컨테이너 영상 전처리 및 식별자 인식 시스템의 설계

박 준표° , 이 주표, 황 대훈
경원대학교 컴퓨터공학부

Design of Container Image Preprocessing And Identifier Recognition System

Joon-Pyo Park° , Joo-Pyo Lee, Dae-Hoon Hwang
Dept. of Computer Engineering, Kyungwon Univ.

요 약

오늘날 컨테이너의 과다한 물동량 증가로 인하여 수작업으로 이루어지는 컨테이너를 처리하는데 어려움을 겪고 있다. 따라서 식별자로 컨테이너를 자동 인식하고 그 결과를 항만 물류처리 자동화 시스템에 적용하고자 하는 필요성이 대두되고 있다.

이에 본 논문에서는 항만 물류처리 자동화 시스템을 사용하기 위하여 컨테이너의 인식 처리를 자동화하는데 그 방안으로 컨테이너의 RGB를 이용하여 바탕색과 문자색을 검출하고 바탕색과 문자색의 차를 이용해 가장 큰 차이를 보이는 RGB 값 중 하나로 영상을 이진화 하였다. 컨테이너의 식별자를 인식하기 위해서 신경망 알고리즘의 하나인 Back-propagation을 적용하여 기존의 식별자 인식 방법보다 신속하고 정확한 처리가 가능하도록 구현하였다.

1. 서론

오늘날 컨테이너의 과다한 물동량 증가로 인하여 그동안 수작업으로 하던 컨테이너 처리를 자동화 하고자하는 필요성이 대두 되었다.

항만에서 컨테이너의 선적 및 야적 등과 같은 물류처리를 자동화하기 위하여 영상처리 컨테이너 식별자 인식 시스템을 사용하게 된다. 이 시스템은 기존의 설비 장비만으로 모든 번호판의 판독이 자동으로 처리될 수 있으며, 신규 컨테이너 증가와 번호판 규격의 변경 시에도 일부 소프트웨어의 변경만으로 기존 장비를 100% 활용할 수 있다는 장점을 가진다.

이에 본 연구에서는 컨테이너 식별자 인식 시스템을 사용하기 위해서 컨테이너의 RGB 색상을 이용하여 문자를 검출 하였다. 보통의 경우는 Gray 레벨의 영상을 사용하여 문자를 검출하였으나 이 방법은 컨테이너가 오래돼서 컨테이너의 외장이 벗겨지고 충돌에 의해 컨테이너가 손상을 입게 되면 정확한 식별자를 인식 할 수가 없었다. 따라서 콘테이너의 RGB 값을 사용하여 식별자를 인식을 하는데 이 방법의 이점은 RGB 값을 사용하기 때문에 컨테이너 색상의 특징을 좀 더 이용을 해서 문자를 검출 할 수 있다는 것이다. 우선 배경색과 바탕색을 찾는데 문자가 없는 부분에서 8개의 배경색을 추출해서 평균값을 구해 배경색을 찾게 되고 그 배경색을 이용해서 다시 문자색을 찾게 된다. 검출된 배경색과 문자색의 차를 이용하여 가장 적합한 RGB 값 중 하나로 영상을 이진화 시켜 문자를 검출을 한다.

그 검출된 4개의 문자와 6개의 숫자를 신경망의 하나인 Back-propagation을 식별자 인식에 적용함으로써 보다 정확하고 신속한 식별자 인식이 가능하도록 하였다.

2. 유효영역 추출

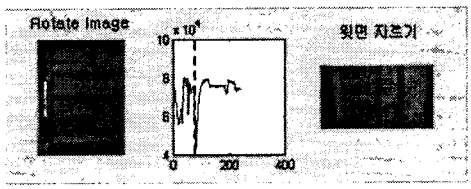
RGB를 이용하여 컨테이너에서 문자를 추출하는 방법은 다른 방법보다 잡음에 강하다는 장점을 가지고 있어서 외부 노출이 심한 컨테이너의 문자를 추출하는데 적합하다.

지금까지 문자를 추출했던 방법은 칼라 값을 이용하지 않고 영상을 GRAY 레벨로 변환을 한 후 히스

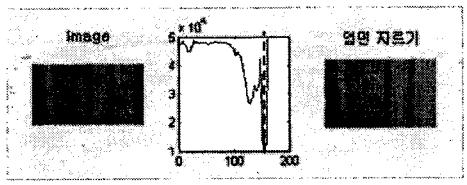
토그램을 이용하여 문자와 배경을 분리했었다. 하지만 컨테이너를 오래 사용할수록 컨테이너의 외장이 벗겨지고 많은 충돌에 의해서 컨테이너가 손상이 되기 때문에 히스토그램을 그렸을 때 나타나야 하는 쌍봉성이 잘 나타나지 않고 문자와 잡음과의 구별이 힘들게 되었다.

이에 본 논문에서는 잡음에 영향을 덜 받기 위해서 컨테이너의 RGB 값을 이용해서 문자를 검출하였다. RGB 값을 이용하는 방법은 컨테이너가 색이 벗겨지고 충돌에 의해 컨테이너가 손상을 입어도 컨테이너에는 그런 잡은 보다는 원래 가지고 있던 컨테이너의 색상이 가장 많이 남는 다는 점을 이용하게 된다.

문자를 추출하는 방법은 크게 배경색을 찾는 부분과 글자색을 찾는 부분으로 나누게 되는데 컨테이너의 배경과 문자를 더욱 정확하게 구별하기 위해서 영상에서 컨테이너 외에 다른 부분들은 없애고 컨테이너만 영상에 남겨야 한다. 컨테이너와 다른 부분들을 분리하기 위해서 컨테이너 상단에 있는 진한 그림자를 이용하게 되는데 전체 영상을 <그림 1>과 같이 수평 히스토그램 그려서 윗 배경을 제거하고 <그림 2>와 같이 수직 히스토그램을 그려서 오른쪽 배경을 제거한다. 수직, 수평 히스토그램을 그리게 되면 그림자 부분에 뚜렷한 경계가 생기게 되기 때문에 그 경계 부분으로 영상을 분리하게 되면 원하는 컨테이너 부분을 얻을 수 있다.

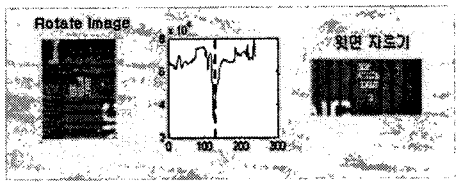


<그림 1> 윗 배경 제거

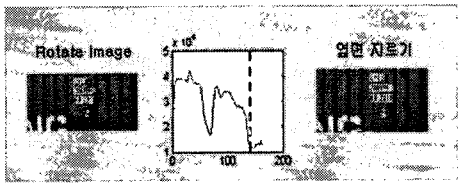


<그림 2> 오른쪽 배경 제거

컨테이너에 문자를 쓸 때 컨테이너에 바로 쓰는 경우도 있지만 눈에 더 잘 띄게 하기 위해서 레이블을 붙이고 그 위에 문자를 쓸 때가 있기 때문에 전처리 과정에서 항상 고려를 해야 한다. 레이블 위에 글자가 써져 있는 부분의 히스토그램이 컨테이너에 글자가 써져 있는 부분보다 다소 복잡하게 나타나 있지만 배경을 제거하는 부분에서는 <그림 3>과 <그림 4>처럼 최소점을 기준으로 영상을 분리하면 된다.



<그림 3> 윗 배경 제거(레이블이 있는 영상)

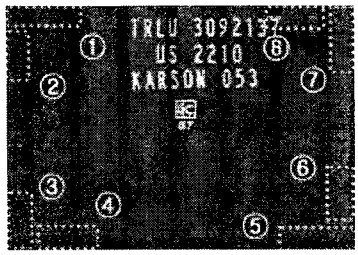


<그림 4> 오른쪽 배경 제거(레이블이 있는 영상)

3.문자추출을 위한 전처리

3.1 배경색 추출

분리된 영상을 얻은 후 먼저 배경색을 찾아야 한다. 배경색을 찾기 위해 컨테이너의 특징을 살펴보게 되면 거의 모든 컨테이너에서 글자 부분은 항상 가운데 있다는 것을 알 수 있다. 그래서 항상 모서리 부분에서는 배경색을 찾을 수 있다는 점을 이용하게 된다.



<그림 5> 배경 영역 검출

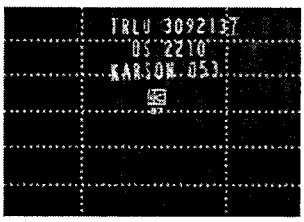
<그림 5>와 같이 컨테이너의 각 모서리 부분에서

각각 2부분씩 모두 8개의 RGB 값을 얻어 가지고 온다. 8개의 값을 가지고 오는 이유는 컨테이너가 충돌에 의해 손상을 입었거나 그림자등 다른 잡음이 있을 수 있기 때문이다. 그래서 8개의 값 중 잡음이 있을 최대값과 최소값을 제외한 값들의 평균값을 구하게 된다. 그 평균값이 배경이 되는데 위 방법으로 구한 값과 영상의 배경의 값을 비교해보면 항상 영상의 배경보다 진한 값이 나오기 때문에 평균값에 1보다 작은 값을 곱해서 원래의 배경색과 가장 가까운 값을 찾게 된다.

3.2 문자색 추출

그 다음은 문자색을 찾아야 한다. 모든 컨테이너에 있는 문자는 사람이 한 눈에 알아 볼 수 있도록 하기 위해서 바탕색과 대비되는 색으로 글자를 쓰기 때문에 문자를 찾을 때는 배경색을 이용하게 된다. 영상의 라인을 스캔하면서 각각의 RGB 값과 배경색의 RGB 값을 비교하게 되는데 그 중 가장 차이가 많이 나는 값을 문자색으로 하게 된다.

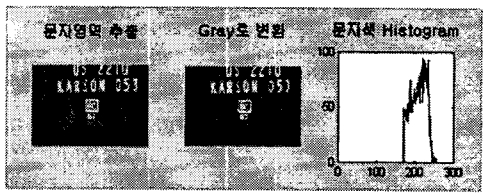
우선해야 할 일은 문자가 어디에 위치에 있는지를 먼저 찾아야 한다. 문자의 위치를 찾는 과정을 하기 위해서 <그림 6>과 같이 영상을 좌우로 3등분을 하고 상하로 6등분을 해서 좌우의 가운데 부분과 상하의 2~3 번째 부분을 사용해서 항상 글자가 쓰여 있는 부분을 사용하게 되는데 이렇게 영상의 가운데 부분만 사용을 하는 이유는 전체 영상을 전부 라인 스캔을 하게 되면 많은 시간이 걸리기 때문에 글자가 쓰여진 부분만을 이용하는 것이다.



<그림 6> 문자 영역 검출

그 다음 <그림 7>과 같이 배경색과 사용할 가운데 부분을 Gray 레벨로 바꾼 후 컨테이너 영상에서 배경색보다 50 이상 큰 값의 개수와 배경색보다 50 이상 작은 값의 개수를 구하게 된다. 그렇게 큰 값과 작은 값을 구분한 후 개수가 많은 부분을 선택하게 된다. 개수가 많은 부분은 글자부분이 되는 것이고 개수가 적은 부분은 잡음을 나타내게 된다. 그 택한 부분의 가장 큰 값을 선택하게 되는데 그 부분이 잘라낸 영상에서 가장 많은 부분을 차지하고 있기 때문에 글자로 간주하고 그 부분의 한 픽셀의 위치 값을 저장하게 된다.

그 다음 선택한 영상을 다시 칼라 영상으로 바꾼 후 찾은 위치에 있는 값을 추출하면 그 부분이 글자색을 나타낸다



<그림 7> 문자색 추출

3.3 이진화

이진화를 하기 위해서는 영상에 임계값을 적용하여 그 임계값을 기준으로 흰색과 검은색으로 분리를 할 수가 있다.

그 임계값을 구하는 방법은 <식 1>을 이용하여 구하게 된다.

$$T = a \times u$$

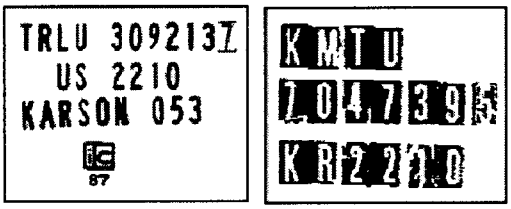
$$u = \frac{1}{N \times m} \sum_{j=1}^{M} \sum_{i=1}^{N} f(i,j) \qquad \text{<식 1>}$$

$$M' = M / t$$

여기서 T는 임계값, a는 임의의 상수, u는 미분연산자에 의해 $f(i,j)$ 영상의 평균, $N \times M$은 입력된 영상의 크기, t는 미분연산자를 적용한 일정한 간격, 그리고 M'는 t로 M을 나눈 영상의 세로 크기이다.

그러나 본 논문에서는 이진화를 하는데 임계값을 구할 때 <식 1>을 사용하지 않았다. 왜냐면 본 논문에서 사용한 전처리 과정에서는 문자색과 배경색을 정확하게 찾아냈기 때문이다. 이진화 시키는데 이진화를 보다 쉽고 정확하게 하기 위해서 전체 영상의 RGB 중 하나의 값만을 사용하는데 RGB 값 중 하나만 사용하는 이유는 RGB 값 중 하나는 배경과 문자의 차가 가장 크게 나타나기 때문에 이진화 시켰을 때 가장 정확히 원하는 값을 얻을 수 가 있다. 이 과정을 위해서는 이진화 시킬 영상에서 가장 적합한

RGB 중 하나를 선택해야 하는데 그 선택하는 방법은 그 영상에서 가장 많은 색을 차지하고 있는 배경색을 이용하게 된다. 그 배경의 RGB 중 R < 120, G < 120, B > 120 일 경우는 RGB 중 R 값만을 사용하게 되고 R > 120, G < 120, B > 120 일 경우는 B 값만을 사용하게 된다. 그 외의 영상은 G의 값만을 사용한다. 그 찾은 RGB 중의 하나의 값만을 가지는 영상으로 변환한 후 영상을 이진화를 시키는데 그때 사용되는 임계 값은 글자색과 배경색의 중간 값을 사용하게 된다.

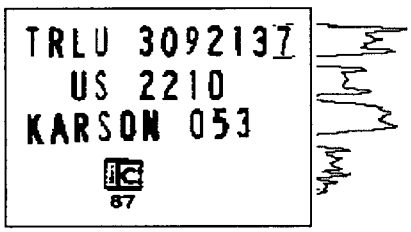


<그림 8> 이진화된 영상

4 각자 분할

우선 각자 분할을 하기 위해서는 우선 컨테이너에 문자의 위치가 어떻게 되어 있는지 알아야 한다. 문자가 쓰인 위치는 크게 4종류로 나뉠 수 있는데 문자가 가로로 한 줄에 다 쓰여 있을 경우가 있고 문자부분과 숫자 부분이 두 줄로 나뉘어 있는 경우, 문자와 숫자 부분이 3줄로 되어 있는 것 그리고 문자가 세로로 한 줄로 되어 있는 경우가 있다.

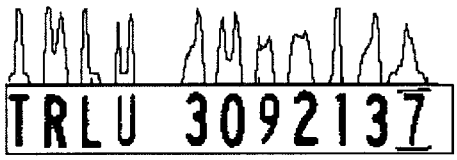
문자 추출은 일단 모든 이미지의 위, 아래, 왼쪽 오른쪽 부분의 공백을 없앤 후 <그림 9>와 같이 수평히스토그램을 그려서 문자열을 세게 된다.



<그림 9> 수평히스토그램으로 문자열 분할

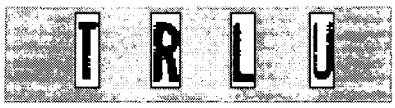
만약 위, 아래의 문자가 붙었을 경우 영상의 문자열의 최소 글자수가 4개로 보고 비율을 통해서 잘라내게 된다. 그 후 <그림 10>과 같이 잘라낸 문자열마다 다시 한번 아래, 위, 왼쪽, 오른쪽의 공백을 없애고 수직히스토그램을 그려서 문자를 잘라내게 된다.

영상에 따라 히스토그램이 다르게 나오지만 이진화가 잘 되었을 경우 위와 같이 글자와 글자 부분에 확실한 경계가 생기게 된다. 위에서 필요한 문자는 영문자 4자 숫자 6개이다. 따라서 수직 히스토그램을 그려서 문자를 잘라낸 후 원하는 10개의 문자가 나올 때까지 계속 문자를 검사하게 된다.

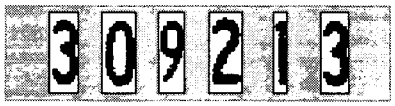


<그림 10> 수직히스토그램으로 각자 분할

각자 분할을 할 경우 작은 노이즈가 문자열로 인식 될 경우가 있으므로 일단 조금한 히스토그램 값이 계산되면 주변의 히스토그램 값을 조사해서 문자인지 조금 진한 노이즈인지 판단을 하게 된다.



<그림 11> 각자 분할된 문자



<그림 12> 각자 분할된 숫자

레이블이 있는 경우는 <그림 8>의 오른쪽과 같이 글자 부분에 나타나야 될 검정색이 배경 부분에 나타나게 되고 흰색이 글자가 되게 된다. 그렇게 때문에 레이블이 있는 이미지는 수평히스토그램을 그려서 문자열을 자르고 색의 한번 반전 시켜 줘야 한다.

우선 이미지에 레이블이 있는 이미지 인지 그렇지 않은 이미지 인지 먼저 구별을 해야 하는데 분할한 문자열을 수평으로 라인 스캔을 하게 되면 글자 부분에 나타나야 할 검정색 부분이 바탕에 나타났기

때문에 긴 영역을 차지하고 있다. 그 부분은 문자라고 보기 힘들기 때문에 문자열의 색을 반전 시켜 주게 된다. 그 후 글자 부분이 검정색이 되어 다른 이미지들과 같은 방법으로 수직히스토그램을 그려서 문자를 각자 분할 해주면 된다.

5. 신경망을 이용한 문자인식

5.1 Scaling

각자 분할된 컨테이너 식별자는 보통 입력이 50*80정도의 크기로 들어온다. 그러나 이것은 상당히 유동적이며 식별자들은 약간의 차이를 보이고 있다.

입력된 문자를 신경망 알고리즘에 적용하기 위해서는 가로*세로의 크기를 정형화하여야 한다.

Scaling 시 얻어지는 장점으로는 문자가 가지고 있는 noise를 적절히 감쇄시켜 인식률을 높일 수 있다는 것이다.

입력되는 각 문자들은 다양한 크기로 scaling이 적용되는데 여기에 적용된 공식들은 다음과 같다.

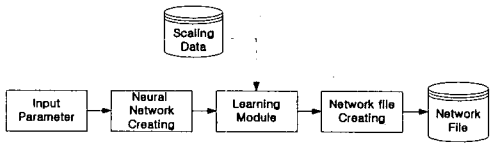
영역 픽셀수 × 0.5 ≤ 검정 픽셀수

가 만족하면 scaling 영역을 1로 만족하지 않으면 0으로 처리한다. 예를 들어 55×80의 이미지에 대하여 7×15의 scaling을 적용한다면 7×5의 기본 패턴을 가지고 scaling을 적용할 것이다. 그럼 6×15의 패턴이 남게 되므로 7×15가 되도록 가상공간을 만들게 된다. 이때 남아있는 부분이 기본패턴의 중간값보다 작으면 무시를 하고 중간값보다 클 때만 가상공간을 만들게 된다.

5.2 학습 DB구축

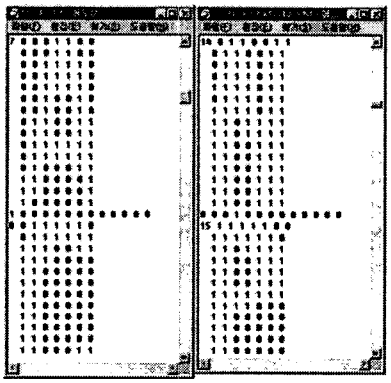
Scaling을 통하여 수집된 문자들을 학습시키기 위하여 학습 DB를 구축하게 된다. 학습 DB를 구축하기 위해서는 <그림 13>과 같이 먼저 컨테이너에 구성된 문자들을 영역별로 나누어 학습 DB를 만들게 되는데, BP1과 알파벳의 일부 글자 BP2, 나머지 알파벳의 BP3의 세 부분으로 나누어 구성한다.

Target Value는 보통의 경우에는 4개로도 충분히 각 패턴을 표현할 수가 있다. 그러나, 본 본문에서는 각 부분을 더욱 세밀히(숫자의 경우 10개, 알파벳의 경우 각각 13개씩) 할당한 이유는 유사한 문자 패턴으로 인한 오인식을 줄일 수 있을 뿐만 아니라 학습 시간을 단축할 수 있다.



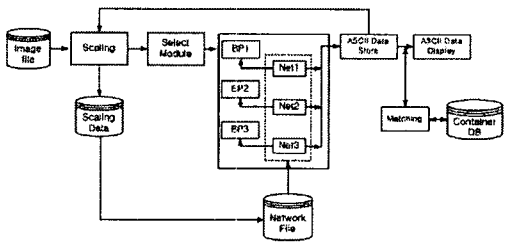
<그림 13> 신경망 학습

다음 <그림 14>는 7×15 비율로 스케일링된 문자를 메모장을 이용하여 학습 패턴별로 학습 file을 만든 것을 보이고 있다. 문자의 앞에 붙어있는 숫자는 각 학습 패턴의 갯수를 나타내고, 마지막에 붙어 있는 숫자들은 학습 패턴의 목표값이다.



<그림 14> 7×15 학습 데이터베이스 구축

5.2 문자인식



<그림 15> 신경망 실행

먼저 전처리 과정을 거친 컨테이너 식별자는 스케일링 과정을 거치고 학습시킨 후 <그림 15>와 같이 각 영역별로 구분하여 BP 과정을 수행하게 된다. 즉 숫자 부분인 BP1, 알파벳의 BP2와 BP3의 세 부분으로 나누어 인식 과정을 수행하게 되는데 알파벳의

경우 BP2와 BP3으로 나누어 다중 뉴럴넷으로 구성한다. 이 이유는 학습에 걸리는 시간을 줄이고 유사한 알파벳으로 인한 인식의 오류를 방지하기 위함이다.

BP 수행과정이 끝난 후에는 각 단계별로 인식 과정을 거쳐 원격지의 데이터베이스와 매칭을 통해 최종적인 인식을 얻게 된다. <그림 16>은 신경망의 실행 알고리즘을 나타 내고 있다.

```
// Running algorithm of BP Neural network

Run the preprocessing step                    // 전처리 과정
    for(i = 0; i < 4; i ++){
     Get a alphabet character;
     Scale the character;
     Run the neural network BPII;
     Run the neural network BPIII;
     Compare the similarity degree of BPII with BPIII;
     // 유사도 매칭
     Store the result;
     }
    for(i = 0; i < 7; i ++){
     Get a number character;
     Scale the character;
     Run the neural network BPI;
     Store the result;
     }
     Match the result with DB;
     Output the matching result;
```

<그림 16> 신경망 실행 알고리즘

5.결론

본 논문에서는 항만 자동화를 위하여 컨테이너의 바탕색과 문자색의 RGB 차이를 이용하여 문자를 검출 하였고, Back-propagation 알고리즘을 이용하여 식별자 인식에 적용함으로써 보다 정확하고 신속한 식별자 인식이 가능하도록 하였다.

검출된 배경색과 문자색으로 이진화 시키는데 이진화를 시킬 때는 영상의 RGB 값을 이용하여 배경색을 찾고 그 배경색을 이용하여 문자색을 검출하였다. 이진화를 시킬 때는 영상의 RGB 중 하나의 값만을 사용하는데 문자색과 배경색의 차이가 가장 많이 나는 값을 선택해서 이진화 시켰다.

각자분할 할 때는 레이블이 없는 이미지는 글자색과 문자색의 중간 이미지를 사용하였고 레이블이 달린 이미지는 레이블이 없는 이미지를 이진화시킨 결과를 반전 시켰다. RGB를 이용한 문자 검출은 기존의 Gray 레벨을 이용한 방법보다 더 나은 검출 결과를 보였다.

문자 인식에 있어서는 Back-propagation을 사용하였다. 숫자의 경우는 Target Value를 10개, 알파벳의 경우는 13개씩 할당을 하였고 숫자는 BP1 알파벳은 BP2와 BP3로 나누어 학습에 걸리는 시간을 줄이고 유사한 알파벳으로 인한 인식의 오류를 방지했다.

앞으로 알고리즘의 보완으로 인식하는데 걸리는 시간을 단축시키고 보다 많은 패턴을 학습시키면 더 나은 결과가 기대 대리라 예상된다.

[참 고 문 헌]

[1] Stephen P.Banks, *"Signal Processing, Image Processing and Pattern Recognition,"* Prentic Hall, 1990

[2] Tsuji, et al., "Document image analysis based on split detection method, paper of the Technical Group on pattern Recognition and Learning," IEICE, PRL85-17, pp. 63-70, 1985.

[3] 이성근, "항만 물류처리 자동화를 위한 컨테이너 식별자 처리 및 인식에 관한 연구", 경원대학교 대학원 석사논문, 1997

[4] Container Identifier Code, http://www.ean.be/html/SSCC.html

[5] 이만형, 허도영, 이성근, 황대훈, "항만 물류처리 자동화를 위한 컨테이너 식별자의 영상 전처리에 관한 연구" 1998년 4월, 정보처리학회 춘계학술발표대회 논문지

[6] R.Hecht-Nielsen, Nrurocomputing : Picking the Human Brain, IEEE Spectrum 25(3), pp. 36-41, March, 1988.

[7] P.D.Wasserman, *"Nerual Computing : Theory and Practice,"* Van Nostrand Reinhold, 1989

[8] R. Beale and T. Jackson, *"Neural Computing An Introduction,"* Adam Hilger Bristol, Philadelphia and New York, pp. 74~79, 1990

[9] 오 창석, "*뉴로 컴퓨터*," 지성 출판사, 1996

[10] 이만형, "Back-propagation 알고리즘을 이용한 컨테이너 식별자 인식 시스템의 구현 및 분석", 경원대학교 대학원 석사논문, 1999