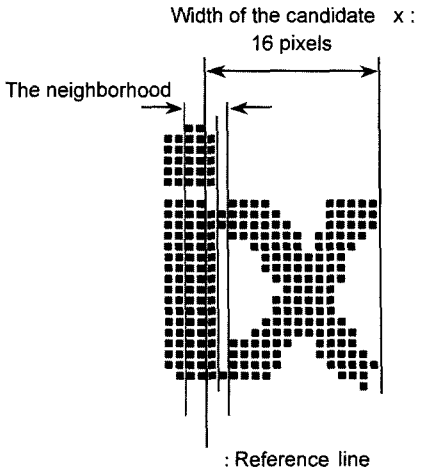

그림 3. 접합 문자의 분할 비용(라인 ①의 분할 비용은 22, 라인 ②의 분할 비용은 21, 라인 ③의 분할 비용은 3, 라인 ④의 분할 비용은 라인 ③과 같은 3)

라인 ③이 참조 라인 ②에 더 근접함으로 라인 ③이 최종적으로 접합 문자 'ix'의 분할 패스로 결정된다. 접합 문자 'ix'로부터 분할된 패턴은 프로토타입 단일 문자 x의 왼쪽 방향의 측면 윤곽, 위쪽 방향의 측면 윤곽, 아래쪽 방향의 측면 윤곽의 히스토그램과 비교하여 문자 분할의 정확성을 검증한다. 만일 왼쪽 방향의 측면 윤곽, 위쪽 방향의 측면 윤곽, 아래쪽 방향의 측면 윤곽의 히스토그램의 거리차의 합이 임계값보다 클 경우, 그 다음 후보 문자를 선택하여 위의 문자 분할 과정을 반복한다.

5. 테스트와 결과

본 연구에서 제안된 문자 분할 알고리즘은 SUN Sparc에서 C언어로 구현되어져 여러 접합 문자들에 대해 테스트되어졌다. 접합 문자 rf, fr, fi, tt, ff, rt, rp와 II 등은 접합 문자 폭이 단일 문자 폭과 적거나 같아서 접합 문자로 발견되기도 어려울 뿐 아니라 분할 패스를 발견하는 것도 어렵다[3]. 본 연구에서 제안된 문자 분할 알고리듬은 위의 접합 문자들도 성공적으로 분할하였다. 또한 접합 문자가 단일 문자의 패턴과 흡사한 Ll(U와 흡사), cl(d와 흡사), rn(m과 흡사) 등도 성공적으로 분할하였다. 그림 4는 본 연구에서 제안된 문자분할 알고리즘으로 가변피치 폰트로 프린트된 접합 문자들의 성공적 문자 분할의 예를 보여준다.

6. 결 론

본 연구에서는 접합 문자를 분할하는 새로운 알고리

그림 4. 접합 문자의 성공적 분할

듬이 제안되었다. 본 연구에서 문자분할을 위해 사용한 접근법은 문자 인식에 기초한 문자 분할법으로 부분적인 문자 인식 결과를 이용하여 문자 분할을 함으로 난해한 문자 분할도 성공리에 수행하였다. 실험 접합 문자의 에러율은 4.23%이다. 단점으로 OCR 시스템이 문자 분할에 시간을 많이 소모하여 시스템의 전체적인 인식 속도가 저하되었다.본 연구에서 제안된 문자 분할 알고리듬은 영문자와 숫자에 테스트되어졌지만, 한글의 문자 분할에도 적용되어질 수 있으며 그 외 다른 언어의 문자 분할에도 응용되어질 수 있다..또한 본 논문에서 정의한 네 방향의 측면 윤곽은 다른 패턴 인식 분야에도 응용될 수 있다[4].

참고문헌

[1] M. Bokser, "Omnidocument Technologies", Proceedings of the IEEE, Vol. 80, No. 7, pp. 1066-1078, 1992.

[2] H. Fujisawa, Y. Nakano and K. Kurino, "Segmentation Methods for Character Recognition: from Segmentation to Document Structure Analysis", Proceedings of the IEEE, Vol. 80, No. 7, pp. 1079-1092, 1992.

[3] Y. Lu, "On the Segmentation of Touching Characters", 2nd International Conference on Document Analysis and Recognition, pp. 440-443, 1993.

[4] Minchul Jung, "Multifont Classification Using Typographical Attributes", 5th-International Conference on Document Analysis and Recognition, pp. 353-356, 1999.