論 文
38~8~6

산업용 비젼 시스템을 위한 체인코더의 제작

Development of A Hardware Chain Coder for Industrial Vision Systems

李 炳 日*·申 有 植*·林 俊 弘**·徐 一 弘§·卞 增 男§§
(Byungil Rhee · You-Shik Shin · Joonhong Lim · Il-Hong Suh · Zeungnam Bien)

요 약

산업용 비젼 시스템에서 주된 문제중의 하나는 방대한 양의 영상정보를 어떻게 효과적으로 처리하며 그 시간을 단축시키는가 하는 것이다. 본 논문에서는 이 정보 처리시간을 단축시키기 위해 체인코딩 알고리즘을 하드웨어화에 알맞게 정리하고 수정하였고, 제안된 알고리즘에 의해 제작된 하드웨어 체인코더를 사용한 결과 소프트웨어에 의한 것보다 코딩시간이 대폭 단축되어 산업현장에서의 실시간 응용에 이용 가능함을 보였다.

Abstract- One of the major issues of the industrial vision systems is lengthy processing time due to bulky data for image coding. To reduce the processing time, a *modified chain coding algorithm* is proposed in such a way that it is more suitable for hardware implementation. A *hardware chain coder* is developed and used for learning and recognizing objects by extracting several features. It is shown that the desired vision system is much faster than a typical software based system so that it may be applicable to real-time industrial operations.

1. 서 론

비젼 시스템(vision system)은 사람의 눈에 해당하는 감지 계통(sensory system)으로서 가장 다양한 정보를 비접촉이며 수동적인 방법으로 얻어 낼 수 있기 때문에 전자공학이 발달하면서 그 응용분야는 넓은 범위로 확대되었다. 최근에 개발된 여러가지 고체소자 비젼센서, 예를들면 CCD, MOS등은 종래의 촬상관과 비교하여, 왜곡이 적고, 작고 가벼우며, 충격에 강하고, 저전압 하에서 동작할 수 있으며, 적은 양의 빛에서도 동작할 수 있는 등 여러가지 장점을 가지고 있을 뿐아니라 가격이 저렴하기 때문에 산업현장에서의 응용이 보다 손쉬워졌다. 산업 현장에서 이용될 수 있는 비젼시스템의 응용분야로는 첫째, 생산 라인에서 생산되거나 사용되는 각종 물체를 식별하여 분

*正 會 員 : 韓國科學院 電氣 및 電子工學科 博士課程
**正 會 員 : 漢陽大 工大 電子工學科 助教授·工博
§正 會 員 : 漢陽大 工大 電子工學科 副教授·工博
§§正 會 員 : 韓國科學院 電氣 및 電子工學科 教授·工博
接受日字 : 1988年 3月 29日
1次修正 : 1989年 1月 4日
2次修正 : 1989年 6月 26日

류하는 일, 둘째, 사람에 의해 행해지던 각종 검사를 비젼 시스템을 통하여 제품의 양, 불량 여부를 판정하는 일, 세째, 물체의 위치를 알아내어 매니플레이터로 하여금 집어 내거나, 제조, 조립 등 적당한 작업을 할 수 있게 하고, 매니퓰레이터의 다음 위치나 현재의 위치등을 계획, 검출하는 일등을 들수 있다.[1]

이러한 목적과 응용분야를 통해서 알아볼 수 있는 산업 현장에서 사용될 비젼 시스템이 가져야 할 특성으로서는[2,3], 첫째, 2차원 영상을 처리할 수 있을 것, 둘째, 영상으로부터 적당한 처리를 통하여 유용한 정보들을 추출해 낼 수 있을 것, 즉 물체의 위치 및 자세에 대한 정보을 알아내고, 물체를 인식하여 종류를 판별하며, 양, 부를 판정할 수 있을 것, 세째, 제조공정에 응용 가능하며 저 가격일것 등이다.

본 논문에서는 비젼 시스템을 산업현장에서 독립적으로 운용하거나, 로보트 제어 시스템의 시각 센서로도 사용할 수 있도록 하는 비젼 시스템을 개발하는 데에 그 목적을 두고 있다. 따라서 저 가격이면서도 실시간응용이 가능해야 한다. 하지만 일반적인 비젼시스템은 그 기능이 실시간 처리에 부적합하거나 매우 고가이어서 산업현장에의 응용이 어렵다. 산업 현장에서 가장 문제가 되는 것은 생산성(productivity)이며[4]이며, 이 생산성을 결정 짓는데는 여러 가지 요소가 있지만 그 중 큰 비중을 차지하는 것이 그 생산에 소요되는 시간이다. 따라서 생산성을 향상 시키려면 그 처리 시간을 단축시켜 실시간 처리가 가능하도록 하는 것이 가장 효과적이다. 산업용 비젼 시스템의 처리 시간을 느리게하는 요인은 크게 두 가지로 구분해볼 수 있다.[5] 첫째 요인은 2차원 디지탈 영상으로 표현되는 방대한 양의 데이터들이다. 현재 영상 입력매체로서는 비데오 카메라(video camera)를 가장 많이 사용하고 있는데, 그 정보량은 흑백 TV의 경우 하나의 영상, 약 8백만 bit가 1/30초마다 생성되는데, 현재의 마이크로프로세서 기술로는 256×256의 해상도를 가지는 하나의 영상을 탐색하는데만 0.3초 이상의 시간이 소요된다. 둘째 요인으로 지적될 수 있는 것은 영상 데이타를 충분히 빠르고도 견실하게(robust)처리할 수 있는 알고리즘의 부재이다.

산업 현장이라는 제한 조건은 인공 광원이나 기타 환경을 적절히 조절하여 명암의 구별을 뚜렷이 할 수 있다는 가정을 세울 수 있도록 해준다. 따라서 자연 영상(gray level image)으로 부터 2진 영상(binary image)을 얻을 때 물체의 모양에 대한 정보가 손상되지 않는다고 가정하고, 이렇게 하여 만들어진 2진 영상에서 다시 물체에 대한 정보만을 뽑아 내기 위한 부호화(Coding) 방법으로는 SRI Moduule에서 사용한 Run Length Coding과 Freeman이 제안한 Chain Coding[6]이 주로 사용되고 있다.

Run Length Coding은 그 구조가 간단하고 한 화면의 영상을 받아들이면서 동시에 Coding이 가능하기 때문에 SRI Module의 경우는 이 방법을 채용하고 있다. 그러나 이러한 Run Length Coding은 만들어진 Code로 부터 다시 물체의 경계를 추출해 내야 하며 이런 물체를 구분하는데는 다소 복잡함이 있다. 한편 Chain Coding은 그 특성상 한 화면의 영상이 마련된 상태에서만 가능한 방법으로써, 각 물체의 구분(isolation)을 따로 할 필요가 없으며, Code가 곧 경계경보의 나열이므로 따로 구해낼 필요가 없고, 각종 특징량(features)들을 찾아내는 방법이 간단해진다. 그러나 그 구조상 영상정보의 흐름을 곧바로 이용한 온라인 처리가 곤란하므로 Hardware화가 어려워지는 단점이 있다. 최근에는 반도체산업의 발다로 빠른 처리속도를 가지는 RISC 프로세서들이 등장하여 이를 이용하면 상당히 많은 시간단축을 꾀할 수 있으나, 매우 고가이므로 이들의 채용이 어렵다. 그래서 본 논문에서는 영상정보를 받아들이면서 동시에 경계점에 대한 정보를 만들어 저장하고 이로부터 경계의 시작점을 찾아서 경계의 진행방향과 조합하여 한 스텝만으로 코드를 생성하도록 Chain Coding 알고리즘을 Hardware화에 알맞는 형태로 구성하여 이를 저 가격의 범용 소자들로 구현하고 제작된 체인코더가 종전에 범용 마이크로 프로세서에서 software만으로 처리될 때 보다 약 1/8 정도로 정보처리 시간을 감소시킴으로써 실시간 동작이 가능하도록 하였다. 또한 이렇게 Hardware에 의해 생성된 Chain Code List를 적절한 처리를 거쳐 물체의 면적, 둘레의 길이, 모양 복잡도, 위치, 자세 등 특징량을 추출하여 각 물체를 기억시키고, 이를 기억된 물체와의 비교, 판단 근거로 삼아 물체인식을 하는데 이용할 수 있었다.

2. 체인코딩 알고리즘

적절한 역치 선정을 통해 얻어진 이진 영상으로부터 유용한 정보들을 얻기위해 여러가지 방법들이 이용될 수 있는데 체인코딩은 정해진 길이와 방향을 가진 선분들의 나열로써 경계(boundary)를 표현하는데 사용되는 방법이다. 이러한 체인코

딩은 4 또는 8 이웃관계(neighborhood)를 가지는 격자의 선정에 따라 각각 4개 또는 8개의 선분들로써 표현된다. 8-이웃관계를 이용하는 방법은 그 종류가 많아 더 복잡하지만 원래의 경계를 4-이웃관계보다 더 충실히 표현하므로 여기서는 8-이웃관계를 사용한다.

각 영상점(pixel)값이 0과 1로 표현되는 이진 영상에서 영상의 경계는 이웃해 있는 두 영상점 $p(i,j)$, $p(k,i)$에 대해 $|p(i,j)-p(k,l)|=1$인 경우, $p(i,j)$와 $p(k,l)$ 사이를 영상의 경계라 하자. 이러한 경계조건으로부터 중심 영상점 $p(i,j)$에 대하여 8-이웃관계(8-neighborhood)를 가지는 영상점들의 집합은

$$P_b(i,j)=\{p(k,l)|i-1\leq k\leq i+1,\quad j-1\leq l\leq j+1,\ k\neq i,\ l\neq j\}$$

로 표현되고, 만약 $p(k,l)\in P_b(i,j)$일때 $|p(i,j)-p(k,l)|=1$이면, $p(i,j)$를 경계점(boundary point)이라 한다. 경계점으로 정의되어진 점들은 이진 영상에서 물체와 배경의 경계를 이루고 있는 점들이다. 영상점들을 따라 체인코딩을 하기 위해서는 이진 영상 내에서 물체의 경계가 있는 위치를 알아야 한다. 이진 영상을 탐색하면서 경계점을 쉽게 구분해 내기 위하여 한 영상점의 주변에 위치하는 영상점의 값들을 좌측에 있는 영상점부터 반시계방향으로 돌며 여덟개의 영상점 값들을 한개의 바이트(byte)로 구성하여 각 영상점들의 특징값으로 두며 이것을 주변값(neighbor value)이라 부르겠다. 경계점 $p(i,j)$에 대한 주변값 $M(i,j)$는 8bit의 값이고 여기에서 주변값 $M(i,j)$의 각 bit를 $b_7,b_6\cdots b_0$라 하면, 이는 다음과 같이 정의된다.

$$\{b_7,b_6,b_5,b_4,b_3,b_2,b_1,b_0\}=\{p(i+1,j+1),p(i,j+1),p(i-1,j+1),p(i-1,j),p(i-1,j-1),p(i,j-1),p(i+1,j-1),p(i+1,j)\}$$

여기서, $p(i,j)=1$이다. 이 주변값은 체인코딩을 행할 때 물체의 경계를 찾아 코딩을 시작하고 경계를 따라 코딩을 계속해 나아가는데에 중요한 정보를 제공하게 된다. 또 이진 영상 내에서 체인코딩을 하기 위해서는 물체의 경계를 이루고 있는 경계점 중의 한 점을 찾아내어 경계를 따라 시계방향을 또는 반 시계방향으로 돌며 코드를 구하게 된다. 이때 체인코딩을 시작하는 경계점을 시작점이라 정의하자. 그리고 이진 영상에서 시작점을

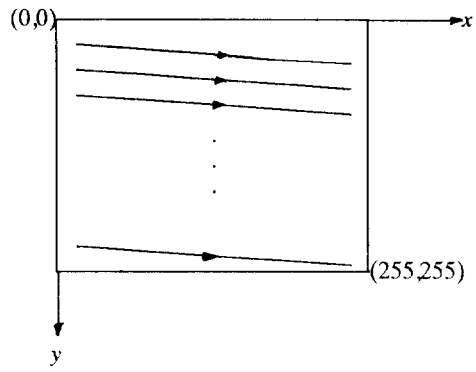

그림 1 영상좌표계
Fig. 1 Image Coordinate

찾아내는 것은 탐색(search)이라 부르자. 한편 시작점으로부터 체인코딩을 하는 과정, 즉 체인코드를 구해 내기 위하여 물체의 경계를 따라가는 것을 추적(tracking)이라 부르자. 이진 영상을 탐색하는 방법은 여러가지가 있을 수 있으나 통상 영상 메모리의 구성이 그림1과 같이 되어있다고 가정하고 메모리 주소를 증가시키며 모든 영상점에 대하여 시작점 여부를 점검하는 방법을 사용하였다. 이와같은 순차적인 방법은 매우 단순하면서도 체인코딩의 전 과정중 가장 많은 시간을 요하고 있다. 여기서 이진 영상을 미리 주변값 정보로 바꾸어 놓고 256경우의 주변값중 위와같은 탐색방법에 주변값이 시작점과 일치하는가를 점검하면 탐색 시간을 절약할 수 있다.

영상점의 값이 '0'에서 '1'로 바뀔때의 시작점(type A)들을 그림2(a)에, 그리고 영상점의 값이 '1'에서 '0'으로 바뀔때의 시작점(type B)들은 그림2(b)에 나타내어져 있다. 이 시작점들의 주변값은 영상탐색 순서 때문에 그림2(a)의 경우는 bit 1, 2, 3, 4 값이 모두 '0'이다. 그림 2(b)에서는 뒤에서 언급하는 체인코딩의 방향 추적규칙 때문에 발생하는 무한 추적의 경우를 만들지않게 하기 위하여 첫째, 3×3 격자(window)내의 상측 및 좌측 영상점의 값은 모두 '1'이어야하고(bit 1, 2, 3, 4, 5), 둘째, $p(i+1,j)=1$인 경우는 $p(i,j+1)=0$이어야 하는 조건을 만족하는 경계점을 시작점으로 두었다.

시작점의 위치를 알고나면 그때부터 경계점을 따라 추적하며 체인코드를 생성한다. 앞서 언급한 바와같이 추적방향을 반 시계방향으로 결정하였다. 한 경계점으로부터 다음 경계점은 경계점의 주변값과 전 단계의 체인코드(경계의 진행방향)

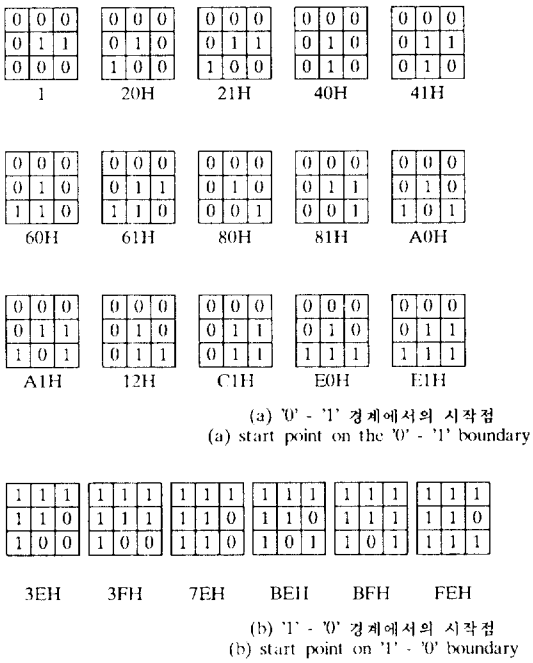

그림 2 시작점의 종류
Fig. 2 Start points

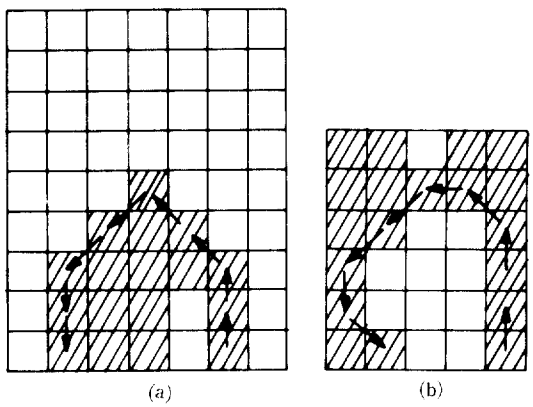

그림 3 추적의 종류
Fig. 3 Tracking Types

그리고 그때의 체인코딩 모우드(mode)를 이용해 결정 할 수 있고 이때 경계점 사이를 연결하는 방향벡터가 체인코드가 된다.

추적 방법은 크게 두가지로 나누어 지는데 이는 시작점의 형태에 따라 달라지므로 각각을 A추적, B추적으로 부르겠다. 그림3(a)와 같이 경계점을 중심으로 이전 단계의 경계점으로부터 반 시계방향으로 돌며 이웃관계에 있는 영상점중 경계점을 찾는 방법이 A추적이다. 또한 그림3(b)와 같이 경계점을 중심으로 시계방향으로 돌며 다음 경계점을 찾는 방법이 B추적이다. 그림3에서 보듯이 A추적은 type A의 시작점으로 시작되는 경계의 추적에 사용되며 B추적은 type B의 시작점으로 시작되는 경계의 추적에 사용된다. 다만, type B 시작점으로 시작되는 경계의 추적중 시작점에서는 영상 탐색 순서때문에 B추적 방법 대신 A추적 방법을 사용한다. 시작점의 경우는 전 단계의 체인코드 대신 영상 탐색 방향값을 사용하여 다음 경계점을 찾는데 사용한다. 따라서 type B 시작점에서 B추적 방법을 사용하면 그림3(b)에서의 시작점에서는 추적하고자 하는 경계가 아닌 바로 위에 있는 경계를 추적하는 결과가 생겨 추적에 실패하게 된다. 마찬가지 이유로 하여 시작점 다음부터는 B추적 방법을 사용해서 추적을 계속한다.

추적의 중단은 추적을 계속하여 추적을 시작했던 점으로 되돌아 온 경우와 추적과정중 주변값이 '0'이 되는 경우에 이루어 진다. 전자의 경우는 정상적으로 추적이 종료되는 경우이나 후자는 경계점이 아니어서 추적을 중단하는 경우이다. type B 시작점이 그림2(a) 같이 여섯가지의 경우만 존재하도록 하는 조건중 첫째, 경계점의 bit 1, 2, 3, 4, 5가 전부 '1'이어야 함은 탐색 순서에 의해 자명하게 얻어지는 조건이며, 두번째, $b_0=1$이면 $b_6=0$인 조건은 그림4와 같이 $b_0=1$이고 $b_6=1$인 경우를 생각해보면 시작점은 $p(i,j)$이나 추적을 계속하여 $p(i+1,j)$에 도달했을 경우 B추적 방법을 사용하게 되므로 다음 경계점은 $p(i,j)$가 아니라 $p(i,j+1)$이 선택되어 무한히 추적을 계속하게 되므로 추적의 중단이 일어나지 않게되어 목적을 달성할 수 없게된다. 따라서 이러한 조건을 고려한 시작점의 종류는 그림4의 경우를 제외한 여러가찌가 된다. (그림2(b))

이상 앞에서 언급된 경계점, 시작점, 그리고 추적을 이용한 이진 영상의 체인코딩 과정은 그림5에 표시되어 있다.

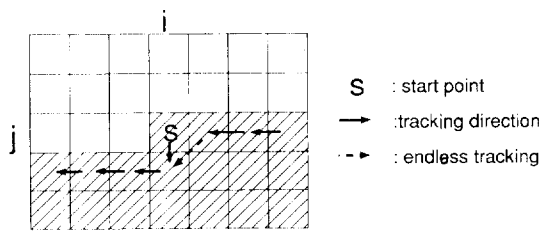

그림 4 무한 추적의 발생
Fig. 4 Endless Tracking

3. 하드웨어 체인코더(Hardware Chain Coder)

체인코더는 그 입력과 출력이 조합형 논리로 구성되는 부분은 룩업 테이블(look up table)로 구현하고 단순 반복형태의 알고리즘을 하드웨어화 하여 속도의 증가를 꾀하였다.

그림5의 흐름도를 이용하여 알고리즘을 하드웨어화 한 체인코더의 전체 구성도를 그림6에 보였다.

체인코더는 크게 주변값 변환부(neighbor-value coder, EM), 주변값 영상 메모리(neighbor-value image memory, NIM), 체인 테이블(CTM), 체인 리스트 메모리(CLM), 주변값 영상 address generator 및 마이크로 프로그램 제어부와 address decoder 등으로 나누어 진다. 또 CPU와 접속하기 위해 그림7과 같은 address map을 가지고 있다.

체인코더의 마이크로 프로그램 제어부는 총 네 가지로 구분되는 상태(state)를 가진다. 내장된 마이크로 프로그램은 각 상태 내에서의 순서(sequence)및 상태천이등을 제어하며 사용자의 명령에 의해 다른 상태로 천이될 수 있도록 프로그램 되어 있다. 각 상태간의 천이 관계와 사용자의 명령에 대한 상태도를 그림 8에 보였다. 먼저, 대기 상태(Idle)는 CPU로 부터 명령을 받거나, CPU가 체인코더 내에 있는 각종 레지스터 및 메모리를 엑세스할 수 있도록 하는 상태이다. 이 상

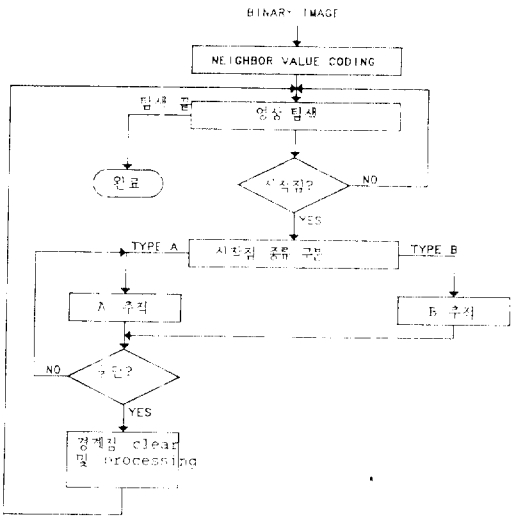

그림 5 체인코딩 과정
Fig. 5 Chain Coding Flow Chart

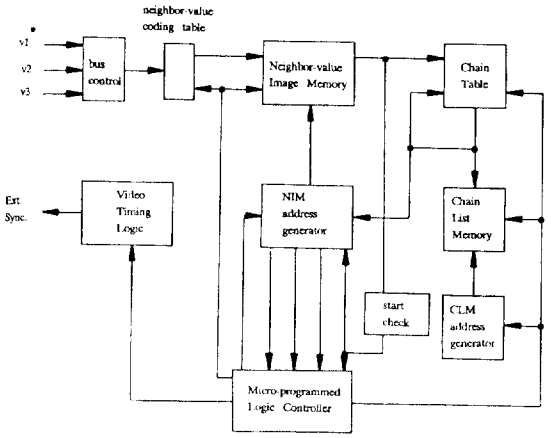

그림 6 체인코더의 전체 구성
Fig. 6 Chain Coder Structure

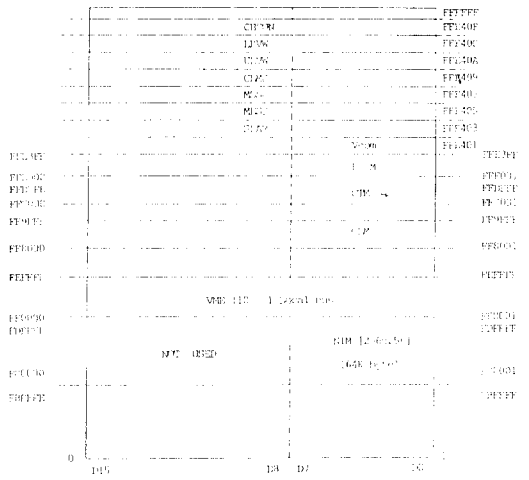

그림 7 체인코더의 Address Map
Fig. 7 Address Map Chain Coder

태에서는 체인코더 내에 있는 모든 메모리와 레지스터들의 주소및 내용은 각각 시스템의 address와 data bus에 접속된다. 따라서 CPU는 이 상태에서 주변값 코딩 테이블을 쓰고 NIM 상에서 추적된 부분의 주변값을 지우며 체인코드의 길이를 읽고 체인코드의 시작점 좌표값을 읽고 체인코더 상태 레지스터를 읽거나 쓰고 주변값 코딩 시작 명령과 메모리 명령을 내리며 체인 테이블과 체인코드 리스트를 쓰고 읽어낸다.

대기 상태에서 CPU로 부터 주변값 코딩 시작명령(MESC)이 내려지면 체인코더는 주변값 코딩상

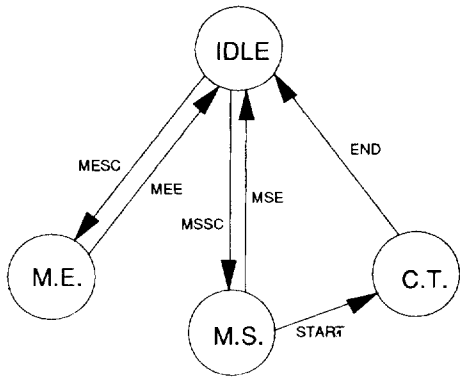

그림 8 체인코더의 상태천이
Fig. 8 State Transition of Chain Coder

태(ME)로 천이되며 3개의 비전 버스로 부터 입력되는 이진영상 정보는 이 중 하나를 선택하여 3×3 격자 및 주변값 코딩 테이블을 거쳐 주변값이 NIM에 저장된다. (그림9) 영상점의 좌표값이 (255,255)가 되면 발생되는 MEE 신호에 의해 제어부는 대기 상태로 되돌아 간다.

대기상태에서 CPU로 부터 메모리 탐색 시작명령(MSSC)이 내려지면 CPU에 의해 미리 지정된 탐색범위(좌상귀(UL), 우하귀(LR))내를 탐색하는 시작점 탐색상태(MS)로 된다. 그림2에 표시된 시작점들은 256×4 *bit*의 PROM에 입력되어 있어 그림10과 같이 시작점을 구분한다. 시작점이 발견되면 START 신호가 발생되고 제어부는 경계추적 상태로 상태천이를 행한다. 시작점이 검출되지 않

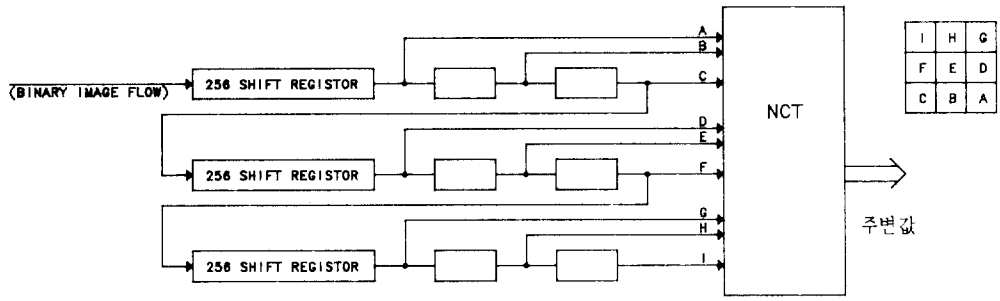

그림 9 주변값 변환부
Fig. 9 Neighvor-value coding part

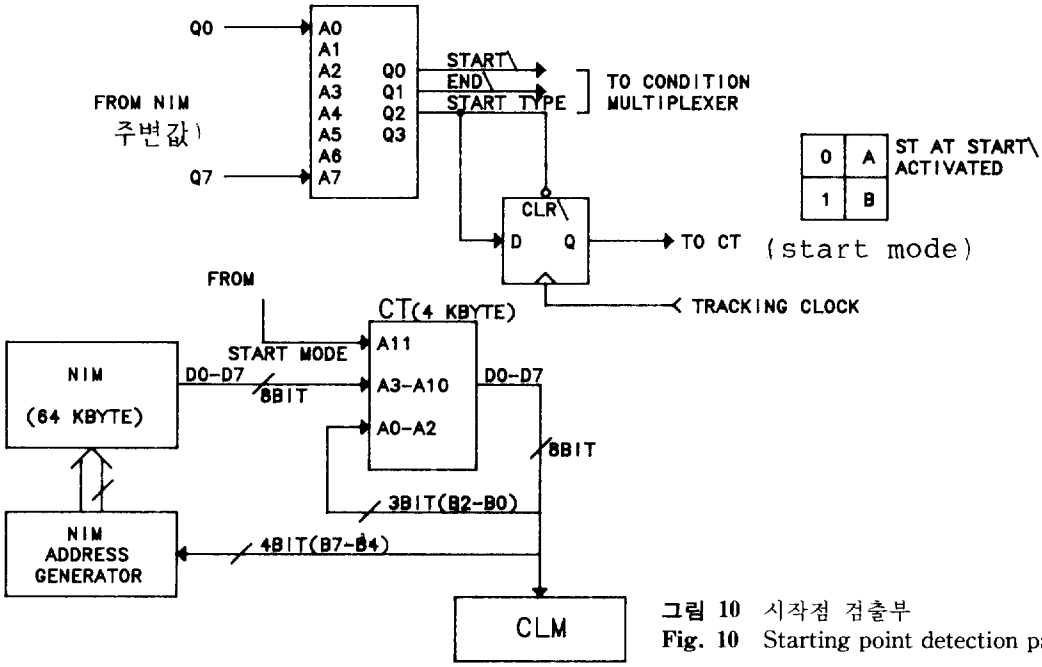

그림 10 시작점 검출부
Fig. 10 Starting point detection part

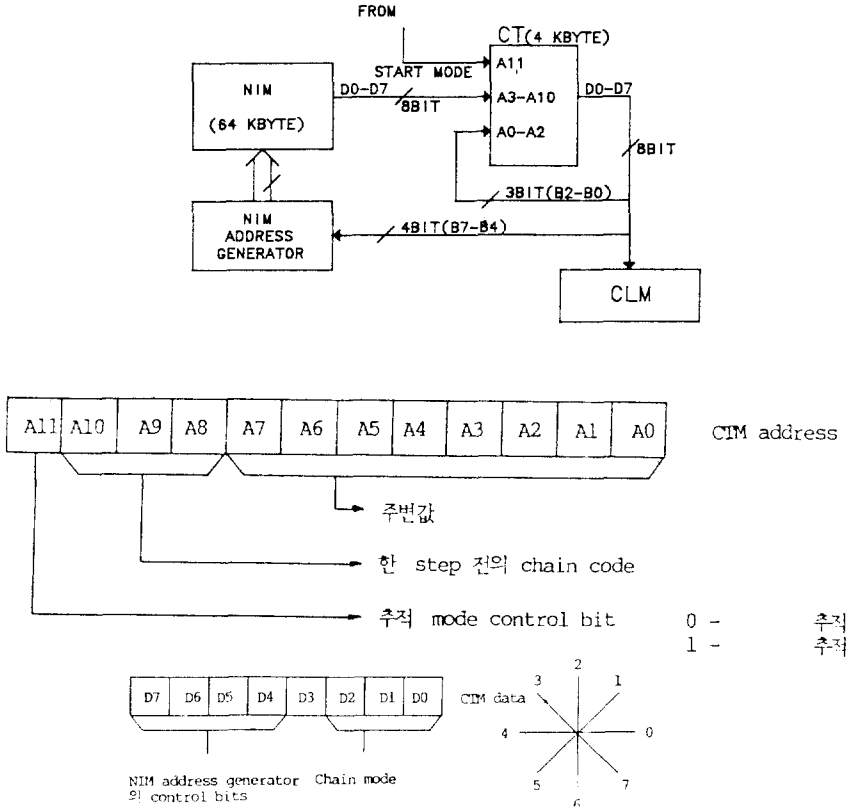

그림 11 경계 추적부의 구성및 체인 테이블의 출력
Fig. 11 Structure of boundary tracking part & output of CT

고 탐색범위의 끝(LR값)에 도달하면 NIM address와 LR값이 같게되어 MSE 신호가 발생되면서 대기 상태로 상태 천이가 일어난다. 한편, 탐색범위는 불필요한 부분의 탐색을 줄이기위해 사용자가 임의의 위치와 크기를 가진 사각형으로 지정할 수 있다. 즉 탐색은 UL위치부터 시작하여 LR위치까지 UL, LR에 의해 결정되는 사각형 내에서만 이루어진다. 또 시작점이 검출되고 경계추적이 끝난뒤 다시 CPU로 부터 탐색 시작명령이 내려지면 경계 추적이 끝난 그 지점부터 다시 탐색을 계속하게 된다.

메모리 탐색 상태로 부터 START 신호에 의해 경계 추적 상태로 상태 천이가 이루어지면, NIM으로 부터 출력되는 주변값과 바로 전 단계의 체인코드, 그리고 시작점 발견시에 결정되는 시작점 형태(type) 선택 비트로 구성되는 체인 테이블에서 다음의 체인코드를 얻고, 이 체인코드를 토대로 NIM address generator를 제어하게 된다. 체인 테이블은 그림11과 같이 구성되어 있다. Mode I의 2K byte는 각 주변값에 따른 A추적에 대한 결과 테이블을, Mode II의 2K byte는 각 주변값에 따른 B추적에 대한 결과 테이블을 형성하고 있다. 그림11에서 추적시의 클럭은 마이크로 프로그램에 의해 발생되는 것으로 추적 순서의 뒷부분에서 발생되기 때문에 B type의 시작점에서도 최초 추적은 A 추적을 행하도록 되어있다. 한편, 얻어지는 체인코드들은 순서적으로 체인코드 리스트내에 저장되며, 이때 체인의 길이도 같이 증가된다. 또, 경계 추적 상태로 상태 천이가 이루어질때 시작점의 address를 기억시켜, 추적이 되는 각 경계점의 address와 비교하여 그 값이 같아지게 되거나, 주변값이 '0'이 되게끔 address가 지정되면 END 신호(그림10)가 발생하며 대기 상태로 상태 천이를 일으켜 CPU로 하여금 추적된 부분의 주변값을 '0'으로 만들어 중복되게 추적하지 않도록 한다.

마이크로 프로그램에 의해 동작되는 제어부는 그림12와 같은 전형적인 구조를 가지고 있다. 프

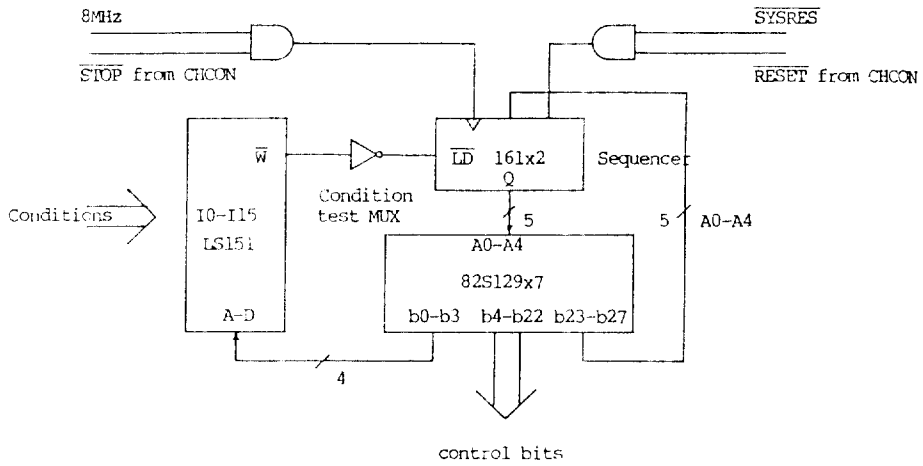

그림 12 마이크로 프로그램 제어부
Fig. 12 Micro programed controller

b4	b5	b6	b7	b8	b9	b10	b11	b12	b13
NIM $\overline{OE}$	NIM R/$\overline{W}$	CIM $\overline{OE}$	X	X	CIM R/$\overline{W}$	NIM addr bit	NIM addr 0-$\overline{LD}$	NIM addr 5-$\overline{LD}$	$\overline{OE_0}$

b14	b15	b16	b17	b18	b19	b20	b21	b22
$\overline{OE_1}$	CIM addr $\overline{CLR}$	CIM addr ENT	Sb_0	Sb_1	MUX	CIM control		CLK

그림 13 마이크로 프로그램 필드의 구성
Fig. 13 micro program field Structure

로그램 필드(program field)의 길이는 28bit로서 그림13과 같이 구분되어 있고 프로그램은 총 23 step으로써 각 상태별로 대기상태 5step, 탐색상태 7(5)step, 추적상태 6step, 그리고 주변값 코딩상태 5(3)step이다. (괄호안은 루프 구성시 step수) 대기상태는 서로 다른 두 부분으로 구성되어 각각 3step, 2step으로 나누어 진다. 제작된 체인코더는 사용된 IC의 동작속도 제한 때문에 8MHz의 클럭으로 동작하므로 기존의 pixel clock과 맞지 않아서 별도의 video timing logic을 구성하여 주변값 코딩시 외부 기억장치로 부터 입력 영상을 읽어들여 사용하였다. 그러나 보다 빠른 속도를 보장하는 IC나 ASIC등을 이용한다면 이 부분에 대한 고려는 할 필요가 없다.

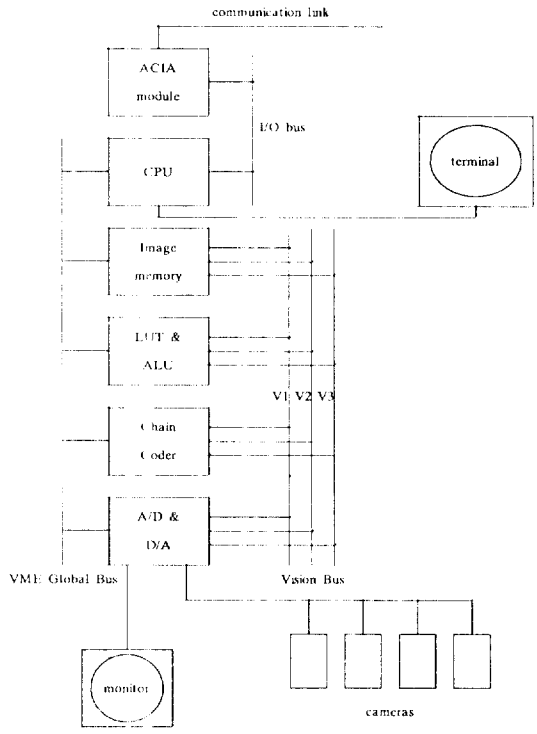

그림 14 전체 시스템의 구조
Fig. 14 Overall system structure

4. 실험 및 검토

4.1 비전 시스템

제작한 체인코더를 이용한 비전 시스템의 구성은 그림14에 있다. 시스템에 연결되어 있는 video camera로 합성영상신호가 입력되면, A/D, D/A 모듈내에서 순수한 영상 신호와 등기 신호를 구분하여 시스템 내에서 사용하는 동기신호를 만들어

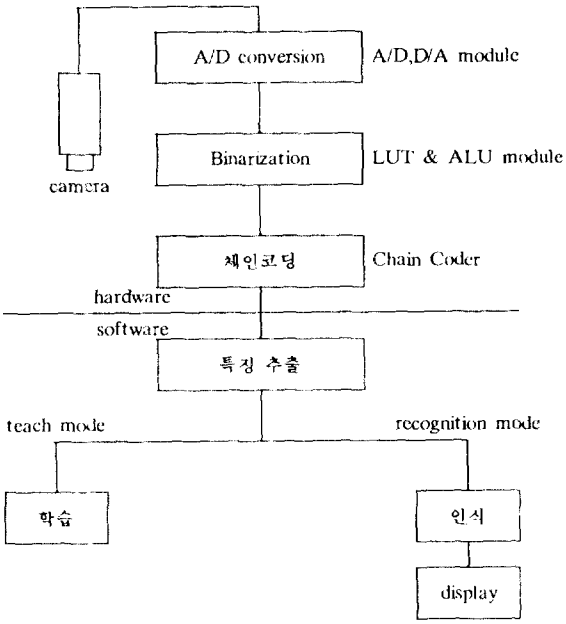

그림 15 전체 시스템의 동작과정
Fig. 15 Overall operation system

내며, 그 신호로써 영상메모리 및 LUT & ALU 모듈 그리고 D/A에서 모니터를 구동하는 합성영상신호를 만들어 내는데 사용한다.

한편 영상신호는 A/D 변환기에서 디지탈 값으로 변환되어 비전 버스를 통해 64K byte의 영상 메모리내에 저장된다. 이와 같은 방법으로 256×256 화소(pixel)의 영상 영역 및 256 단계의 명암 단계를 가지는 자연 영상을 저장할 수 있고, 이진 영상을 얻기 위하여는 LUT(Look-Up Table)에 역치(threshold)를 기준으로 하여 2진값을 가지도록 테이블을 만들고, 이 테이블을 이용하여 이진 영상을 얻는다. 이 때의 역치를 설정하는 방법으로는 모니터에 표시된 이진 영상을 사용자가 보면

4.2 학습 및 인식

제안 모델을 이용한 비전 시스템의 응용으로 물체의 학습및 인식기능을 구현하였다. 산업 환경에서 인식을 구하는 물체의 종류는 사용자마다 그 종류가 달라지기 때문에 인식 대상의 변경에 따른 적응성을 높이기 위해 인식 대상을 사전에 학습시키고 이 학습에 의해 형성되는 자료(database)를 토대로 대상체를 인식하는 방법이 필요하다. 물체를 학습하기 위해서는 학습 대상체의 이름을 먼저 가르쳐주고 반복적으로 특징량을 추출하여 자료를 형성한다. 학습시 필요한 특징량으로서는 물체의 면적, 둘레의 길이, 모양 복잡도, hole의 갯수, hole의 총 면적, 그리고 hole의 총 둘레의 길이를 선정하였다.[6~8] 여기서 물체의 면적과 둘레의 길이 그리고 모양 복잡도는 다음과 같이 정의하였다.

ㅇ물체의 면적 : 물체를 형성하고 있는 영상점의 수

ㅇ둘레의 길이 : 물체의 경계를 이루는 경계점의 수

ㅇ모양 복잡도 $= \frac{\text{둘레의 길이}}{(\text{물체의 면적})^2}$

학습 시에 영상 내부에는 한개의 학습 대상체만 있어야 하며, 학습 대상체를 여러가지 자세로 하여 학습 시킨 후 각 특징량의 평균치를 취함으로써 대상체의 자세변화에 따르는 특징량의 변화를 보상한다. 한편, 학습되지 않은 대상체를 분류하기 위해 학습시에 학습 표현에 의해 구해진 특징량들의 최소, 최대값을 구하고, 이로 부터 20%의 이유 변화를 허용하여 이 안에 들지않는 것은 학습되지 않은 물체로 간주한다.

학습 과정에 대한 대상체의 database가 만들어지면 곧바로 인식을 할 수 있다. 인식 대상체를 화면 안에 두고 인식 시작 명령을 내리면 인식 대상체의 특징량이 구해지며 이 특징량을 비교하여 인식하는 과정에서 벡터 비교(vector matching)법

이를 이용하여 인식 대상체의 특징량으로부터 J_i들을 구하고 이들중 가장 작은 값을 가지는 database i를 구한 후 각 특징량의 최소값과 최대값 사이에 인식대상체의 특징량이 들어 있는가를 확인하여 범위 안에 있으면 인식 대상체는 i^*번째 database에 등록된 물체로 인식한다. 또한 범위안에 들어있지 않는 경우에는 그 물체를 기억되어있지 아니한 물체로 간주한다.

특징 공간을 구성할 때 각 특징간의 가중치가 다르기 때문에 이에 대한 고려가 필요하다. 그러나 이와같은 가중치를 얻어 내는 것은 특징의 종류와 인식 대상체의 부류등에 따라 그때 그때에 적절히 조절하는 것이 필요하며 가중치를 설정하기 위한 확실한 측정자(measure)도 없기 때문에 그 최적치를 구한다는 것은 매우 어려운 일이다. 따라서 본 논문에서는 모든 가중치는 각 학습 대상체의 특징량중 같은 종류의 특징량들에 대한 최대치를 구하고 그 최대값을 1로 정규화(normalize)하여 특징공간상에서 각 물체의 위치를 결정한다. 또 물체와 구멍에 대한 각각의 특징공간을 구성하여 구멍이 있는 경우에 한해서 구멍에 대한 특징공간을 이용하도록 하였다.

4.3 결과 및 검토

체인코딩 시간을 탐색과정을 포함하는 시간으로써 software로만 처리할 경우 많은 시간을 요한다. Software에 의한 체인코딩시 시작점을 탐색하기 위하여 매 영상점 마다 영상점 값을 읽고 이를 비교 판단하며 address의 비교등을 수행하여야 하므로 1MIPS의 속도를 가진 프로세서를 사용하여 9개의 명령어가 필요하다고 보면 256×256의 영상을 탐색하는 시간이 $1\mu sec \times 9 \times 64Kbyte = 0.60sec$ 정도 소요된다. 또한 600개의 체인코드를 추출해 내기 위해서는 영상점 값을 읽는 것 뿐만 아니라 이를 가지고 다음 경계점의 위치를 알아내는 처리시간을 감안하여 탐색시간의 3배가 소요된다고 하면 추적시간은 $1\mu sec \times 3 \times 9 \times 600 = 16.2msec$가 되어 총 처리시간은 약 0.62초가 된다. 이에 비하여 제작된 체인코더는 마이크로 프로그램의 한 step이 한 클럭 주기안에 수행되고 클럭 주파수는 8 MHz이므로 탐색상태의 소요시간은 $5step \times 0.125\mu sec \times 64K = 0.9msec$, 추적상태의 소요시간 $6step \times 0.125\mu sec \times 600 = 450\mu sec$ 그리고 경계점을 다시 지우는데 필요한 software처리 소요시간이 영상 탐색시간 정도 걸린다고 보면 $1\mu sec \times 9 \times 600 = 5.4msec$ 도합 $46.81msec$이 소요된다. 따라서 software의 한 코딩보다 13배의 속도향상이 이루어 짐을 알 수 있다. 실체 측정시간은 그림16에 나타내어져 있다. 제작된 체인코더는 범용의 TTL-IC를 사용한 관계로 속도를 높이는 데에는 한계가 있었으나, 보다 고속의 소자들과 custom화된 PAL, ASIC등을 사용하여 제작된다면 보다 빠른 체인코더를 제작할 수 있으리라 생각된다. 주 프로세서로는 MC68000(8MHz)을 사용하였고 SSM-16 컴퓨터의 C 컴파일러를 이용하였다. C 언어를 이용한 관계로 보다 빠르게 프로그램을 최적화한다면 30% 이상의 software처리시간 감소가 예상된다.

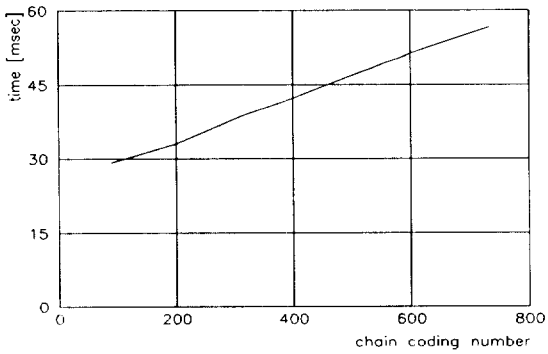

그림 16 체인코드 갯수별 코딩시간
Fig. 16 Number of chain code vs. the coding time

5. 결 론

산업 현장에서 유용하게 사용될 수 있는 비젼 시스템은 저가격이면서 사용자의 다양한 요구에 대처할 수 있는 유연성과 신뢰성을 지닌 시스템이어야 한다. 본 논문에서는 이와같은 욕구를 충족시킬 수 있는 비젼 시스템의 설계를 주 과제로 하여, 이러한 일을 수행하는데에 있어서 기존의 시스템이 안고있는 처리시간의 단축문제를 해결하기 위해 이진 영상처리를 할때, hardware화에 보다 적합한 체인코딩 알고리즘을 제안하고 체인코더를 설계, 제작하였으며, 실험을 통해 체인코딩 시간이 software에 의한것 보다 매우 단축되었음을 알 수 있었다. 앞으로의 연구 과제로는 제안된 체인코더에서의 제약조건을 제거하고 물체의 국부적인 특징을 이용해 인식을 할 수 있게하는 체인코더의 보완 및 이를 이용해서 많은 특징량을 찾아내어 더욱 많은 물체를 인식할 수 있도록 하는 알고리즘의 개발이 필요하다고 하겠다.

참 고 문 헌

[1] Eugene Charniak and Drew McDermott, Arificial Intelligence, Addision Wesley, 1985.

[2] 월간 자동화 기술 1985년 10월호.

[3] 월간 자동화 기술 1985년 3월호.

[4] Gerald J. Agin, "Computer Vision Systems for Industrial Inspection and assembly," IEEE Computer, May, 1980.

[5] H. Otsu, "A Threshold Selection Method from Gray-level Histogram," IEEE Transaction on System, man and Cybernetics, vol. SMC-9, pp. 62-66, 1979.

[6] Alan Pugh, Robot Vision, IFS Ltd. U.K.

[7] Carl Helmers, Robotics Age in the beginning, HAYDEN BOOK COMPANY Inc. p. 81-149.

[8] Herbert Freeman, "Computer Processing of Line-Drawing Images," Computing Surveys, vol. 6, no. 1, March, 1974.

[9] 이병일, "물체인식을 위한 산업용 비젼 시스템의 설계," 한국과학기술원 1987.

[10] 이정환, "Run-length code를 이용한 새로운 Chain coding 알고리즘에 관한 연구," 한국과학기술원 1986.