문서 영상에서의 테이블 벡터화에 관한 연구

심진보* 김우성* 박용범** 오원근***

* 호서대학교 컴퓨터 공학과
** 단국대학교 전자계산학과
*** 한국과학기술연구원 시스템공학연구소 인공지능 연구부

요 약

본 논문에서는 문서 인식 시스템에서 정확한 문서 인식의 기본이 되고 인식 결과에 중요한 영향을 미치는 전처리 알고리즘 중 테이블 입력의 효율적인 처리 방법을 연구한다. 테이블 내의 문자를 인식하기 위해서는 테두리선과 문자 부분을 먼저 분리하는 작업이 필요하다. 왜냐하면, 테이블을 인식하기 위해서는 테두리선에 의해 블록화된 테두리선 안의 문자를 인식해야 하며 또한 테두리선을 효율적으로 벡터화하는 방법이 필요하다.

따라서 테이블을 벡터화하는 방법으로 8방향 체인 코드를 이용하여 테이블 선 성분을 추출하는 방법과 히스토그램을 이용하여 테이블의 수평, 수직 성분을 추출하여 얻어진 교차점을 이용하여 대각선 성분을 찾아내는 방법 및 화소의 run-length를 이용하여 수평, 수직 성분을 추출하여 얻어진 교차점을 이용해 대각선 성분을 찾아내는 방법 등이 있다. 본 논문에서는 문서 영상 내의 테이블을 효율적으로 벡터화하기 위한 방법을 연구한다.

1. 서론

본 논문은 문서 인식 시스템에서 정확한 문서 인식을 위해, 테이블 입력의 효율적인 처리 방법을 연구한다. 이러한 처리 알고리즘으로 본 논문에서 택한 알고리즘은 3가지이다. 첫 번째로 8방향 체인 코드를 사용한 테이블 영상의 벡터화 방법이다 [1,2]. 두 번째로는 테이블 영상의 수직, 수평의 투영을 통한 히스토그램을 사용하여 각각 수직선 성분과 수평선 성분을 추출하고, 그 교차점을 구하여 교차점 정보를 가지고 대각선 성분을 추출하였다. 세 번째로 흑화소의 run-length를 이용하여 수직, 수평 성분을 추출하여 교차점을 찾고, 대각선 성분을 찾아낸다

이러한 세가지 방법은 각각 그 특성이 있다. 체인 코드를 이용한 첫 번째 방법은 모든 pixel에 대한 정보를 가지고 문자와 테이블을 구별함으로써, 영상 정보를 최대한 이용한 알고리즘을 구현하였다는 것이다. 그러나 이러한 처리를 했을 때 가장 문제가 되는 것은 처리 시간이 였다. 모든 pixel에 대한 조사를 하기 때문에 그 처리 시간이 많이 소요되었다. 그러나 결과는 문자와 테이블을 확실히 구별하여 벡터화 한 데이터를 가지고 테이블을 재구성 할 수 있었다. 첫 번째 알고리즘의 문제인 처리 시간의 단축을 위하여 히스토그램을 이용한 방법과 run-length를 이용한 테이블의 추출을 연구하였다. 또한 영상의 어떤 보정없이 문자와 테이블이란 특수한 상황을 고려하여 연구하였다. 문자와 테이블의 구별은 그 연속성에 있는데, 문자를 구성하는 연속된 흑화소의 수는 테이블을 구성하는 연속된 흑화소보다 작다는데, 착안을 두었으며, 같은 생각에서 이러한 영상을 투영했을 때, 문자가 있는 곳보다는 테이블의 선 성분이 있는 곳에 더 많은 투영값이 생겼다. 이러한 처리는 체인 코드를 사용한 알고리즘보다 처리 시간은 많이 줄어들었으나, 제한적이라는 문제를 가지고 있었다.

2. 테이블 영상의 벡터화

테이블 영상을 추출하는 본 연구에서 연구한 알고리즘은 각기 세가지가 있다. 체인 코드를 이용한 알고리즘과 히스토그램을 이용한 알고리즘, 그리고 run-length를 이용한 방법이 있다. 체인 코드는 처음 만나는 흑화소를 기준으로 이웃 8화소를 검사하여 흑화소가 있으면 그 흑화소에 8방향의 체인 코드를 부여하는 것이다. 그래서 이러한 방법으로 수직 방향과 수평 방향의 체인 코드를 갖는 흑화소를 찾는 것이다. 그러나 모든 흑화소에 대한 이러한 처리는 바람직하지 않기 때문에, 이러한 처리 이전에 영상을 이진화하고, 이진화된 영상을 다시 세선화 과정을 거침으로써 처리할 데이터를 줄였다. 이러한 전처리는 체인 코드를 사용한 처리에 필수적이다. 그러나 문제가 되는 것은 이러한 체인 코드를 이용한 처리를 하기전의 전처리에서 많은 시간이 든다는 것이다. 이진화 과정은 처리 시간이 거의 들지 않지만 세선화 과정은 많은 소요 시간을 갖는다. 이러한 이유는 세선화를 위한 마스크를 사용하기 때문이다. 이러한 세선화 과정에서 평활화가 일어나는데, 본 연구에서 택한 SPTA알고리즘은 이러한 심한 평활화가 없다. 그리고 영상 처리에서 영상데이터를 보다 효과적으로 세선화 할 수 있다.

2-1 8방향 체인 코드를 이용한 벡터화

1) 전처리

8방향 체인 코드 알고리즘을 사용하기 전에, 그 전처리로써 gray-level value를 갖는 영상 데이터를 흑, 백의 두 값을 갖는 binary 데이터로 만든다. 배경에서 객체를 추출하기 위한 그레이 레벨의 적당한

임계치를 선정하는 것이 영상 처리에선 중요하다. 이상적인 경우에 히스토그램은 배경과 객체를 나타내는 두 정점 사이에 깊고 날카로운 골짜기를 가지므로 이 골짜기의 바닥값으로 선택되어질 수 있다. 이러한 임계치를 선정하는 것은 골짜기가 평평하고 넓으며, 노이즈가 많거나, 또는 두 정점이 두드러지게 높아서 종종 골짜기를 발견할 수 없어 어려울 때가 있다. 본 연구의 이진화를 위한 임계치 선택에서 otsu 알고리즘을 사용하였다. otsu알고리즘은 최적의 임계치를 자동적으로 선택하는 것에 접근한다. otsu 알고리즘은 Gray level histogram을 정규화하고 확률 분포처럼 간주하여, 주기적인 시간 변수를 사용하지 않고 이를 통해 통계되지 않는 임계치를 사용하여 이진화를 수행한다. (nonparametic & unsupervised) 이것은 local minimum T_0를 자동적으로 설정한다. 이렇게 처리된 이진화 데이터는 다음 전처리를 거치게 되는데, 이것이 세선화 과정이다. 세선화 과정은 체인 코드 사용하기 위한 것이다. 체인 코드 사용할 때 주변의 많은 흑화소를 둔다면 특정한 특징점을 찾는데 많은 조건이 필요하기 때문에 이를 간소화하기 위해서이며, 실제적인 문자와 데이블의 골격만을 처리하기 위한 것이다. 이러한 세선화 과정에 SPTA알고리즘을 사용했다 [3,4]. SPTA알고리즘의 특징은 패턴의 연결성을 깨지 않는다는 것이다. 기존 세선화 알고리즘은 세선화를 하는 과정에서 영상의 연결성이 끊어지는 경우가 있었으나, 본 논문에서 택한 SPTA 알고리즘은 이러한 문제에 강하다. 그리고 심한 평활화를 일으키지 않는다. 더 나아가서 이 SPTA알고리즘은 가능한 실제 패턴과 유사한 패턴을 재구성하기 위한 충분한 정보를 얻어낸다. 이러한 SPTA의 조건은 아래와 같다.

* N(p)는 <그림 2> 또는 <그림 2>를 회전시킨 윈도우 중 어느 하나를 만족시킨다.
* N(p)는 정확히 2개의 검은 4-neighbor pixel들을 포함한다.

위의 조건들이 아래 식으로 만족될 경우 왼쪽 윤관 점 P가 안전점이 된다.

① left safe point

$$S_4 = n_0 \cdot (n_1 + n_2 + n_6 + n_7) \cdot (n_2 + n_3) \cdot (n_6 + n_5) \quad (1)$$

② right safe point

$$S_0 = n_4 \cdot (n_5 + n_6 + n_2 + n_3) \cdot (n_6 + n_7) \cdot (n_2 + n_1) \quad (2)$$

③ up safe point

$$S_2 = n_6 \cdot (n_7 + n_0 + n_4 + n_5) \cdot (n_0 + n_1) \cdot (n_4 + n_3) \quad (3)$$

④ down safe point

$$S_6 = n_2 \cdot (n_3 + n_4 + n_0 + n_1) \cdot (n_4 + n_5) \cdot (n_0 + n_7) \quad (4)$$

n_3	n_2	n_1
n_4	P	n_0
n_5	n_6	n_7

<그림 1 마스크의 위치 >

*		X
	P	X
X	X	X

X	X	X
	P	X
*		X

X		
	P	X
X		

X	X	X
	P	
Y	Y	Y

< 그림 2 세선화를 위한 마스크 >

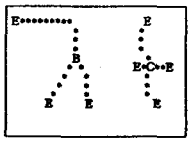
이 알고리즘은 경계점을 제거하기 위하여 미리 정의된 그림 2의 4개의 마스크에 하나라도 부합되면 이 점을 safe point라하고 지우지 않는다. 그렇지 않은 점은 flag라하여 에지점으로 생각하여 지운다. 그림 2의 마스크는 왼쪽 에지를 찾기 위한 마스크이며, 그림 2의 마스크를 회전함으로써 오른쪽, 위쪽, 아래쪽의 에지를 찾는다. 그림 2에서 x, y는 don't care이고, *는 흑화소이며 빈 영역은 백화소를 나타낸다. 이러한 과정을 Bloolean 표현으로 나타내기 위한 마스크의 위치를 그림 1에 정의하였다. 식 (1), (2), (3), (4)에서 " ' "는 complement를 나타내고, 각 Blooean 변수는 흑화소인 경우 true이고, S4, S0, S2, S6이 각각 false인 경우 왼쪽, 오른쪽, 위쪽, 아래쪽의 safe point가 된다. 이 알고리즘의 구현은 tree구조 형태의 비교를 통하여 비교 회수를 줄여 수행 시간을 줄일 수 있다[4].

2) Feature extraction

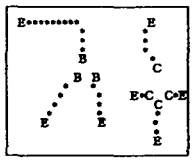
영상 데이터는 SPTA skeleton algorithm 과정를 거치는 동안 끝점, 분기점, 교차점이 특징점으로 추출된다. 이러한 특징점 데이터는 다음과 같은 조건을 거쳐 특징점으로 추출이 된다. 주변 8점을 원형으로 돌면서 흑백화소의 변화가 4번이면 끝점으로 간주하였다. 주변 8점의 변화가 6번이면 분기점으로 간주하였다. 주변 8점의 변화가 8번이면 교차점으로 간주하였다. 이렇게 처리된 예를 그림 3에 보였다.

·3) Stroke extraction

특징점 추출 과정을 거쳐 나온 특징점 데이터를 가지고 특징점과 특징점 사이를 연결하는 화소들의 집합을 한 개의 stroke로 정의하여 그림 4에 보여지는 스크로크를 추출하였다.



<그림 3 >
세선화 후 특징점 추출 결과



<그림 4>
그림 3을 이용한 Stroke 추출 결과

4) Table extraction

스트로크 추출 과정에서 얻어진 데이터를 가지고, 각 스트로크의 arc길이와 시작점과 특징점 사이의 거리를 계산하고, arc와 시작점과 특징점 사이에서 얻어진 거리의 비가 0.9 이상이면서, 흑화소의 개수가 20 이상이면 table의 segment로 판별하였다. 구부러진 stroke일 경우 다각형 근사(Polygonal approxima-tion)기법을 이용하여 직선 segment로 추출하였다 [5,6]. 이러한 다각형 근사기법은 이산적인 좌표값으로 표시되기 때문에 컴퓨터에서는 처리하기 힘들다. 따라서 이러한 이산적인 좌표값 대신에 8방향 체인코드 이용해 좌표값을 체인 코드로 나타내어 처리한다. 8방향 체인 코드의 체인 코드값 0~7은 45° 간격으로 기울기 각도 0° ~ 315° 도를 나타낸다. 이러한 이산적인 좌표를 8-방향 체인 코

드화 할 때 오는 불연속을 없애기 위해서 unwrap시켰다 [7].

2-2. 히스토그램을 이용한 테이블 벡터화

테이블을 인식하려면 크게 두 가지 방법으로 접근할 수가 있는데, 첫째는 일반적 형태의 접근이고, 둘째는 국소적 형태의 접근 방법이 있다. 히스토그램을 이용한 알고리즘은 앞에서 제시한 두 가지 접근 방법 중에서 전자인 일반적 형태의 접근 방법에 대해서 설명한다.
테이블을 인식하려면 테이블을 이루는 선분들을 구별해 내야 한다. 이 선분들을 구별하기 위한 방법으로 크게 나누어 세 가지 단계로 이루어진 방법을 사용하였다.

* 첫째, 선이 존재할 만한 곳을 찾는다.
* 둘째, 선이 존재할 가능성이 있다면 그것이 올바른 선인지 아니면 글자를 구성하는 화소들의 집합인지를 가려낸다.
* 셋째, 찾아낸 선을 전에 발견한 선과 비교하여 연속적으로 존재하면 세선화의 과정 대신에 전에 찾은 선을 삭제하고 나중에 찾은 선의 정보만을 기억한다.

위와 같이 세 가지 단계를 거쳐서 나오게 되는 정보는 선분의 존재 위치와 길이가 되며, 여기에서 선들의 교차점을 알아내고, 찾아낸 교차점을 바탕으로 대각선분을 찾아냈다.

1) 전처리

입력 영상은 400dpi의 테이블 영상으로 취득하였다.테이블을 처리하기 위해 저장한 데이터는 스캐너로 입력받은 문서의 이미지가 되며, 그 이미지의 화소들을 처리하기 쉽게 이진화하여 흑화소는 1로 백화소는 0으로 표현하는 저장 방식을 사용하였다. 이진화의 과정에서 얻어지는 흑화소의 수를 많게 하여 테이블이 더 정확하게 나타나도록 하였다. 문서 이미지는 처리 과정을 거친 후에 다시 같은 문서 이미지로서 저장된다. 수평선과 수직선의 획득은 영상의 처리 전 입력 시에 순수 이미지 영역을 얻어내므로서 얻을 수 있었다.

2) 수평선의 획득

수평선의 획득은 전체 영상의 수평축으로 스캔을 하면서, 전처리 과정에서 얻어진 데이터 영역을 처리하게 되는 데, 선분인지를 판명하는 방법은 수평축에 존재하는 전체 화소수로 기준을 삼았다. 만약에 수평축 전체에 존재하는 화소의 수가 어느 일정 수준 이상이 되면 그 줄에는 선이 존재할 가능성이 있다고 가정하고 다음 처리 단계로 넘어가게 된다. 그리고, 그 줄에 선이 존재할 가능성이 있다고 판명되었을 경우에는 다시 스캔하여 연속된 화소의 수 중 가장 긴 것을 찾아내고 연속화소의 수가 어느 수준 이상으로 판명이 되면 그 라인은 선분으로 인정하고 선분으로 인정된 부분의 가장 처음 화소와 가장 마지막 화소를 찾게 된다. 이 두개의 화소를 찾았으면 테이블을 구성하는 정보로서 선분이 존재하는 현 라인의 y좌표와 그 길이를 함께 저장하게 되고, 만약에 선분이 아니라고 판명되면 그 줄은 아무것도 존재하지 않는 공백으로 채우게 된다.

따라서, 마지막에는 테이블을 구성하는 선분만이 남게 된다. 찾아낸 선분을 그 선분의 시작 위치에서부터 마지막 위치까지 흑화소로 채워 똑바른 직선으로 재구성하게 되며, 원래 문서 이미지의 예각으로 기울어진 선분을 보정하게 된다. 실험적으로 결과가 제대로 나온 것은 예각 중에서도 10도 안의 각도로 기울어져 있는 경우였으며, 거의 원화상에 가까운 정보가 나오게 되나, 그 이상의 각도로 기울어져 있으면 원화상에 충실하지 못한 정보가 나오게 되고, 전체적으로 선분의 길이는 짧아지게 된다.

3) 수직선의 획득

수직선의 획득은 수평선의 획득과 거의 같은 방법으로 선분에 대한 정보를 획득하게 되나, 몇 가지 상이한 점이 있는 부분을 위주로 설명을 하도록 한다. 수직선과 수평선의 차이는 수평선의 주위에 다른 문자가 오는 경우는 거의 없지만, 수직선의 경우에는 수직축 스캔을 기준으로 할 때, 수직선 주위에 문자를 구성하는 화소가 존재할 가능성이 높다.

이러한 점에서 볼 때 수직선의 처리가 수평선의 처리보다 글자를 처리하는 부분이 더 강화되야만 한다는 것을 암시하여 준다. 여기에서 사용한 방법은 수직선을 검색시 연속된 화소의 수가 한 문자를 구성하는 화소의 수평이나 수직의 길이보다 적으면 문자를 구성하는 흑화소로 인식하고 그 흑화소는 선분의 범위에서 제외하는 방법이다. 이 방법은 스캐너로 입력받은 이미지가 불량할 때 치명적이라는 단점이 있으나 일반적인 데이터를 놓고 봤을 때, 원래 이미지에 충실하다. 그리고 또 다른 문제점은 테이블 내의 한 격자 안에 수직선이 존재할 경우에는 역시 위에서와 마찬가지로 선분과 문자의 구별이 어렵다는 것이다.

여기에서 얻고자 하는 정보는 수평선의 획득과 마찬가지로 수직선의 x좌표와 길이가 되며, 연속된 수직선 중 가장 긴 것을 택하여 연속된 수직선들의 맨 마지막 위치 정보만을 저장하였다.

4) 수평선과 수직선이 이루는 교차점의 획득

교차점의 획득은 수평선과 수직선에서 얻어진 y좌표와 x좌표를 조합하여 얻는데, 여기에서 두 개의 선이 한 테이블 내에서 만나는가의 여부를 따질 필요가 있다. 하나의 테이블에서 만나지 않는 부분을 제외하고 실질적으로 각각의 테이블을 구성하는 교차점들의 정보를 저장하고 대각선을 찾을 때 넘겨준다. 교차점의 획득에서 중요한 점은 두 개 이상의 테이블이 존재할 경우 각 테이블 별로 교차점에 대한 정보가 있어야 한다는 점이다. 그래야만 교차점의 리스트로 두 개 이상의 테이블을 표현할 수 있기 때문이다.

5) 교차점 리스트를 이용한 대각선의 획득

대각선의 획득은 바로 위에서 얻은 교차점의 정보를 갖고 처리한다. 대각선은 교차점과 교차점을 잇는 선분이라 가정하고, 각각의 두 교차점을 잇는 대각선이 존재하는지의 여부를 살펴 대각선 정보를 얻어내는 것이다. 대각선을 표현하는데 필요한 정보들은 대각선이 존재하는 두 교차점이 된다. 대각선을 찾는 방법은 두 교차점을 잇는 직선의 방정식을 구하고, 그 직선의 방정식을 따라가면서 화소가 존재하는지 여부를 살펴보는 방법을 사용한다.

2-3. Run-Length를 이용한 테이블 추출

2-2에서 언급한 처리는 영상의 일반적인 처리였다. Run-length를 이용한 테이블 입력 처리는 지역적인 처리 방법으로 메모를 최소한 사용하며, 문자와 테이블 선 성분을 구별하기 위해 연구하였다. 일반적인 문서에서 문자와 테이블의 구분은 2-2의 방법에서 언급 한대로 연속된

화소 수의 많고 적음이 있다. 연속된 화소의 수가 클수록 테이블 선 성분의 가능성이 높고, 작을 경우는 문자일 경우가 큰 것이다. 문자 font의 기본적인 크기는 대개 16×16, 32×32, 64×64 의 화소 단위로 되어 있다. 이러한 문자의 크기 정보는 문자의 연속된 화소의 수를 알 수 있게 한다. 즉, 16×16의 크기를 갖는 문장인 경우 수평선, 수직 선 가운데 가장 큰 획의 화소수는 16을 넘지 않는다는 것이다. 따라서 16개 이상의 연속된 화소를 검사하면 문자와 테이블을 구성하는 선 성분과의 구별이 가능하다는 것이다. 그러나 실제 문서에서는 큰 제목이나, 작은 제목 등 그 크기가 다양하다, 보다 큰 크기의 32×32를 기준으로 알고리즘을 작성하여 실험을 하였다.

결과적으로 문서에서 문자와 테이블을 확실히 가려낼 수 있었다. 또한 그 사이즈를 64×64, 128×128로 확장하여 실험한 결과 보다 확실한 영역 구분이 가능하였다.

1) 전처리

본 연구에서는 실험 테이블의 종류를 다섯 가지로 정의 하고 연구를 하였다. 정의한 다섯 가지 문서는 아래와 같다.

* 문서 내에 테이블이 두개 존재하는 경우
* 이중 외곽선을 갖는 테이블
* 대각선을 갖은 테이블
* 수평선 성분과 수직선 성분만을 갖은 테이블
* 기울기가 큰 테이블과 작은 테이블

위와 같은 구분의 이유는 문서 내에 있는 테이블의 수에 상관없는 알고리즘을 생각했기 때문이다. 또한 이중 외곽선을 갖는 특수한 경우는 테이블 작성시 있을지도 모르는 테이블의 꾸밈에 상관없는 알고리즘을 위한 것이었다. 그리고 수치 기입 테이블이나 일정 계획 등 테이블의 형식에는 대각선이 존재하는 경우가 있으므로 이에 대한 처리도 필요하기 때문이었다. 그리고 가장 단순한 구조의 수평선, 수직선을 갖는 테이블과 입력시 입력자의 잘못으로 인한 바르지 못한 스캔상태에서의 알고리즘 적용을 위해 기울어진 테이블을 선택하였다. 이러한 생각에서 취득한 테이블 영상은 400 dpi의 해상도로 취득하였다. 그리고 문서 내에서 테이블 영역만을 실험 영상으로 취득하였다. 이러한 이유는 테이블 영역 외의 문자 영역은 처리 시간의 문제와 연관이 있을 뿐이고, 테이블 내의 문자와 테이블 선 성분의 구분이 가능하면 테이블 영역 외의 문자 영역 내의 문자와 테이블과의 구별이 가능하다고 전제하였다. 이렇게 취득된 영상 데이터는 처리 과정 시 취득된 gray-level의 영상을 흑화소와 백화소로만 나타낸 이진화 영상으로 변화시켰다. 이것은 취득 영상에서 잡음을 제거하기 위해서였고, 또한 테이블의 선 성분을 이루는 흑화소 가운데 가장 연결성이 좋은 흑화소의 집합만을 남겨놓기 위해서였다.

2) 수평선 성분 취득

문서내에 존재하는 어떤 한 문자의 크기를 16×16, 32×32, 64×64로 설정을 하고, 최대 128×128로 가정을 하였다. 우선 처리될 영상 화일에서 테두리를 제외한 실질적으로 처리 되야 할 영상 영역을 취득하였다. 이것은 이진화 시 처리 시간을 생각해서, 실질적인 테이블 영역만을 테이블 영상에서 분리 함으로써 처리 시간을 높이기 위해서였다. 16, 32, 64, 혹은 128의 연속된 화소를 수평선 성분으로 처리하였다. 처음엔 32개의 연속된 흑화소의 run을 발견하면, 그 흑화소의 집합을 테이블을 구성하는 선 성분의 일부로 생각하여 그 부분의 이진화 데이터를 재 저장하고 문자로 생각되는, 즉 32화소의 run을 만족시키지 못하는 집합의 모든 흑화소를 제거하였다. 테이블의 선 성분을 찾는 도중에 테이블의 선 성분임에도 이진화 시 끊어진 부분이 있을 것을 가정하여 끊어진 부분을 보정하기 위해, 연속된 흑화소의 개수가 임계치를 2/3 만족되었다면, 그리고 그 이후에 백화소가 체크되었다며는 백화소의 출현 시점에서 서브루틴으로 들어가 현재까지 체크된 부분이 문자 영역인지 혹은 테이블의 선 성분 영역인지를 판단하게 하였다. 이는 앞에서 언급한데로 테이블의 선 성분임에도 이진화시 임계치 값에 의해 손상된 영상일 가능성도 있으며, 동시에 어떤 문자의 한 획을 이루는 연속된 흑화소의 영역일 수도 있기 때문이다.

3) 수직선 성분 취득

테이블 영상이 스캔되어 저장되었을 때는 연속된 메모리로 정보를 갖는다. 이러한 영상 데이터를 수직으로 처리하기 위해선 같은 방법으로 문자와 테이블의 수직선 성분을 구별하지만 수직의 데이터를 임시 저장할 저장 장소가 따로 있어야 한다 [8]. 이 버퍼는 연속된 영상에서 윤곽을 가지고 하나의 수직 영상을 찾아 임시 저장 장소에 저장을 한다. 그런 후 수직 성분의 취득은 수평 성분 취득과 같은 방법으로 연속된 흑화소의 수를 16, 32, 64, 128로 그 비교 값을 갖고 취득한다. 그러나 실제 구현에서 이러한 저장 장소 핸들에 문제가 있어서 수직선 성분의 취득 결과는 볼 수 없었다.

4) 교차점의 좌표 취득

교차점의 취득은 우선 수평선 성분을 취득하여 임시 저장을 한 후 수직선 성분을 취득해 가면서 수평선 성분이 존재하는 부분을 교차점으로 하여 좌표 값을 취득하려 했다. 이러한 처리에서 예상되는 것은 수평선 성분의 처리된 데이터를 담아 두는 데 소요되는 메모리 문제였다. 그러나 이러한 저장 장소는 수평선이 존재하는 열만을 연속으로 저장하여, 그 좌표값, 즉 x, y좌표만을 알 수 있게 하였고, 수직선 취득시 흑화소의 존재가 확인된 곳만이 교점의 좌표로 교차점 리스트에 재저장되도록 한다.

5) 대각선 성분 취득

일반적인 테이블의 경우 대각선 성분이 존재할 것이므로, 이 부분을 처리해야 한다. 일반적인 테이블의 형태인 경우 대각선이 대부분 가장 왼쪽의 상위 Cell에 존재하는 경우가 많다. 따라서 교차점의 좌표가 취득이 되면, 변화된 좌표를 이은 블럭영역을 Cell로 정의하고, 가장 왼쪽 Cell부터 아래로만, 즉 아래에 존재하는 다음 셀로만 대각선을 취득을 위해 검사를 한다.

여기서 말하는 Cell이란 수직선과 수평선이 만나는 교차점 4개를 연결하여 하나의 box를 형성하고 이러한 영역을 Cell이라 정의한다. 따라서 2개의 수직선을 이루는 2개의 y좌표와 2개의 수평선을 이루는 2개의 x좌표로 이러한 셀 영역을 지정할 수 있다.

3. 실험 결과

8방향 체인 코드 이용할 경우 모든 픽셀에 대한 마스킹을 행해야 했기 때문에 그 처리 시간이 많이 소요되는 문제를 가지고 있었다. 그러나 그 결과는 문자와 테이블을 구별하였으며, 또한 벡터화된 데이터를 가지고 재구성을 할 수 있었다. 그림 5에 그 실행 결과를 보였다. 히스토그램에 의한 실험 결과에서는 그림 6에 보여지듯이 스캔 된 이미지가 단순하고 오차가 적을수록 영상은 깨끗하게 처리가 되며, 또한 영상의 끊긴 부분이 적을 수록 거의 원영상에 가까운 영상이 추출될 수가

있다. 그리고 아직 처리가 되지 못한 부분인 두꺼운 이중선은 최대한 올바른 각도로 스캔을 하거나 어느 정도의 각도 보정된 이미지에 대해서 원영상에 가까운 결과를 얻는다. 다음에 보일 입력 데이터 1과 2의 예는 어느 정도 보정된 이미지를 처리한 것이고, 마지막 입력 데이터 3의 예는 각도 보정을 하지 않은 영상을 처리한 결과이다. Run-length를 이용한 방법을 실험한 결과는 만족스럽지 않았다. 그림 7에 그 결과를 보였다. 그 이유는 문서가 똑바로 입력되지 않았기 때문에 발생된 문제이기도 하고 연속된 흑화소의 수를 임계치로 하였기 때문이다. 이러한 문제는 우선 임계치를 동적으로 할당하도록 하는 해결 방법이 있으며, 아래와 같은 문제가 발생했는데, 각각 그 해결 방법을 생각해 보았다.

●첫 번째로 발생한 것은 실험 대상 문서의 스캔상태가 양호하지 않아서 글자는 물론 테이블의 영상이 끊긴 경우 어느 정도 보정은 할 수 있었으나, 연속된 화소의 수를 기본으로 한 본 알고리즘에서는 그 한계가 있다.

●두 번째로 발생한 문제는 기울기 문제이다. 스캔시 영상의 취득 단계에서 문서가 우로 혹은, 좌로 기울어져 있는 것이 문제였다. 실제 아날로그 상태에선 문제가 되지 않는 이런 기울기가 디지털 영상에서는 커다란 문제가 되었다. 즉, 연속된 화소의 체크에 한계가 있었다. 열의 선 성분임에도 불구하고 기울기 때문에 잘려져서 한 열 위로 확장된 선 성분이 잡히거나, 화소의 수가 모자라서 선 성분임에도 불구하고 삭제가 되는 경우가 있었다. 이러한 문제의 해결을 위해 다음과 같은 해결 방법으로 개선할 예정이다. 먼저 두 번째 문제를 보완하려면, 인식 알고리즘을 사용하기 전에 hough 변환 같은 알고리즘을 사용하여 영상을 바로 잡아 주는 것이다. 그러나 이런 한 변환 알고리즘을 사용한다면 전체적인 프로그램의 실행 시간이 길어지므로 바람직하지 못하다고 생각된다. 기울기를 보정하지 않고, 스캔된 본 영상에서 테이블 인식을 시도하려 한다. 취득된 수평선, 수직선 부분의 시작과 끝 좌표를 취득하여 그중 수평선, 수직선 성분이 가장 긴 부분의 좌표만을 취득하여 이 문제를 해결한다. 그리고 나서, 첫 번째 문제와 같은 경우 어느 정도 문제를 해결할 수 있다. 스캔 영상에서 테이블영역의 좌표 값을 얻어내는 것을 최종 목표로 잡았는데, 이런 한 목표로에 다다를 수 있는 정도로 보정할 수 있음을 알았다. 화소의 run의 값을 조절하여 어느 정도 보정 효과를 얻었지만, 만족스럽지 못했다. 따라서 이 부분의 처리는 수평선과 수직선의 시작점과 끝점의 좌표를 가지고 그 사이의 끊긴 부분을 잇는 것으로 처리할 수 있다.

1) 8방향 체인 코드를 이용한 결과

1 Pixel Wide Image

M_1	M_2	M_3
4.630 424	0.896 186	103.159 993
3.006 356	0.937 627	28.497 881
3.853 814	0.925 847	39.431 143
0.760 393	1.000 000	0.647 775

입력데이터

결과 데이터

<그림 5. 8방향 체인코드를 이용한 결과>

2) 히스토그램을 이용한 테이블 벡터화 결과

1	A	J
2	B	K
3	C	L
4	D	M
5	E	N
6	F	O
7	G	P
8	H	Q
9	I	R
0		1

a) 입력 데이터 1

b) 결과 데이터 1

위의 영상은 가장 상태가 양호한 것을 처리한 결과이다. 원영상과 거의 오차가 없는 것을 볼 수 있다.

	SNRin	SNRout
scalar quantization	7.2	6.0
vector quantization	11.5	8.9

TABLE II
UNIFORM BIT ALLOCATION VERSUS OPTIMAL BIT ALLOCATION

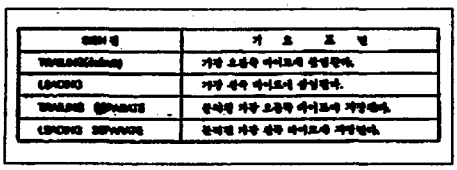
	SNRin	SNRout
uniform bit allocation	7.8	6.5
optimal bit allocation	11.5	8.9

(a) Uniform bit allocation (b) Optimal bit allocation

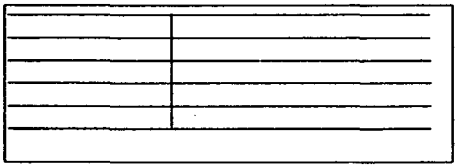
c) 입력 데이터 2

d) 결과 데이터 2

위의 영상은 보정되지 않은 스캔된 영상 그대로를 처리한 결과이다. 두꺼운 이중 외곽선과 세로선의 처리가 완벽하지 않은 것은 원영상이 기울어져 있기 때문이다.



e) 입력 데이터 3

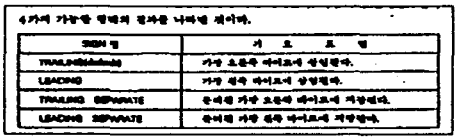


f) 결과 데이터 3

<그림6. 히스토그램을 이용한 결과>

위의 영상에서 양쪽 외곽선이 나타나지 않은 이유는 선성분이 자주 끊겨 올바른 선으로 인식되지 않기 때문이다.

3) Run-length에 의한 결과



입력데이터 1

결과데이터 1

<그림 7. Run-length를 이용한 결과>

4. 결론 및 연구 방향

8방향 체인 코드를 이용한 방법은 문서 내의 테이블을 완벽하게 처리를 하여 테이블의 재구성까지 가능하지만 그 처리 시간이 너무 많이 걸린다. 히스토그램을 이용한 방법은 5가지 종류의 테이블에 적용하여 얻은 결론은 일반적인 접근 방법은 속도의 향상을 가져올 수는 있으나 원래 테이블을 충실하게 표현하지 못한 단점이 있었고, 테이블을 구성하는 수직선과 수평선의 위치가 원화상과는 조금씩 다르게 되는 것이 있으며, 또한 이중으로 된 외곽선의 경우는 선이 가늘지 않으면 이미지는 약간 기울어진 상태이므로, 다른 선들과 하나로 포함되어 이중라인 대신에 단일선으로 나올 수가 있다. 또한, 복잡하거나 복수개의 테이블이 문서 이미지에 저장되어 있을 경우에 필진 더 섬세한 처리 과정이 필요하다는 것이다. 원영상과 거의 유사한 결과를 얻기 위해 더 연구되야할 부분은 이중 외곽선의 올바른 처리 방법과 영상에서 테이블의 기울어진 각도가 커도 어느 정도 무시하고 결과를 신뢰할 수 있도록 하는 것과 마지막으로 올바른 과표의 값을 산출해 내는 것이다. 또 속도를 향상시키기 위한 방법은 이진화의 과정을 거친 데이터를 사용하지 않고 스캐너로 받아들인 원 이미지를 그대로 사용하는 것이 현 속도에 비해 50%정도의 향상을 가져올 것 것이다. 더 수정, 보완되어야 할 점도 많지만 본 알고리즘은 처리 속도와 메모리 효율성 면에서 좋다고 생각된다. 완벽한 사선 처리를 위해 run-length를 이용한 테이블 처리 알고리즘과 결합하여 수정 보완된 알고리즘을 계속 연구할 예정이다. Run-length를 이용한 방법의 결과에선 많은 문제가 있었으나, 이는 프로그램 구현시 좀 더 상세히 세부 처리 루틴을 구현하지 않은 결과였다. 앞으로 이러한 프로그램을 수정 보완하고, 많은 테이블 영상을 가지고 본 알고리즘을 검증하는 것이다. 여기서 좀 더 보완되어야 할 문제는 전체 문서 내에서 문자 영역과 테이블 영역 내의 문자 영역 데이터를 문서 인식 알고리즘에 넘겨주는 것이며 앞에서 제시된 히스토그램을 이용한 테이블 인식 알고리즘과 결합하여 알고리즘을 수정, 보완하는 것이다.

5. 참고문헌

[1] H. Asada and M. Brady , "the curvature primal sketch," IEEE Trans. Pattern Analysis Machine Intell., vol. PAMI-8, pp. 2-14, Jan. 986.

[2] F. Mokhtarian and A. Mackworth, " Scale-based description and recognition of planar cuurves and two-dimaensional shapes," IEEE Trans. Pattern Analusis Machine Intell., vol. PaMI-8, pp. 34-43, Jan. 1986.

[3] N. J. Naccache and R. Shinghal, " SPTA : A proposed slgorithm for thinning binary patterns," IEEE Trans. Syst, Man, Cybern., vol. SMC-19, no. 2, pp.409-418, May, 1984.

[4] 서강대학교 산업기술 연구소 " 복합 정보의 자료 파일 처리 기법에 관한 연구," 한국전자통신연구소.

[5] BEN YACOUB Souheil, Jolion jean-Michael "On curv approximation and hierarchical Hough transform" IEEE Trans. Pattern Anal. Mach, Intell., vol. PAMI-6, no. 7, pp. 386-405, July 1984.

[6] 지용화, 박래홍, "여러 스케일을 이용한 다각형 근사화," 신호처리 합동 학술대회 논문집, 제 4권 제 1호, pp. 470-473, 1991.

[7] J. J. Rodriguez and J. K. Aggarwal, "Matching aerial images to 3-D terrain maps, " IEEE Trans. Pattern Analysis Machine Intell., vol. PAMI-12, pp. 1138-1149, Dec. 1990.

[8] 한국과학기술원 전산학과 김인중, 서장원, 김형진 "Run-length code와 Hoough Transform을 이용한 Vectorizer "., 제 7회 영상 처리 및 이해에 관한 워크샵., pp.53-58, 1995.

[9] R. C. Gonzalez and P. Wintz, Digital Image Processing. Addision-Wesley, second ed..

[10] F. Y. Feng and T. Pavlidis, "Finding Vertices in a picture," Computer Graphics Image Processing, vol. 2, pp. 103-117, 1973.

[11] A. Rosenfeld and E. johnston, "Angle detection on digital curves," IEEE Trans. Computers, vol. C-22, pp. 875-878, Sept. 1973.