바코드 인식/검사를 위한 영상인식 알고리즘

An Image Recognition Algorithm for Bar-Code Recognition/Inspection

김기순, 최종문, 김준식

Kee-Soon Kim, Choung-Moon Choi, Joon-Seek Kim

요 약

본 논문에서는 산업용으로 주로 사용되고 있는 바코드(code 93)를 비전시스템을 이용하여 자동으로 인식할 수 있는 알고리즘을 제안하였다. 제안된 알고리즘은 입력된 바코드 영상에 대해 회전에 관계없이 각도를 자동 추출하여, 모듈(module)을 구성하는 화소들을 추출한다. 각 모듈에 대해 적응적인 방법으로 바(bar)와 스페이스(space)를 구성하는 엘리먼트(element) 값을 구하고, 심벌 문자들의 엘리먼트 값을 9개의 그룹으로 나누어 바코드 값을 인식한다. 여러 종류의 바코드 영상을 대상으로 모의실험을 수행하여, 제안한 알고리즘의 성능을 검증하였다.

Abstract

In this paper, we proposed the automatical recognition algorithm of bar-code using a vision system, which can be used in the industrial application(code 93). The proposed algorithm extracts the pixels which consist of the bar-code modules unrelated to rotation, then that obtains the elements which consist of bars and spaces. After the obtained elements are divided by nine group, the value of bar-code is recognized. The performance of proposed algorithm is verified through the simulation. The proposed one has good performance.

Keywords: 영상인식, 바코드, 비젼 시스템, 바코드 인식, 회전보정

I. 서 론

최근 바코드는 백화점이나 편의점, 슈퍼마켓 등 유통 분야에서 판매 관리와 자동 수·발주 관리 업무로 우리의 일상 생활과 밀접하고 광범위하게 사용되고 있으며, 공장에서는 생산 관리, 품질 관리, 자재 및 완제품 입출고 관리 등에 적용되어 생산성 향상은 물론 원가 절감에 크게 기여하고 있다. 또한 사무 자동화 분야에서는 출퇴근 관리, 자산 관리, 자료 관리 등에 적용되어 큰 효과를 보고 있다[1]. 또한 ID카드에도 바코드를 부착하여 출입관리, 도서와 도면 대여 관리 등에 활용하고 있다. 그 외에도 운송, 병원, 우체국, 국방부 등 그 응용범위가 날로 확대되고 있는 추세이다[2]. 전 세계적으로 사용되고 있는 바코드의 종류만 해도 수십 종류가 되고 있으며, 각 분야, 또는 국가마다 다른 바코드를 사용하고 있다. 따라서 본 논문에서는 세계적으로 어느 정도 표준화가 이루어져 사용되고 있는 바코드 중에서 산업용, 특히, 자동차, 의료, 국방, 선박 등에서 사용되고 있는 바코드(code 93)를 자동 인식하는 알고리즘에 대해 연구하였다. 지금까지 바코드 리더(reader)라는 기기를 이용하여 바코드를 인식하고 있다. 바코드 리더는 크게 스캔부분과 바코드를 인식하는 디코더 부분으로 나누어진다. 바코드 리더의 원리를 보면 광원으로부터 주사된 빛은 심벌의 바와 스페이스에 의해 반사되고, 수광부에서 이 빛의 양을 감지하여 그 크기에 따라 전류를 발생시켜 인식하게 되어있다. 바코드 리더의 스캔 거리에 따라 분류하면, 접촉형과 비접촉형 스캐너로 나누어지며, 판독 가능한 거리는 접촉형 스캐너의 경우 2㎜이하, 비접촉형 스캐너의 경우 5m이상까지 가능하다. 바코드 디코더는 스캔된 바코드 데이터 신호를 받아 해독을 하고, 해독된 데이터를 용도에 따라 컴퓨터에 전송하거나 자체 내의 응용 프로그램으로 응용하는 기능을 가지고 있다. 실제 2㎝이상 스캔거리를 갖는 바코드 리더의 경우 상당히 고가이다. 하지만 자동화에 사용하는 바코드 리더의 스캔 거리는 그 이상이 되어야 하기 때문에 가격도 더 고가가 된다. 또한 판독률을 최대한 높이기 위해 심벌방향과 스캔방향을 상대적으로 조정해야한다. 스캔거리가 5m이상인 고성능 바코드 리더는 레이저 스캐너를 장착하고 있으며, 바코드의 스페이스 부분에 검은 점, 바 부분에 흰 점이 존재할 경우, 바나 스페이스를 오독할 수 있으며, 또한 바코드의 변이 고르지 않으면 바와 스페이스가 정상 폭보다 넓거나 좁게 오독되어 에러를 발생시킬 수 있다. 반면, 본 논문에서 제안한 비전시스템에 의한 바코드 인식 방법은 바코드의 회전에 무관하게 바코드의 영역을 찾아 여러 라인으로 스캔한 값을 참조하

기 때문에 오독률을 최소화할 수 있다. 줌 렌즈에 의한 확대가 가능하기 때문에 스캔거리에 무관하고, 바와 스페이스 넓이의 변화에 대한 영향을 최소화시킬 수 있으며, 바와 스페이스상의 백색(salt & pepper)잡음이나, 흐리게 인쇄된 바코드의 경우는 영상처리 알고리즘을 적용하여 정확한 값을 추출할 수 있다. 또한 공장 자동화 시스템과 같이 반복적인 작업에 본 논문에서 제안한 시스템을 적용하게 되면 고가의 바코드 리더를 대체하는 효과를 얻을 수 있으며, 바코드 인식률도 높일 수 있고, 인식된 바코드 정보를 바로 데이터 베이스화하거나 응용할 수 있는 장점이 있다.

II. 바코드의 구조

1. 바코드 구조

바코드는 다양한 폭을 가진 바(bar, 검은 막대)와 스페이스(space, 흰 막대)의 배열 패턴으로, 정보를 표현하는 부호(code) 또는 부호체계이다. 바코드 심벌의 구조상 최소단위는 모듈(module) 또는 X 디멘젼이라고 부르며, 1비트의 값을 가진다. 1개 또는 여러 개의 모듈이 모여서 바와 스페이스를 만드는데, 이들을 엘리먼트(element)라고 한다. 즉, 엘리먼트들은 1개 또는 복수 개의 2진수 비트(0이나 1)를 포함하고 있다. 엘리먼트는 흑백의 색깔로 표현되므로 눈으로 구분할 수 있는 기본 요소가 된다. 엘리먼트들은 모여서 심벌 문자(symbol character)를 만드는데 이는 1개의 ASCII 문자에 상응한다. 심벌 문자들은 모여서 완전한 심벌을 만드는데 이는 1개의 레코드(record)에 상응하는 정보를 포함한다[2]. 그리고 정보를 바코드로 표시하는 방법에는 여러 가지가 있는데, 이를 정한 규칙(또는 원리)을 바코드 심벌로지(symbology)라고 한다. 심벌로지는 데이터 표현의 연속성에 따라 불연속형과 연속형으로 나누게되며, 불연속형은 각 심벌과 심벌 사이에 갭(gap)이 존재하고, 연속형은 갭이 존재하지 않고, "바"로 시작하여 "스페이스"로 끝나고, 다음 심벌의 시작 엘리먼트에 의해 구별된다.

1. Code 93의 심벌로지 구조

Code 93은 모든 ASCII 데이터를 표현할 수 있는 연속형 심볼로지이며, 1982년 인터멕(Intermec)사에 의해 개발되었는데, 현재 고밀도의 데이터 응용분야와 code 39를 보완하는 응용 분야에 사용이 증가하고 있다. Code 93이라는 이름은 그림 1에서와 같이 심벌 문자를 구성하는 9개의 모듈(9X) 중 3개의 바가 존재한다는 데서 연유했다.

심볼의 필드 구조는 그림 2에서 보는 것 같이 시작 문자(start), 데이터, 검증 문자(C), 검증 문자(K), 종료 문자(stop), 종료 바(termination bar)로 구성된다. 시작 문자와 종료 문자는 사각형(□)으로 표시되는데, 종료 문자로 사용될 경우에는 뒤에 종료 바(1X)가 추가되어 비트 패턴이 1010111101(10X)이 된다. 심볼 문자들은 9개의 모듈(9X)로 구성되며, 3개의 바와 인접한 스페이스로 조합된다. 바는 1X, 2X, 3X의 요소폭을 가지며(시작/종료 문자는 예외적으로 4X의 바를 포함) 스페이스는 1X, 2X, 3X, 4X의 요소폭을 가진다. 바의 모듈(1X)은 비트 1로 스페이스 모듈은 비트 0로 표시된다.

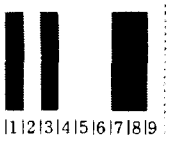


그림 1. 심벌 문자 구조

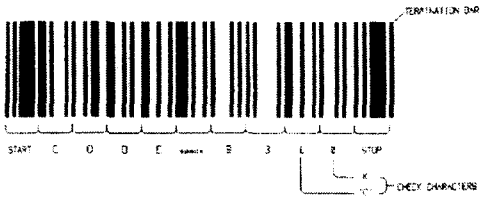


그림 2. Code 93의 심볼 구조

III. 제안한 바코드 인식/검사 알고리즘

1. 전체적인 구성

그림 3에 본 논문에서 제안한 알고리즘의 전체적인 흐름도를 나타내었다.

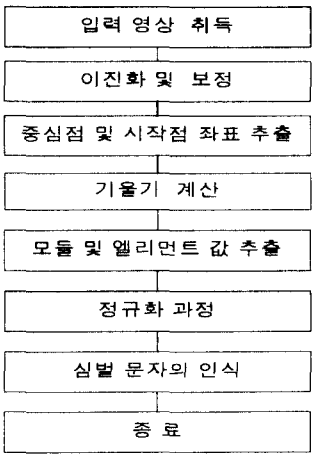


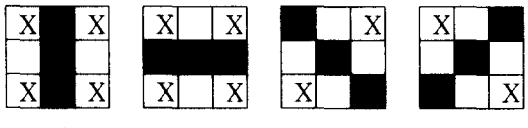
그림 3. 제안한 알고리즘의 전체 흐름도

바코드 영상을 입력한 후 이진화 과정을 통해 이진영상으로 변환한다. 이 과정에서 발생하는 오류를 여러 가지

형태의 마스크를 이용하여 보정 하게된다. 바코드 영상의 외각라인을 검색하여 중심점의 좌표를 구한 후 중심점 좌표를 중심으로 31×31 블록을 적용하여 블록의 외각라인을 시계방향으로 검색하여 시작점(바)에 해당하는 좌표를 구한다. 시작점의 좌표에서부터 chain code의 방향성분을 이용하여 바의 양단 좌표((x1, y1), (x2, y2))를 구한 후 두 좌표를 이용하여 기울기를 구하게 된다. 기울기가 45° <θ<135° 일 경우, 바의 양단 좌표를 지나가는 법선을 구해 법선의 y축 접점 두 개를, 기울기가 0° ≤θ≤45° 또는 135° ≤θ<180° 인 경우, x축 접점 두 개를 이용하여, 두 접점을 11등분한 후 바코드를 라인 스캔 할 10개의 좌표를 구하게 된다. 이렇게 얻은 바코드에 관한 모듈 값 및 엘리먼트 값을 이용하여 바코드의 시작과 종료의 바가 정상적인지 확인 후, 비정상적일 경우, 앞 뒤 데이터를 반대로 입력한 후 정규화과정을 거치게 된다. 즉, 모듈값의 배수에 관한 수로 정규화 값을 구하게된다. 이렇게 얻어진 정규화 값을 이용하여 심벌 문자를 데이터 베이스와 비교하여 바코드를 인식한다.

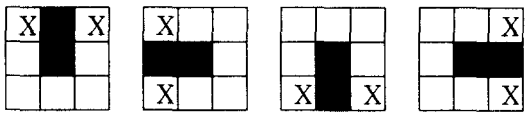
2. 이진화 및 보정

본 논문에서는 비트 패턴을 모듈로 변환 후 모듈의 "X"문자를 제거하고, 정규화 과정을 통해 모듈의 배수만으로 심벌을 구분하는 알고리즘을 제안하였다. 그림 2를 예로 들면, 비트 패턴은 "101000110"이고, 이 비트 패턴을 모듈의 배수로 나타낸 값은 바와 스페이스를 구성하는 엘리먼트라고 부르며, 엘리먼트 값은 "1X, 1X, 1X, 3X, 2X, 1X"이다. 여기서 X를 제거하는 정규화 과정을 거치게 되면, "1, 1, 1, 3, 2, 1"이 되고, 이들을 정규화 값이라 부른다. 이 정규화 값의 합은 9가 되고, 정규화 값을 이용하여 심벌 문자를 인식한다. 입력된 바코드 영상은 256 level을 갖는 명암 영상으로, 제안된 알고리즘을 적용하는데 용이하도록 입력영상을 이진화하여[3, 4] 이진영상으로 변환하였다. 바코드 영상은 이진화를 거치면서 또는 회전에 의해 바 성분 이외의 잡음이 발생하게 되고, 그대로 바의 방향성분을 이용해 좌표를 찾으면 좌표값의 오차가 있을 수 있다. 따라서 정확한 좌표값을 얻기 위하여 방향성분에 영향을 줄 수 있는 잡음을 제거하고, 오목한 부분은 흑화소를 추가하는 과정을 거치게 된다. 그림 4는 한 화소 두께의 바를 복원하는 여러 형태의 마스크를 도시했다. 그림 4(a), 그림 4(b)에서는 수직/수평 방향으로 한 화소 두께로 존재하는 바를 복원하고, 그림 4(c), 그림 4(d)는 대각/반대각 방향으로 한 화소 두께로 존재하는 바를 복원한다. 여기서 X는 don't care이다. 복원 방법은 마스크에서 don't care가 아닌 부분을 흑화소로 변환한다. 그림 5에서는 볼록한 부분을 제거하는 여러 형태의 마스크를 나타내고 있으며, 복원방법은 중심의 흑화소를 제거한다. 그림 6에서는 대각/반대각 방향으로 놓여있는 흑화소를 제거하는 마스크의 형태를 도시했으며, 중심의 흑화소를 제거하여 복원한다. 그림 7은 오목한 부분을 복원하기 위한 마스크로, 오목한 부분에 한 화소만 백화소로 존재할 경우 적용한다.



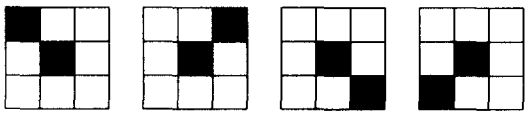
(a) (b) (c) (d)

그림 4. 한 화소 두께의 바 복원 마스크



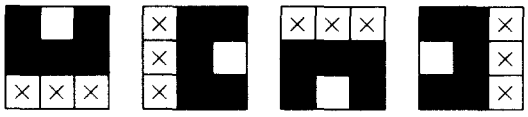
(a) (b) (c) (d)

그림 5. 볼록한 부분 제거 마스크



(c) (d) (e) (f)

그림 6. 독립적인 대각/반대각 성분 제거 마스크



(a) (b) (c) (d)

그림 7. 오목한 부분 복원 마스크

3. 중심점 및 시작점 좌표 추출

입력영상에 대해 바코드와 문자영역을 포함하는 최소 사각형의 좌표를 구하고, 최소 사각형의 좌표에 대한 중심좌표(x_{middle}, y_{middle})를 구한다. 이 중심좌표를 기준으로 31×31블록에 해당하는 영상의 외각부분만을 시계방향으로 검색하면서 화소값이 백화소에서 흑화소로 변하는 두 개의 최외곽화소 좌표((x_{start1}, y_{start1}), (x_{start2}, y_{start2}))를 구한다. 이 좌표는 바코드를 구성하는 바의 외각점이 되며, 이 좌표를 시작점으로 하여 chain code를 적용한다. 두 번째 좌표(x_{start2}, y_{start2})는 첫 번째 찾아진 바가 끊어진 경우를 대비해 구한 좌표이다. 여기에서 31×31블록을 사용한 이유는 바코드를 구성하고 있는 엘리먼트의 최대 넓이의 화소수는 대략 14화소 정도가 된다. 따라서 중심좌표(x_{middle}, y_{middle})를 중심으로 31×31블록을 검색하면 반드시 두 개 이상의 바를 검색할 수 있다.

4. 기울기 계산

바의 양단 좌표((x_{start1}, y_{start1}) 또는 (x_{start2}, y_{start2}))를

시점으로 그림 9에 보이는 3×3 마스크를 적용하여 chain code[5-7]를 실행한다. 실행방법은 0에 해당하는 화소부터 시계방향으로 흑화소를 검색하면서 존재하는 첫 번째 흑화소의 방향값을 저장하게 된다. 그림 9에서 p는 바로 이전에 찾아진 흑화소의 좌표가 되며, 첫 번째 p좌표는 (x_{start1}, y_{start1})(또는 (x_{start2}, y_{start2}))가 된다. 계속 반복 실행하다가 새로 찾은 좌표가 (x_{start1}, y_{start1})(또는 (x_{start2}, y_{start2}))와 같으면 chain code를 종료한다. Chain code는 바가 끊어진 경우 정확한 좌표를 구할 수 없다. 따라서 끊어짐 보정을 하게되는데, 바의 평균적인 길이는 460화소 정도이다. Chain code에서 찾아진 바의 길이와 평균길이의 오차가 ±5% 이상이면, 앞에서 구한 두 번째 중심좌표 (x_{start2}, y_{start2})를 이용하여 chain code를 실행하게 된다. 실행 방법은 앞의 방법과 동일하다.

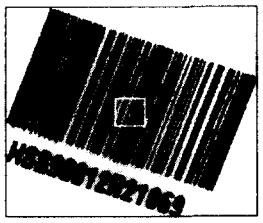


그림 8. 시작점을 찾기 위한 마스크

7	0	1
6	p	2
5	4	3

그림 9. 3×3 chain-code 방향성분 마스크

Chain code에 의해 구해진 검은색 바의 방향 성분값을 이용하여 검은색 바의 양 끝점의 좌표를 구한다. 이때, (x_{start1}, y_{start1})(또는 (x_{start2}, y_{start2}))에서부터 방향 성분값을 이용하여 좌표값을 찾아간다. 표 1은 방향 성분값과 정방향으로의 좌표값 변환을 나타내고 있으며, 표 2는 방향 성분값과 역방향으로의 좌표값 변환을 나타내고 있다.

표 1. 정방향 좌표변환

방향성분	좌표값 변환
0	x-1, y
1	x-1, y+1
2	x, y+1
3	x+1, y+1
4	x+1, y
5	x+1, y-1
6	x, y-1
7	x-1, y-1

표 2. 역방향 좌표변환

방향성분	좌표값 변환
0	x+1, y
1	x+1, y-1
2	x, y-1
3	x-1, y-1
4	x-1, y
5	x-1, y+1
6	x, y+1
7	x+1, y+1

그림 10(a)는 정방향 좌표복원 마스크를 나타내고 있으며, 그림 10(b)는 역방향 좌표복원 마스크를 나타내고 있다.

7	0	1
6	p	2
5	4	3

(a) 정방향 좌표 복원

3	4	5
2	p	6
1	0	7

(b) 역방향 좌표 복원

그림 10. 좌표복원 마스크

$$x_1 = x_{start1} - f_0 - f_1 + f_3 + f_4 + f_5 - f_7$$
$$y_1 = y_{start1} + f_1 + f_2 + f_3 - f_5 - f_6 - f_7$$
$$x_2 = x_{start1} + b_0 + b_1 - b_3 - b_4 - b_5 + b_7$$
$$y_2 = y_{start1} - b_1 - b_2 - b_3 + b_5 + b_6 + b_7 \quad \cdots\cdots (1)$$
$$f_n,\ n=0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7$$
$$b_n,\ n=0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7$$

여기서 f_n은 정방향으로 진행하면서 방향성분 n이 125° 이상의 방향 바뀜이 발생하는 좌표까지, 방향성분 n을 갖는 화소의 개수이며, b_n은 역방향으로 진행하면서 방향성분 n이 125° 이상의 방향 바뀜이 발생하는 좌표까지, 방향성분 n을 갖는 화소의 개수이다. 식 (1)에서 바의 양 끝점 $((x_1, y_1), (x_2, y_2))$을 구하고, 바코드의 기울기 $\theta = \tan^{-1}\{(y_2 - y_1)/(x_2 - x_1)\}$를 구할 수 있다. 기울기가 증가하면 y축과의 접점을 구하기 어려워지므로, 기울기가 $45° < \theta < 135°$ 인 바코드의 경우 아래에서 설명한 방법 1을 사용하고, 기울기가 $0° \le \theta \le 45°$ 또는 $135° \le \theta < 180°$ 인 바코드의 경우, 방법 1를 사용하여 b/a값을 구하면 된다. 하지만 b값이 증가하기 때문에 구하기 어렵다. 따라서, $x \rightarrow y$, $y \rightarrow x$로 치환한 후 아래의 방법 2를 적용하여 바코드의 회전 보정 없이 자동으로 기울어진 각도를 인식하여 바코드의 값을 검색한다.

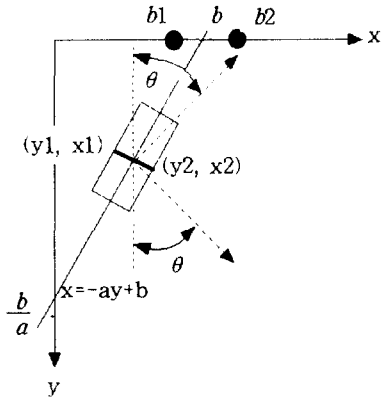
◆ 방법 1.

그림 11(a)에 도시한 것과 같이 직선의 방정식은 $y = ax + b$, 법선의 방정식은 $y = -ax + b$, y축과의 접점은 $b = ax + y$, 법선 방정식의 기울기 $-a = (y_2 - y_1)/(x_2 - x_1)$ 이다. 법선의 방정식이 검은색 바의 양 끝점의 좌표((x_1, y_1), (x_2, y_2))를 통과할 때 y축과의 접점은 $b_1 = ax_1 + y_1$과 $b_2 = ax_2 + y_2$이 된다. 여기서 구해진 b_1과 b_2사이를 11등분하게 되면 10개의 라인 스캔을 할 수 있으며, $0 \le x < 700$, $y = -ax + b_1 + (b_2 - b_1)/11n$, $0 \le y < 700$, $n=1, 2, \ldots 10$의 좌표값을 이용하여 바코드를 10개의 라인으로 스캔한다.

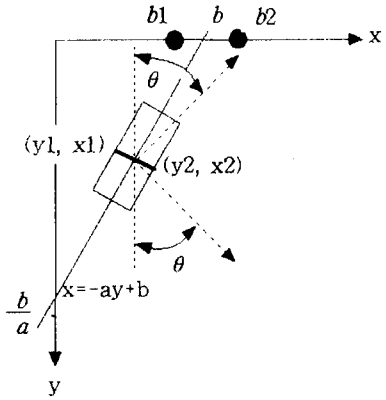
◆ 방법 2.

그림 11(b)에 도시한 것과 같이 직선의 방정식은 $x = ay + b$, 법선의 방정식은 $x = -ay + b$, x축과의 접점은 $b = x + ay$, 법선 방정식의 기울기 $-a = (x_2 - x_1)/(y_2 - y_1)$

이다. 법선 방정식이 바의 양 끝점의 좌표((x1, y1), (x2, y2))를 통과할 때 y축과의 접점은 $b_1 = x_1 + ay_1$과 $b_2 = x_2 + ay_2$이 된다. 여기서 구해진 b_1과 b_2사이를 11등분하게 되면 10개의 라인 스캔을 할 수 있으며, $x = -ay + b_1 + \{(b_2 - b_1)/11\}n$, $0 \le x < 700$, n=1, 2, ...10, $0 \le y < 700$의 좌표값을 이용하여 바코드를 10개의 라인으로 스캔을 한다.



(a) 45° < θ<135°



(b) 0° ≤θ≤45° 또는 135° ≤θ<180°

그림 11. 기울기에 따라 시작점 찾는 방법

5. 모듈 및 엘리먼트 값 추출

Code 93의 심벌로지에 의한 심벌은 "바"로 시작해서 "바"로 끝난다는 특징을 적용하여, 10개의 바코드 라인 스캔값을 저장하고 있는 각 배열에서 첫 번째 흑화소와 마지막 번째 흑화소를 각각 찾고, 첫 번째 흑화소로부터 바를 구성하는 흑화소 개수를 계수한 후 저장한 다음, 스페이스를 구성하는 백화소 개수를 계수한 후 저장한다. 이 과정을 마지막 번째 흑화소까지 실행한다. 결과적으로 10개의 배열에는 바에 해당하는 흑화소의 계수값과 스페이스에 해당하는 백화소의 계수값을 저장하고 있다. 10개 배열을 순서에 따라 각각의 평균값을 구하여 하나의 배열에 저장한다. 이때 각 배열값이 평균값의 50-150% 사이의 값을 갖지 않을 경우, 해당 배열값을 평균값으로 대치한 후 평균을 다시 계산한다. 이러한 과정은 바코드를 이진화하는 과정에서 나타날 수 있는, 이웃하는 두 바가 서로 붙어 넓어지는 경우와, 바의 넓이가 좁아지는 경우를 보정하기 위한 것이다. 다음으로 심벌의 시작 바와 마지막 바를 찾아 정상인지, 앞뒤가 바뀌었는지를 구별하게 된다. 그림 2에서 보듯이 바의 최대 요소폭은 3X이다. 즉, 최대 넓이를 갖는 바는 시작 바에서부터 다섯 번째에 위치하고, 마지막 바에서부터 뒤로 3번째에 존재한다. 이때, 바의 최대 넓이는 모듈의 3배 (거의 평균값의 3배와 같음)값을 갖기 때문에 쉽게 찾을 수 있고, 앞뒤가 바뀌었을 경우, 배열값을 수정하게 된다. 본 논문에서는 심벌 전체를 한번에 인식하지 않고, 심벌 문자별로 인식한다. Code 93의 심벌로지에서 심벌 문자를 나타낼 때, 바와 스페이스를 구성하는 엘리먼트는 6개로 이루어져 있다. 심벌 문자별로 인식하기 위해 배열에 저장 되어있는 바와 스페이스를 구성하는 엘리먼트(바와 스페이스를 구성하는 화소의 계수값)를 6개씩 나눈 후 정규화 과정을 거치게 된다. 우선 전체 평균값을 구한 후 각 배열의 값과 비교하여 평균값보다 작은 경우, 평균값으로 대치시킨다. 전체 평균을 대치시키므로 각 엘리먼트의 에러가 적어지게 된다. 여기서 평균값은 엘리먼트의 개수에 의한 평균값이 아니라, 모듈의 개수에 의한 평균값이다. 즉, code 93의 심벌로지에서 심벌 문자는 6개의 엘리먼트, 9개의 모듈로 구성되어 있다. 여기서 모듈값은 식 (2)에서와 같이 심벌 문자를 구성하는 모든 화소의 합과 심벌 문자를 구성하는 모듈 개수 9의 비로 정의된다.

$$\text{모듈값}(X) = \frac{\text{심벌 문자를 구성하는 모든 화소의 합}}{9} \quad \cdots\cdots (2)$$

6. 정규화 과정

이진화 과정에서 바의 넓이가 변하면 이웃하는 스페이스의 넓이에 영향을 준다. 반대로 스페이스의 넓이가 변하면 이웃하는 바의 넓이에 영향을 준다. 즉, 이웃하는 엘리먼트 값에 가장 영향을 많이 받게 된다. 따라서 적응적인 방법을 적용하여 각 6개의 엘리먼트 값을 정규화 시킨다. 간단히 말하면, 엘리먼트 값(1X, 2X, 3X, 4X, 5X)에서 수식적으로 X를 제거하는 것을 말하며, 본 논문에서는 정규화 값이라 부른다. 하지만, 실제 카메라로 취득한 바코드 영상의 엘리먼트 값에서 나오는 몫은 정수가 되지 않는다. 따라서 이 값들을 정수화 시키는 과정을 말한다. 식 (2)에서 모듈값을 얻을 수 있으며, 6개 엘리먼트의 값을 모듈값과 비교하여 모듈값보다 작을 경우, 정규화값은 1이 되며, 엘리먼트 값과 모듈값 차의 절반을

각각 이웃하는 엘리먼트 값에서 빼주고, 이웃하는 엘리먼트 값이 모듈값보다 작을 경우 빼지 않고 그대로 둔다. 다음으로 6개의 엘리먼트를 순서대로 2개씩 나누어 합을 구한 후, 이 합을 모듈값과 비교하여 근사적으로 몇 배수인지 계산한다. 우선, 2배수(2X)의 경우 정규화값은 유일하게 1과 1로 나타날 수 있다. 다음으로 3배수(3X), 4배수(4X)의 경우 식 (3)에 의해 근사화된 엘리먼트값을 구한 후 정규화 한다.

$$\text{element}_1\text{의 근사화값} = \frac{\text{element}_1}{(\text{element}_1 + \text{element}_2)} \times nX$$

$$\text{element}_2\text{의 근사화값} = \frac{\text{element}_2}{(\text{element}_1 + \text{element}_2)} \times nX$$

$$n=3, 4 \cdots\cdots\cdots (3)$$

여기서 n은 배수, X는 모듈값, element_1, element_2는 인접한 두 엘리먼트의 값이다.

표 3. 바코드 정규화의 예

[11110011000011111100000000]	← 바코드의 이진 패턴
[4 2 2 4 6 8]	← 바와 스페이스의 계수 값
[3.56 2.89 2.89 3.56 6 8]	← 심벌의 모듈값과 비교하여 조정
[3.51 2.99 2.99 3.51 6 8]	← 심벌 문자의 모듈(X)값과 비교하여 조정
[1X 1X 1X 1X 2X 3X]	← 엘리먼트 값
[1 1 1 1 2 3]	← 정규화

표 3에 바코드 정규화 예를 나타내었다. 라인 스캔된 바코드의 이진 패턴 데이터에서, 바와 스페이스의 값을 각각 계수한다. 이 계수한 값은 엘리먼트 값이 되고, 심벌의 전체 모듈값을 구한 후, 모듈값보다 작은 엘리먼트 값을 모듈값으로 대치한다. 실제는 전체 심벌의 모듈값이 되지만, 여기에서는 현재 심벌 문자의 모듈값을 구해 사용했다. 심벌의 모듈값은 (4+2+2+4+6+8=26)/9 = 2.89이다. 이 모듈값 보다 엘리먼트 값이 작을 경우, 두 값의 차의 절반(2.89-2=0.89/2=0.45)을 구하고, 이웃하는 엘리먼트 값과 모듈값을 비교하여 엘리먼트 값이 클 경우, 엘리먼트 값에서 이 값을 뺀 값(4-0.45=3.56)으로 대치하고, 적을 경우 그대로 둔다. 이번에는 심벌 문자의 모듈값을 구하고 각 엘리먼트 값과 비교하여 모듈값보다 작은 경우 모듈값으로 대치시키고 모듈값과 현재 엘리먼트 값의 차의 절반(2.99-2.89=0.1/2=0.05)을 구한 후, 현재 모듈값과 이웃하는 엘리먼트 값과 비교하여 엘리먼트 값이 클 경우, 엘리먼트 값에서 이 값을 뺀 값(3.56-0.05=3.51)으로 대치시키고, 작을 경우 그대로 둔다. 다음으로 각 배수 1X = (3.56+2.89+2.89+3.56+6+8=26.9)/9 = 2.99, 2X=5.98, 3X=8.97, 4X=11.96, 5X=14.95를 구하고, 최종적으로 앞에서부터 두 엘리먼트 값의 합(6.5, 6.5, 14)을 구하여 식 (3)에 의해 근사화된 엘리먼트값을 구하고, 가까운 정수 값으로 정규화한다.

7. 심벌문자의 인식

모든 심벌을 정규화값으로 변환 후, 정규화값을 이용하여 심벌 문자를 쉽게 찾기 위해 각 심벌 문자의 정규화값 6개 중에서 첫 번째, 두 번째 자리의 정규화값을 이용하여 9개의 그룹((1, 1), (1, 2), (1, 3), (1, 4), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2))으로 나눈다. 여기서 괄호 안의 숫자는 첫 번째, 두 번째 자리의 정규화값이며, 모든 심벌 문자들은 9개의 그룹중 한 그룹에 속하게 된다. 나머지 4자리의 정규화값으로 고유값을 만들고, 그림 12에 의해 미리 만들어놓은 심벌 문자의 고유값과 비교하여 심벌 문자로 인식한다. 그림 12에는 code 93의 심벌 문자들과 패턴 그리고 이진수로 나타낸 심벌 문자를 보여주고 있다.

표 3을 예로 들면, 정규화값이 111123이므로 (1, 1)그룹에 속하고, 나머지 4개의 정규화값에 의해 고유값 ($1\times2^3+1\times2^2+2\times2^1+3\times2^0=14$)을 구하고, 고유값에 해당하는 심벌문자 H를 얻을 수 있다. 이 방법에 의해 모든 심벌 문자들을 구할 수 있다. 이렇게 구한 모든 심벌 문자들 중 뒤에서 2번째와 3번째 심벌 문자를 사용해 검증문자로서 인식된 심벌의 에러 유무를 확인할 수 있다. 따라서, 심벌 문자들을 뒤에서 4번째까지 가중치를 주어 구한 심벌 문자와 뒤에서 3번째 검증문자와 비교하고, 다시 앞에서부터 이 검증 문자까지 가중치를 주어 심벌 문자를 구하고 뒤에서 2번째 검증문자와 비교하여 에러의 유무를 확인하고, 시작/종료 문자와 마찬가지로 해당값을 표시하지 않는다.

아래에 검증 문자를 산출하는 예를 보이고 있다.

① 심벌 인식값:

H S S 9 0 8 1 2 R 2 1 0 6 4

② 문자값:

40 51 51 25 16 24 17 18 50 18 17 16 22 20

③ 첫 번째 검증문자 가중치:

14 13 12 11 10 9 8 7 6 5 4 3 2 1

④ 문자값과 가중치를 곱한 후 합산:

$(40\times14)+(51\times13)+(51\times12)+(25\times11)+(16\times10)$
$+(24\times9)+(17\times8)+(18\times7)+(50\times6)+(18\times5)$
$+(17\times4)+(16\times3)+(22\times2)+(20\times1)=3318$

⑤ 첫 번째 검증문자 3318

Modulo 47=28, 문자값 28에 해당하는 심벌 S

⑥ 문자값:

40 51 51 25 16 24 17 18 50 18 17 16 22 20 14(검증문자)

⑦ 두 번째 검증문자 가중치:

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

⑧ 문자값과 가중치를 곱한 후 합산:

$(40\times15)+(51\times134+(51\times13)+(25\times12)+(16\times11)$
$+(24\times10)+(17\times9)+(18\times8)+(50\times7)+(18\times6)$
$+(17\times5)+(16\times4)+(22\times3)+(20\times2)+(28\times1)=3731$

⑨ 두 번째 검증문자 3731

Modulo 47=18 문자값 18에 해당하는 심벌 I

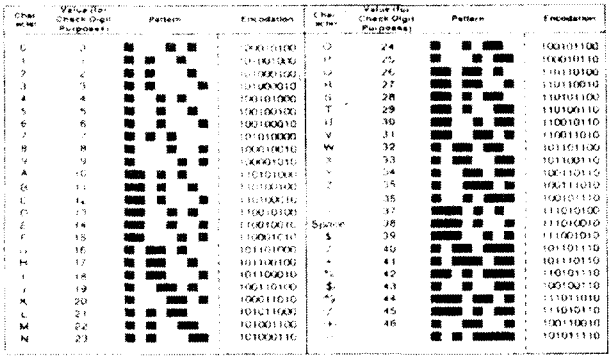


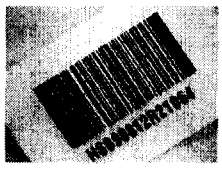
그림 12. Code 93의 심벌 문자 패턴

Ⅳ. 모의 실험 및 결과

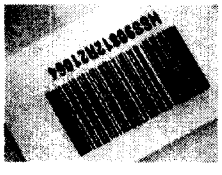
1. 실험 환경

본 논문에서는 산업용 code 93 바코드를 삼성 NetScan 카메라 (SNC-80/320)를 사용하여 거리를 달리하면서 영상의 크기는 700×700이고, 256 level을 갖는 명암 영상을 취득하여 사용하였다. 그림 13에는 여러 형태로 기울어진 입력 영상들이 나타나 있다. 여기서 기울기는 바코드 영상의 왼쪽에서 수직축으로 라인 스캔을 했을 때 가장 먼저 찾아지는 흑화소에서 수직선과 바코드 간 시계 반대방향으로의 기울어진 정도로 말한다.

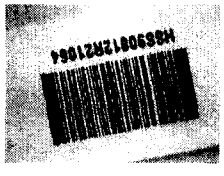
(a) 입력 영상1 (173°)



(b) 입력 영상2 (23°)

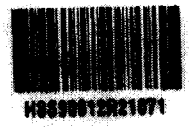


(c) 입력 영상3 (20°)



(d) 입력 영상4 (7°)

(e) 입력 영상5 (0°)



(f) 입력 영상6 (0°)

그림 13. 입력 영상

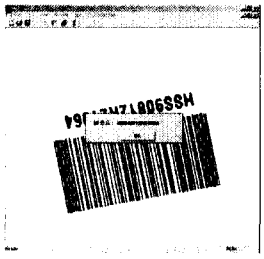
입력 영상 중에서 그림 13(c), 그림 13(d)에서 구한 데이터는 앞, 뒤가 바뀐 데이터가 된다. 따라서, 모듈과 엘리먼트 값을 구한 후 바코드의 앞, 뒤의 데이터를 바꾸면 된다. 입력영상은 code 93 심벌로지를 갖는 10개의 바코드 영상을 임의로 회전시켜 각각을 20개씩 취득하여 총 200개의 영상을 사용하였으며, Microsoft사의 visual C++ 6.0을 이용하여 알고리즘을 구현하고 모의 실험을 하였다.

2. 결과

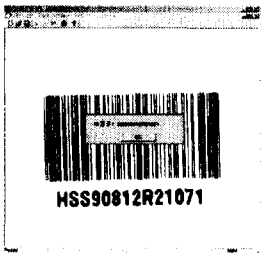
다음 영상들은 본 연구에서 제안한 알고리즘을 구현하기 위해 Microsoft의 Visual C++ 6.0을 이용하여 구축한 소프트웨어에서 바코드를 인식한 결과를 나타내고 있다. 영상 중간에 있는 다이얼로그 박스에 인식된 심벌 문자들을 표시하였다. 결과영상들을 보면 회전에 무관하고, 앞 · 뒤의 바뀜이 있는 영상(그림 14(c), (f))에서도 정확한 바코드 값을 인식하는 것을 볼 수 있다. 제안한 알고리즘을 적용하여 모의 실험한 결과 98% 인식률과 0.4초 정도의 처리시간이 걸렸으며, 바코드의 인쇄가 흐린 경우와, 표면이 불균일한 경우 잘못 인식된 결과가 나오기도 하였다.

(a)

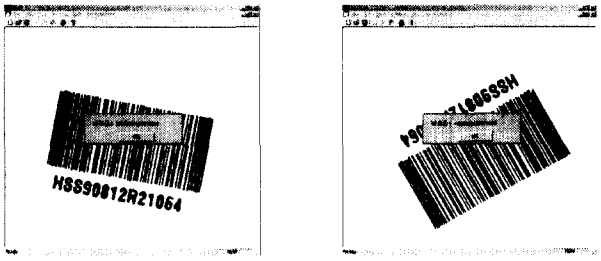
(b)



(c)



(d)



(e) (f)

그림 14. 결과 영상

V. 결론

본 논문에서는 산업용(특히, 자동차, 의료, 국방, 상업용 등)으로 주로 사용되고 있는 바코드를 비전시스템을 이용하여 자동으로 인식할 수 있는 알고리즘을 제안하였다. 비전시스템을 이용한 본 알고리즘은 바코드 영상을 거리에 무관하게 입력받을 수 있고, 회전된 바코드 영상은 보정없이 기울기를 계산하여, 10개의 라인으로 스캐닝하여 바코드 값을 추출하기 때문에 바코드의 부착 방향, 인쇄상태 및 이물질에 의한 오독을 최소화하였다. 또한, 본 알고리즘은 개인용 컴퓨터에 장착할 수 있는 비전시스템을 기반으로 하기 때문에 기존 바코드 인식 시스템에 비해 간단하고, 저렴하게 구축할 수 있으며, 검사 결과를 데이터 베이스화하여 손쉽게 이용할 수 있는 장점이 있다. 본 알고리즘을 모의 실험한 결과 인식율은 98%, 한 장의 바코드 처리시간이 0.4초정도 소요되었다. 기존 바코드 인식 시스템에 의한 인식율은 96%, 처리시간은 0.8초 정도 소요되었다. 인식율은 거의 비슷하게 나왔으며, 처리시간은 기존 바코드 인식시스템의 경우, 스캐너의 수동작에 의한 이동시간 때문에 약 2배 정도의 차이를 보였다. 따라서, 물류 입출고 관리 분야에 본 알고리즘을 적용한 인식 시스템을 이용하여 자동화하면, 처리비용 및 인건비를 절감할 수 있을 것이다.

앞으로, 바코드의 인쇄상태 및 표면의 불균일에 따른 오인식을 제거하기 위한 전처리/후처리 알고리즘에 대한 연구가 추가적으로 이루어져야 할 것이다.

접수일자 : 2001. 6. 26 수정완료 : 2001. 7. 18

참고 문헌

[1] 오 호근, 최신 바코드 기술 및 응용, 성안당, 1997년.
[2] 오 호근, 바코드 기술 및 응용, 성안당, 1997년.
[3] Howard E. burdick, *Digital Imaging*, McGraw-Hill, 1997.
[5] E. Bribiesca, "A New Chain Code," *Pattern Recognition*, Vol. 32, No. 2, pp. 235-251, 1999.
[6] A. Y. Wu, S. K. Bhaskar, "Parallel Processing of Region Boundaries," *Pattern Recognition*, Vol. 22, No. 1; pp. 165-172, 1989.
[7] Hong-Chih Liu and M. D. Srinath, "Corner Detection from Chain-Code," *Pattern Recognition*, Vol. 23-A, No. 1, pp. 51-67, 1990.

김기순(Kee-Soon Kim)
準會員
1997년 호서대학교 전자공학과
1999년 호서대학교 대학원 전자공학과 (공학석사)
1999년~현재 호서대학교 대학원 전자공학과 박사과정
관심분야 : GIS, 영상신호처리, 컴퓨터 비젼, 영상인식, 영상압축 등

최종문 (Choung-Moon Choi)
正會員
1974년 연세대학교 전기공학과
1977년 연세대학교 대학원 전기공학과 (공학석사)
1984년 연세대학교 대학원 전기공학과 (공학박사)
1979년 3월~1984년 2월 동양공업전문대학 전임강사
1984년 3월~현재 호서대학교 전기정보통신공학부 전기전공 교수
관심분야 : 제어시스템, 로보틱스 및 응용, 회로 및 시스템, 영상신호처리 등

김준식 (Joon-Seek Kim)
正會員
1987년 서강대학교 전자공학과
1989년 서강대학교 대학원 전자공학과 (공학석사)
1993년 서강대학교 대학원 전자공학과 (공학박사)
1993년 서강대학교 부설산업기술연구소 박사후 연구원
1994년~현재 호서대학교 전기정보통신공학부 전자전공 교수
관심분야: 영상신호처리, 영상압축, 컴퓨터 비젼, 영상인식, GIS 등