컨테이너 번호 추출을 위한 영상 처리 시뮬레이터 설계 및 구현*

A Design and Implement of Image Processing Simulator for Extraction of Container License Number

최창훈**
Chang Hoon Choi

Abstract

With the recent outstanding advance in computer software and hardware, a number of researches to enhance the processing speed and the process accuracy has been undertaken in the field of container terminal industry. In this paper, we propose a simulator for container image processing that can be used mainly for the development of automatic container recognition system. The purpose of this study is to simulate many different algorithms and factors to extract information accurately on the captured container image.

* 이 논문은 2000년도 상주대학교 산업과학기술연구소 학술연구비 지원에 의하여 연구되었음.
** 국립 상주대학교 컴퓨터공학부 교수

1. 서 론

급변하는 해운 환경 속에서 국가 경쟁력의 우의를 확보하기 위해 항만 시설의 첨단 기술을 필요로 하고 있다[7],[13]. 그러나 현재 국내 항만(부두)에서 컨테이너(Container)의 입반출되는 시간 등에 대한 관리가 수동으로 이루어지기 때문에 물류 유통 시간이 많이 소요되어 정시에 컨테이너들을 선적하는데 있어 장애 요인으로 대두되고 있다. 정부에서도 항만운영에 있어 항만의 생산성을 높이고, 국내 및 해외 항만 이용자들에 대한 서비스를 향상시키기 위하여 항만 산업에 기업 경영 방식을 도입하고 있다. 선박대기 시간을 단축하고, 단계적인 근로자 인원 감축과 대고객 서비스를 강화하여 국가 경쟁력을 높이기 위해서는 국내 항만 운영 관리에 대한 전산화의 실현이 반드시 이루어져야 할 것이다.

그러나 현재 컨테이너 터미널 게이트(terminal gate)당 근무 요원들이 직접 입반출되는 컨테이너 정보에 대한 기록 관리를 모두 수 작업으로 처리하고 있기 때문에 게이트에 다량의 컨테이너가 집중될 경우 병목 현상이 발생하게 되어 컨테이너 게이트로의 통과 시간이 지연될 뿐만 아니라 근무 요원의 컨테이너 정보의 수기 오류를 범하는 경우도 발생하게 된다. 따라서 이를 자동화함으로써 컨테이너의 게이트 체류시간을 최소화하고, 입반출되는 컨테이너에 대한 정확한 정보관리를 가능하게 될 것이다. 이러한 자동화를 위해서는 컨테이너 번호판에 대한 영상 처리 기술이 필요하다.

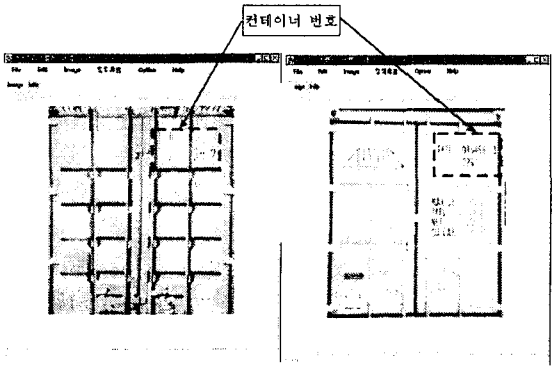
영상처리에 관련된 연구들 중에서 자동차 번호판 영상처리에 관한 연구가 있다[11],[16-19]. 그러나 현재 국내에서의 자동차 번호판 영상처리 기술에 대한 연구는 국내용 자동차 번호판 규격에 한정된 연구로 국한되어 왔다. 즉 일정한 규격의 번호판 크기 규격, 동일한 문자체 그리고 번호 및 기호들의 상대적 위치 등이 모두 규격화되어 있다[19]. 그러나 컨테이너 번호는 국가별, 회사별로 색상 및 문자체 등이 서로 다르기 때문에 자동차 번호 추출을 위한 영상처리기법과 다른 컨테이너 번호 추출을 위한 영상처리 기술이 필요하다. 그러나 현재 컨테이너 번호에 대한 영상처리 연구는 국내외적으로 전무한 상태이다. 따라서 본 연구에서는 국가별, 회사별로 다양한 형태를 갖는 컨테이너 번호를 추출하기 위하여 선형탐색법을 제안하여 이를 구현하였다. 제안된 선형 탐색법은 다양한 문자체 및 크기 색상 등으로 이루어진 컨테이너 번호에서 사용되는 문자들을 효율적으로 추출하는데 매우 유용한 방법이다.

또한 추출된 컨테이너 번호 영상을 본 연구의 결과물인 영상 처리 소프트웨어 도구를 통하여 사용자가 추출된 결과들을 직접 확인할 수 있게 하였고, 또한 추출된 문자를 차후에 신경망으로 학습시킬 수 있도록 후처리를 수행하였다. 따라서 이러한 개발 도구를 활용함으로써 국가별 회사별로 다양한 형태를 갖는 컨테이너 번호판 자동 추출뿐만 아니라, 컨테이너에 여러 종류의 복잡하고 다양한 영상 처리 기법[1-6]들을 손쉬운 방법 즉, 그래픽적인 사용자 인터페이스(graphical user interface)를 통하여 사용자들로 하여금 시각적으로 직접 확인하며 실험할 수 있도록 개발되어 대학등에서 영상처리 학습 도구로 활용될 수 있을 것으로 기대할 수 있다. 본 논문의 2장에서는 현재 해상에서 국제적으로 시행되고 있는 컨테이너 번호에 대한 규격을 설명하였고, 또한 기존의 자동차 번호판에서의 영상처리와의 다른 점을 기술하였다. 제3장에서는 컨테이너 번호의 영역을 추출하는 과정을 각 절에서 설명하였고, 제4장에서는 추출된 문자를 분리시키고 신경망에서 이를 학습하게 할 수 있도록 후처리 단계에 이르기까지 각 과정별로 설명하였다. 제5장에서는 컨테이너 영상물에 대한 농담 등 다양한 영상처리 알고리즘들을 실험하여 그 처리 결과를 직접 보여주었다.

2. 컨테이너 번호 영역 구성 및 특성

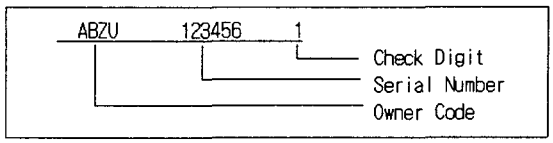
현재 자동차에서 사용되고 있는 번호판과 해상운송에 사용되는 컨테이너 번호판에 대해 먼저 살펴보기로 하자. 일반적으로 자동차 번호판 상단

부에는 관할 관청기호와 차종별 기호가 있고 하단부에는 용도별 기호와 등록 번호로 구성되어 있다. 또한 이들 국내 번호판들이 문자체 및 그의 크기가 동일하게 제작되어 있다. 그러나 컨테이너에서는 컨테이너 번호판에 대한 마킹(marking) 규정은 있으나, <그림 1>과 같이 문자체 및 문자 크기 등은 규정이 없어 국가별 또는 회사별로 다양하게 구성되어 있어 자동차 번호판 인식과 다른 처리 방법이 요구된다.



<그림 1> 두 종류 컨테이너의 전면 모습

국제적인 해상 운송에 사용되는 각 컨테이너 번호는 ISO 규격 및 조약[12]에 의거하여 아래 <그림 2>와 같이 규정되어 있다.

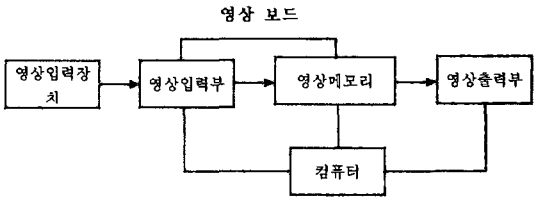


<그림 2> 컨테이너 마킹에 대한 ISO 규정

이러한 번호판은 컨테이너 전면의 우측 상단에 위치하고 있으며, 번호 앞의 네 개의 문자는 컨테이너의 소유주 회사명을 나타내고, 그 뒤의 숫자들은 일련번호 및 체크 디지트(check digit)를 나타낸다.

3. 컨테이너 번호 영역 추출

우선 본 연구에서 사용한 컨테이너 번호 영상을 획득할 수 있는 하드웨어(hardware) 구성도는 <그림 3>과 같다.



<그림 3> 영상처리 하드웨어 구성도

영상입력 보드에서는 영상 입력장치[5]로부터 들어온 아날로그 영상 데이터를 디지털로 변환시켜 주는 기능을 한다. 이렇게 디지털화된 영상을 보조기억장치에 저장시킬 수도 있고, 또한 사용자들이 시각적으로 직접 확인할 수 있도록 모니터에 나타나게 할 수도 있다.

획득된 영상으로부터 원하는 영역을 효율적으로 추출하기 위해서 영상에 대한 전처리 단계로서 그레이(gray) 영상 처리가 필요하다[8]. 현재까지는 모든 영상 대해 공통적으로 적용이 가능한 영상처리 기법은 존재하지 않고 있으며, 영상마다 그에 적합한 처리가 필요하다[9].

3.1 영상의 이진화 처리

영상의 특징을 해석하기 위해서는 영상에서 대상물을 추출하여 대상과 배경을 분리한 이진 영상으로 처리한다. 원 영상에서 좌표(m,n)에 대한 속성 정보 f(m,n)을 이진 영상 B(m,n)(B∈{0,1})으로 변환하는 것을 영상의 이진화라고 한다[1]. 영상의 속성값, 즉 명도값을 f로 하고, 임계값(threshold)을 T라고 하였을 때, 이진화값은 다음식에 의하여 결정된다[13].

$$B(m,n)=\begin{cases}1, & f(m,n) \geq T\text{의 경우}\\ 0, & f(m,n) < T\text{의 경우}\end{cases}$$

일반적으로 B(m,n) = 1 인 픽셀 집합을 대상물(object)영역이라 하고, B(m,n) = 0 인 픽셀의 집합을 배경(background)영역이라고 한다. 이렇게 함으로써 영상의 대상물 영역에 대한 영역 추출을 가능하게 할 수 있게 된다. 위에서 언급한 방법을 이용한 이진화 값을 결정하는 프로그램은 [코드 1]과 같다.

```
public ImageProducer toThresHold(int value, int max, int min)
{
    int x, y, bin;

    for ( y=0 ; y<height ; y++)
        for ( x=0 ; x<width ; x++)
            workPixel[y][x]= (pixels[y][x]<value) ? min : max ;
    return toImage(workPixel);
}
```

[코드 1] 이진화 값 결정 코드

이진화 처리 후의 영상에서 농담 분포의 차이가 많을 경우에는 임계값을 결정이 어렵기 때문에 그 영상의 히스토그램(histogram)을 평균화 필터를 사용하여 히스토그램 곡선을 평활화시켰다[10]. 이에 대한 구현 코드는 [코드 2]에서 볼 수 있다.

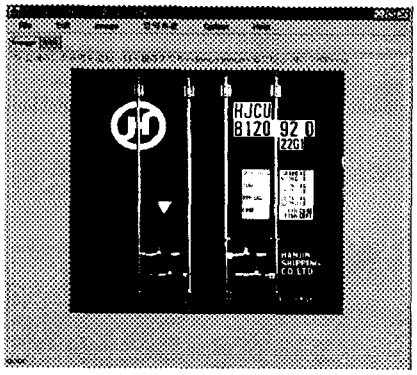
```
static void filterHist(){
    float sum;
    int j;
    for ( int a=0 ;  a<2 ;a++){
        for(int y=0; y<=255; y++) {
            sum = 0.0f;
            j = 0;
            for(int x=-15; x<=15; x++) {
              j++;
              if( (y-x) >= 0 && (y-x) <= 255)
                sum += hist[y-x];
            }
            hist[y] = (int)(sum / (float) j);
        }
    }//end of for{a}
}//end of filterHist()
```

[코드 2] 평활화 코드

컨테이너 영상에서 문자 영역을 추출하기 위한 임계값을 얻기 위해 (식 1)과 같은 방법을 사용하였다. 이는 문자 영역만을 뚜렷이 추출하기 위해 이진화시킨 영상의 히스토그램상에서의 픽셀값이 1인 수를 계산하여 그 영상의 크기로 나눈 값, 즉 영상의 평균 값을 선형함수로 하여 임계값을 결정한 것이다[19].

$$\begin{aligned}&\text{문자영역} \quad g(x,y) \geq T\\ &\text{배경영역} \quad otherwise\\ &T = f(\mu) = k \cdot u\\ &\mu = \frac{1}{N \cdot M}\sum_{y=1}^{M}\sum_{x=1}^{N} g(x,y)\end{aligned} \qquad \text{(식 1)}$$

여기서 $g(x,y)(1 \leq x \leq N, 1 \leq y \leq M)$는 번호판 영상 픽셀, T는 임계값, $f(\mu)$는 μ에 대한 선형함수, k는 임의의 상수, μ는 $g(x,y)$영상의 평균, 그리고 $N \cdot M$은 입력된 영상의 크기를 나타낸다. 이렇게 입력된 컨테이너 영상에 대한 이진화 처리된 결과는 <그림 4>에서와 같은 모습으로 나타나게 된다.



<그림 4> 이진화 처리후의 모습

3.2 영상의 잡음 제거 처리

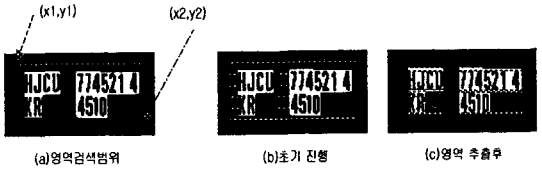
임의의 한 픽셀에 대한 주변 8개의 픽셀의 값을 조사한 후 검은 색이 2개 이하이면 잡음으로 간주하고 이를 제거한다. 이렇게 영상에 포함된 잡음 제거 방법을 구현한 형태가 [코드 3]에 나타나 있다.

```
public void  binaryMedian()
{
    int med=0;
    for ( int y=1;  y<height-1 ; y++){
      for ( int x=1 ; x<width-1 ; x++){
        med=0;
        for ( int i=y-1 ; i<=y+1 ; i++){
          for ( int j=x-1 ; j<=x+1 ; j++ )
            med+=rangeData[i][j]; //주변 색의 합을 구한다.
        }
        if(med<=2) rangeData[y][x]=0;  //잡음으로 간주
      }
    }
}
```

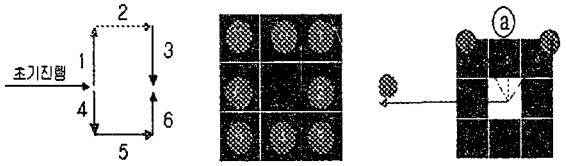
[코드 3] 잡음제거 구현 코드

3.3 선형 탐색법을 이용한 컨테이너 번호 영역 추출

영상에 포함된 잡음 제거 처리가 완료된 컨테이너 영상에서 우리가 원하는 번호 영역만을 추출하기 위해서는 선형 탐색법을 사용하였다. 선형 탐색법은 먼저 추출할 영역부터 결정하게 된다. <그림 5>의 (a)는 추출할 영역의 검색 범위를 설정하는 방법을 나타낸다. 검색범위의 설정은 임의의 좌측 상단(x1, y1)부터 시작해서 일정한 간격으로(본 연구에서는 10픽셀) 왼쪽에서 오른쪽 방향으로, 그리고 위에서 아래 방향으로 진행하면서 영역추출을 시작해야 할 부분을 찾는다(<그림 5>의 (b)). 이러한 영역 추출은 <그림 6>의 (b)에서와 같이 8개 방향으로 주변 픽셀들의 값을 조사하면서 이루어지며, 만약 흰 색의 픽셀 수가 3개 이상이면 그 지점부터 영역 추출에 들어간다. 따라서 이러한 영역 추출을 통하여 얻어진 영역 추출 결과는 <그림 5>의 (c)와 같이 나타나게 된다. 영역 추출은 <그림 6>의 (a)처럼 6가지 방향으로 테두리 쪽으로 밀착되어 진행하게 하면서 그 영역의 시작 지점(좌측 상단)과 영역의 끝점(우측 하단)을 찾게 된다. <그림 6>의 (a)에서 각각의 방향의 진행은 현재 지점이 이미 추출된 영역 내에 있거나, 또는 현재 지점에서 주변에 흰색이 1이하로 존재 할 경우에 중단된다.



<그림 5> 영역 추출 과정



(a) 검색진행 방향 (b) 인접 픽셀 (c) 검색될 픽셀

<그림 6> 영역 추출 결정 방법

<그림 6>의 (a)에서 영역 추출을 위해 1번 방향으로 진행할 경우에 <그림 6>의 (c)에서 보는 것처럼 3가지 방향으로 검색을 하고 다음 조건에 따라 진행 방향이 결정된다. 만약 <그림 6>의 (b)에서 6번 방향에 흰색이 있다면, 6번 방향으로 이동하고, 만약 6번이 검은색이고 7번이 흰색일 경우 7번 방향으로 이동하게 된다. 그러나 6번이 검은색이고, 7번이 검은색, 0번이 검은색이고, 1번이 흰색일 경우라면, 1번 방향으로 이동한다. 그러나 위의 조건에 맞는 경우가 없을 때는 기본 방향인 ⓐ번 방향으로 진행하게 한다. <그림 6>의 (c)에서 ① 번 방향이 다른 방향보다 긴 것은 <그림 6>의 (a)에서는 1번 방향의 경우 왼쪽 벽을 따라 진행하기 때문에 왼쪽으로

```
for(x=startx,y=starty;x>10&&x<width-10&&y>10&&y<height-10;y--){
    rangeSearch(y, x, color);
    white=whiteNumber(color);
    if((earlyDetected=earlyRanged(x,y)) !=-1) return earlyDetected;
    if(white<2) break;        //흰색이 두개 이하이면 진행을 멈춘다.
    if(color[6]==255 ) { x--; y++; } //좌 수평이동
    else if(color[6]==0 && color[7]==255 ) x--; //좌상
    else if(color[6]==0 && color[7]==0  && color[0]== 0
          && color[1]==255 ) x++; //우상
    x1=(x<x1) ? x : x1;
    y1=(y<y1) ? y : y1;
    x2=(x>x2) ? x : x2;
    y2=(y>y2) ? y : y2;
}
```

[코드 4] 영역 추출 검색 코드

는 무한히 진행할 수 있도록 함으로써 기울어진 영역도 추출 가능하다는 것을 나타낸다. 위와 같은 컨테이너 번호 영역 추출 방법을 프로그램 언어로 구현한 코드가 [코드 4]에 잘 나타나 있다.

3.4 추출된 영역 관리

추출된 영역은 그 영역의 데이터와 영역의 시작 지점(x1,y1), 영역의 폭과 높이를 하나의 객체로 링크드 리스트(linked list)를 이용하여 관리한다. 추출된 영역이 객체로 저장되면 잡음을 제거하고 그 영역에 몇 행의 픽셀들이 존재하는 지를 조사한다. 국부적으로 잡음 제거를 함으로써 이미지 전체를 처리하는 것보다 속도면에서 향상시킬 수 있게 된다. 추출된 영상이 2개행 이상일 경우에 그 행수만큼 영역을 분할하여 원 객체를 제거하고 분할된 객체를 링크드 리스트에 연결시키게 된다. [코드 5]에서는 추출된 영역을 링크드 리스트로 관리하는 기법을 프로그램 언어로 구현한 것을 보여주고 있다.

```
int[][] rangePD= new int[rangeHeight][rangeWidth];
for (int i=0 ;  i<rangeHeight ; i++)       //행
      for (int j=0 ; j<rangeWidth ; j++ )      //열
                  rangePD[i][j]= pixelData[y1+i][x1+j];

v.addElement(new RangePixelData(x1, y1, rangeWidth, rangeHeight, rangePD));
```

[코드 5] 추출 영역 관리 코드

<그림 7> 추출된 번호판 문자의 연결

<그림 7>은 위와 같은 알고리즘을 사용하여 추출된 번호 영역만 분리시켜 연결하여 관리할 수 있는 형태로 만든 결과를 확인시켜 주고 있다.

4. 문자 영역 분할과 후처리

4.1 개별 문자 분할

추출된 컨테이너 번호 영역에서 개별 문자를 추출하는 방법으로는 히스토그램을 이용하였다. 문자 사이의 공백 영역의 픽셀의 양에 따라 개별 문자를 분할하게 된다. 잡음으로 인한 공백 영역에 픽셀이 존재하게 될 때는 전체 픽셀 분포에 따른 비율로 임계값을 조정하게 된다. 아래 [코드 6]은 문자를 분할하는 방법을 구현한 프로그램을 나타낸다.

```
void countCharacter()
{
   int sum=0, startChar, endChar;
   int max=0, x=0, y=0 ;
   for ( x=0; x<width; x++){
       for ( ; x<width; x++){// 글자가 시작하는 부분을 찾는다.
                sum=0;
for ( y=0 ; y<height ; y++)
                    if(rangeData[y][x]==1)sum++;  //검은색이면 증가
                if(sum>=1 )break;
            }
            startChar=x;
       max=0;
       for (   ; x<width ; x++ ){// 글자가 끝나는 부분을 찾는다.
                sum=0;
                for ( y=0 ; y<height ; y++)
                    if(rangeData[y][x]==1)sum++;  //검은색이면 증가
          if(sum>max)max=sum;
                if(sum<1 )break;
            }
            endChar=x;
       if( endChar-startChar>=1 && max>8)
         if(checkIsCharacter(startChar, endChar) ) charCounter++;
   }
 }
boolean checkIsCharacter(int startX, int endX) {
    boolean isCharacter=false;
    int black, x, y;
    int charWidth=endX-startX;//글자의 폭
    int size=height*charWidth;
    for ( x=startX ; x<endX ; x++ )
        for ( y=0 ; y<height  ; y++)
            if(rangeData[y][x]==1)black++;
    if( black>(int)(size*0.2) && black<(int)( size *0.8) )
      isCharacter=true;
    return isCharacter;
 }
```

[코드 6] 문자 분할 코드

4.2 후 처리

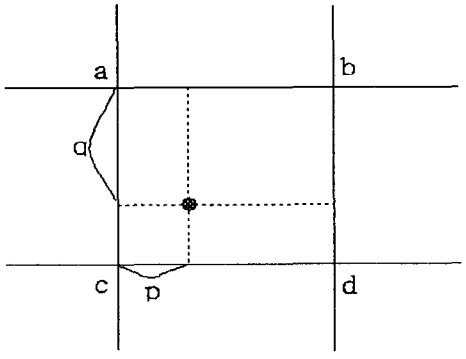
문자 추출 및 분할을 통하여 추출된 글자들이 서로 다른 크기로 형성될 가능성이 있다. 이를 위하여 추출된 글자들을 동일한 크기로 맞추어 조정하는 스케일링(scaling) 처리가 필요하고, 또한 선의 굵기가 불완전한 문자의 윤곽선을 동일한 굵기의 선으로 정리하기 위한 세선화(thinning) 과정이 필요하다.

4.2.1 스케일링(Scaling)

문자 추출을 위한 학습시키는 데이터의 양을 줄이기 위해서는 동일한 크기로 맞추어 줄 필요가 있다. 본 논문에서 사용한 스케일링 방법은 기존의 여러 가지 기하학적 처리 방법 중에서 처리 후의 화질상태 보존을 위한 정확성 및 처리 속도를 고려하여 쌍일차 보간법(Bilinear Interpolation) [3],[4],[14]을 활용하였다. 쌍일차 보간법은 대상 픽셀이 위치할 곳의 주변의 네 개의 픽셀들에 대한 밝기의 평균값을 취하는 것이다. 평균값 계산의 가중치는 원하는 픽셀의 위치에 이웃하는 픽셀들의 근접도에 비례한다.

예를 들어, 입력영상에 대한 임의의 점에 있어 주변의 네개 픽셀의 명도값이 a=10, b=40, c=5, d=15 이고 p=0.3, q=0.6 일 때 새로운 픽셀 O(x,y)의 명도 값은 다음과 같이 구할 수 있게 된다.(<그림 8>)

$$O(x,y) = (1-q) * ((1-p)*a + p*b) + q * ((1-p)*c + p*d)$$
$$O(x,y) = 0.4 * (0.7*10 + 0.3*40) + 0.6 * (0.7*5 + 0.3*15)$$
$$= 12.4 \simeq 12$$



<그림 8> 쌍일차 보간법의 예

[코드 7]은 위와 같은 스케일링 기법을 이용하여 프로그램 코드로 구현한 것을 보여 주고 있다.

```
public static void scaleLinear(int[][] in, int[][] out, float zx, float zy)
{
   int width = in[0].length;
   int height = in.length;
   int i, j, xBuf, yBuf;
   float x, y, p, q;
   int xs = (int)Math.floor(width/2);
   int ys = (int)Math.floor(height/2);
   int data;

   for (i = -ys; i < ys; i++) {
     for (j = -xs; j < xs; j++) {
      y = (float)Math.floor(i/zy);
      x = (float)Math.floor(j/zx);
      yBuf = (y < 0) ? (int)y : (int)y-1;
      xBuf = (x < 0) ? (int)x : (int)x-1;
      p = x - xBuf;
      q = y - yBuf;
      if((yBuf >= -ys) && (yBuf < ys-1) && (xBuf >= -xs)
                                        && (xBuf < xs-1)) {
       data = (int)((1.0f-q)*( (1.0f-p)*in[yBuf+ys  ][xBuf+xs]+
        p*in[yBuf+ys][xBuf+xs+1])+q *((1.0f-p)*in[yBuf+ys+1][xBuf+xs]
         +p*in[yBuf+ys+1][xBuf+xs+1]));
      }else{data = 255;//여백값}
      if(data < 0) data = 0;
      if(data > 255) data = 255;
      out[i+ys][j+xs] = data;
     }
   }
}//end of scaleLinear()
```

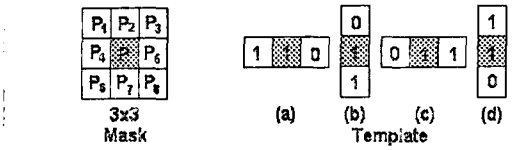
[코드 7] 스케일링 보정 코드

4.2.2 세선화

추출된 글자에서 선의 구조를 해석하기 위해 선의 폭을 섬세하게 하여 한 픽셀로 이루어진 중심선을 추출하는 세선화 작업이 필요하다. 본 논문에서 사용되는 세선화 알고리즘은 영상 안의 각 픽셀을 3×3의 이웃 윈도우 안에서 조사하여 각 영역이 세선화할 때까지 각 영역의 경계선을 한번에 한 픽셀 두께씩 벗겨내는 것이다. 본 논문에서는 문자의 본질적인 구조를 보존하면서 선도형으로 변형시키는데 효과가 있어 자동차 번호판 등 문자를 추출할 때 많이 활용되고 있는 Zhang-Suen 알고리즘[3][4]을 사용하여 세선화를 시켰다. 아래 예시는 본 논문에서 사용한 세선화

알고리즘을 설명한 것이고, [코드 8]은 추출된 문자를 세밀하게 작업하기 위한 세선화 처리 방법을 토대로 하여 프로그램 언어로 구현한 코드이다.

<3x3 마스크를 이용한 세선화 알고리즘>



[Critical point 알아내기 결정 방법]

[1] P is critical point if (P2=0 and P3=1) or (P7=0 and P8=1)
[2] P is critical point if (P4=0 and P1=1) or (P5=0 and P3=1)
[3] P is critical point if (P2=0 and P1=1) or (P7=0 and P6=1)
[4] P is critical point if (P4=0 and P6=1) or (P5=0 and P8=1)

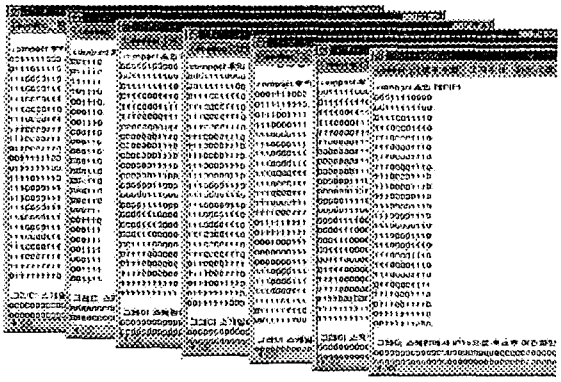
[End point 결정 방법]

[1] P1 + P2 + P3 + P6 + P7 +P8 = 0
[2] P1 + P4 + P6 + P3 + P5 +P8 = 0
[3] P1 + P2 + P3 + P6 + P7 +P8 = 0
[4] P1 + P4 + P6 + P3 + P5 +P8 = 0

[세선화 과정 절차]

Step 1. 입력패턴에서 네 단계에 따라 해당하는 모든 픽셀을 지운다.
[1] Remove all pixels that match (a) and are removable.
[2] Remove all pixels that match (b) and are removable.
[3] Remove all pixels that match (c) and are removable.
[4] Remove all pixels that match (d) and are removable.

Step 2. 삭제점이 없을 때까지 위의 과정을 반복.



<그림 9> 문자를 분리시킨 형태의 모습

```
/** 세선화*/
public static int[][] thinning(int[][] inputImage, int width, int height)
{
   final int maxLoop = 100;
   boolean remain = true;
   int iteration = 0;
   int[] tmp = new int[9];
   int[][] pixelArray = new int[height][width];
   int[][] tmpArray = new int[height][width];
   int i, j, k, count, i2, j2;

  /* inputImage[][] 을 tmpArray[][] 및 pixelArray[][]에 복사한다. */
  /* 리턴되는 값은 변경된 pixelArray[][] 이다. */
   for (j = 0; j < height; j++)
      for(i = 0; i < width; i++)
         pixelArray[j][i] = tmpArray[j][i] = inputImage[j][i];
   while(remain && iteration < maxLoop) {
      iteration++;
      remain = false;
      for (k = 0; k < 4; k++) {
        for (j = 1; j < height - 1; j++){

          for (i = 1; i < width - 1; i++) {
            count = 0;
            for(j2 = j-1; j2 <= j+1; j2++)
              for(i2 = i-1; i2 <= i+1; i2++)
                tmp[count++] = pixelArray[j2][i2];
            if(match(k,tmp) && !critical(k,tmp) && !endpoint(k,tmp)){
                  tmpArray[j][i] = 0;
                  remain = true;
              }
          }
        }
          for (j = 0; j < height; j++)
              for (i = 0; i < width; i++)
                  pixelArray[j][i] = tmpArray[j][i];
      }
   }
    return (pixelArray);
}
```

[코드 8] 세선화를 위한 코드

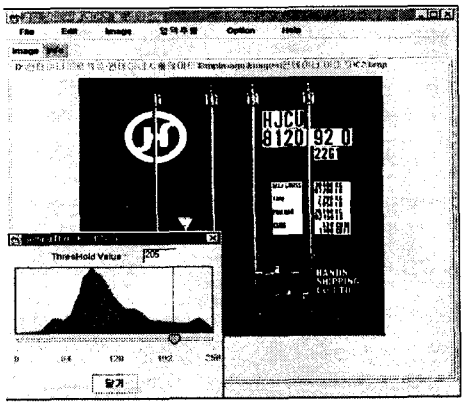
<그림 9>는 위와 같은 방법을 통하여 문자 영상을 분리시킨 결과를 보여주고 있다. 이 결과물은 컨테이너 문자를 인식할 수 있는 신경망 회로의 입력에 그대로 넣어 학습시킬 수 있는 학습 패턴(pattern)으로 사용할 수 있다.

5. 교육 도구로서의 영상처리 S/W

본 컨테이너 영상 처리 시뮬레이터는 영상 처리를 통해 컨테이너 번호를 추출할 수 있을 뿐만 아니라 여러 가지 다양한 종류의 영상 처리 기법들을 그래픽적인 사용자 인터페이스를 사용하여 간편한 방법으로 다양한 알고리즘을 적용시켜 실험할 수 있도록 개발되었다.

5.1 히스토그램에 의한 농담 분포 실험

이미 앞에서 임계값을 기준으로 한 영상 테이터를 처리하는 과정을 살펴보았다. 히스토그램 방법을 통한 농담 분포를 사용한다면, 각 농담값에 픽셀이 몇 개가 존재하는 지에 대한 빈도 수를 알아 볼 수가 있어, 적절한 임계값 설정을 수식이 아닌 사용자가 시각적으로 직접 확인하며 결정하게 하는 것이 가능하다. <그림 10>은 획득된 컨테이너의 영상에 대한 히스토그램을 나타낸다. 이러한 히스토그램을 나타내는 윈도우(window)에서 조절 버튼(button)을 사용하여 사용자가 임의로 임계값을 직접 조절하며 실험할 수 있다.



<그림 10> 임계값 조절 실험

5.2 윤곽선 찾기 위한 영상처리 실험

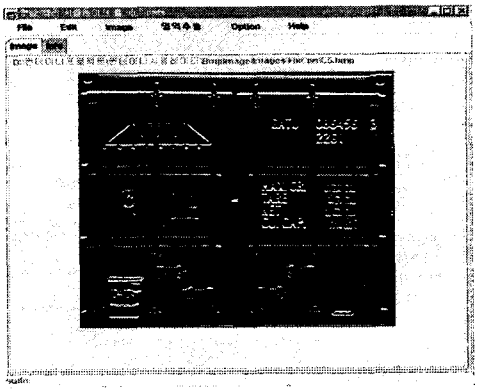
윤곽선은 물체의 외곽을 나타내는 선으로서 영상처리의 차원에서 영상을 특징 짓는 선으로서 매우 중요한 요소이다. 윤곽선 추출을 통하여 특정 물체를 추출한다든지 또는 면적과 주위의 크기를 측정할 수 있게 하는 것이다.

5.2.1 수평 방향 미분 영상 처리 실험

이 실험은 영상의 윤곽이 수평으로 농담치가 급격하게 변화하는 부분에 대해 함수의 변화량을 취하여 미분 연산이 윤곽선 추출에 이용되는 것이다. 윤곽선을 추출함에 있어서 수평 방향의 미분을 취하려면 다음과 같은 알고리즘으로 처리하게 된다[12].

$$\Delta F(X, Y) = |F(X, Y-1) - F(X, Y+1)| \quad (식\ 2)$$

즉, 수평 방향으로의 미분은 어느 한 점 $F(X,Y)$을 기준으로 할 때, X좌표는 같고, Y좌표만 중심 픽셀의 상하에 해당하는 픽셀들의 차이값을 구할 수 있게 된다. <그림 11>에서는 이러한 방법을 이용하여 주어진 컨테이너 영상을 수평 방향 미분 실험한 결과를 보여주고 있다.



<그림 11> 수평방향 미분 영상 처리

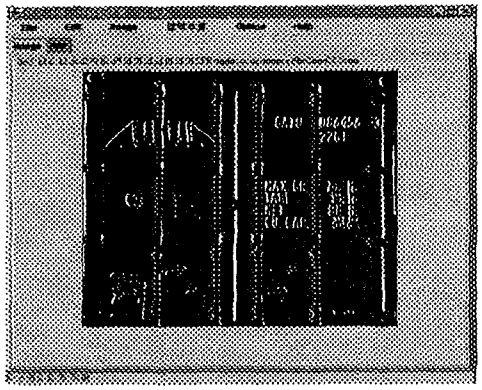
5.2.2 수직 방향 미분에 의한 영상 처리 실험

이 실험은 영상의 윤곽이 수직으로 농담치가 급격하게 변화하는 부분에 대해 함수의 변화량을 취하여 미분 연산이 윤곽선 추출에 이용되는 것이다. 윤곽선을 추출함에 있어서 수직 방향의 미분을 취하려면 다음과 같은 알고리즘으로 처리하게 된다. 수직 방향의 미분은 다음과 같다[12].

$$\Delta F(X, Y) = |F(X+1, Y) - F(X-1, Y)| \quad (식\ 3)$$

즉, 수직 방향으로의 미분은 어느 한 점 $F(X,Y)$을 기준으로 할 때 Y좌표는 같고, X좌표만 중심 픽셀의 좌우에 해당하는 픽셀들의 차이 값을 구할 수 있게 된다. 이러한 방법을 이용하여 주어

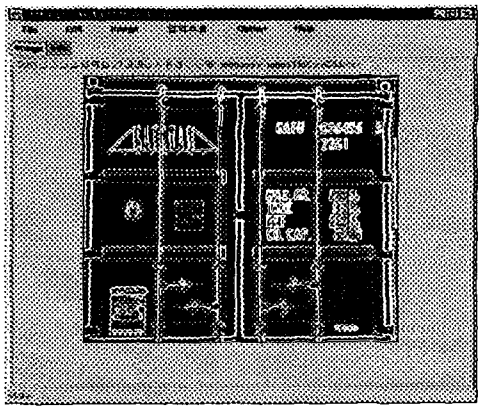
진 컨테이너 영상을 수직 방향 미분 실험한 결과를 <그림 12>에서 보여주고 있다.



<그림 12> 수직 방향 미분 영상 처리

5.2.3 수직/수평 방향 미분 영상 처리 실험

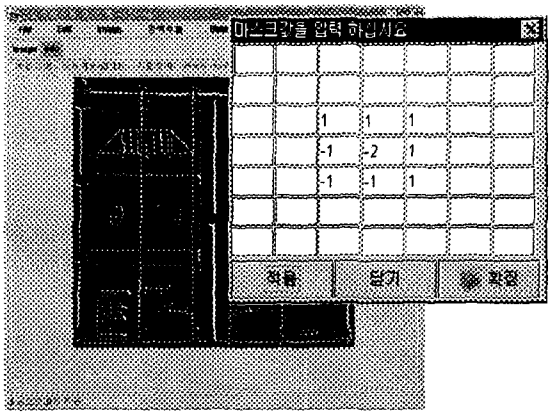
수직 수평 방향의 미분은 앞에서 설명한 두 가지 미분 방법을 이용하여 구할 수 있다. <그림 13>은 수직 수평 미분에 의한 영상 처리한 결과를 보여주고 있다.



<그림 13> 수직/수평 미분 영상 처리

5.2.4 Template matching에 의한 윤곽선 추출

Template matching[15]이란 윤곽을 나타내는 패턴을 미리 정하여 사용하고 영상의 각 화소 값에 근사한 값을 선택하는 방법이다. <그림 14>는 Prewitt 방법[1],[14]의 행렬값 중의 한 종류의 마스크를 이용하여 처리한 모습을 보인 것이다. 또한 <그림 14>와 같은 마스크 입력 메뉴 윈도우에 다양한 마스크를 입력시켜 실험자가 개발한 여러 가지 마스크 값들을 입력시키면서 간편하게 실험 결과를 시각적으로 확인할 수 있다.



<그림 14> Prewitt 방법을 이용한 영상처리

5.2.5 영상의 2차 미분 실험

2차 미분은 윤곽의 강도만을 검출하는데 이용되며, 디지털 영상에서는 다음과 같은 식에 의하여 처리된다[10].

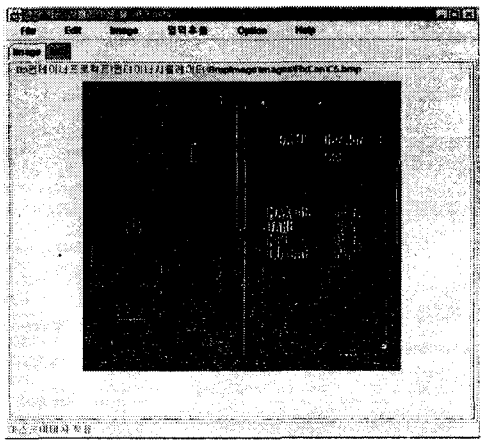
$$L(x,y) = 4f(x,y) - (f(x,y-1) + f(x,y+1) + f(x-1,y) + f(x+1,y))$$

(식 4)

디지털 영상은 데이터가 일정 간격으로 나란히 형성되어 있기 때문에, 인접 픽셀간의 차이를 연산으로서 미분을 근사시키는 방법을 사용하였다. <그림 15>는 (식 4)를 이용한 2차 미분을 실행한 결과를 보여주고 있다.

6. 결 론

본 연구에서는 입반출되는 컨테이너 정보의 자동 관리를 위한 컨테이너 자동 인식 시스템 개



<그림 15> 영상의 2차 미분 실험

발에 활용될 수 있는 컨테이너 영상 처리 시뮬레이터를 개발하였다. 본 연구의 결과물인 컨테이너 영상 처리 시뮬레이터를 통하여 국가별 회사별로 다양한 형태를 갖는 컨테이너 영상에서 컨테이너 번호판 자동 추출뿐만 아니라, 그 추출 번호를 사용자가 시각적으로 직접 관찰할 수 있게 하였고, 또한 추출된 문자를 차후에 신경망으로 학습시킬 수 있도록 후처리 단계까지 수행시켰다. 더우기 그래픽적인 사용자 인터페이스를 사용하여 간편한 방법으로 획득된 영상에 다양한 영상 처리 알고리즘을 적용시키면서, 그 결과물을 시각적으로 즉시 확인하며 실험할 수 있도록 개발되어, 대학등에서 영상처리 학습 도구로 활용될 수 있을 것으로 기대된다.

참고문헌

[1] Ahmed, M., Image Processing, McGraw-Hill, 1995.

[2] Barid, H.S., Bunke, H. K. Yamamoto, Structured Document Image Analysis, Spring-Verlag, 1992.

[3] Bassmann, H. and Besslich, P.W., Ad Oculus: Digital Procsssing, International Thomson Publishing, 1995.

[4] Baxes, G. A. : Digital Image Processing, Jhon Wilesy & Sons, 1994.

[5] Blissett, R.J., Stennett C. and Day R. M., "New Techniques for Digital CCTV Processing in Automatic Monitoring," Ottawa-VNIS '93, pp.137-140, Oct. 1993.

[6] Canny. J., "A Computational Approach to Edge Detection," IEEE Trans. Pattern Anal. Mach. Intell., PAMI-8(6), pp. 679-698, 1986.

[7] Dougherty, E. J., Simmons, G. R., "GRAIL the Container terminal of the Future", 14th Ship Technology and Research (STAR) Symposium, Apr. 1989.

[8] Floyd, R.W., and Steinberg, L., "An Adaptive Algorithm for Spatial Gray Scale," Society for Information Display 1975 Symposium Digest of Technical Papers, pp. 36-37, 1975.

[9] Gose, E., Johnsonbaugh, R. and Jost, S.: Pattern Recognition and Image Analysis. Prentice-Hall PTR, 1996.

[10] Haralick, R.M., " Digital Step Edges from Zero Crossing of Second Order Directional Derivatives," IEEE Trans. Pattern Anal. Mach. Intell., PAMI-6(1), pp. 58-68, 1984.

[11] Hirano M., "Development of Vehicle-following Distance Warning System for trucks and Buses," Ottawa-VNIS '93, pp.513-516, Oct. 1993.

[12] HuerTas, A., Medioni, G., "Detection of Intensity Changes with Subpixel Accuracy Using Laplacian-Gaussian Masks," IEEE Trans. Pattern Anal. Mach. Intell, PAMI-8(5), pp651-664, 1986.

[13] Neil Davidson, World Container Terminal: Global Growth and Private Profit,1998, Drewy.

[14] Pratt, W.K., Digital Image Processing, 2nd edition. New York: John Wiley &

Sons, 1991.

[15] Proakis, J.G., and Manolakis, D.G., "Introduction to Digital Signal Processing", Macmillan, 1988.

[16] Taktak R., Dufaut M, and Husson R., "Road Modeling and Vehicle Detection By Using Image Processing," 1994 IEEE Internaitonal Conference on System, Man and Cybermetics, pp.2153-2158, Oct. 1994.

[17] Yoshida M., "Optical Vehicle Detector for Traffic Control," Ottawa-VNIS '93, pp.154 -156, Oct. 1993.

[18] 최영진, 오영환, 나가지마, "컴퓨터 영상처리를 이용한 차량번호판 추출 방법," 전자공학회논문지, 제24권, 제2호, pp.309-314, 1987.

[19] 조보호, 정성환, "특정 영역 기반의 자동차 번호판 인식 시스템," 한국정보처리학회논문지, 제6권 제6호, pp.1686-1692 1999.

저자소개

최창훈(崔昌勳)

1988년 2월 명지대학교 전자계산학과(학사)
1990년 2월 서강대학교 대학원 전자계산학과(공학석사)
1997년 8월 서강대학교 대학원 전자계산학과(공학박사)
1990년 1월~1990년 9월 대우통신(주)
1995년 10월~1996년 2월 미국 AT&T(NCR) 기술파견 연구원
1997년 9월~현재 국립상주대학교 컴퓨터공학부 전임강사/조교수
관심 분야 : 컴퓨터구조, 병렬처리시스템, 컴퓨터시뮬레이션