Back-propagation 알고리즘을 이용한 컨테이너 식별자 인식 시스템의 구현 및 분석

이만형°, 황상훈, 정신규, 황대훈
경원대학교 전자계산학과

An Implementation and Analysis of the Container Identifier Recognition System using Back-propagation Algorithm

ManHyung Lee°, SangHoon Hwang, ShinKu Jung, DaeHoon Hwang
Dept. of Computer Science, Kyungwon Univ.

요 약

오늘날 컨테이너의 과다한 물동량 증가로 인하여 수작업으로 이루어지는 컨테이너 식별자를 처리하는데 어려움을 겪고 있는 가운데, 이를 자동으로 인식하고 그 결과를 항만 물류처리 자동화 시스템에 적용하고자 하는 필요성이 대두되고 있다.

이에 본 논문에서는 컨테이너의 인식 처리를 자동화하기 위한 방안으로 컨테이너의 식별자 인식에 신경망 알고리즘의 하나인 Back-propagation을 적용하였으며, BP 알고리즘을 적용하기 위해서 적절한 scaling 비율을 구하고, 학습 DB를 구축하여 기존의 식별자 인식보다 신속하고 정확한 처리가 가능하도록 구현하였다.

Ⅰ. 서론

현재 일부 항만에서 컨테이너의 자동 처리를 위해 채택되고 있는 시스템은 크게 바-코드(bar-code) 시스템과 영상 처리를 기반으로 하는 컨테이너 식별자 인식 시스템으로 대별된다.

바-코드 시스템은 컨테이너에 바-코드를 부착하거나 소지하여 처리하는 것으로 훼손이나 분실시에는 컨테이너를 자동으로 처리할 수 없을 뿐 아니라, 차량 번호판이나 컨테이너 ISO 번호가 변경될 때마다 바-코드를 교체해야 하는 문제점 등을 갖고 있다. 따라서 이 시스템은 추가 경비 지출이 계속하여 발생하고 관리비용이 과다하게 지출되는 문제점 등으로 인하여, 오늘날 항만에서의 컨테이너 처리를 위한 시스템으로는 영상 처리 컨테이너 식별자 인식 시스템을 적용하는 추세에 있다[1][2].

영상 처리 컨테이너 식별자 인식 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[3].

이에 본 연구에서는 항만에서 컨테이너의 선적 및 야적 등과 같은 물류처리를 자동화하기 위하여 신경망 알고리즘의 하나인 Back-propagation을 식별자 인식에 적용함으로써 보다 정확하고 신속한 식별자 인식이 가능하도록 하였다.

Ⅱ. 컨테이너 식별자

2.1 컨테이너 식별자의 ISO 규격

컨테이너의 식별자 코드는 다음과 같은 다섯 개의 구분된 영역으로 구성되어 있다. 이중 Owner code

는 국제 컨테이너 관리국에 의해 전세계적으로 고유하게 부여된 4개 문자 코드이다[4].

2.2 컨테이너 식별자의 유형 및 특징

컨테이너에 기록하는 문자 유형에는 어떤 표준이나 강제적인 규약이 없으므로, 어떤 문자는 이런 폰트를 쓴다고 한 마디로 단언할 수는 없다.

또한 문자들은 모두 컨테이너의 외부에 노출되어 있기 때문에, 문자의 유실이나 오물 등과 같은 잡음으로 인하여 문자를 인식하는데 어려움이 따른다. 아울러 컨테이너의 바탕 즉 문자가 쓰여있는 부분이 굴곡되어 있는 특징으로 인하여 문자의 왜곡 등과 같은 형태 변형과 명암 등으로 인한 문자의 잡음 개입과 유실 등의 영향을 미칠 수 있다. 다음 그림 1은 컨테이너 식별자의 특징을 나타내고 있다.

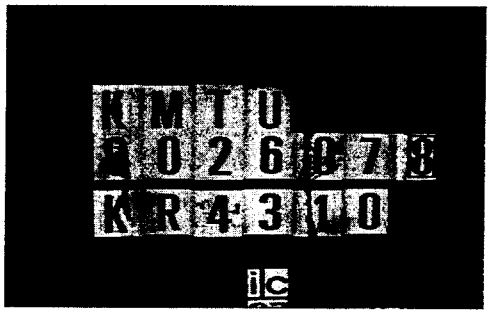

그림 1 컨테이너 식별자의 예

III. 시스템 설계

3.1 전체 시스템 구성

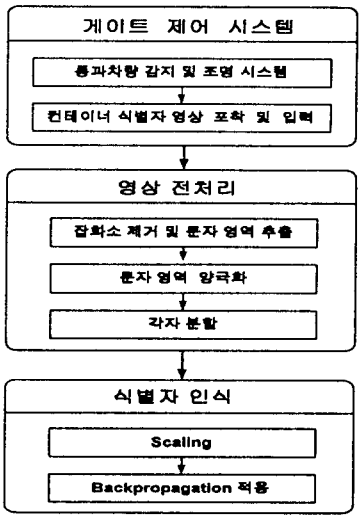

그림 2 식별자 인식 시스템의 구조

컨테이너 식별자 인식을 위한 전체 시스템 구조는 그림 2와 같이 세 부분으로 나눌 수 있다.

먼저 게이트 제어 시스템에서는 컨테이너 식별자의 영상을 처리하기 위하여 통과 차량의 감시 및 조명 시스템, 식별자 입력등을 처리하고, 영상 전처리에서는 잡화소 제거 및 문자 영역을 추출하고, 문자 영상을 양극화하여 각자분할을 수행한다. 마지막으로 식별자 인식에서는 scaling된 문자를 BP 알고리즘에 적용하는 단계로 나눌 수 있다.

본 게이트 제어 시스템에서는 최상의 차량의 위치를 감지하기 위하여 정밀하고 신뢰성이 보장되는 광센서를 사용하였는데, 감지 과정은 그림 3과 같은 과정을 통하여 실행된다.

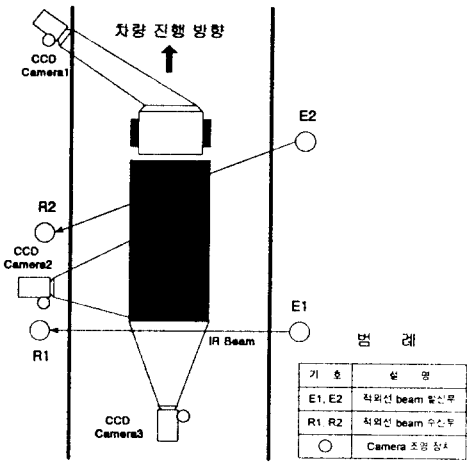

그림 3 게이트 제어 시스템 구성

차량이 감지 차선을 통과할 때 한 쌍의 송신기와 수신기 사이를 통과하는 눈에 보이지 않는 적외선 빔(beam)을 차단하게 되는데, 본 시스템에서 2쌍의 센서를 사용하는 것은 서 있는 사람이나 동물이 차량으로 감지되는 것을 방지하기 위한 것이다.

조명 시설의 설치는 식별자의 인식을 어렵게 하는 그림자를 제거하기 위한 목적으로 설치된 것으로, 주간에 빛을 차단하는 자연, 빌딩, 사람, 구름 및 차량의 일부에 의해 생성되며 야간의 그림자는 다른 차량의 전조등, 시설물의 조명 장치 등으로부터 발생한다. 따라서 조명 기기를 설치함으로써 식별자 인식에 영향을 미치는 그림자를 제거할 수 있게 된다.

3.2 영상 전처리

CCD 카메라에 포착된 컨테이너 영상이 입력으로

들어오면, 우선 컨테이너의 배경색 및 문자색을 알아내기 위하여 영상에 라인-스캔(line-scan)을 실시한다. 이때 각 라인-스캔의 화소값들을 자료구조에 기억시키며, 이 기억된 값들은 차후 문자 영역을 추출하는 과정에서 다시 쓰이게 된다. 그리고 추출되어진 영역에서 문자인 부분은 검은 색으로 배경색인 부분은 흰색으로 색상을 양극화시킨다. 이 과정을 하기 위해서 휘도 히스토그램을 그리는데, 이 휘도 히스토그램을 이용하면 보다 쉽게 색을 양극화 할 수 있고 아울러 이 결과를 통하여 잡화소를 제거할 수도 있다. 이 양극화된 이미지를 다시 수평, 수직 line-scan을 실시하여, 최소 문자 영역을 추출하고, 흑픽셀의 발생 빈도를 나타내는 히스토그램을 구하여 문자의 좌우를 구하는 각자분할을 수행한다[5]. 이 각자 분할된 식별자들은 비트맵 이미지로 저장되어 문자인식 단계로 넘겨지고, scaling 및 학습 DB 구축, BP 알고리즘을 적용한다.

3.3 문자 scaling

전처리 과정을 거친 식별자를 BP 알고리즘에 적용하기 위해서는 크기를 정형화하여야 한다. 그렇게 하기 위해서는 scaling 과정이 반드시 필요하다.

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

(1) Scaling

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 그림 4와 같이 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. 즉, 7×5가 기본 패턴이므로 4×3이상일 때만 가상공간이 만들어지게 되는 것이다. 그림과 같이 가상공간을 설정한 후에는 재 scaling을 하게 되는데 그림 전체 scaling 수는 7×15가 아니라 8×16이 되며 이때 문자의 변형이 가장 적은 중간 부분에서 기본 패턴을 하나씩 삭제하면 최종적으로 7×15의 scaling이 된다.

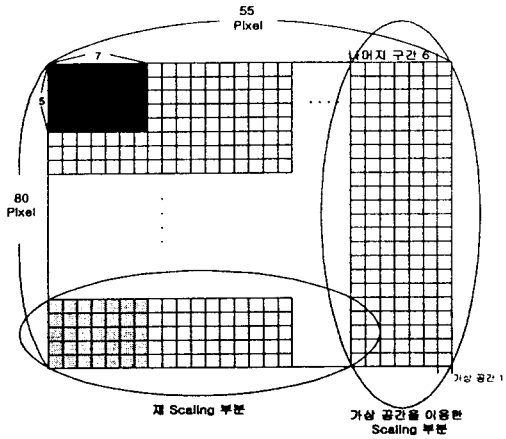

그림 4 55×80에서 scaling을 적용한 예

다음 그림 5는 scaling을 통하여 얻어진 문자를 나타내고 있다.

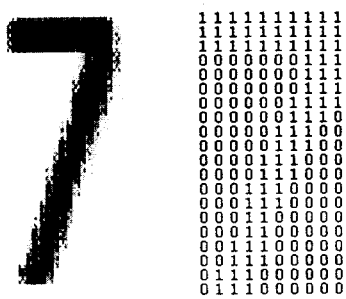

그림 5 Scaling을 통하여 얻어진 문자

(2) Scaling시 나타나는 문제점

Scaling시 발생하는 문제점은 크게 두 가지로 나눌 수 있는데, 먼저 scaling을 작게 적용했을 때 발생하는 문제점과, scaling을 크게 적용했을 때 발생하는 문제점으로 나뉠 수 있다.

먼저 scaling을 작게 적용할 경우에는 획 두께보다 설정된 임계치 값이 너무 작으면 제대로 된 scaling이 이루어지지 않고, 반대로 크게 했을 때 문제점은 획두께가 scaling 기본 패턴보다 작거나 문자의 크기가 현저히 작을 경우 발생된다.

이러한 현상을 없애기 위해서는 식별자 이미지를 camera에 꽉 차도록 입력받아 적용하면 이러한 오류를 줄일 수 있다. 그러므로 적당한 scaling 비율을 구하는 것이 인식에 있어 가장 중요한 일이라 할 수

있다.

3.4 학습 DB 구축 및 식별자 인식

(1) 학습 DB 구축

Scaling을 통하여 수집된 문자들을 학습시키기 위하여 학습 DB를 구축하게 된다. 학습 DB를 구축하기 위해서는 먼저 컨테이너에 구성된 문자들을 영역별로 나누어 학습 DB를 만들게 되는데, BP1과 알파벳의 일부 글자 BP2, 나머지 알파벳의 BP3의 세 부분으로 나누어 구성한다. 다음 표 1은 각 영역별 학습 패턴의 목표값을 나타내고 있다.

표 1. 학습 패턴의 목표값

패턴	Target Value	10진값	패턴	Target Value	10진값
0	0000000001	1	5	0000100000	32
1	0000000010	2	6	0001000000	64
2	0000000100	4	7	0010000000	128
3	0000001000	8	8	0100000000	256
4	0000010000	16	9	1000000000	512

(a) BP1 학습 패턴의 목표값

패턴	Target Value	10진값	패턴	Target Value	10진값
A	00000000000001	1	J	0000010000000	128
B	00000000000010	2	K	0000100000000	256
C	00000000000100	4	L	0001000000000	512
D	00000000001000	8	Q	0010000000000	1024
E	00000000010000	16	V	0100000000000	2048
H	00000000100000	32	Z	1000000000000	4096
I	00000001000000	64			

(b) BP2 학습 패턴의 목표값

패턴	Target Value	10진값	패턴	Target Value	10진값
F	00000000000001	1	S	0000010000000	128
G	00000000000010	2	T	0000100000000	256
M	00000000000100	4	U	0001000000000	512
N	00000000001000	8	W	0010000000000	1024
O	00000000010000	16	X	0100000000000	2048
P	00000000100000	32	Y	1000000000000	4096
R	00000001000000	64			

(c) BP3 학습 패턴의 목표값

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 각 패턴이 유효한 출력값을 하나만을 가짐으로 오인식이 발생할 경우에 이를 방지할 가능성을 더욱 높였다. 또, BP2와 BP3의 알파벳의 경우에도 두 가지의 경우로 나누었는데 이것은 문자 인식시에 적용되는 라인 스캔의 유사도에 따라 구분함으로써 학습에 걸리는 시간과 오류를 줄이기 위함이다.[6]

숫자의 경우 나타날 수 있는 출력값은 총 1024 가지이고 이 중에서 유효한 출력값의 경우는 10개이므로 10개를 제외한 나머지 1014개의 값은 오류임을 알 수 있다. 알파벳의 경우도 마찬가지이다. 오류가 발생하게 되면 오류에 해당하는 문자를 인식하지 않고 "?"로 남겨둔 뒤, 후처리 과정(DB 매칭) 수행시에 이를 제외시킴으로써 매칭도를 높일 수 있다.

현재 학습을 위한 파라미터 값은 다음과 같다.

Rate : 0.013 (학습률을 결정하는 에러율을 0.013으로 학습)

Momentum : 0.9 (학습 수렴구간 중 국부 최소점에 빠지는 것을 막기 위하여 출력값에 변화를 줌)

Tolerance : 0.2 (학습 에러율이 0.2 이상이면 인식으로 결정하고 0.2 이하이면 ?를 출력)

Interval : 10 (10단위로 결과를 출력)

Input_Node : 105 (입력 노드의 갯수로 입력 노드가 10×20일 경우 200, 7×15일 경우 105로 입력 노드에 따라 변경)

Middle_Node : 80 (입력 노드와 출력 노드 사이의 노드로 유동적으로 갯수를 설정)

Output_Node : 10 (출력 노드의 갯수를 정의하는 노드로 숫자일 경우는 10, 알파벳일 경우는 13으로 설정)

학습 DB를 완성한 후 식별자 인식을 위해서는 그림 6과 같은 흐름에 따라 식별자를 인식하게 된다.

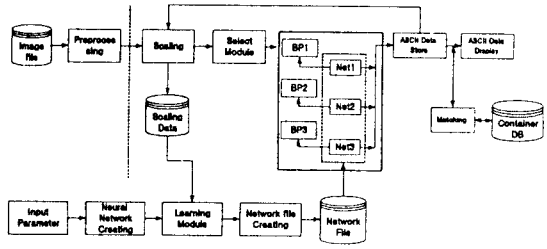

그림 6 인식 처리 과정의 흐름도

(2) BP 알고리즘의 적용

BP 알고리즘을 적용하기 위해서는 각 문자를 정형화하여 학습 DB를 만들고, 각 학습 DB에 해당하는 Neural Network에 매칭하게 된다[7][8].

다음 그림 7은 BP 알고리즘을 적용하기 위한 전체적인 구성을 나타낸 것이다.

먼저 전처리 과정을 거친 컨테이너 식별자는 scaling 과정을 거치고 학습시킨 후 각 영역별로 구분하여 BP과정을 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.

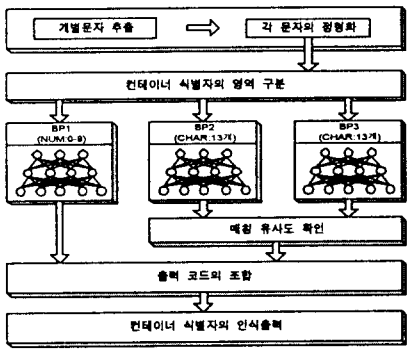

그림 7 Back-propagation 수행 과정

BP 수행과정이 끝난 후에는 각 단계별로 인식 과정을 거쳐 원격지의 DB와 매칭을 통해 최종적인 인식을 얻게 된다.

IV. 실험 평가 및 결과

4.1 Scaling에 따른 분석 결과

각각의 scaling 비율로 문자를 인식할 경우 학습시간과 인식률에서 많은 차이를 보이고 있는데, 이는 input node의 증가와 scaling시 나타나는 문제점으로 인해 발생하는 것이고 현재 7×15와 10×20, 12×24, 15×30의 경우에 인식률을 구하였다. 7×15일 경우 약 30%의 인식률을 보였고, 10×20의 경우에는 이 보다 좋은 95.7%의 인식률을 나타내고 있다. 하지만 12×24에서는 20%의 인식률을, 15×30의 경우에는 5%의 저조한 인식률을 나타내고 있다.

7×15의 경우 인식률이 저조하게 나타나는 이유는 앞서 설명한 것처럼 scaling할 때 입력되는 문자의 크기에 비해 획의 두께가 얇거나 camera에 식별자 이미지를 꽉 차게 입력받지 못하여 문자의 크기가 현저히 작을 때 발생한다. 15×30 이상의 경우에서는 식별자의 이미지가 scaling 비율에 비해 작을 경우 오류를 발생시킨다. 이러한 문제를 해결하기 위해서는 식별자를 입력받을 때 이미지에 꽉 차게 입력받고 적당한 scaling율을 구해 적용하여야 한다.

다음 표 2는 각 scaling별 학습 데이터의 갯수, 학습시간 및 인식률 등의 분석결과를 나타낸 것이다.

7×15의 scaling으로 하여 학습시킬 때 걸리는 시간은 실행 시마다 조금씩 차이가 있으나 평균 BP1(194)인 경우 22분 25초, BP2(42)인 경우 25분 43초, BP3(74)인 경우 36분 16초의 시간이 걸렸다. 그러나 인식률에서는 30%에도 미치지 못하는 저조한 결과를 나타내었다. 실행 시마다 학습 시간에서 차이를 보이는 이유는 초기 연결 강도를 -0.5 ~ +0.5 범위 안에서 random하게 설정하므로 이 설정값에 따라 학습이 이루어지는 경우가 다르기 때문이다.

표 2 각 Scaling별 분석 결과

		7×15	10×20	15×30
Input node		105	200	450
Middle node		80	150	350
Output node	숫자	10	10	10
	알파벳	13	13	13
인식률		30%	95.7%	5%
학습시간 (데이터 갯수)	BP1	22분23초(194)	56분27초(125)	Scaling시 오류로 인한 문제점 발생
	BP2	9분42초(42)	21분39초(27)	
	BP3	36분16초(74)	55분52초(47)	

10×20의 경우에 예상한대로 7×15보다 좀더 많은 시간이 소요된 것을 볼 수 있다. input node와 middle node의 변화로 인해 학습에 걸리는 시간이 증가하였기 때문이다. 10×20보다 12×24가 시간이 많이 걸리는 이유도 마찬가지이다. 즉, scaling을 세분하게 실행하면 학습에 걸리는 시간이 보다 증가함을 나타내고 있다.

15×30에서는 거의 인식이 이루어지지 않음을 알 수 있는데, 이는 앞서 설명한 것과 같이 입력되는 문자의 크기가 scaling 비율에 미치지 못함으로 발생되는 현상으로 scaling 비율의 한계를 나타낸다. 즉, 이 이하의 크기로 scaling을 적용하여야만 인식에서 좋은 결과를 얻을 수 있는 것이다.

다음 그림 8은 scaling의 크기에 따른 인식률 및 학습 시간을 도표로 나타낸 것이다.

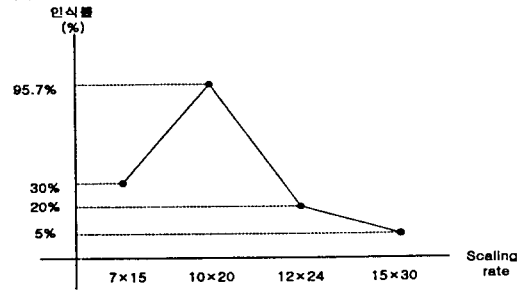

그림 8 Scaling율에 따른 인식 결과

그림 9는 scaling 비율이 인식에 얼마나 큰 영향을 미치는지를 잘 나타내고 있다. 7×15 이하의 경우와 12×24 이상의 경우에 scaling시 나타나는 문제점으로 인해 scaling 비율이 10×20 전후에서 가장 좋은 scaling 비율이 된다는 것을 알 수 있다.

또, 12×24 이상의 경우에는 인식률이 저조할 뿐만 아니라 학습에 걸리는 시간도 상당히 오래 걸리는 것을 알 수 있었다. 10×20에 비해 많은 시간이 걸리는 것으로 알 수 있듯이 input node의 갯수가 학습에 많은 영향을 미치고 있음을 알 수 있다.

다음 표 3은 10×20의 전후에서의 scaling 비율에 따라 인식률을 비교한 것이다.

표 3 10×20 전후에서의 scaling별 분석 결과

		9×18	10×18	11×22
Input node		162	180	242
Middle node		120	130	200
Output node	숫자	10	10	10
	알파벳	13	13	13
인식률		50%	96%	30%
학습시간 (데이터 갯수)	BP1	1시간20분(148)	2시간08분(147)	4시간52분(139)
	BP2	7분39초(30)	7분22초(32)	55분59초(28)
	BP3	16분31초(62)	2시간01분(64)	7시간15분(61)

10×20 사이에서의 scaling 비율에 따른 인식률을 살펴보면 scaling 한계에 의한 오류는 발생하지는 않았다. 전체적인 인식의 결과도 그리 높게 나타나지는 않았지만, 10×18의 경우 10×20의 경우보다 다소 높은 인식률을 나타내고 있다.

표 3에서 나타난 바와 같이 scaling 비율이 커질수록 학습시간이 늘어나는 것을 볼 수 있다. 또한 scaling 비율을 얼마로 하느냐에 따라 데이터의 갯수에도 차이를 보이고 있다.

V. 결 론

본 논문은 컨테이너 문자인식에 Back-propagation을 이용하여 인식률을 높이기 위한 알고리즘을 적용한 것이다.

기존의 컨테이너 문자 인식의 경우 하나의 컨테이너 식별자를 인식하는데 걸린 시간은 2~3초의 처리시간이 소요되었으나, Back-propagation을 사용한 컨테이너 문자인식에서는 약 17초 정도의 시간이 소요되고 있다. 이렇게 인식 시간이 많이 걸리는 이유는 수행시마다 네트워크망을 메모리에 적재하는데 이로 인한 시간 손실이 크기 때문이다. 따라서 현재는 이를 보완하여 한번만 네트워크망을 메모리에 적재하여 수행할 수 있도록 하여, 인식하는데 걸리는 시간은 약 1초로 기존의 문자인식보다 다소 적은 시간이 소요되었다.

인식률의 경우 기존 컨테이너 인식의 경우 약 96%의 인식률을 보였고, Back-propagation을 이용한 경우 이와 비슷한 96%의 인식을 보이고 있다. 이는 오인식 할 수 있는 문자의 경우 학습을 통해 원래의 문자로 인식할 수 있게 하였지만, 학습률의 크기를 크게 조정한다면 학습 시간은 다소 많이 소요되겠지만 좀더 정확한 학습을 할 수 있으므로 인식률에서 좀더 나은 결과를 얻을 수 있을 것이다[9].

앞으로 학습률에 따른 결과와 middle node의 변경에 의한 학습 시간을 비교하여 학습시간이 적게 걸리며 인식률의 향상을 가져오는 학습 파라미터를 찾아 적용시켜야 할 것이다.

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