영상해석용 직선 Hough Transform 연산기의 아키텍쳐 설계

박 영 준[†] · 송 낙 운[††]

요 약

본 논문에서는 영상인식을 위한 직선 HT(Hough transform) 알고리즘 연산의 하드웨어 구조를 제안하였다. 이 연산기는 기울기연산을 위한 필터링부위와 HT 연산부위로 이루어졌으며, 이때 각도에 관한 정보는 메모리 테이블에 저장하였다. 제안된 구조는 C 언어를 이용한 알고리즘 시뮬레이션을 수행하며 동작의 확인과 계산의 정밀도를 결정한 다음, 전체블록에 대하여 VHDL 언어에 의한 아키텍쳐 시뮬레이션을 수행하였다. 각 실험결과에 의하면, 연산된 데이타 값이 유사하게 얻어졌으며, 영상의 선명도와 사용 비트수가 커질수록 연산값의 차이가 적어짐을 확인하였다.

Architecture design of the straight-line Hough Transform processor for image analysis

Youngjune Park [†] · Nagun Song [††]

ABSTRACT

In this paper, a hardware architecture to calculate straight-line Hough transform algorithm for image recognition is suggested. This processor consists of the filtering module for gradient calculation and the HT calculation module, and the angle information are stored in memory table. For the suggested architecture, firstly, algorithm simulation is executed using C language to confirm the operation and to decide the precision of calculation, and secondly, architecture simulation is executed using VHDL language for the total blocks. According to C & VHDL simulation results, it is confirmed that the calculated data value is similarly obtained and the calculation difference is decreased as image clarity and bits increase.

1. 서 론

근래에 이르러 영상해석과 인식분야는 영상처리분야와 더불어 영상관련의 중요한 연구분야로 많은 관련 연구가 진행중이다. 영상해석은 그 자체로서만 아니라 영상인식의 전처리 과정으로 사용되며, 일반적으로 다양한 영역(예로, feature extraction, segmentation, classification 등)으로 같이 분류된다[1]. 이러한 영상해석의 분야에서 HT는 feature extraction 및 일부 segmentation 분야에 속하며 그 결과를 여러 형태의 인식에 대한 정보로 사용할수 있다.

1962년 Paul Hough에 의해 제안된 HT는 영상에서의 윤곽선을 검출해 내는 방법으로, 영상인식과 로봇

† 정 회 원:세원 텔레컴(주)
†† 정 회 원:홍익대학교 전자공학과
논문접수:1997년 5월 6일, 심사완료:1997년 9월 5일

시각에 광범위하게 사용된다. HT는 초기에 Duda and Hart와 O'Gorman and Clowes에 의해 연구되었으나 [2, 3], 계산량이 너무 많아 알고리즘의 연구에 그치다가, C. Kimme가 기울기방향(gradient direction)을 HT에 도입하면서 계산량의 획기적인 감소를 가져오게 되었다[4]. 그리고 스캐닝된 사진같은 정지영상에 대한 HT를 범용 컴퓨터를 이용해 처리하게 되었고, 원과 타원에 대한 HT 알고리즘도 제시되었다. 이후의 연구들로는 아날로그 계측기의 보정에 HT를 적용한 연구[5]나 vehicle guidance 및 로보트에의 적용에 대한 연구[6-8], 문자인식 그리고 임의의 형태에 대한 영상 인식[9]에 이르는 여러 문제에 적용하는 방법이 연구되었다.

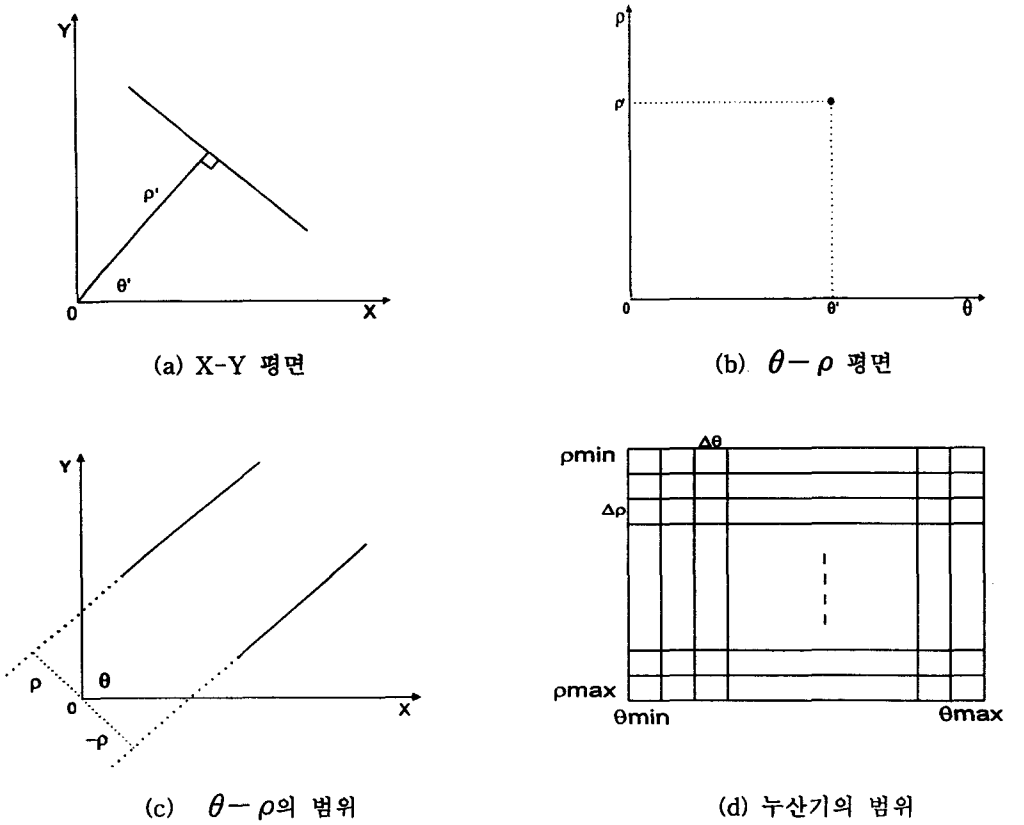
1980년 후반 이후 이에 관한 하드웨어에 대한 연구가 활발히 이루어졌는데, 68000을 사용하며 주변 로직회로를 설계하여 전용 연산보드를 만든 연구[10]가 있으나, 이의 경우 프로그램에 많은 부분을 의존하고 하드웨어 크기가 증가되는 문제점이 있다. 하드웨어 구현시에 중요한 개선점은 연산과 저장 부위의 성능 향상이다. 이를 위한 효율적인 알고리즘에 관한 연구[11, 12]가 있으며 관련 아키텍처에 관한 연구가 활발하게 이루어지고 있다. 한편 이중 연산기 성능을 개선하기 위하여 파이프라인 및 systolic, SIMD[13-15] 병렬 연산구조를 채택한 연구 등 디지탈 연산방식과 아날로그[16, 17] 연산 방식에 의하여 구조에 관한 연구가 이루어졌으며 아울러 단일 VLSI 칩 구현을 위한 연구가 이루어졌다. 한편 vehicle guidance를 위한 VLSI 칩의 구현 연구[18]에서 HT을 부분적으로 사용한 예가 있었으나, HT를 부분적으로만 사용했으며 90도보다 크고 작은 두가지 경우만을 판단하기 위한 것으로 윤곽선의 각도를 계산하는데 HT의 개념을 일부 도입했을 뿐이었다.

이에 관한 연구들은 대부분 알고리즘의 적용에 관한 것으로, 실용화(즉, 로봇 또는 차량의 자동운전장치, 분류 등)에 사용할수 있는 보다 소형이며 튼튼한 하드웨어의 연구가 필요했다. 이를 위하여 본 논문에서는 영상을 신속히 처리할수 있고 소형의 전용 하드웨어 구조로, 향후 FPGA 및 커스텀칩 등으로 구현하기 위한 구조를 제안한다. 이는 크게 기울기필터링부위와 HT 부위를 단일칩화할수 있는 구조로 이루어져 있으며 다양한 각도의 정보를 수용하며 이를 테이블화하여 연산시간을 줄이도록 설계하였다. 이에 직선 HT 알고리즘을 택하여 주어진 영상에서의 윤곽선의 특징을 추출하기 위한 하드웨어 구조를 설계하고, C 및 VHDL 언어로 시뮬레이션하여 이의 타당성을 검증하였다. 이를 위한 논문의 구성은 먼저, 2장에서 HT 알고리즘과 계산과정에 대해 상세히 다루었으며 3장에서는 하드웨어의 구조를 제시하고 4장에서는 제안 구조에 대하여 관련 시뮬레이션을 수행하며 결과를 검토하였으며 5장에 결론을 제시하였다.

2. 직선 Hough 변환 알고리즘의 연산

Hough 변환은, 직각좌표계의 정보를 다른 좌표계(예로, θ-좌표계)로 매핑하는 과정으로 볼수있다. 이제 직각좌표계의 경우, 두 점을 지나는 한 직선 $y=kx+c$가 있다고 하면, 이 위의 모든 점들을 K-C 평면으로 옮기면 같은 점 (k, c)를 지나게 된다. 이로부터 여러 직선이 교차하는 한 점 (k, c)가 X-Y 평면에서의 한 직선을 나타낸다[1, 2]. 그러나 그림 1(a)에서 X-Y 평면에서의 직선이 수직 또는 수직에 가까운 경우 k 값이 무한히 커지는 문제가 있으므로, 이의 정보를 무난히 변환하기 위해서는 일반적으로 그림 1(b)과 같은 θ-ρ 평면을 사용한다. θ-ρ 평면을 사용하면 크기가 NxN인 영상에서 원점부터 직선까지의 거리 와 원점을 지나는 수선의 각 θ는 모두 ($|\rho| \le N\sqrt{2}$, $0° < 360°$)의 범위 안에 있기 때문에 유한한 크기를 갖는다. 이때 X-Y 평면에서 동일한 와 를 갖는 직선이, 일부 단절선을 포함하여 여럿 존재할수 있으며, 실제 디지탈 영상에서의 직선은 점들의 집합으로 이루어져 있기 때문에, 이의 기록을 위한 누산기(accumulator)가 필요하다. 한편 영상의 왼쪽 아래를 원점으로 정하는 경우 ρ와 θ의 범위는 ($0 \le \rho \le N\sqrt{2}$, $-90° < \theta \le 90°$) 또는 그림 1(c)처럼 ($-N < \rho \le N\sqrt{2}$, $0° \le \theta < 180°$)의 범위가 되는데 본 논문에서는 모두 후자의 경우를 기준으로 하였으며 그림 1(d)는 이를 적용한 누산기이다.

이의 연산을 위하여 디지탈화한 영상을 기울기필터링과 HT를 수행한다. 우선 영상의 어느 점이 윤곽선에 해당하는가를 찾고자 이를 수행하는데, 이를 위하여 먼저 주어진 점들을 기준으로 주변의 점들에 대한 기울기를 계산하며 와 를 얻게 된다. 먼저, 디지탈



(그림 1) HT 연산 평면
(Fig. 1) HT calculation plane

화된 영상을 기울기 필터링을 통하여 X 방향 기울기 grad_x와 Y 방향 기울기 grad_y를 구하는데 이때 사용한 3x3 필터윈도우는 그림 2와 같다(3장의 그림 5 (b)참조).

A	B	C
D	E	F
G	H	I

(그림 2) 필터 윈도우의 구성도
(Fig. 2) Structure of filter window

한편 이에 의한 기울기의 식은 다음과 같다.

$$\text{grad\_x} = \frac{1}{2+k}(C + k \cdot F + I - A - k \cdot D - G) \quad (1.1)$$

$$\text{grad\_y} = \frac{1}{2+k}(A + k \cdot B + C - G - k \cdot H - I) \quad (1.2)$$

(Prewitt : k = 1, Sobel : k = 2)

이들로부터 기울기크기는 다음식을 사용하여 구한다.

$$m = |\text{grad\_x}| + |\text{grad\_y}| \quad (2)$$

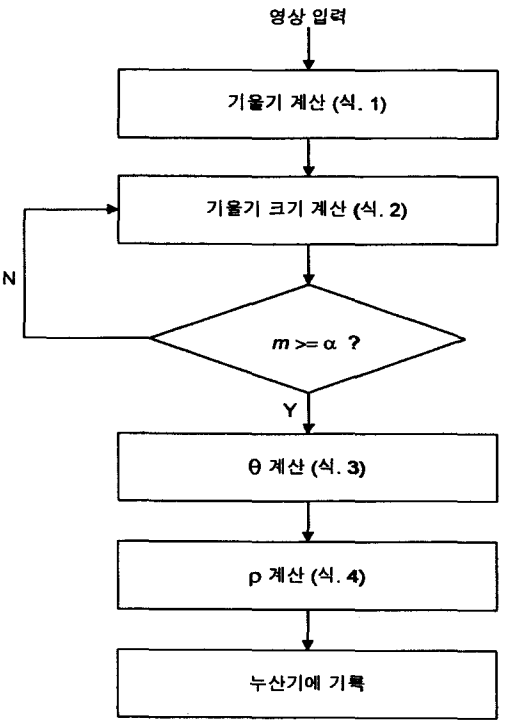
만약 한 점에서의 기울기크기가 기준값보다 큰 경우에, 그 점은 윤곽선을 나타낸다고 판단한다. 기울기방향 는 식 (3)으로 구해지며 윤곽선이 변화하는 방향, 즉 윤곽선에 대한 수선의 방향이 된다.

$$\theta = \text{atan}(\frac{\text{grad\_y}}{\text{grad\_x}}) \quad (3)$$

θ가 어느 사분면에 오는지는 grad_x와 grad_y의 부호에 따라 결정되는데 기울기방향은 그 방향이 윤곽선에 대한 수선의 방향으로 하며, 결과적으로 기울기필터링은 윤곽선의 검출과 θ값의 계산을 동시에 수행한다. 실제 계산과정에서는 기울기방향이 θ의 범위 $0° \le \theta < 180°$ 안에 오도록 하고 grad_y가 음수인 경우는 의 범위를 벗어나므로 무시한다. 여기서, 의 범위를 제한하는 것은 한 개의 선에 대해 나타날 수 있는 두 기울기방향 중 하나만 선택하는 역할을 겸하고 있다. 이제 이들 윤곽선 데이타로부터 다음식에 의하여 HT 값 ρ를 계산한다.

$$\rho = x\cos\theta + y\sin\theta \quad (4)$$

이때 한 직선에 속한 점들에 대한 계산결과는 약간씩 다른 값을 갖게 되므로, 윤곽선에 속한 각 점에서 계산된 결과를 모두 기록한 다음 기록된 결과를 사용하는 과정에서 상대적으로 큰 값을 갖는 점 (θ, ρ)를 찾아 이를 이용하게 된다. 이제 이를 종합한 HT 연산의 흐름도는 다음 그림 3과 같다.



(그림 3) HT 연산의 흐름도
(Fig. 3) Flowchart of HT calculation

누산기에서 $\Delta\theta$와 $\Delta\rho$는 HT의 정밀도를 결정하는데, 여기서 $\Delta\theta$는 임의로 정할수 있는 샘플링 값이고 $\Delta\rho$는 $\Delta\theta$에 의해 결정되는 양자화된 값이며, 이들간에 oversampling이 안 생기도록 고려하며 이는 다음 식으로 계산된다[19].

$$\Delta\rho \ge l_{max}\sin(\frac{1}{2}\Delta\theta) \quad (5)$$

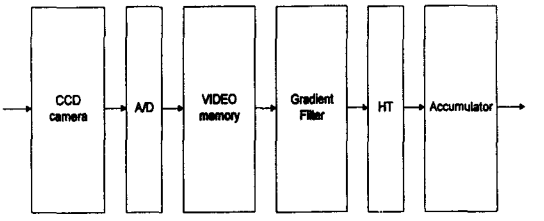
식 (5)에서 l_{max}는 서로 이웃한 θ_n과 θ_{n+1} 사이의 가장 먼 거리에 해당하고, 이 식은 선의 각도 θ가 θ_n과 θ_{n+1} 사이에 있는 경우 같은 값을 갖도록 하는 조건을 나타낸다.

3. 하드웨어 설계

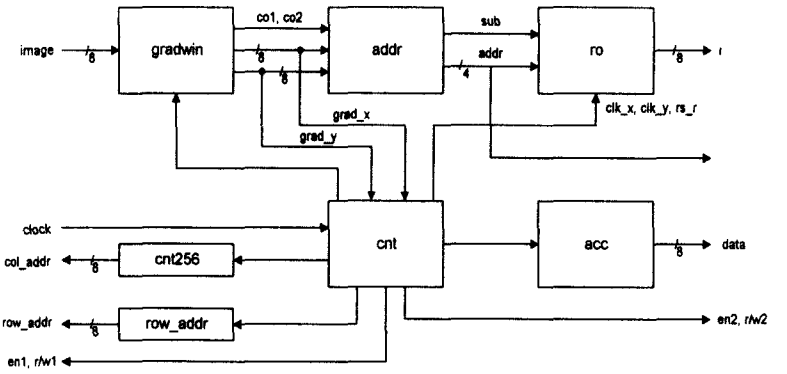
그림 4는 전용 하드웨어를 사용한 HT 시스템의 블록도로, 흑백 CCD 카메라로부터 입력된 영상을 아날로그/디지탈 변환기에서 256 그레이단계의 디지탈 영상으로 바꾼 다음 dual port SRAM에 저장하는 일반적인 영상획득 구조와 본 논문에서 제안한 HT 하드웨어를 연결한 구성도이다.

이중 기울기필터와 HT 블록부문의 본 논문에서 제안된 구조를 그림 5(a)에 보였으며, 이중 gradwin, ro, acc 블록은 그림 5(b, c, d)에 자세히 나타내었다.

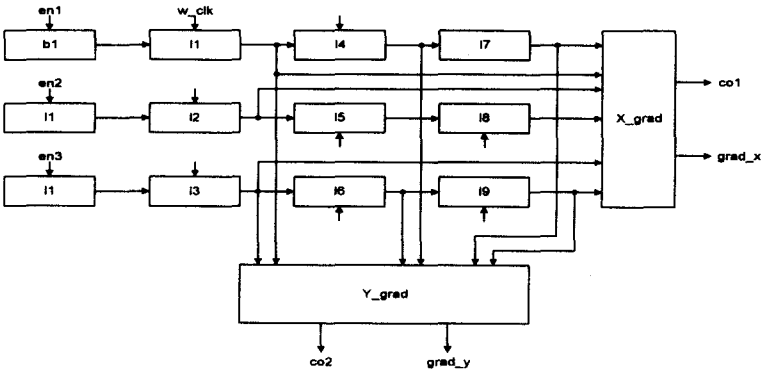
우선 gradwin(gradient window) 블록(그림 5(b))에서는 래치윈도우, X_grad, Y_grad 블록으로 이루어



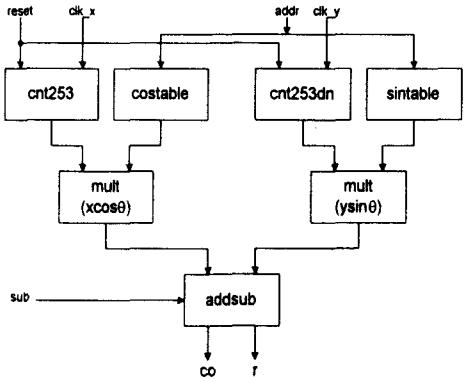
(그림 4) HT 시스템의 전체 구조
(Fig. 4) The total architecture of HT system



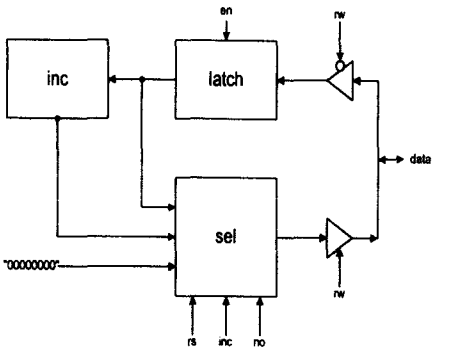
(a) 전체 블록도



(b) gradwin 블록



(c) ro 블록



(d) acc 블록

(그림 5) 제안된 구조
(Fig. 5) Suggested architecture blocks

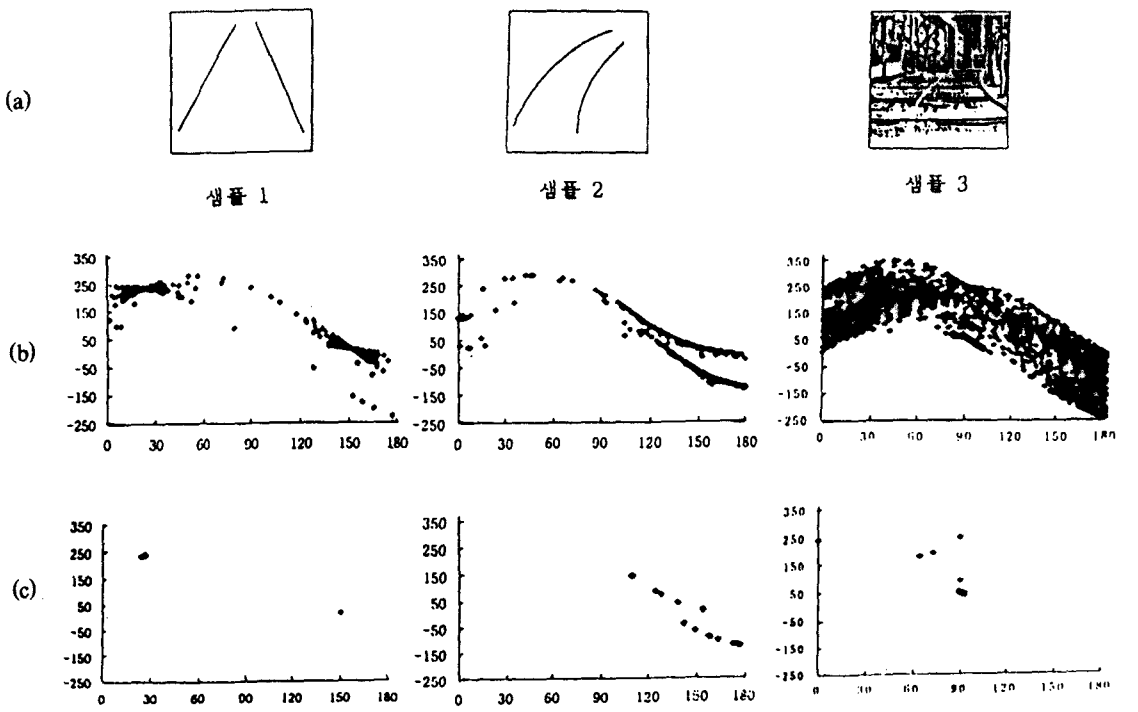
지며, 전체 영상에 대해 3x3 크기의 윈도우를 이동시키며 기울기를 계산한다. 한 윈도우에 대한 계산이 끝나면 데이터가 옆으로 이동하도록 되어, 메모리에서 읽혀진 데이터는 버퍼로 들어가며, 윈도우로 보내는 데이터를 3개씩 병렬로 만드는 역할을 한다. 래치로 구성된 윈도우 내에서는 기울기가 계산되는 동안 데이터를 유지하다가 계산이 끝나면 왼쪽 래치의 데이터를 받아들인다. X_grad, Y_grad 부위는 기울기를 계산하는 곳으로, 입력만 다를 뿐 동일한 연산을 한다. 계산된 결과는 θ값을 만들기 위한 회로와 기준값과 비교하기 위한 비교회로에서 사용된다. 다음으로, 그림의 addr 블록은 gradwin에서 계산된 기울기 값에서 θ에 해당하는 값을 만들어내는 부분이다. θ는 실제 arctangent 값이 아닌 sine, cosine 테이블의 위치를 찾기 위한 메모리 주소를 만들며, 이 값은 나중에 누산기의 주소를 정하는 데에도 사용된다. 출력되는 신호 addr은 각도 θ에 대응되는 메모리의 주소를, sub는 addr 값이 90에서 180도 사이의 각도임을 알려주기 위한 신호이다.

다음으로, 그림 5(c)의 ro 블록은 ρ 값을 계산하는 곳으로 여기서는 addr 블록에서 만들어진 주소와 계산중인 점의 좌표를 이용해서 식 (3)의 ρ 값을 계산한다. 여기에는 각각 열과 행의 좌표를 만들어주는 카운터, 코사인 및 사인값의 저장테이블, 이들로부터 $X\cos\theta$, $Y\sin\theta$를 만드는 곱셈기, 최종적으로 를 연산하는 덧셈기블록 등이 있다. Costable과 sintable 부위에는 $\Delta\theta=6$인 코사인과 사인값이 들어있으며, 그 내용으로는 실제 삼각함수 값에 128을 곱한 숫자를 입력하였다. 0에서 180도 사이의 사인과 코사인은 0에서 90도 사이의 사인곡선을 수평 또는 수직으로 반전시켜 나머지 곡선을 만들수 있으므로, 메모리 테이블도 이를 응용한다. 따라서, 실제 메모리 테이블에 들어있는 숫자도 0도에서 90도 사이의 사인 값이면 되는데 여기서는 두 개의 테이블을 사용하므로 sintable에는 0-90도까지의 사인 값이, costable에는 코사인 값이 들어있다. 만약 계산된 θ가 90에서 180도에 해당하는 경우는 addr 블록에서 테이블을 반대의 순서로 읽도록 신호를 만들어준다. 이때 cosine 값은 음수가 되므로 addsub 블록에서는 addr 블록에서 만들어진 sub 신호에 따라 뺄셈을 수행한다. 한편, 곱셈기는 combinational array 구조[20]와 관련 adder는 carry-save adder의 구조를 택하였으며 최종단에 carry-lookahead adder를 사용하였다. 다음의 addsub 블록은 ρ의 계산에서 덧셈을 수행하며, 제어신호에 따라 뺄셈도 수행한다. 이는 코사인 테이블에 양수만 넣어두고도 $\cos\theta$가 음수인 경우를 계산할 수 있도록 하기 위한 것으로, 만약 $\cos\theta$가 음수인 각도일 때에는 addr 블록에서 sub 신호를 '1'로 만들고 이에 따라 뺄셈을 수행한다.

다음으로, 그림 5(d)의 acc 블록은 HT의 결과를 최종적으로 누산기에 기록하는 부분이다. 누산기의 주소는 addr 블록과 ro 블록에서 만들어지고 acc 블록에서는 해당 주소의 데이터를 누산기로부터 읽어들여 1을 증가시키거나 초기화한다. 그림에서 inc는 adder이며 sel은 mux가 된다. 여기서 rs는 초기화를 위해 입력중 "00000000"을 선택하며, inc는 누산기의 값을 1 증가시킨 값을, no는 읽어들인 누산기 값을 그대로 되돌리기 위한 신호이다. 다음으로, 그림 5(a)의 cnt256, row_addr 두 개의 블록은 영상 메모리의 주소를 만들어 주는 카운터이다. 이중 cnt256은 X 좌표에 해당하는 주소를 만들어 주고 row_addr은 Y 좌표에 해당하는 주소를 만들어 준다. Row_addr은 발생한 주소를 지연시키기 위한 래치와 이중 어떤 값을 출력할 것인지를 결정하는 mux를 포함하고 있다. 다음으로, 그림의 제어부블록에서는 각 부분에 필요한 클럭을 발생시키고 누산기와 메모리를 위한 제어신호를 발생시킨다. 또, 기울기크기를 기준 값과 비교하고 grad_y가 음수인지를 확인하여 현재 계산중인 결과를 누산기에 기록할 것인지 버릴 것인지 결정한다.

4. 시뮬레이션 및 결과 검토

본 연구의 시뮬레이션에 사용할 영상데이터는 임의의 영상을 만들거나 사진으로부터 입력받을 수 있도록 PCX 그래픽 파일형식을 사용하였다. 그러나 PCX 파일은 영상을 압축해 놓았기 때문에 이를 풀고 헤더와 팔레트를 분리하는 작업이 필요하여 시뮬레이션 과정에서 바로 불러오기에 적합하지 않으므로, 시뮬레이션에는 영상부분만을 분리시킨 다음 압축을 풀고 저장한 파일을 사용한다. PCX 파일로부터 영상데이터만을 분리해 내는 과정은 pcx2dat, 처리된 데이터를 다시 PCX 형식으로 바꾸는 과정은 dat2pcx라는 프로그램을 만들어 사용하였다[21, 22]. 입력 영상



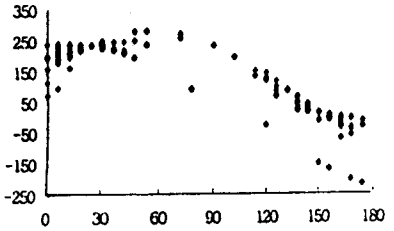
(a) 원영상 (b) 누산기에 기록된 결과 (c) 누산기의 값 $A(\theta, \rho) \geq 3$인 위치

(그림 6) 샘플 1, 2, 3에 대한 C 시뮬레이션의 결과
(Fig. 6) C simulation results of sample 1, 2, 3

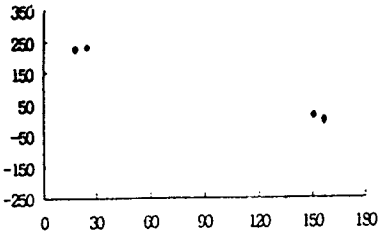
데이터는 256 단계로 양자화하였으며, 기울기는 -255부터 255까지의 정수로 표현하기 위해 스케일링비를 곱하는데 계산에서 영상의 가장자리는 제외되므로 크기가 NxN인 경우 실제로 기울기를 계산하는 점의 수는 (N-2)x(N-2)개가 된다. 3x3 윈도우에 Prewitt와 Sobel template을 적용하여[23], grad_x와 grad_y를 구한다. 기울기 값은 부호를 제외하고 8 비트로 표시되나, 실제 θ의 계산에는 이중 상위 4 비트만을 사용하였다. 이는 기울기방향의 하드웨어에서 $\Delta\theta = 1$인 경우 θ는 180개, $\Delta\theta$를 1보다 크게 잡으면 θ의 개수는 이보다 적어지므로, 두 개의 8 비트 데이터를 모두 사용하지 않아도 각도를 계산할수 있기 때문이다. 이때 제한된 비트수 만으로 연산을 할 경우 이의 오차를 계산한 결과에 의하면, 사용비트수가 많을수록 그리고 영상의 윤곽선이 뚜렷할수록 오차는 적어짐을 발견하였다. 즉, addr 블록의 구조시뮬레이션으로 비트수에 따른 θ 값을 구한 다음, 이를 알고리즘 시뮬레이션에서 구해진 θ와 비교하면, 시뮬레이션 방법의 차이에 의한 에러값은 비트 수가 커질수록 줄었다.

이제 전체 모듈을 통합한 HT 시뮬레이션을 임의개의 영상샘플(영상크기:256x256)에 대해 수행하였는데, 그중 주어진 영상에 대한 C 언어에 의한, 3장의 전체 구조시뮬레이션 결과를 그림 6에 보였다. 그림의 (θ, ρ) 위치의 누산기 값을 $A(\theta, \rho)$라 할때, 각 그림에서 (a)는 원영상이며, 여기에서 HT을 수행하여 $m > 80$, $\Delta\theta = 1$, $\Delta\rho = 1$일때 (b)는 $A(\theta, \rho) \geq 1$인 위치, (c)는 $A(\theta, \rho) \geq 3$인 위치를 나타내었다.

그림에서 보는 바와 같이, 일정크기 이상의 누산기 위치를 직선의 특성으로 판단할수 있음을 확인하였다. 특히, 샘플 3의 경우 여러 방향의 선이 혼재한 관계로 차이가 발생하였는데 이는 응용용도에 따라 방향등에 비중을 두는 등의 방법으로 개선할 수 있으리



(a) A(θ,ρ)≥1인 위치



(b) A(θ,ρ)≥20인 위치

(그림 7) 샘플 1에 대한 VHDL 시뮬레이션의 결과
(Fig. 7) VHDL simulation results of sample 1

라 본다. 다음으로 3장의 전체구조를 VHDL(V-SYSTEM)로 시뮬레이션한 결과를 다음에 보였다. 여기에서 영상데이타는 메모리에 저장된 형태를 VHDL로 기술하여 전체 블록에 연결하였다. 그림 7은 그림 6의 샘플 1에 대한 시뮬레이션 결과인 누산기의 값을 보였으며, 이때 $\Delta\theta=6$, $\Delta\rho=4$로 하였다.

이의 결과는 C 시뮬레이션 결과와 대체적으로 일치하였다. 아울러, VHDL 시뮬레이션의 오차도 C 언어에 의한 구조시뮬레이션과 같은 범위 안에 나타났는데, 이는 같은 범위의 정수형 데이터를 사용했기 때문이다.

5. 결 론

본 논문에서는 크기가 256x256인 그레이단계 영상에 대한 직선 HT 알고리즘을 구현할 수 있는 하드웨어의 구조를 제안하여 영상처리에 보다 용이한 실험방법을 통하여 다양한 영상데이터의 이용이 가능하도록 하였다. 이를 위해 일단 주어진 영상을 기울기 필터링을 행한후 일정한 크기 이상의 기울기만을 택한후 이들로부터 (θ, ρ) 정보를 추출하여 누산기에 기록하는 구조를 택하였으며 이때 ρ 연산에 필요한 각도 정보는 테이블화하여 구하였다. 제시된 구조는 C 언어, VHDL을 통해 모델링하여 시뮬레이션하였는데, 이에 의하면 주어진 영상으로 HT에 의한 연산으로 영상해석을 위한 단순화하여 집약된 데이타가 도출되었으며 이에 의해 영상인식에 필요한 DB 구축이 가능함을 확인하였다. 아울러 θ의 계산에서 발생한 오차가 ρ의 계산에도 영향을 주므로, θ의 정밀도가 전체적인 성능에 영향을 줌을 알수 있었다.

향후, 제안된 구조를 FPGA 및 커스텀칩 등으로 구현하여 실제 회로에 탑재할수 있도록 하며, 보다 정교한 θ 계산을 수행하도록 하드웨어 구조의 성능을 높이는 작업을 수행하며, 설계면에서는 입력영상의 실시간처리속도의 향상을 위한 구조의 개선을 수행하며, 규칙적인 영상처리에 관련된 수평, 수직동기신호를 비롯한 외부 인터페이스 기능을 보강하고자 한다. 아울러 DB 구축과 더불어 영상인식의 후처리부문을 보강하며 궁극적으로는 차량 guidance 등에 탑재할수 있는 시스템설계에 적용하고자 한다.

참 고 문 헌

[1] A. K. Jain, Fundamentals of digital image processing, Prentice-Hall International, Inc., 1989.

[2] R. O. Duda, P. E. Hart, "Use of Hough transformation to detect lines and curves in pictures," Comm. ACM, v. 15, n. 1, pp. 11-15, 1972.

[3] O'Gorman, M. B. Clowes, "Finding picture edges through collinearity of feature points," IEEE T-comp., v. 25, n. 4, pp. 449-454, 1973.

[4] C. Kimme et al., "Finding circles by an array of accumulators," Comm. ACM, v. 18, n. 2, pp. 120-122, 1975.

[5] C. R. Dyer, "Gauge inspection using Hough transform," IEEE transactions of pattern analysis and machine intelligence, v. 5, pp. 621-623, 1983.

[6] R. M. Inigo, E. S. McVey, B. J. Berger and M. J. Wirtz, "Machine Vision Applied to Vehicle Guid-

ance," IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 6, n. 6, pp. 820-826, Nov. 1984.

[7] G-S Cheng et al., "Autonomous land vehicle guidance by line and road following using clustering, HT, and model matching technique," 1994 Int. Computer Sym. Conf., v. 1, pp. 89-94.

[8] J. Forsberg et al., "Mobile robot navigation using the range-weighted HT," IEEE R&A Mag., v. 2, n. 1, pp. 18-26, March 1995.

[9] D. C. W. Pao, H. F. Li and R. Jaykumar, "Shapes Recognition Using the Straight Line Hough Transform: Theory and Generalization," IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 14, n. 11, pp. 1076-1089, Nov. 1992.

[10] K. Hanahara, T. Maruyama and T. Uchiyama, "A Real-Time Processor for the Hough Transform," IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 10, n. 1, pp. 121-125, Jan. 1988.

[11] A. S. Aguado et al., "Ellipse detection via gradient direction in the Hough transform," 5'th cof. on Image processing and its applications, pp. 375-378, July 1995.

[12] K. Nakashima et al., "A fast Hough transform hardware employing reduced size table look-up," Tr. IEEJ, v. 115-D, n. 5, pp. 591-597, May 1995.

[13] A. Rosenfeld, J. Ornelas, Jr. and Y. Hung, "Hough Transform Algorithms for Mesh-Connected SIMD Parallel Processors," Computer vision, graphics, and image processing, v. 41, pp. 293-305, 1988.

[14] G. Seetharaman, "A simplified design strategy for mapping image processing algorithm on a SIMD torus," J. TCS, v. 140, n. 2, pp. 319-331, April 1995.

[15] S. M. Bhandarkar et al., "A reconfigurable architecture for image processing and computer vision," IJ on pattern recognition and AI, v. 9, n. 2, pp. 201-229, Apr. 1995.

[16] D. Ben-Tzvi et al., "Analog implementation of the Hough transform suitable for single chip implementation," IEEE ISCAS, pp. 53-56, May 1990.

[17] R. M. Inigo et al., "Fast analog architecture for high-order curve recognition," J. Electronic Imaging, v. 4, pp. 114-122, April 1995.

[18] J. Sch nfeld and P. Pirsch, "Compact hardware realization for Hough based extraction of line segments in image sequences for vehicle guidance," ICASSP-93 v. 1 of V, pp. I397-I400, 1993.

[19] T. M. Van Veen and F. C. A. Groen, "Discretization errors in the Hough transformation," Pattern recognition, v. 14., n. 1-6, pp. 137-145, 1981.

[20] J. P. Hayes, 'Computer architecture and organization,' pp. 246-248, MH 1988.

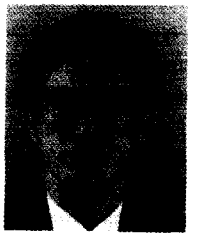
[21] 양진석, PC 그래픽파일 프로그래밍, 정보문화사, 1993.

[22] 원유헌, 하수철, C 프로그래밍, 정익사, 1991.

[23] R. M. Haralick and L. G. Shpiro, Computer and Robot Vision v. 1, Addison-Wesley Company, Inc., 1992.

박 영 준

1995년 2월 홍대전자공학과 졸업(학사)
1997년 2월 홍대대학원 전자공학과 졸업(석사)
1997년 2월~현재 세원 텔레컴(주)
관심분야: 통신 시스템 설계



송 낙 운

1975년 서울대학교 전자공학과 졸업(학사)
1986년 Univ. Texas Austin(Ph.D)
1986년~1989년 금성반도체 근무
1989년 홍익대 전자공학과 부교수
관심분야: VLSI시스템 자동화 설계