논문 13-38A-09-06

한국통신학회논문지 '13-09 Vol.38A No.09
http://dx.doi.org/10.7840/kics.2013.38A.9.773

개선된 보팅 정책을 적용한 허프 변환 하드웨어 구조

이 정 록◆, 배 경 렬*, 문 병 인°

A Hardware Architecture of Hough Transform Using an Improved Voting Scheme

Jeong-Rok Lee◆, Kyeong-ryeol Bae*, Byungin Moon°

요 약

허프 변환은 데이터 손실 및 왜곡이 포함된 영상에서도 직선 정보 추출에 용이한 장점이 있어 컴퓨터 비전 분야의 응용분야에 널리 사용되어 왔다. 그러나 허프 변환의 보팅 과정은 비효율적인 연산구조와 많은 메모리 접근 횟수로 인해 실시간 처리 임베디드 비전 시스템에 적용하는데 한계가 있다. 이에 본 논문에서는 허프 변환의 개선된 보팅 정책을 제시하고, 이를 적용하여 적은 하드웨어 자원 사용량으로 실시간 성능을 만족하는 허프 변환의 하드웨어 구조를 제안한다. 제안된 보팅 정책은 인접한 픽셀 간의 관계를 이용하여 보팅 연산 과정의 오버헤드를 줄였으며, 하드웨어 재사용성을 높임으로서 효율적인 연산구조를 가진다. 이러한 개선된 보팅 정책을 적용한 제안된 하드웨어 구조는 인접한 픽셀들의 보트 값을 병렬적으로 연산하고 저장하여 시간당 처리량을 높인다. 제안 구조의 장점은 순차적 연산 방식 대비 매우 적은 추가 하드웨어 자원만으로 이러한 성능 향상을 위한 병렬화를 달성한다는 것이다.

Key Words : Hough Transform, line detection, Computer Vision, Voting scheme, Parallelization

ABSTRACT

The Hough transform for line detection is widely used in many machine vision applications due to its robustness against data loss and distortion. However, it is not appropriate for real-time embedded vision systems, because it has inefficient computation structure and demands a large number of memory accesses. Thus, this paper proposes an improved voting scheme of the Hough transform, and then applies this scheme to a Hough transform hardware architecture so that it can provide real-time performance with less hardware resource. The proposed voting scheme reduces computation overhead of the voting procedure using correlation between adjacent pixels, and improves computational efficiency by increasing reusability of vote values. The proposed hardware architecture, which adopts this improved scheme, maximizes its throughput by computing and storing vote values for many adjacent pixels in parallel. This parallelization for throughput improvement is accomplished with little hardware overhead compared with sequential computation.

※ 본 연구는 미래창조과학부 및 정보통신산업진흥원의 IT융합 고급인력과정 지원사업의 연구결과로 수행되었음 (NIPA-2013-H0401-13-1005)
◆ 주저자 : 경북대학교 IT대학 전자공학부, trainee@ee.knu.ac.kr, 학생회원
° 교신저자 : 경북대학교 IT대학 전자공학부, bihmoon@knu.ac.kr, 종신회원
* 경북대학교 IT대학 전자공학부, puris1@ee.knu.ac.kr
논문번호 : KICS2013-04-160, 접수일자 : 2013년 4월 5일, 최종논문접수일자 : 2013년 9월 9일

Ⅰ. 서 론

정보산업 기술의 발전에 힘입어 다양한 분야에서 디지털 영상 데이터를 활용한 서비스들이 등장하였고, 고성능 영상 장치가 널리 활용됨에 따라 고해상도 비디오 영상 처리 기술의 요구가 증대 되었다. 방대한 양의 데이터를 갖는 고해상도 영상에 대한 실시간 처리를 위해서는 조명, 잡음 등에 강인하며 대상의 특성, 구조 등을 잘 나타낼 수 있는 정보를 선별하고, 특징적이지 못한 불필요한 정보를 제거하여 처리하여야 할 데이터양을 줄이는 것이 매우 중요하다. 이러한 처리 과정으로서, 외곽선 검출 및 직선 정보 검출은 그림 1과 같이 핵심적인 정보만을 남겨 데이터양을 줄이는 것에 효과적이다.

그림 1. 영상 데이터에서 외곽선 검출 및 직선 정보 검출
Fig. 1. Edge detection and line detection form image data

특히 직선 정보 검출은 차선 인식, 데이터 분류 등에서 데이터양을 줄이는 전처리 과정으로서 영상처리 분야와 컴퓨터 비전 분야에서 많은 연구가 되어왔다[1-3]. 다수의 직선검출 방법 중 가장 널리 사용되고 있는 허프 변환(Hough transform) 기법은 데이터 손실 및 왜곡이 포함된 영상에서도 직선 정보 추출에 용이한 장점을 가진다[4,5]. 하지만 허프 변환을 통해 직선 정보를 검출하기 위해서는 삼각함수, 곱셈, 나눗셈 등과 같은 오버헤드가 큰 연산들의 반복적 수행이 필요하고, 변수 공간을 양자화하는 해상도와 검출 정확성 사이의 트레이드오프를 고려해야 하기 때문에 실시간 성능을 보장하기 어렵다는 단점이 있다.

이러한 허프 변환의 단점을 보완하여 실시간 성능을 만족시키기 위한 많은 알고리즘 개선 연구가 이루어져 왔으며, 그 예로서 효율적인 자료구조와 동적 기법들을 사용하여 변수 공간을 일정한 간격으로 양자화하지 않고 직선이 위치할 근처에서 높은 해상도를 갖게 하는 방법을 사용하거나[6-8], 영상 공간을 블록단위로 분할하여 각 블록 내에서 직선을 검출한 후에 계층 구조를 이용하여 클러스터링을 수행하는 알고리즘을 제안하여 연산속도를 향상 시키는 연구들[9,10]이 이루어졌다.

하지만, 이러한 알고리즘의 고속화 연구에도 불구하고 실시간 처리를 요구하는 일부 임베디드 응용시스템에서는 임베디드 프로세서의 성능 한계로 인해 전용 하드웨어 가속기 기반의 구현방법이 요구되었다. 전용 하드웨어 가속기 기반의 방법은 고해상도 영상의 실시간 처리를 요구하는 시스템에서 기존의 임베디드 프로세서만으로는 연산량을 감당하기 어렵기 때문에 시스템의 성능 및 구현 비용 측면에서 효과적이다.

이에 임베디드 시스템에서 고해상도의 영상을 실시간으로 처리하기 위한 전용 하드웨어 가속기 개발에 관한 연구가 지속적으로 진행되었으며, 허프 변환의 복잡한 수식 및 반복 연산의 비효율성과 요구되는 많은 메모리 용량 및 메모리 접근 빈도를 고려한 하드웨어 구조가 연구되어 왔다. 초기에 곱셈기를 이용하여 구현된 구조는 삼각함수와 곱셈들의 오버헤드가 큰 연산들을 직관적으로 구현하여 하드웨어 자원적인 측면을 고려했을 때 효율적이지 못하였다[11]. 이에 삼각함수와 곱셈기 연산을 비트 시프트 연산으로 치환하는 CORDIC(COordinate Rotation DIgital Computer)을 적용하여 구현한 구조가 제안되었다. 이를 통해 하드웨어 자원 사용량은 감소시켰지만 정확도 높은 해상도를 얻기 위한 반복 연산은 여전히 남아있으며, LUT크기의 증가로 인한 부하 문제가 추가로 발생한다[12]. 비슷한 연구로 비트 시프트와 덧셈기를 이용한 DA(Distributed Arithmetic) 구조가 제안되었으나 CORDIC의 한계를 극복하지 못했다[13]. 이후 누산기 기반의 적은 하드웨어 자원 사용량만으로 병렬화를 수행하여 매 클럭 사이클(clock cycle) 마다 한 픽셀에 대한 허프 변환 값을 출력하도록 하는 구조가 제안되었다[14,15]. 하지만 영상 전체의 모든 픽셀에 대해 픽셀 상호간의 관계를 통한 연산이 수행되어야하므로 시간당 처리량에 한계점을 가진다.

따라서 본 논문에서는 허프 변환기반 실시간 라인 검출 시스템의 구현을 위해, 오버헤드가 큰 연산을 반복 수행해야하는 허프 변환의 연산 비효율성을 개선하고 픽셀 상호간의 특성을 이용하여 연산 과정을 병렬화함으로써 시간당 처리량을 높이는 하드웨어 구조를 제안한다. 본 논문의 2장에서는 허프 변환의 연산 비효율성을 개선하여 하드웨어 구조에 최적화된 알고리즘을 제안한다. 3장에서는 픽셀 상호간의 특성을 이용하여 적은 하드웨어 자원만으로 연산과정을 병렬화하

는 높은 시간당 처리량의 하드웨어 구조를 제안한다. 4장에서는 제안한 개선된 허프변환 보팅 정책 기반 실시간 직선 검출 하드웨어 구조의 성능을 분석하고, 마지막으로 5장에서 본 논문의 결론을 맺는다.

Ⅱ. 개선된 보팅 정책 기반 허프 변환

2.1. 허프 변환 기법

일반적인 허프 변환 기법은 그림 2와 같이 수식 (1)을 사용하여 영상의 (x,y) 좌표 공간에 존재하는 각 픽셀들을 허프 공간이라 일컫는 (r,θ) 좌표 공간으로 사상(mapping)하여 직선 정보를 검출한다. 그림 2와 같이 영상의 (x,y) 좌표 공간에 각 픽셀들은 (r,θ) 좌표 공간에 곡선으로 사상되며, (x,y) 좌표공간에서 같은 직선상에 존재하는 픽셀들의 경우 (r,θ) 매개변수 공간에서 교점을 가지게 된다. 따라서 허프 변환은 그림 2와 같이 (r,θ) 좌표 공간에 누적된 빈도수를 같은 직선상에 존재하는 픽셀의 수로 볼 수 있으며, 임계값 이상의 빈도수를 가지는 교차점의 (r,θ)들을 직선 성분으로 정의할 수 있다. 이때, 수식 (1)을 통해 연산되는 (r,θ) 값을 보트(vote)라 정의하고, 연산된 보트들의 빈도수를 2차원 배열에 누적시키는 과정을 보팅(voting)이라 정의한다.

$$x \cdot cos\theta + y \cdot sin\theta = r \tag{1}$$

보팅 과정은 2차원 배열의 θ를 표본화하고 그에 따른 보트를 구하는 과정을 거친다. 이때, 보팅 과정은 θ를 표본화한 해상도에 따라 $\Delta\theta$ 단위마다 해당 보트를 연산하여 총 $\theta/\Delta\theta = n$ 만큼 반복 수행하게 된다. 즉, 영상에서 직선 성분 검출을 위한 입력 픽셀의 개수를 K라고 할 때 허프 변환의 보팅 과정을 수행하기 위한 시간 복잡도는 $O(Kn)$로 표현된다. 이때, $O(Kn)$의 높은 시간 복잡도를 가지는 수식 (1)의 오버헤드는 시스템 성능 저하에 큰 영향을 미치게 되며, 허프 변환에서는 실시간 성능 만족을 위해서 수식 (1)의 오버헤드를 줄이는 연구가 필요하다.

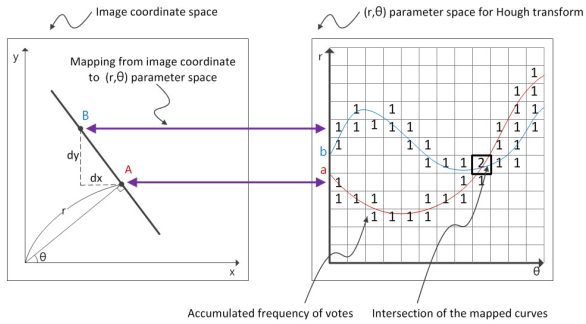

그림 2. 허프변환 : 영상 좌표평면에서 (r,θ) 매개변수 공간으로 사상
Fig. 2. Hough transform : mapping from image coordinates to (r,θ) parameter space

2.2. 인접 픽셀 간 보트의 상관관계

그림 2에서 (x,y) 좌표 공간의 픽셀 A와 픽셀 B는 x축으로의 변위 dx, y축 방향으로의 변위 dy를 가지는 인접한 상관관계에 있다. (x,y) 좌표 공간에 존재하는 픽셀 A가 (r,θ) 좌표 공간으로 사상되는 보트들을 $A(x,y)$라고 할 때, 픽셀 B가 (r,θ) 좌표 공간으로 사상되는 보트인 $B(x+dx,y+dy)$는 수식 (1)과 같이 픽셀 B의 (x,y) 좌표 값을 사용하여 삼각함수, 곱셈, 나눗셈을 포함한 연산을 하는 대신 $A(x,y)$ 값과 dx, dy만을 고려하여 수식 (2)와 같이 표현할 수 있다[13].

$$\begin{aligned} &B(x+dx,y+dy) \\ &= A(x,y) + dx \cdot cos\theta + dy \cdot sin\theta \end{aligned} \tag{2}$$

또한, 허프 변환을 위한 보팅 연산의 목적은 각 보트들의 단순 크기를 비교하여 빈도를 측정하는 것이므로 수식 (2)의 양변을 $cos\theta$로 나누어도 각 보트의 빈도는 동일하게 누적되므로 동일한 보팅 결과값을 얻을 수 있다. 따라서 그림 2에서 픽셀 B의 보트는 인접한 픽셀 A의 보트와 dx, dy만을 고려하여 수식 (3)과 같이 표현할 수 있다.

$$\begin{aligned} &B'(x+dx,y+dy) \\ &= A'(x,y) + dx + dy \cdot tan\theta \end{aligned} \tag{3}$$

이때 dx, dy의 값이 예측 가능하면 인접 픽셀의 보트로부터 $B(x+dx,y+dy)$를 구하는 것은 단순 덧셈 연산만으로 가능하게 된다. 또한 수식 (2)의 dx, dy 항의 계수가 모두 소수부를 가지는데 반해 수식 (3)은 dx 항의 계수가 정수 1을 가지게

되므로 dy 항이 0이 되는 x축 방향으로의 연속적인 픽셀 입력에 대한 보트 연산이 단순해진다.

단, $\frac{\pi}{4} < \theta < \frac{3\pi}{4}$ 범위에 한하여 $tan\theta$ 값의 경우 $sin\theta$ 및 $cos\theta$ 값과는 달리 θ가 커짐에 따라 그 함수 값이 기하급수적으로 증가하는 문제가 있으며 이로 인해 하드웨어 구현 시에 하드웨어 자원 사용량이 크게 증가할 수 있다. 이에 $tan\theta$ 및 $cot\theta$ 값의 증감 폭이 대칭 관계를 가진다는 특성을 이용하여 $tan\theta$의 오버헤드가 커지는 $\frac{\pi}{4} < \theta < \frac{3\pi}{4}$ 범위의 $\Delta\theta$에 대한 보트 연산은 수식 (2)의 양변을 $sin\theta$로 나누어 수식 (4)를 이용해 수행한다.

$$B''(x+dx, y+dy) = A''(x,y) + dx \cdot cot\theta + dy \quad (4)$$

이때 수식 (2)의 dx, dy 항의 계수가 모두 소수부를 가지는데 반해 수식 (4)는 dy 항의 계수가 정수 1을 가지게 되므로 dx 항이 0이 되는 y축 방향으로의 연속적인 픽셀 입력에 대한 보트 연산이 단순해진다.

이에 본 논문에서는 수식 (3)과 수식 (4)를 이용하여 직선 검출의 정확성에 영향을 미치지 않으면서도 허프 변환의 보팅 연산의 연산량을 최소화 하여 하드웨어 구현에 최적화된 개선된 허프 변환 보팅 정책을 제안한다.

2.3. 개선된 허프 변환 보팅 정책

기존의 제안된 허프 변환 보팅 정책은 연속적으로 입력되는 인접한 픽셀들의 보트가 가지는 상관관계를 이용하여 수식 (2)와 같이 보트 연산을 단순화 한다. 하지만 수식 (2)의 하드웨어 구현은 시간당 처리량을 높이기 위해 각 픽셀들을 병렬적으로 보팅함에 있어 많은 오버헤드를 가진다는 한계가 있다. 본 논문에서는 적은 오버헤드만으로 인접한 픽셀들의 보트 연산이 높은 시간당 처리량을 가지게 하기 위하여 인접한 N개의 픽셀을 묶어 병렬 픽셀 블록(parallel block) 단위로 보트 빈도를 누적하는 보팅 정책을 제안한다. 그림 3은 제안한 개선된 허프 변환 보팅 정책의 블록 다이어그램을 나타낸다.

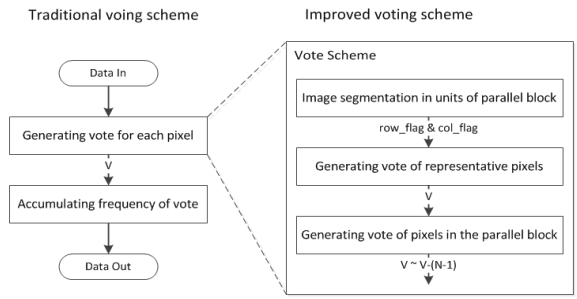

그림 3. 개선된 허프 변환 보팅 정책의 블록 다이어그램vote
Fig. 3. Block diagram of the improved hough transform voting scheme

그림 3은 개선된 허프 변환 보팅 정책은 기존의 보팅 정책에서 입력되는 한 픽셀에 대한 보트를 연산하고 보트 값에 대응되는 보트 메모리 주소에 빈도수를 누적하는 과정을 개선하여 추가연산 없이 다수 픽셀의 보트를 연산하고 빈도를 누적할 수 있도록 하는 정책이다.

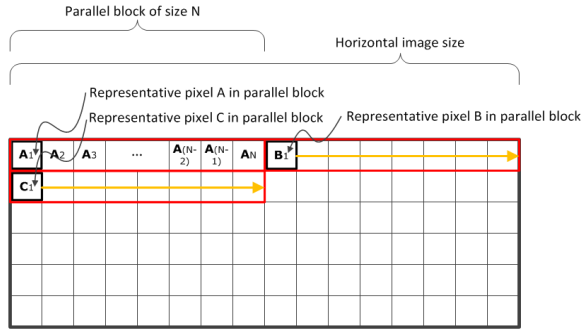

그림 4. 수식 (3)기반 병렬 픽셀 블록 간 상관관계
Fig. 4. Correlation between parallel blocks based on Equation (3)

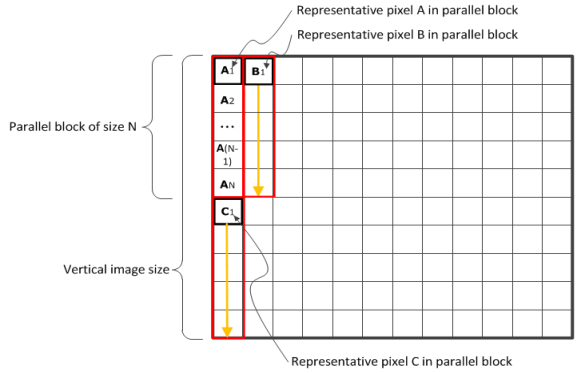

그림 5. 수식 (4)기반 병렬 픽셀 블록 간 상관관계
Fig. 5. Correlation between parallel blocks based on Equation (4)

개선된 보팅 정책에서 'Image segmentation in units of parallel block'은 병렬적으로 보팅을 수행할 픽셀 N개를 묶은 병렬 픽셀 블록 단위로 입력 영상을 구분하는 과정이다. 이는 그림 4에서와 같이 (x,y) 좌표 공간에서 영상 가로 좌표 크기의 약수(divisor)로 N을 정하여 1*N 블록 단위로 보팅 과정을 수행하는 것이다. 상기 병렬 픽셀 블록에서 처음에 위치한 A_1, B_1, C_1 등은 병렬 픽셀 블록 내의 대표 픽셀로서 dx는 N, dy는 1의 인접한 상관관계를 가지며, 'Operation vote of representative pixels' 과정에서 수식 (3)을 이용해 이전 대표 픽셀의 보트로부터 정수 N과 미리 계산된 상수 $tan\theta$의 단순 덧셈만으로 인접 대표 픽셀의 보트 연산을 수행한다. 이는 상수 $cos\theta$와 상수 $sin\theta$를 누산하는 방식의 수식 (2)에 비해 x축 방향의 누산이 정수 N의 가산으로만 이루어지므로 하드웨어 구현 시에 누산 레지스터의 크기를 줄일 수 있어 하드웨어 자원 사용량 측면에서 효율적이다. 다음으로 개선된 보팅 정책에서 'Operating vote of pixels in te parallel block'은 각 $\Delta\theta$에 대한 대표 픽셀의 보트를 기준으로 병렬 픽셀 블록 내 N개의 모든 픽셀에 대한 보트를 생성하는 과정이다. 그림 4에서 병렬 픽셀 블록 내 N개의 픽셀 A_1~A_N은 dx는 1, dy는 0의 인접한 상관관계를 가지므로 인접 보트의 연산 수식 (3)을 수식 (5)와 같이 변형하여 적용할 수 있고, 수식 (5)를 적용하여 병렬 픽셀 블록 내 각 픽셀들의 보트는 정수 1차이의 연속된 값으로 단순하게 구해진다.

$$B'(x+dx,y+dy) = A'(x,y) + 1 \quad (5)$$

따라서 병렬 픽셀 블록 내 N개의 픽셀들에 대한 보트 빈도를 갱신함에 있어서 대표 픽셀 A_1의 보트가 V라면 추가연산 없이 픽셀 A_2~A_N의 보트는 V+1~V+(N-1)의 값을 가짐을 알 수 있고, 보트 V에 해당하는 메모리 주소부터 연속된 N개의 주소 값에 누적 빈도를 갱신함으로써 대표 픽셀의 보트만으로 모든 보트의 빈도를 갱신하게 된다. 이는 기존 허프 변환 보팅 정책에서 각 픽셀들의 보트를 수식 (2)를 이용하여 독립적으로 구하는 방식에 비하여 오버헤드가 매우 낮으며, 하드웨어 구현 시에 병렬 픽셀 블록 내 각 픽셀들의 보트를 별도의 보트 연산기 없이 가산기 하나로 구현할 수 있어 병렬화 및 하드웨어 자원사용량 측면에서 매우 효율적이다.

단, $\frac{\pi}{4} < \theta < \frac{3\pi}{4}$ 범위의 $\Delta\theta$에 대해서는 그림 5와 같이 영상 세로 크기의 약수로 N을 정하여 N*1 블록 단위로 보팅 과정을 수행한다. 그리고 그림 5에서 대표 픽셀의 보트 연산은 수식 (4)를 적용하여 탐색방향을 세로방향으로 함으로써, y축 방향으로 이웃한 대표 픽셀의 보트 연산을 소수 값의 누산으로 수행하는 기존 수식 (2)의 방식에 비해 정수 N의 누산만으로 수행할 수 있다. 이는 하드웨어 구현 시에 누산 레지스터의 크기를 줄일 수 있으므로 하드웨어 자원 사용량 측면에서 효율적이다. 또한 병렬 픽셀 블록 내 N개의 픽셀 A_1~A_N가 dx는 0, dy는 1의 인접한 상관관계를 가지므로 보트 빈도 갱신 시에 수식 (4) 대신 수식 (6)과 같은 형태로 적용할 수 있으며, 수식 (6)을 적용하여 병렬 픽셀 블록 내 각 픽셀들의 보트는 정수 1차이의 연속된 값으로 단순하게 구해진다. 그리고 상기 과정과 동일하게 연속된 메모리 주소에 누적 빈도를 갱신함으로써 대표 픽셀의 보트만으로 모든 보트의 빈도를 갱신하게 된다. 이는 기존 허프 변환 보팅 정책에서 각 픽셀들의 보트를 수식 (2)를 이용하여 독립적으로 구하는 방식에 비하여 오버헤드가 매우 낮다. 따라서 하드웨어 구현 시에 기존 보팅 정책에서 누산을 위한 두 개의 레지스터와 두 개의 가산기를 필요로 하는데 반해 병렬 픽셀 블록 내 각 픽셀들의 보트를 가산기 하나로 구현할 수 있어 병렬화 및 하드웨어 자원사용량 측면에서 효율적이다.

$$B'(x+dx,y+dy) = A''(x,y) + 1 \quad (6)$$

이에 본 논문에서는 제안한 개선된 허프 변환 보팅 정책을 이용하여 직선 검출의 정확성에 영향을 미치지 않고 허프 변환의 보팅 연산의 연산량을 최소화 하면서 시간당 처리량을 높이는 하드웨어 구조를 제안한다.

Ⅲ. 개선된 허프 변환 보팅 정책을 적용한 하드웨어 구조

본 장에서는 그림 6과 같이 개선된 허프 변환 보팅 정책을 적용한 효율적인 하드웨어 구조를 제안한다. 본 논문에서 제안하는 개선된 허프 변환 보팅 정책 기반 하드웨어 구조는 수식 (3) ~ 수식 (6)을 이용하여 각 $\Delta\theta$에 대해 독립적으로 보팅 연산을 수행하는 병렬처리 구조이며, 매 클럭 사이클

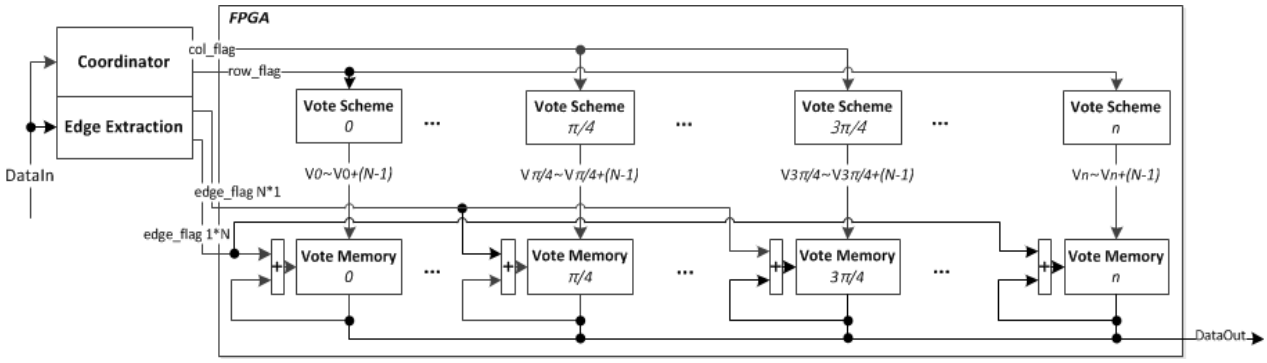

그림 6. 개선된 허프 변환 보팅 정책 하드웨어 구조
Fig. 6. Proposed hardware architecture of the improved hough transform voting scheme

(clock cycle) 마다 N개의 픽셀에 대한 보팅을 동시에 수행할 수 있는 구조이다. 먼저 Coordinator 모듈은 입력된 데이터를 바탕으로 줄바꿈 신호인 row_flag 신호와 유효 프레임 신호인 frame_flag 신호를 생성한다. 그리고 Edge Extraction 모듈에서는 영상의 외곽선 정보를 추출한 뒤, Coordinator 모듈과 동기를 맞추어 edge_flag 신호를 생성한다. 개선된 보팅 정책에 따른 Vote Scheme 모듈은 입력받은 신호들을 사용하여 원점에서부터 인접한 좌표를 이동해 가며 대표 보트를 연산하며, $0 < \theta < \frac{\pi}{4}$, $\frac{\pi}{4} < \theta < \frac{3\pi}{4}$ 범위의 $\Delta\theta$에 대해서는 수식 (3)을 기반으로 연산을 수행하고 $\frac{3\pi}{4} < \theta < \pi$ 범위의 $\Delta\theta$에 대해서는 수식 (4)를 기반으로 연산을 수행한다. 또한 병렬적으로 각 대표 픽셀과 인접한 N개 픽셀의 모든 각도에 대한 보트를 병렬적으로 생성한다. 보트 메모리에서는 생성된 N개 픽셀에 대한 보트 빈도를 메모리에 갱신하는 보팅 과정을 반복한다. 이를 위해 병렬적으로 인접한 N개 픽셀의 보트를 생성하는 Vote Scheme 모듈은 모든 $\Delta\theta$에 대해서 각 각 독립적으로 존재해야 한다.

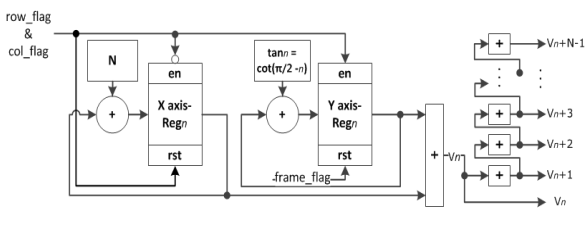

그림 7. Vote Scheme 모듈의 하드웨어 구조
Fig. 7. Proposed hardware architecture of the Vote Scheme module

그림 7은 Vote Scheme 모듈에서 대표 픽셀의 보트 연산을 위한 하드웨어 구조이며, 대표 픽셀 들이 dx는 N, dy는 1의 관계를 가지므로 수식 (3)을 이용하여 이전 보트를 기억하는 두 개의 레지스터와 변위에 따라 정수 N과 상수 $tan\theta$를 각 레지스터에 가산하는 누산기만으로 단순하게 구성한다. 이는 누산에 요구되는 두 개의 레지스터가 모두 소수부를 포함하는 수식 (2)를 이용한 하드웨어 구조와 비교하였을 때 하드웨어 자원사용량 측면에서 유리하다. 이때 row_flag는 y축 방향의 다음 입력을 나타내고, frame_flag는 다음 프레임의 입력을 나타내며, Vn은 상기 대표 픽셀의 보트를 의미한다. 그리고 Vn ~ Vn+N-1은 병렬 픽셀 블록 내에 N개 픽셀의 보트를 의미하며, 블록 내 픽셀들의 보트는 연속된 값을 가진다. 따라서 제안한 구조에서 병렬 픽셀 블록 내 픽셀의 병렬적인 보트 연산기 하드웨어는 추가적인 정수부 가산기 하나로 구현이 가능하므로, 참조문헌 [15]에서 병렬 픽셀 블록 내 픽셀의 병렬적인 보트 연산기 하드웨어 구현에 추가적인 가산기, AND 게이트 및 디코더가 요구되는데 반해 병렬화에 필요한 자원을 최소화한다.

또한 $tan\theta$ 값은 $cot\left(\frac{\pi}{2}-\theta\right)$ 값과 같이 $\frac{\pi}{4}$를 기준으로 대칭을 이루므로 수식 (3)기반의 Vote Scheme 모듈의 대표 보트 출력 값 Vn과 수식 (6) 기반의 Vote Scheme(90-n) 모듈의 대표 보트 출력 값 V(90-n)은 동일한 값을 가진다. 따라서 수식 (3) 기반 대표 픽셀의 보트 연산기에서 row_flag 입력신호를 x축 방향의 다음 입력을 나타내는 col_flag 입력신호로 입력신호만을 달리하여 수식 (6)기반 대표 픽셀의 보트 연산기를 구현할 수 있어 하드웨어 재사용성을 높인다.

Ⅳ. 성능 검증 및 분석

본 장에서는 제안하는 개선된 허프 변환 보팅 정책 하드웨어 구조가 기존의 허프 변환 기법과 검출 정확성에 차이가 없음을 증명하고, 사용한 보트 연산기가 기존 연구들에서 사용한 연산기에 비해 하드웨어 사용량 및 시간당 처리량 면에서 우수함을 증명한다.

먼저, 제안한 하드웨어 구조의 검출 성능을 비교 분석하기위해 실제 구현된 논문 중 검출 성능이 우수하고, 시간당 처리량과 연산속도가 가장 높은 참조문헌 〔15〕와 검출 성능을 비교한다. 참조문헌 〔15〕에서는 제안하는 구조의 검출성능을 비교분석하기 위해 HDL 기반으로 하드웨어 구조를 모델링 하였고, Quartus Ⅱ Version 9.1 with SP2, EP2S180F1508C3 환경에서 고정 소수점 자리 비트수(fractional bit)를 가변하며 보트 연산 값의 오차율(maximum error of vote)을 측정하였다. 표 1은 참조문헌 〔15〕와 제안한 하드웨어 구조의 고정 소수점 자리 비트수를 가변함에 따른 보트 연산 값의 오차율을 나타낸다. 표 1에서 보는바와 같이 제안하는 하드웨어 구조는 기존 연구에 비해 검출 성능 측면에서 차이가 거의 없음을 확인하였다. 이는 새로운 정책을 제시하는 과정에서 기존 알고리즘을 근사화, 간략화 하는 것이 아니라 형태만 변형시켰기 때문이며, 이로 인한 연산 값의 손실이 없어 검출 성능이 떨어지지 않음을 확인하였다.

또한 표 2는 병렬 픽셀 블록의 크기 N을 증가시켜가며 Vote Scheme 모듈의 하드웨어 자원 사용량을 비교한 결과를 보여준다. 대표 픽셀의 보트를 연산하는 오버헤드에 비해 병렬 픽셀 블록 내 인접한 픽셀들의 보트를 생성하는 오버헤드가 작으므로 N의 크기가 늘어날수록 보트 하나를 연산하는데 드는 자원사용량(ALUTs/N)이 줄어들어 효율적으로 병렬화가 이루어짐을 확인할 수 있다. 이는 참조문헌 〔15〕에서 병렬화를 위해 가산기, AND 게이트, 디코더가 요구되는 데에 반해 제안하는 하드웨어 구조에서는 병렬화를 위해 가산기만이 요구되고 그 수가 훨씬 적기 때문이다.

표 1. 고정 소수점 자리 비트수에 따른 오차율 비교
Table 1. Error rate comparison according to the number of fractional bits

Number of Fractional Bits	Maximum Error of Vote ([15])	Maximum Error of Vote (Proposed)
10	0.375	0.375
11	0.188	0.188
12	0.094	0.094
13	0.047	0.047

표 2. 병렬 픽셀 블록 크기 N에 따른 자원 사용량 비교
Table 2. Hardware resources comparison according to the size of parallel blocks

N	ALUTs	ALUTs/N	Maximum Frequency
2	57	28.50	449
4	79	19.75	449
8	122	15.25	449
16	208	13.00	425

표 3. 기존 연구들과의 보트 값 연산기의 성능 비교
Table 3. Performance comparison between the vote architectures with different approaches

Performance Index	Pipelined CORDIC [12]	DA [13]	Chern's Method [14]	Zhong-Ho's Method [15]	Proposed
Error	0.177	0.125	0.125	0.094	0.094
ALUTs	520	13	55	416	208
Throughput per Cycle	2	1/9	1	16	16
Maximum Frequency (MHz)	384	500	387	333	425
Throughput (K pixels * angles)	768	56	387	5328	6800
Throughput/ ALUTs	1.477	4.274	7.036	12.808	32.692

다음으로, 제안하는 하드웨어 구조가 시간당 처리량과 하드웨어 자원사용량 측면에서 우수함을 증명

하기 위해 기존의 다른 연구들과 비교분석 하였다. 표 3은 제안하는 Vote Scheme 모듈의 하드웨어 구조와 참조문헌 [15]에서 분석한 기존 연구들의 보트 연산을 수행하는 하드웨어 부분에 대한 검출 오차율(Error), 하드웨어 자원사용량(ALUTs), 클럭당 처리량(Throughput per Cycle), 최대 동작 주파수(Maximum Frequency), 시간당 처리량(Throughput), 하드웨어 자원사용량 대비 처리량(Throughput/ALUTs)을 나타낸다. 표 3의 기존 연구들 중 검출 오차율과 하드웨어 자원 사용량 대비 시간당 처리량이 가장 우수한 참조문헌 [15] 연구와의 객관적인 성능 비교를 위하여, 제안한 하드웨어 구조의 클럭당 처리량을 참조문헌 [15]와 동일한 16으로 설계하여 성능을 비교 분석한다. 제안한 하드웨어 구조는 참조문헌 [15]와 픽셀 간 상관관계를 이용한 누산기의 구조가 동일하여 동일한 검출 오차율을 가진다. 하드웨어 비용 측면에서, 참조문헌 [15]의 하드웨어 구조는 누산에 필요한 두 개의 레지스터가 모두 소수부를 포함해야 하는데 반해, 제안한 구조에서는 하나의 레지스터를 정수부만 가지는 레지스터로 대체할 수 있다. 추가로 표 2에서와 같이 처리량을 높임에 따른 오버헤드가 매우 적어, 참조문헌 [15]와 동일한 클럭당 처리량을 가질 때 요구되는 하드웨어 자원사용량이 절반에 그친다. 또한 참조문헌 [15]에서는 병렬 픽셀 블록 내 픽셀의 보트를 연산하기 위해서 소수부를 포함한 가산을 수행한 후 이의 정수부만을 취하기 위해 추가적인 가산기 및 디코더를 순차적으로 거쳐야하지만, 제안하는 하드웨어 구조에서는 병렬 픽셀 블록 내 픽셀의 보트를 연산하기 위해서 한 번의 정수부 가산만이 요구되어 참조논문 [15]에 비해 임계 경로가 개선되었기 때문에 우수한 최대 동작 주파수를 가진다.

따라서 제안하는 하드웨어 구조는 높은 동작 주파수를 만족하며 적은 하드웨어 자원으로 높은 처리량을 가지므로, 하드웨어 자원사용량 대비 시간당 처리량이 기존의 다른 연구들과 비교하여 가장 높은 것을 확인할 수 있다.

V. 결 론

본 논문은 하드웨어 자원사용량 및 시간당 처리량 면에서 우수하면서도 기존의 허프 변환 기법과 직선 검출 정확도 측면에서 동일한 성능을 가지도록 보팅 정책을 개선한 허프 변환의 하드웨어 구조를 제안하였다. 새로운 정책을 제시하는 과정에서 기존 알고리즘을 근사화, 간략화 하는 것이 아니라 형태의 변형만 있었기 때문에 연산 값의 손실이 없어 검출 성능이 떨어지지 않음을 확인하였다. 또한, 허프 변환 기법의 병목(bottle neck) 부분인 보트 값 연산이 기존의 소수점 이하의 값을 포함하는 연산에서 정수 연산으로 대체되므로 인해 하드웨어 자원사용량이 감소하였다. 제안된 구조는 픽셀 간 병렬화 및 다수의 보트 빈도를 메모리에 누적하는데 사용되는 추가 하드웨어 자원을 최소화하였기 때문에, 병렬화를 통해 시간당 처리량을 높이기에 유리하다. 게다가, 모든 θ에 대한 보트를 독립적으로 연산하기 위한 보트 연산기의 구현을 동일하게 하여 하드웨어 재사용성을 높인다. 동일한 실험 환경에서 기존의 연구들과 비교 분석 결과 제안한 하드웨어 구조는 동일한 자원사용량에서 시간당 처리량이 가장 우수함을 확인하였다.

References

[1] C. T. Ho and L. H. Chen, "A high speed algorithm for line detection," *Pattern Recognition Lett.*, vol. 17, no. 5, pp. 467-473, May 1996.

[2] Y. S. Lee, H. S. Koo, and C. S. Jeong, "A straight line detection using principal component analysis," *Pattern Recognition Lett.*, vol. 27, no. 14, pp. 1744-1754, Oct. 2006.

[3] M. H. Chang and J. A. Park, "Information extraction of the moving objects based on edge detection and optical flow," *J. Korea Inform. Commun. Soc. (KICS)*, vol. 27, no. 8A, pp. 822-828, Aug. 2002.

[4] P. V. C. Hough, "Methods and means for recognizing complex patterns," *U.S. Patent 3.069.654*, Dec. 1962.

[5] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11-15, Jan. 1972.

[6] F. O'Gorman and M. B. Clowes, "Finding picture edges through collinearity of feature points," *IEEE Trans. Comput.*, vol. C-25, no. 4, pp. 449-456, Apr. 1976.

[7] W Niblack and D Petkovic, "On improving the accuracy of the hough transform," *Machine Vision Applicat.*, vol. 3, no. 2, pp. 87-106, Mar. 1990.

[8] L. A. F. Fernandes and M. M. Oliveira, "Real-time line detection through an improved hough transform voting scheme," *Pattern Recognition*, vol. 41, no. 1, pp. 299-314, Jan. 2008.

[9] J. Princen, J. Illingworth, and J. Kittler, "A hierarchical approach to line extraction based on the hough transform," *Comput. Vision, Graphics, Image Process.*, vol. 52, no. 1, pp. 57-77, Oct. 1990.

[10] S. B. Yacoub and J. Jolion, "Hierarchical line extraction," *IEE Proc. Vision, Image, Signal Process.*, vol. 142, no. 1, pp. 7-14, Feb. 1995.

[11] K. Hanahara, T. Maruyama and T. Uchiyama, "A real-time processor for the hough transform," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 1, pp. 121-125, Jan. 1988.

[12] F. Zhou and P. Kornerup, "A high speed hough transform using CORDIC," in *Proc. Int. Conf. Digital Signal Process. (DSP 1995)*, pp. 27-39, Limassol, Cyprus, June 1995.

[13] K. Mayasandra, S. Salehi, W. Wang, and H. M. Ladak, "A distributed arithmetic hardware architecture for real-time hough-transform-based segmentation," *Canadian J. Elect. Comput. Eng.*, vol. 30, no. 4, pp. 201-205, May 2005.

[14] M. Y. Chern and Y. H. Lu, "Design and integration of parallel hough transform chips for high-speed line detection," in *Proc. 11th Int. Conf. Parallel Distributed Syst. Workshops*, vol. 2, pp. 42-46, Fukuoka, Japan, July 2005.

[15] Z. H. Chen, A. W. Y. Su, and M. T. Sun, "Resource-efficient FPGA architecture and implementation of hough transform," *IEEE Trans. VLSI Syst.*, vol. 20, no. 8, pp. 1419-1428, Aug. 2012.

이 정 록 (Jeong-Rok Lee)

2011년 8월 경북대학교 IT대학 전자공학부 학사 졸업
2011년 9월~현재 경북대학교 IT대학 전자공학부 석사과정
<관심분야> SoC, Computer Vision, 컴퓨터 구조

배 경 렬 (Kyeong-ryeol Bae)

2009년 2월 경북대학교 전자전기컴퓨터학부 학사 졸업
2011년 2월 경북대학교 전자전기컴퓨터학부 석사 졸업
2011년 3월~현재 경북대학교 IT대학 전자공학부 박사과정
<관심분야> SoC, Stereo Vision, 컴퓨터구조

문 병 인 (Byungin Moon)

1995년 2월 연세대학교 전자공학과 학사 졸업
1997년 2월 연세대학교 전자공학과 석사 졸업
2002년 2월 연세대학교 전기전자공학과 박사 졸업
2002년~2004년 하이닉스 반도체 선임연구원
2004년~2005년 연세대학교 연구교수
2005년~현재 경북대학교 IT대학 전자공학부 부교수
<관심분야> SoC, 디지털 VLSI, 컴퓨터 구조