문서 영상 내의 테이블 벡터화 연구

김 우 성[†] · 심 진 보[††] · 박 용 범[†††] · 문 경 애[††††] · 지 수 영[††††]

요 약

본 논문에서는 문서인식 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.

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 수평선 성분과 수직선 성분을 추출하여 얻어진 교차점을 이용해 대각선 성분을 찾아내는 방법이 있다. 또한 규칙성을 이용한 테이블 추출 방법은 테이블을 구성하는 수직선 성분과 수평선 성분의 규칙성을 찾아내 이를 이용하여 테이블을 벡터화 시킨다. 본 논문에서는 문서 영상 내의 테이블을 효율적으로 벡터화하기 위한 방법으로 규칙성을 이용한 방법을 제안한다.

Research on the Table Vectorization in the Document Image

Woo Sung Kim[†] · Jin Bo Shim[††] · Yong Bhum Park[†††] · Kyeong Ae Moon[††††] · Soo Yeong Ji[††††]

ABSTRACT

In this paper. we develop an efficient algorithm which vectorize the table input for mixed document recognition system. It is necessary to separate character and line for recognizing the character in the table. For recognizing table, we have to recognize the character which is blocked by table line and develop the efficient vectorization method for table line.

For vectorizing table, we develop several methods. The first method is to extract table line part using 8-direction chain codes. The second method is to extract horizontal and vertical lines using histogram of lines. The third one is to extract diagonal lines of table by using the cross points of horizontal and vertical lines. Finally we also develop the table vectorization method which finds the regularity characteristics of horizontal and vertical lines composing table. In this paper, we suggest a regularity method for efficient table vectorization.

1. 서 론

본 논문은 문서 인식 시스템에서 정확한 문서 인식을 위해, 테이블 입력의 효율적인 처리 방법을 연구한다. 이러한 처리 알고리즘으로 본 논문에서 제시한 알고리즘은 기존의 3가지 방법과 제안한 히스토그램을 응용한 규칙성을 이용한 방법을 포함한 4가지 방법이다[1, 2, 3, 4]. 본 논문에서는 기존의 방법들과 제안한 규칙성을 이용한 방법들의 실험 결과를 보이고,

※이 논문은 1995년도 교육부지원 한국학술진흥재단의 지방대육성과제 학술조성비에 의하여 연구되었음.

† 정 회 원:호서대학교 컴퓨터공학과
†† 준 회 원:호서대학교 컴퓨터공학과 석사과정
††† 정 회 원:단국대학교 전자계산학과 교수
†††† 정 회 원:시스템공학연구소 영상처리 인공지능 연구부

논문접수:1996년 3월 15일, 심사완료:1996년 5월 30일

장단점을 비교한다. 첫 번째 방법은 8방향 체인 코드를 이용한 테이블 영상 벡터화 방법이다. 두 번째 방법은 테이블 영상의 수직, 수평의 투영을 이용한 히스토그램을 사용하여 각 각 수직선 성분과 수평선 성분을 추출하고, 그 교차점을 구하여 교차점 정보를 가지고 대각선 성분을 추출한다. 세 번째 방법은 흑화소의 런(run)의 길이를 이용하여 수직선 성분과 수평선 성분을 추출하여 교차점을 찾고, 대각선 성분을 찾아낸다. 네 번째 방법은 문서 영상에서 존재하는 테이블을 구성하는 수평선 성분과 수직선 성분들이 테이블을 구성하는 특징적인 규칙성을 추출하여 벡터화한다.

8 방향 체인 코드를 이용한 방법은 처리 시간이 많이 소요된다는 단점이 있는데, 이는 문서 영상을 이루는 모든 화소를 조사하여, 화소의 정보를 이용하기 때문이다. 그러나 그 결과는 문자와 테이블을 확실히 구별하였으며, 벡터화를 함으로써 재구성 할 수 있었다. 히스토그램(histogram)을 이용한 방법과 Run-length를 이용한 방법은 8 방향 체인 코드를 이용한 방법의 문제인 처리 시간의 단축을 위해 연구되었기 때문에, 그 처리 내용을 단순화하였다. 즉 히스토그램을 이용한 방법과 Run-length를 이용한 방법은 문자를 이루는 화소의 수와 테이블을 구성하는 수평선, 수직선의 화소수의 차이를 이용한 것이다. 모든 화소의 정보를 이용하는 것이 아니므로 처리 시간은 단축되어졌다. 히스토그램을 이용한 방법은 문자와 테이블의 선 성분을 흑화소의 누적 정도로 우선 테이블을 구성하는 선 성분이 있을 만한 곳을 탐색하고, 탐색된 영역을 재 탐색한다. Run-length를 이용한 방법은 흑화소의 연속된 수를 이용하여 문자와 선 성분의 런의 차이를 임계치로 하여 선 성분과 문자를 구별한다. 그러나 이러한 알고리즘은 영상의 입력 형태에 의존한다. 즉 영상의 심한 굴곡이 처리 결과에 나쁜 영향을 미친다. 제안한 규칙성을 이용한 방법은 문서 내의 테이블을 이루는 선 성분의 규칙성을 이용하여 벡터화하는 모델을 제시한 것이다.

2. 테이블 영상의 벡터화

알고리즘의 검사를 위해서 실험 데이터는 문서 내의 테이블 영상만을 대상으로 하였다. 이는 테이블 외곽의 문자보다는 테이블 내의 문자와의 구별에 더 많은 중점을 두었기 때문이다. 그리고 테이블 내의 문자를 구별할 수 있다면, 테이블 외곽의 문자를 구별하는 것은 어려운 것이 아니므로, 실험 데이터의 크기를 줄여 실험에 사용하였다.

2.1 8 방향 체인 코드를 이용한 벡터화

1) 전처리

8 방향 체인 코드 알고리즘을 사용하기 전에, 그 전처리로써 그레이 레벨 값을 갖는 영상 데이터를 흑, 백의 두 값을 갖는 이진 데이터로 만들었다. 배경에서 객체를 추출하기 위한 그레이 레벨의 적당한 임계치를 선정하는 것이 영상 처리에선 중요하다. 이상적인 경우에 히스토그램은 배경과 객체를 나타내는 두 정점 사이의 깊고 날카로운 골짜기를 가지므로 이 골짜기의 바닥 값으로 선택되어질 수 있다. 이러한 임계치를 선정하는 것은 골짜기가 평범하고 넓으며, 노이즈가 많거나, 또는 두 정점이 두드러지게 높아서 종종 골짜기를 발견할 수 없어 어려울 때가 있다. 이진화를 위한 임계치 선택에서 Otsu 알고리즘은 최적의 임계치를 자동적으로 선택하는 것에 접근한다[5]. Otsu 알고리즘은 그레이 레벨 히스토그램을 정규화하고 확률 분포처럼 간주하여, 주기적인 시간 변수를 사용하지 않고 이를 통해 통제되지 않는 임계치를 사용하여 이진화를 수행한다. 이것은 지역적 최소치 T0을 자동적으로 설정한다. 이렇게 처리된 이진화 데이터는 다음 전처리를 거치게 되는데, 이것이 세선화 과정이다. 세선화 과정은 체인 코드를 사용하기 위한 것이다. 체인 코드를 사용할 때 주변에 많은 흑화소를 둔다면, 특정한 특징점을 찾는데 많은 조건이 필요하기 때문에 이를 간소화하기 위해서이다. 또한 실질적인 문자와 테이블의 골격만을 처리하기 위한 것이다. 이러한 세선화 과정에 SPTA(Safe-Point Thinning algorithm)을 사용한다[6]. SPTA 알고리즘의 특징은 패턴의 연결성이 끊어지는 경우가 있었으나, 본 논문에서 택한 SPTA 알고리즘은 이러한 문제에 강하다. 그리고 심한 평활화를 일으키지 않는다. 더 나아가서 이 SPTA 알고리즘은 가능한 실제 패턴과 유사한 패턴을 재구성하기 위한 충분한 정보를 얻어낸다. 이러한 SPTA의 조건은 아래와 같다.

* N(p)은 〈그림 2〉 또는 〈그림 2〉를 회전시킨 윈도

우 중 어느 하나를 만족시킨다.

* N(p)은 정확히 2개의 검은 4-이웃 화소들을 포함한다.

위의 조건들이 아래 식으로 만족될 경우 왼쪽 윤곽점 P가 안전점이 된다.

㉠ left safe point

$$S_4 = n_0 \cdot (n_1 + n_2 + n_6 + n_7) \cdot (n_2 + n'_3) \cdot (n_6 + n'_5) \quad (1)$$

㉡ right safe point

$$S_0 = n_4 \cdot (n_5 + n_6 + n_2 + n_3) \cdot (n_6 + n'_7) \cdot (n_2 + n'_1) \quad (2)$$

㉢ up safe point

$$S_2 = n_6 \cdot (n_7 + n_0 + n_4 + n_5) \cdot (n_0 + n'_1) \cdot (n_4 + n'_3) \quad (3)$$

㉣ 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
(Fig. 1)Location of mask

*		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
(Fig. 2) Mask for thinning

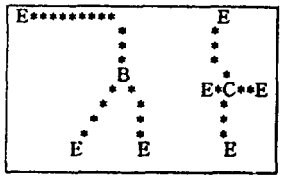
이 알고리즘은 경계점을 제거하기 위하여 미리 정의된 그림 2의 4개의 마스크에 하나라도 부합되면 이 점을 안전점이라 하고 지우지 않는다. 그렇지 않은 점은 flag라 하여 경계점으로 생각하여 지운다. 그림 2의 마스크는 왼쪽 경계를 찾기 위한 마스크이며, 그림 2의 마스크를 회전함으로써 오른쪽, 위쪽, 아래쪽의 경계를 찾는다. 그림 2에서 x, y는 비교 무시이고, *는 흑화소이며 빈 영역은 백화소를 나타낸다. 이러한 과정을 불(Boolean) 표현으로 나타내기 위한 마스크의 위치를 그림 1에 정의하였다. 식 (1), (2), (3), (4)에서 "′"는 보수를 나타내고, 각 불 변수는 흑화소인 경우 참이고, S4, S0, S2, S6이 각각 거짓인 경우 왼쪽, 오른쪽, 위쪽, 아래쪽의 안전점이 된다. 이 알고리즘의 구현은 트리구조 형태의 비교를 통하여 비교 회수를 줄여 수행 시간을 줄일 수 있다.

2) 특징점 추출

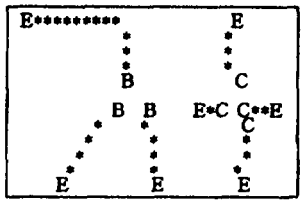
영상 데이터는 SPTA 골격 추출 과정을 거치는 동안 끝점, 분기점, 교차점이 특징점으로 추출된다. 이러한 특징점 데이터는 다음과 같은 조건을 거쳐 특징점으로 추출이 된다. 주변 8점을 원형으로 돌면서 흑, 백화소의 변화가 4번이면 끝점으로 간주하였다. 주변 8점의 변화가 6번이면 분기점으로 간주하였다. 주변 8점의 변화가 8번이면 교차점으로 간주하였다. 이렇게 처리된 예를 그림 3에 보였다.

3) 스트로크 추출

특징점 추출 과정을 거쳐 나온 특징점 데이터를 가지고 특징점과 특징점 사이를 연결하는 화소들의 집합을 한 개의 스트로크로 정의하여 그림 4에서처럼 스트로크를 추출하였다.



(그림 3) 세선화 후 특징점 추출 결과
(Fig. 3) Result of feature extracted after thinning

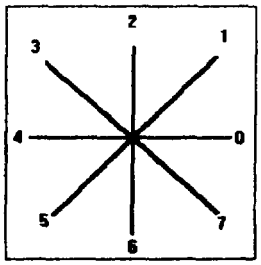


(그림 4) 그림 3의 스트로크 추출 결과
(Fig. 4) Result of Stroke extracted from Figure 3

4) 테이블 추출

스트로크 추출 과정에서 얻어진 데이터를 가지고, 각 스트로크의 호 길이와 시작점과 특징점 사이의 거리를 계산하고, 호의 시작점과 특징점 사이에서 얻어진 거리의 비가 0.9 이상이면서, 흑화소의 개수가 20 이상이면 테이블의 세그먼트로 판별하였다. 구부러진 스트로크일 경우 다각형 근사(Polygonal approximation)기법을 이용하여 직선 세그먼트로 추출하였다[1, 7].

다각형 근사 기법은 이산적인 좌표 값으로 표시되기 때문에 컴퓨터에서는 처리하기 힘들다. 따라서 이러한 이산적인 좌표값 대신에 8방향 체인 코드 이용해 좌표값을 나타내어 처리한다. Liu등은 윤곽선의 각도 변화율인 곡률 $K(t)$를 구해 이 함수의 극소점들 중에서 곡률 값이 양수가 되는 것들을, 극대점들 중 곡률 값이 음수가 되는 것들을 제외한 극점들을 중요점으로 하여 다각형 근사화를 하였다. 곡률은 수학적으로 기울기 각도 $\theta(t)$의 순간 변화율(미분)로 정의된다[2, 8]. 그런데 컴퓨터로 처리하기 위한 윤곽선은 이산적인 좌표값으로 표시되므로 $\theta(t)$대신 체인 코드를 이용한다. 즉 곡률을 구하기 위해서는 먼저 윤곽선을 그림 5와 같은 8-방향 체인 코드로 나타낸다[9]. 8방향 체인 코드의 체인 코드값 0~7은 45°간격으로 기울기 각도 0°~315°를 나타낸다. 이러한 체인 코드는 8방향 코드화에서 오는 불연속을 없애기 위해서 unwrap시켰다[10]. 곡률은 기울기 각도함수의 미분으로 정의되기 때문에 앞서 언급한 unwrap된 체인 코드를 미분함으로써 얻어진다. 그런데 영상의 양자화로 인해 체인 코드값이 이산적이기 때문에 가우스 1차 미분함수와 unwrap된 체인 코드와 convolution함으로써 원하는 평활화된 곡률 함수를 얻는다. 이때 가우스 1차 미분함수의 분산 값에 따라 곡률 함수의 평활화 정도가 변하고 따라서 근사화 되는 점의 수도 변하게 된다.



(그림 5) 8 방향 체인코드
(Fig. 5) 8 direction chain code

2.2 히스토그램을 이용한 테이블 벡터화

히스토그램을 이용한 알고리즘은 그 접근 방법이 일반적 형태의 접근 방법이다. 테이블을 인식하려면 테이블을 이루는 수직선, 수평선 성분들을 구별해 내야 한다. 이 각각의 선 성분들을 구별하기 위한 방법으로, 히스토그램을 이용한 방법에서는 크게 세 가지 단계를 나누어 이루었다.

- 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 대각선 성분을 찾아냈다.

1) 전처리
- 스캐너를 통한 400dpi 영상을 획득함
- 이진화(256 gray-level) 처리 수행
- 전체 영상을 투영
- 수평과 수직의 라인 단위로 흑화소의 수를 평가[11]

2) 수평선의 획득
- 수평축에 존재하는 흑화소의 수가 영상의 수평 길이에 2/3을 넘으면, 수평선이 존재한다고 가정
- 수평선이 존재한다고 가정한 위치에서 문자 성분과 선 성분을 분리

3) 수직선의 획득
- 수직 축에 존재하는 흑화소의 수가 영상의 수직 길이에 1/3을 넘으면, 수직선이 존재한다고 가정
- 수직선이 존재한다고 가정한 위치에서 문자 성분과 선 성분을 분리[12]

4) 수평선과 수직선이 이루는 교차점의 획득
- 수평선 성분과 수직선 성분 처리에서 얻어진 각

성분들의 좌표를 조합하여, 수평선과 수직선 성분의 교차점을 추출

5) 교차점 리스트를 이용한 대각선의 획득

· 대각선의 존재는 수직선 성분과 수평선 성분들의 교차점들 사이에 위치한 것으로 가정하고, 각 교차점들 사이에서 연속적으로 흑화소가 존재하는지 검사

· 존재하면 두 교차점의 좌표를 저장

2.3 Run-Length를 이용한 테이블 추출

2.2 절에서 언급한 처리는 영상의 일반적인 처리였다. Run-length[4, 13, 14]를 이용한 테이블 입력 처리는 지역적인 처리 방법으로 메모리를 최소한 사용하며, 문자와 테이블 선 성분을 구별하기 위해 연구하였다. 일반적인 문서에서 문자와 테이블의 구분은 2.2의 방법에서 언급 한대로 연속된 화소 수의 많고 적음이 있어 가능하다. 연속된 화소의 수가 클수록 테이블 선 성분의 가능성이 높고, 작을 경우는 문자일 경우가 큰 것이다. 문자 자형의 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 문자인 경우 수평선, 수직선 가운데 가장 큰 획의 화소 수는 16을 넘지 않는다는 것이다. 따라서 16개 이상의 연속된 화소를 검사하면 문자와 테이블을 구성하는 선 성분과의 구별이 가능하다는 것이다[15]. 그러나 실제 문서에서는 큰 제목이나, 작은 제목 등 그 크기가 다양하다. 그래서 보다 큰 크기인 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. 또한 영상에 첨가된 여타의 잡영상도 함께 처리할 수가 있었다. 이러한 것은 잡영상의 연속된 흑화소수가 역시 한문자의 수평, 또는 수직을 이루는 한 획보다 작을 경우에 한한 것이다. 즉 일관성이 없는 잡영상을 제거할 수 있었다.

1) 전처리

본 연구에서는 실험 테이블의 종류를 다섯 가지로 정의하였다. 정의한 다섯 가지 문서는 아래와 같다.

①문서 내에 테이블이 두개 존재하는 경우

②이중 외곽선을 포함한 테이블

③대각선을 포함한 테이블

④수평선 성분과 수직선 성분만을 포함한 테이블

⑤기울기가 큰 테이블과 작은 테이블

위와 같은 구분의 이유는 문서 내에 있는 테이블의 수에 상관없는 알고리즘을 생각했기 때문이다. 또한 이중 외곽선을 갖는 특수한 경우는 테이블 작성 시 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.

- 가장 단순한 구조의 수평선, 수직선을 갖는 테이블과 입력 시 입력자의 잘못으로 인한 바르지 못한 입력에 알고리즘 적용 위해 기울어진 테이블 영상을 400dpi 해상도로 취득.
- 취득된 영상 데이터는 처리 과정 시 취득된 그레이 레벨의 영상을 흑화소와 백화소로만 나타낸 이진화 영상으로 변화.
- 취득 영상에서 잡음 제거 및 테이블 선 성분을 이루는 흑화소 가운데 가장 연결성이 좋은 흑화소의 집합만을 남겨 놓기 위함.

2) 수평선 성분 취득

문서 내에 존재하는 어떤 한 문자의 크기를 16×16, 32×32, 64×64로 설정을 하고, 최대 128×128로 가정을 하였다.

- 처리될 영상 파일에서 테두리를 제외한 실질적으로 처리되어야 할 영상 영역만 취득, 그 이유는 이진화 시 처리 시간 문제 때문이며, 실질적인 테이블 영역만을 테이블 영상에서 분해하여 처리시간을 높이기 위함.
- 16, 32, 64, 혹은 128개 이상 연속 화소를 수평선 성분으로 처리함.
- 처음 32개의 연속된 흑화소의 run을 발견하여 그 흑화소의 집합을 테이블을 구성하는 선 성분의 일부로 간주해 그 부분의 이진화 데이터를 재저장, 문자라고 생각되는 화소는 제거.

if(Run_length(n) > th(0))

```
{       line_element;   }
else
{       character_element;}
```

(여기서, Run_length(n)는 연속된 화소의 수이고 th(0)는 임계치이다.)

- 테이블의 선 성분을 찾는 도중에 테이블의 선 성분임에도 불구하고 이진화 시 끊어진 부분 가정하여 보정.
- 현재 나타난 백화소의 성분이 만약 선 성분의 일부이면, 이 부분을 흑화소로 연결하고, 문자 부분으로 판정이 나면 이 부분을 백화소로 저장해 제거.

3) 수직선 성분 취득

- 영상데이터의 수직 처리는 수평선 추출과 같은 방법으로 문자와 테이블의 수직선 성분을 구분함.
- 수평 성분 취득과 같은 방법으로 연속된 흑화소의 수를 16, 32, 64, 128로 그 비교 값을 갖고 취득함.

4) 교차점의 좌표 취득

- 수평선 성분을 취득, 임시 저장 후 수직선 성분을 취득해 가며 수평선 성분이 존재하는 부분을 교차점으로 하여 좌표값을 취득.
- 수평선 성분의 처리된 데이터를 저장에 소요되는 메모리는 수평선이 존재하는 열만을 연속으로 저장하여, 그 좌표값, 즉 x, y좌표만을 알 수 있게 함.
- 수직선 취득 시 흑화소의 존재가 확인된 곳만이 교점의 좌표로 교차점 리스트에 재 저장되도록 함.

5) 대각선 성분 취득

일반적인 테이블의 경우 대각선 성분이 존재할 것이므로, 이 부분을 처리해야 한다.

- 일반적인 테이블 형태인 경우 대각선이 대부분 가장 왼쪽의 상위 셀에 존재하는 경우가 많으므로 교차점의 좌표가 취득이 되면, 변화된 좌표를 연결한 블록영역을 셀로 정의해 가장 왼쪽 셀부터 아래로 존재하는 다음 셀만 검사. (셀이란 수직선과 수평선이 만나는 교차점 4개를 연결하여 하나의 box임)
- 2개의 수직선을 이루는 2개의 y좌표와 2개의 수평선을 이루는 2개의 x좌표로 셀 영역 지정.
- 셀을 구성하는 두 대각선을 잇는 직선의 방정식을 만족하는 좌표에 있는 흑화소의 집합을 찾아 대각선으로 처리.

2.4 규칙성을 이용한 테이블 벡터화 알고리즘

규칙성을 이용한 테이블 벡터화 알고리즘은 테이블을 구성하는 규칙을 생성한다. 기본적인 테이블의 구성은 가로의 선 성분들과 세로의 선 성분들로 구성되어 있다. 이러한 점에 착안하여 다음과 같은 알고리즘을 구현하였다[16].

- 가로축을 구성하는 테이블의 요소인 가로선 성분을 가상의 수직선을 이용하여 흑화소들의 개수가 현저히 적게 나타나는 수직 축을 검사한다.
- 둘째, 세로축을 구성하는 테이블의 요소인 세로선 성분을 가상의 수평선을 이용하여 찾아낸다. 이 방법도 위의 방법과 동일 하나 가로축을 세로축으로 변환하였을 뿐이다.

이 알고리즘은 히스토그램 방법에 비해 검색하게 되는 열이나 행수가 매우 적어 알고리즘은 가로 50, 세로 30인 테이블 크기를 처리할 때 IBM pc 486DX2-66에서 0.05초 걸린다.

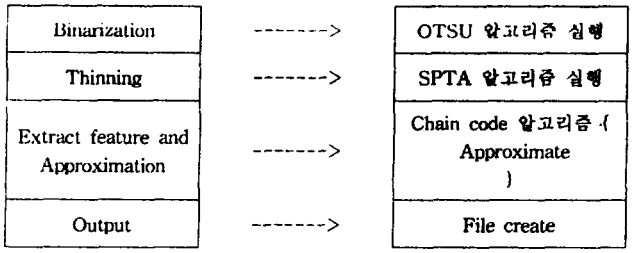
3. 이론적 성능 평가

이 장에서는 2장에서 기술한 알고리즘들의 연산 복잡도를 이론적으로 분석 평가한다.

1) 8방향 체인 코드를 이용한 방법

8방향 체인 코드를 이용한 방법은 처리할 영상을 받아 배열로 저장을 한다. 다음에 이 영상을 전처리 과정에서 otsu알고리즘을 이용하여 Binarization을 행하여 파일로 저장을 하고, 이렇게 처리된 Binary 데이터를 가지고 세선화 과정을 거쳐 처리한다. 이러한 세선화 과정에 SPTA알고리즘을 사용하며, 처리된 데이터를 파일로 저장한다. 그리고 이렇게 처리된 데이터 파일을 가지고 특징점 추출을 하는데, 이때 마스크를 사용하여, 8방향 체인 코드를 가지고 각각의 화소 정보를 재구성하며, 역시 이를 파일로 저장한다. 이때 저장된 파일은 영상의 파일이 아닌 8방향 체인 코드를 갖는 파일이다. 이러한 8방향 체인 코드를 갖는 파일을 가지고 각 각 수평선 성분과 수직선 성분

알고리즘	OTSU	SPTA	CHAIN CODE(다각형 근사화알고리즘)
연산복잡도	O(Level)^2	O((col-1)×(row-1)×64)	O((row×col)×(영상 화소수))



(그림 6) 체인코드를 이용한 벡터화 방법의 연산복잡도
(Fig. 6) Complexity of vectorization method for chain code

을 찾아 벡터화하며, 이 과정에서 다각형 근사화 알고리즘을 적용하여, 각 테이블을 구성하는 스트로크 가운데, 네 군데의 모서리를 찾아, 직선으로 근사화하여 테이블을 구성하는 구성 성분인가를 결정하게 된다. 테이블의 일부라고 판정이 되면, 이를 벡터화한다. 이러한 모서리부분의 근사화는 글자의 "ㄱ"이나 "ㄴ" 혹은 "ㄹ"의 각 굴곡부분과 같기 때문이다. 사용된 알고리즘의 연산 복잡도는 다음과 같다.

2) 히스토그램을 이용한 방법

히스토그램을 이용한 방법은 전체 영상을 배열로 저장하고 수평선 성분과 수직선 성분을 추출하기 위해서 수평과 수직으로 각 각 한 번씩 스캔하게 된다. 수평과 수직을 스캔할 때 걸리는 시간이 알고리즘의 수행에 있어 대부분을 차지하게 된다.

⊙O(row×col) … 히스토그램을 이용한 방법의 연산 복잡도

```
입력의 전처리부
    {
    처리할 영상을 이진 영상으로 만듦;
    영상 전체에서 가장 우측과 좌측에 존재하는 화소와 가
    장 상단과 하단에 위치하는 화소를 추출하여 그 화소들
    의 좌표 안에서 새로운 영역을 선택;
    히스토그램으로 흑화소의 수를 추출;
    }
수평선 처리부
    {
    세로축 좌표의 반복문
        {
        현 세로축의 양 끝 화소를 추출;
        세로축에서 존재하는 전체 화소의 수를 검사;
        화소의 수가 임계치를 넘는 가를 검사;
            /*임계치는 수평 전체 길이의 1/3*/
            임계치를 넘는 수평성분에 대해 연속된 화소의
            개수를 재검사;
            검사된 연속 화소의 개수 중 최대의 수가 글자의
            크기를 넘으면, 그 화소들의 집합은 수평선을 이
            룬다고 가정하고 수평선의 좌표를 기억;
            근접한 연속 수평선이 발견되면 가장 나중에 발
            견된 수평선만을 기억;
            }
    }
  /*수직선 처리를 위해 수평선 처리부와 같은 루틴을 다시
  실행*/
  {
    가로축 좌표의 반복문
    {
        /*임계치는 수직 전체 길이의 1/3*/
        }
    }
대각선 처리부
    {
    테이블을 구성하는 첫 번째 좌표와 두 번째 수직성분과
    두 번째 수평성분이 만나는 좌표 사이에 대각선이 존재
    하는 지를 검사;
    두 점 사이에 기울기를 계산하여 일차원 방정식을 만들
    어 연속된 화소의 수를 평가;
    연속된 화소들이 선 성분이라고 평가되면 대각선의 기
    울기와 대각 선분을 이루는 두 양 끝점의 좌표를 저장;
    }
최종 처리부
    {
    각 처리 루틴에서 추출된 정보를 파일로 저장하고, 처리
    된 결과 영상을 새로운 파일로 저장;
    }
```

3) Run-length를 이용한 방법

Run-length를 이용한 방법은 처리할 영상을 배열에 저장하고, 배열의 순차적인 검색을 위하여, 두 번의

루프를 수행한다. 두 번의 루프 연산 이후엔 리스트가 생성되어, 이 리스트를 검색하여 선 성분을 추출한다. 이때 Run-length를 이용한 방법은 수평선 성분과 수직선 성분을 추출하기 위해서, 각 각 수평과 수직 스캔을 한다. 즉, 각 스캔을 할 때 한 번씩 두 번의 루프를 수행하는데, 이는 연산 복잡도에 커다란 영향을 주지 않는다.

⊙O(row×col) … Run-length를 이용한 방법의 연산 복잡도

이러한 연산 복잡도가 나타내듯이 처리할 영상의 크기에 따라 연산 시간의 변화가 크다. 즉 처리할 영상이 작을 경우엔 적은 연산 시간이 소요되며, 클 경우엔 많은 연산 시간이 필요하다. 그러나 1024×768의 영상을 실험한 결과 2.5초가 걸린다.

```
  /*처리할 영상을 획득*/
영상 획득 반복문으로 열 크기 비교{
  칼럼 크기 비교{
            획득된 영상 가운데 첫 번째 흑화소의 좌표 추출;
            획득된 영상 가운데 마지막 흑화소의 좌표 추출;
            /*순수 테이블 영역과 보다 가까운 영상 영역을
            만들기 위한 변수 획득*/
            }
     }
     /*성분 추출을 위한 반복문*/
  열 크기 비교{
            칼럼 크기 비교{
            /*임계치를 주고 연속된 흑화소를 검색, 리스트
            를 생성*/
  }
```

위의 루틴을 수직선 성분 추출과 수평선 성분 추출을 위해 각 각 전체 루틴에서 두 번 실행하게 된다.

4) 규칙성을 이용한 방법

수평선, 수직선 성분의 추출을 위해서 이미지 전체를 검색하므로 O(col×row)가 기본이지만 여기에 연결 리스트로 저장하기 위한 처리가 추가되므로,

⊙O(row×col×20) … 규칙성을 이용한 방법의 연산 복잡도

위의 연산 복잡도에서 20은 각 row당 검색하게 되는 최대 화소의 수이다. 처리속도는 규칙성을 이용한 방법이 가장 우수한데, 그 이유는 검색 시 전체 화상을 검색하는 것이 아니라 흑화소가 적게 존재하는 부분 즉 선 성분을 지나치게 되는 부분만이 검색되므로, 검색 횟수가 타 방법에 비해 훨씬 적기 때문이다.

4. 실험 결과

8 방향 체인 코드 방법을 이용할 경우 모든 화소에 대한 마스킹을 행해야 하기 때문에 그 처리 시간이 많이 걸리는 문제를 갖고 있다. 그 결과는 그림 6에서 볼 수 있듯이 문자와 테이블을 구별하였으며, 또한 벡터화된 데이터를 가지고 원영상과 같은 테이블을 재구성을 할 수 있었다. 히스토그램에 의한 실험 결과에서는 그림에 보여지듯이 취득한 이미지가 단순하고 오차가 적을수록 영상은 깨끗하게 처리가 되며, 또한 영상의 끊긴 부분이 적을수록 거의 원영상에 가까운 영상이 추출될 수가 있다. 그리고 아직 처리가 되지 못한 부분인 두꺼운 이중선은 최대한 올바른 각도로 스캔을 하거나 어느 정도의 각도가 보정된 이미지에 대해서 원영상에 가까운 결과를 얻는다. 그림 7의 (a), (c), (e)는 입력 영상이다. 그림 7의 (b), (d)의 예는 어느 정도 보정된 이미지를 처리한 것이고, 마지막 그림 7의 (f)의 예는 각도 보정을 하지 않은 영상을 처리한 결과이다. 그림 8의 Run-length를 이용한 방법의 실험 결과는 만족스럽지 않았다. 그것은 문서가 똑바로 입력되지 않았기 때문에 발생된 문제이기도 하고 연속된 흑화소의 수를 임계치로 하였기 때문이다. 이러한 문제는 우선 임계치를 동적으로 할당하는 해결 방법이 있으며, 아래와 같은 문제가 발생했는데, 각각 그 해결 방법을 제시한다. 발생된 문제는 다음과 같다.

- 실험 대상 문서의 스캔상태가 양호하지 않아서 글자는 물론 테이블의 영상이 끊긴 경우
- 기울기 문제

영상의 취득 단계에서 문서가 우로 혹은, 좌로 기울어져 있는 것이 문제였다. 이러한 기울기가 디지털 영상에서는 커다란 문제가 되었다. 즉, 연속된 화소를 검색하는 데 한계가 있었다. 이러한 문제의 해결을 위해 다음과 같은 방법으로 개선한다.

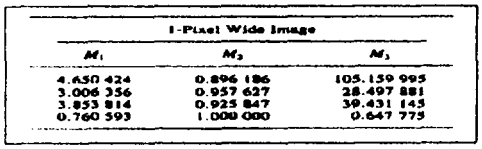
- 화소의 run 값을 조절하여 어느 정도 보정 효과를 얻었지만, 만족스럽지 못했다. 따라서 이러한

문제의 처리는 수평선 성분과 수직선 성분의 시작점과 끝점의 좌표를 가지고 그 사이의 끊긴 부분을 잇는 것으로 처리할 수 있다.

- 기울기를 보정하지 않고, 취득한 원 영상에서 테이블 인식을 시도한다. 취득한 수평선, 수직선 부분의 시작과 끝 좌표를 취득하여 그 중 수평선, 수직선 성분이 가장 긴 부분의 좌표만을 취득하여 이 문제를 해결한다.

규칙성을 이용한 테이블 추출 결과는 그림 9와 같다. 그림 9의 (b)는 Synthetic 입력 데이터(그림 9의 (a))를 처리한 결과이며, 행에서의 열의 수와 열에서의 행의 개수를 얻어낸 것이다.

1) 8방향 체인 코드를 이용한 결과



1-Pixel Wide Image

M_1	M_2	M_3
4.650 424	0.896 186	105.159 995
3.006 356	0.957 627	28.497 881
3.953 814	0.925 847	39.431 145
0.760 593	1.000 000	0.647 775

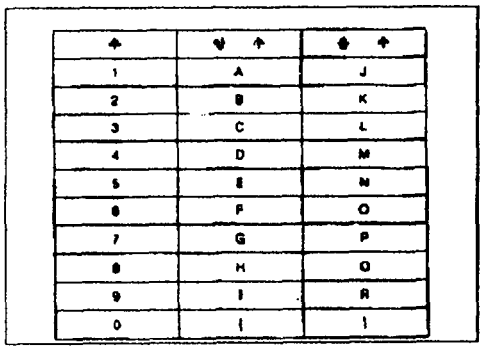
(a) 입력 영상
(a) Input image

(b) 결과 영상
(b) Result image

(그림 7) 8 방향 체인 코드를 이용한 방법의 결과 : (a) 입력 데이터 (b) 결과 데이터
(Fig. 7) Vectorization result using 8 direction-chain code : (a) Iuput Image (b) Result Image

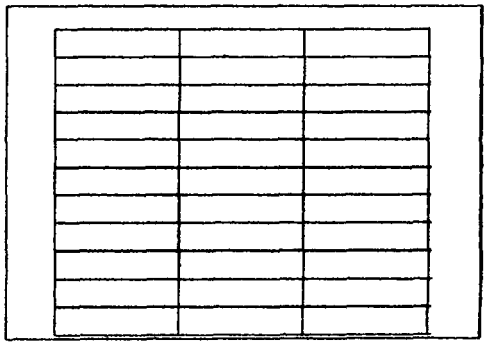
2) 히스토그램을 이용한 테이블 벡터화 결과

- 그림 7의 (b)는 가장 상태가 양호한 것을 처리한 결과이다. 원영상과 비교해 거의 오차가 없는 것을 볼 수 있다.
- 그림 7의 (c)는 보정되지 않은 스캔 영상을 그대로 처리한 결과이다. 두꺼운 이중 외곽선과 세로선의 처리가 완벽하지 않은 것은 원영상이 기울

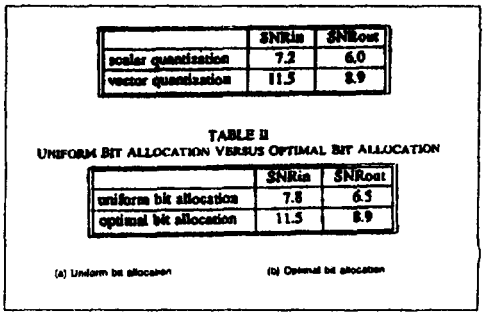


♣	♥ ♠	♦ ♠
1	A	J
2	B	K
3	C	L
4	D	M
5	E	N
6	F	O
7	G	P
8	H	Q
9	I	R
0	(	)

(a) 입력 영상 1
(a) Input image 1



(b) 결과 영상 1
(b) Result image 1



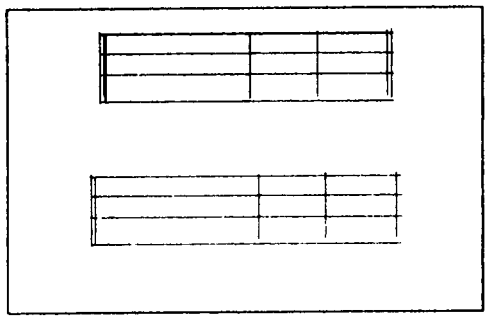
	SNRin	SNRout
scalar quantization	7.2	6.0
vector quantization	11.5	8.9

TABLE II
UNIFORM BIT ALLOCATION VERSUS OPTIMAL BIT ALLOCATION

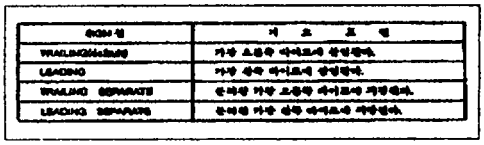
	SNRin	SNRout
uniform bit allocation	7.8	6.5
optimal bit allocation	11.5	8.9

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

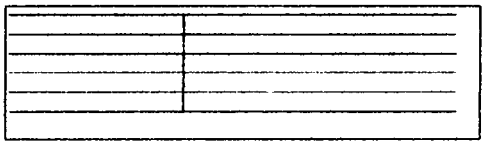
(c) 입력 영상 2
(c) Input image 2



(d) 결과 영상 2
(d) Result image 2



(e) 입력 영상 3
(e) Input image 3



(f) 결과 영상 3
(f) Result image 3

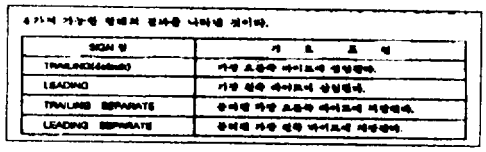
(그림 8) 히스토그램을 이용한 방법의 결과 : (a) 입력 영상 1 (b) 결과 영상 1 (c) 입력 데이터2 (d) 결과 데이터2 (e) 입력 영상 3 (f) 결과 영상 3

(Fig. 8) Result using histogram method : (a) Input image1 (b) Result image1 (c) Input image2 (d) Result image2 (e) Input image3 (f) Result image3

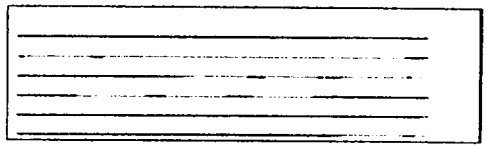
어져 있기 때문이다.

- 그림 7의 (f)는 양쪽 외곽선이 나타나지 않았는데, 이유는 선 성분이 자주 끊겨 올바른 선으로 인식되지 않았기 때문이다.

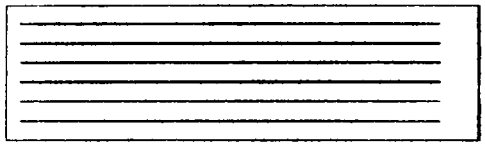
3) Run-length에 의한 결과



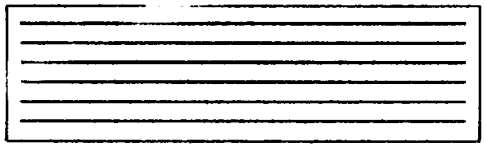
(a) 입력 영상
(a) Input image



(b) 결과 영상 1
(b) Result image 1



(c) 결과 영상 2
(c) Result image 2



(d) 결과 영상 3
(d) Result image 3

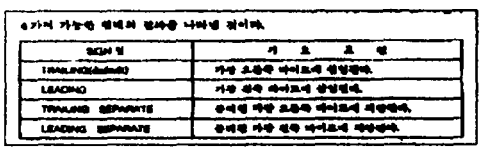
(e) 결과 영상 4
(e) Result image 4

(f) 결과 영상 5
(f) Result image 5

(그림 9) Run-length를 이용한 방법의 결과 : (a)입력 영상 (b)결과 영상 1 (c)결과 영상 2 (d)결과 영상 3 (e)결과 영상 4 (f)결과 영상 5
(Fig. 9) Result using Run-Length method : (a) Input image (b) Result image 1 (c) Result image 2 (d) Result image 3 (e) Result image 4 (f) Result image 5

- 그림 8의 (b)는 스캔된 입력 영상(그림 8의 (a))을 처리한 결과로써 영상 내에서 테이블을 구성하는 수평선 성분만을 추출한 결과이다. 실험 데이터인 그림 8의 (a)는 영상 취득 후 아무런 보정을 하지 않은 영상 데이터이다.
- 그림 8의 (c)는 수평선 선분이 검색된 모든 열을 그룹화 하여 얻어낸 영상이다.
- 그림 8의 (d)는 그림 8의 (c)에서 보는 바와 같이 임계치보다 작은 영역은 잡히지 않고 재구성되지도 않았다. 이러한 점을 수정하여 얻은 결과이다.
- 그림 8의 (e)는 수직선 성분이 검색된 모든 열을 그룹화한 영상이다.
- 그림 8의 (f)는 수평선, 수직선 성분을 합하여 최종 벡터화로 재구성한 결과이다.

4) 규칙성을 이용한 방법의 실험 결과



(a)입력 영상
(a) Input image

```
Wait a minutes.....

start_x = 17          start_y = 24
end_x = 524           end_y = 164

** vertical **
  0   0 -> 17  30 -> 215  5 -> 521
  9 -> 522 30 -> NULL
** horizontal **
  0   0 -> 24   2 ->  80  2 -> 107
  2 -> 135  2 -> 162  2 -> NULL
  /* 연결 리스트에 저장되는 정보 중 앞의 숫자는
     좌표값, 뒤의 숫자는 그 화소가 나온 빈도 수이다. */
```

(b)입력 영상을 처리한 결과 데이터
(b) Processing result of input data

(그림 10) 규칙성을 이용한 방법의 결과 : (a)입력 데이터 (b)입력 영상을 처리한 결과 데이터
(Fig. 10) Vectorization result using rule-based method : (a) Input data (b) Processing result of input data

5) 규칙성을 이용한 방법과 기존방법의 실험 결과 비교

〈표 1〉 벡터화 방법 실험 결과 비교
〈Table 1〉 Comparison of vectorization methods

알고리즘	처 리 속 도	벡터화 정보의 저장 시 소요되는 메모리 크기 (원영상과의 비율)
8방향 체인코드 방법	각 흑화소 단위로 방향성을 검사하고 검사한 화소에 대한 속성들을 저장하기 때문에 처리 시간이 길다.	3.730%
히스토그램을 이용한 방법	수평과 수직으로 투영하여 얻은 히스토그램으로 영상에서의 테이블 정보를 추출하므로 처리시간이 비교적 빠르다.	0.153%
Run-Length를 이용한 방법	영상에서의 흑화소를 수평과 수직으로 런을 추출하기 때문에 일일이 흑화소를 추적하여 처리하지만 실제 실험 영상을 가지고 실험한 결과, 비교적 처리 속도가 빠르다.	0.019%
규칙성을 이용한 방법	수평과 수직으로 투영한 정보를 이용하여 가상의 수직선과 수평선을 사용해 수평과 수직으로 테이블을 구성하는 선의 개수만을 파악하므로 처리시간이 가장 빠르다.	0.013%

- 위의 결과는 실험 영상을 각각의 알고리즘으로 처리 시 나온 결과의 평균적 특징임
- 539×189의 영상에서의 전처리와 재구성을 제외한 순수 알고리즘의 처리 속도는 체인 코드를 이용한 방법에서 5분, 히스토그램을 이용한 방법에서 1.5초, Run-Length 방법에서 1.7초, 규칙성을 이용한 방법에서 0.5초 걸림

5. 결론 및 앞으로의 연구 방향

8 방향 체인 코드를 이용한 방법은 문서 내의 테이블을 완벽하게 처리를 하여 테이블을 재구성하지만 그 처리 시간이 너무 많이 걸린다. 이는 영상을 이루는 화소의 정보를 하나 하나 취득하여 문자와 테이블의 선 성분을 구별하였으며, 이를 토대로 원 영상의 테이블과 같은 테이블을 재구성하기 위한 벡터화 처리 때문에 소요되는 시간이다. 히스토그램을 이용한 방법을 5가지 종류의 테이블에 적용하여 얻은 결론은 속도의 향상을 가져올 수는 있으나 원래 테이블을 충실하게 표현하지 못한 단점이 있었고, 테이블을 구성하는 수직선과 수평선의 위치가 원 영상과는 조금씩 다르게 변화된 것이 있으며, 또한 이중 외곽선의 경우는 선이 가늘지 않으면 영상 자체가 약간 기울어진 상태가 되므로, 다른 선들과 하나로 중첩되어 이중 라인 대신에 단일선으로 나올 수가 있다. 또한, 복잡한 복수 개의 테이블이 문서 이미지에 저장되어 있을 경우에 훨씬 더 섬세한 처리 과정이 필요하다. 원 영상과 거의 유사한 결과를 얻기 위해 더 연구되어야 할 부분은 이중 외곽선의 올바른 처리 방법과 영상에서 테이블의 기울어진 각도가 커도 어느 정도 무시하고 결과를 신뢰할 수 있도록 하는 방법 및 올바른 좌표값을 산출해 내는 것이다. 처리 속도를 향상시키기 위한 방법은 이진화의 과정을 거친 데이터를 사용하지 않고 스캐너로 받아들인 원 영상을 그대로 사용하는 것이 기존 처리 속도에 비해 50%정도의 향상을 가져올 것이다. 더 수정 보완되어야 할 점도 많지만 히스토그램을 이용한 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. 마지막으로 본 논문에서 제안한 규칙성을 이용한 방법은 처리 속도 면에서 가장 뛰어나며 메모리의 효율성 면에서도 히스토그램 방법과 동일하게 나타났다. 따라서 규칙성을 이용한 방법으로 테이블의 표현을 단순화하고 더 빠른 벡터화 방법을 계속 연구할 예정이다. 앞으로의 연구 방향은 이 프로그램을 수정, 보완하여 더 많은 테이블 영상 데이터에 규칙성 알고리즘을 적용하여 검증하는 것이다. 여기서 좀 더 연구되어야 할 문제는 전체 문서 내에서 문자 영역과 테이블 영역 내의 문자 영역 데이터를 문서 인식 알고리즘에 넘겨주는 것이며, 앞에서 제시한 히스토그램을 이용한 테이블 벡터화 알고리즘과 결합하여 알고리즘을 수정, 보완하는 것이다. 본 논문에서 제안한 규칙성을 이용한 테이블 처리 방법은 앞으로 한 화소를 비트 단위로 처리하여 전체 영상의 크기를 더 줄여 입출력 시 소요되는 시간을 더욱 단축시켜야 할 것이며, 재구성 시 좌표 정보만으로 구성하는 것이 아니라 HTML 같은 언어를 이용하여 표현을 하는 것이다.

참 고 문 헌

[1] 서강대학교 산업기술 연구소, "복합 정보의 자료 파일 처리 기법에 관한 연구", 한국 전자통신 연구소, pp. 40-42 1993.

[2] 지용화, 박래홍, "다단계 다각형 근사화 기법," 전자공학논문지, 제 29권 B편, 제 5호, pp. 46-54, 1992년 5월.

[3] S. Chandran and R. Kasturi, "Structural Tabulated Data," *IEEE ICDAR*, pp. 516-519, Oct. 1993.

[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison Wesley, pp 354-356, 1993.

[5] N. Otsu, "A threshold selection method from gray-level histograms", *IEEE Trans. Systems, Man, and Cybern*, vol SMC-9, No.1, pp. 62-66, Jan. 1979.

[6] N. J. Naccache and R. shinghal, "SPTA : A proposed algorithm for thinning binary patterns," *IEEE Trans. Syst, Man, Cybern.*, vol SMC-19, No. 2, pp. 409-418, May 1984.

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

[8] F. Mokhtarian and A. Mackworth, "Scale-based description and recognition of planar curves and two-dimensional shapes," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. PAMI-8, pp. 2-14, Jan. 1986.

[9] J. A. saghri and H. Freeman, "Analysis of the precision of, generalized chain code for the rep-

resentation of planar curves," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. PAMI-3, pp. 533-539, Sep. 1981.

[10] 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.

[11] Moon-soo Chang, Sun-Mee Kang, Woo-Sik Rho, Heok-Gu Kim, and Duk-Jin Kim, "Improved binarization algorithm for document image by histogram and edge detection," *IEEE ICDAR*, pp 636-639, Aug. 1995.

[12] Y. Lu and A. C. Tisler, "Gray Scale Filtering for Line and Word Segmentation," IEEE ICDAR, pp 648-651, Aug. 1995.

[13] I. Pitas, *Digital Image Processing Algorithms*, Prentice Hall, pp 184-185, 1993.

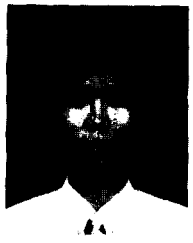
[14] T. Pavlidis, "A Vectorizer and Feature Extractor for Document Recognition", Computer Vision, Graphics, and Image Processing 35, pp 111-127, 1986.

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

[16] K. Itonori, "Table structure recognition based on textblock arrangement and ruled line position," *IEEE ICDAR*, pp 765-768, Oct. 1993.

김 우 성

1980년 서강대학교(공학사-전자공학)
1983년 미국 Texas A&M 대학교(공학석사-컴퓨터공학)
1993년 서강대학교(공학박사-전자공학)
1984년~1985년 한국통신 연구원
1985년~1987년 한국전자통신연구소 연구원
1987년~현재 호서대학교 부교수
1995년 3월~1996년 2월 시스템공학연구소 초빙연구원
1996년 3월 한국과학재단 지정 호서대 반도체 장비 국산화연구센타 연구 2부장
관심분야: 패턴인식, 영상처리, 신경회로망

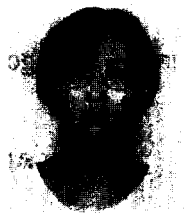


심 진 보

1992년 동양 공업 전문대학 전자과 졸업
1994년 호서대학교 공과대학 컴퓨터공학과 졸업(학사)
1996년 호서대학교 공과대학 대학원 컴퓨터공학과 4학기 재학 중
관심분야: 영상처리, 패턴인식, 인공지능, 멀티미디어

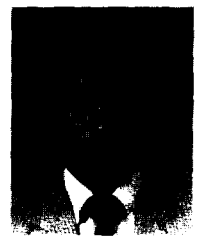
박 용 범

1985년 서강대학교 전자계산학과 졸업(학사)
1987년 미국 폴리텨대학 전산학과(석사)
1991년 미국 폴리텨대학 전산학과(박사)
1991년 미국 NewYork Polytechnic Univ.
1992년 현대전자 연구소 선임연구원
1993년~현재 단국대학교 전산학과 전임강사
관심분야: 뉴럴네트, 음성 및 영상처리



문 경 애

1985년 충남대학교 계산통계학과 졸업
1988년 2월 충남대학교 계산통계학과 석사과정 졸업
1994년 충남대학교 전산학과 박사과정수료
1988년~1991년 충남대학교 전산학과 조교 및 시간강의
1991년~현재 시스템공학연구소 인공지능연구부 근무
관심분야: 패턴인식, 컴퓨터비젼, 병렬처리 등임



지 수 영

1986년 2월 충북대학교 컴퓨터공학과 졸업
1988년 2월 충북대학교 컴퓨터공학과 석사과정 졸업
1991년 1월~현재 시스템공학연구소 인공지능연구부 근무
1995년 9월~현재 고려대학교 전산과학과 박사과정 재학중
관심분야: 패턴인식, 문자인식, 컴퓨터비젼 등임