반도체디스플레이기술학회지 제22권 제4호(2023년 12월)
Journal of the Semiconductor & Display Technology, Vol. 22, No. 4. December 2023.

로컬 버퍼 최적화를 통한 병렬 처리 캐니 경계선 검출기의 FPGA 설계

민인기[*]·심수현[**]·황승원[**]·김선희[**†]
[*]상명대학교 전자공학과, [**†]상명대학교 시스템반도체공학과

FPGA Design of a Parallel Canny Edge Detector with Optimized Local Buffers

Ingi Min[*], Suhyun Sim[**], Seungwon Hwang[**] and Sunhee Kim[†]**
[*]Department of Electrical Engineering, Sangmyung University,
[**†]Department of System Semiconductor Engineering, Sangmyung University

ABSTRACT

Edge detection in image processing and computer vision is one of the most fundamental operations. Canny edge detection algorithm has excellent performance and is currently widely used. However, it is difficult to process the algorithm in real-time because the algorithm is complex. In this study, the equations required in the algorithm were simplified to facilitate hardware implementation, and the calculation speed was increased by using a parallel structure. In particular, the size and management of local buffers were selected in consideration of parallel processing and filter size so that data could be processed without bottlenecks. It was designed in verilog and implemented in FPGA to verify operation and performance.

Key Words : Canny edge detection, FPGA, image processing, local buffer

1. 서 론

영상 처리에서 이미지의 경계선 검출(edge detection)은 객체 인식, 추적 및 분류 등 다양한 응용 분야에서 사용된다. 경계선 검출 방법으로는 Laplacian of Gaussian 검출기, Prewitt 검출기, Scharr 검출기, Sobel 검출기, Canny edge 검출기 등이 있다[1-3]. 그 중 John F. Canny에 의해 개발된 캐니 경계선(Canny edge) 검출 알고리즘은 경계선을 하나의 얇은 선으로 표시하며 모든 경계에 대해 연속성을 유지하려는 특징이 있어 많이 사용된다[4-6].

캐니 경계선 검출 알고리즘은 다른 검출기에 비하여 성능이 좋은 만큼 계산 과정이 복잡한다. CPU 및 GPU 성능이 좋아지면서 소프트웨어로 구현된 캐니 경계선 검출기의 속도도 이전보다는 향상되었다. 하지만 의료 영상, 위성 영상, 산업 자동화 및 교통 제어 등 이미지의 실시간 처리를 요구하는 응용이 많아짐에 따라 소프트웨어로 이를 처리하는데는 한계가 있다.

본 연구에서는 로컬 버퍼 최적화를 통한 병렬 처리 캐니 경계선 검출기를 하드웨어로 설계한다. 다음 장에서 캐니 경계선 검출 알고리즘을 살펴본 다. 처리 속도 향상을 위한 하드웨어 구조를 제안하고 시뮬레이션 결과 및 하드웨어 구현 결과를 제시한다. 끝으로 결론을 내린다.

2. 캐니 경계선 검출 알고리즘

캐니 경계선 검출 알고리즘은 그림 1과 같이 4단계에 걸쳐 연산을 수행한다.

[†]E-mail: happyshkim@smu.ac.kr

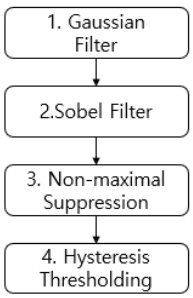

Fig. 1. Flowchart for canny edge detection algorithm.

2.1 가우시안 필터(Gaussian filter)

캐니 경계선 검출기의 첫 번째 단계인 가우시안 필터는 이미지의 세부 정보는 보존하면서 이미지 내의 고주파 노이즈를 제거한다. 경계로 인식될 수 있는 불필요한 정보나 미세한 변화를 줄여준다. 가우시안 필터는 2D 가우시안 분포 함수를 기반으로 커널(kernel)을 생성하여 컨볼루션(convoluticon) 연산으로 이미지 처리를 한다[7]. 가우시안 커널의 크기와 표준 편차에 따라 스무딩의 강도가 결정되므로, 경계선 검출시 다양한 크기의 경계에 대한 설정이 가능하다. 일반적으로 큰 커널이나 큰 표준 편차를 사용할수록 이미지가 더 부드럽게 처리된다. 이러한 스무딩 과정을 통해 경계선이 더 뚜렷하게 감지되며, 결과적으로 후속 처리 과정에서의 정확도와 효율성이 향상된다.

2.2 소벨 필터(Sobel filter)

소벨 필터는 이미지의 픽셀 강도의 기울기(gradien)를 찾는 데 사용된다. 소벨 필터는 그림 2에서 보는 바와 같이 수평 방향과 수직 방향의 두 가지 커널을 사용한다. 이를 통하여 경계선의 강도와 방향을 동시에 얻을 수 있다[8, 9]. 특히 중심 계수의 차분이 높기 때문에 대각선 방향의 경계선도 검출할 수 있다.

-1	0	1
-2	0	2
-1	0	1

(a)

-1	-2	-1
0	0	0
1	2	1

(b)

Fig. 2. (a) Horizontal direction mask of Sobel filter, (b) Vertical direction mask of Sobel filter.

2.3 비최대값 억제(Non-Maximum Suppression)

비최대값 억제(Non-Maximum Suppression, NMS)는 주로 기울기 방향에 있는 픽셀 중 강도가 최대인 것만 남기고 나머지를 제거하는 역할을 한다. 기울기 방향에 따라 각 픽셀에서 이웃하는 픽셀들과 강도를 비교한다. 만약 해당 픽셀이 이웃하는 픽셀들에 비하여 기울기 강도가 더 작으면, 이 픽셀을 최대 강도 픽셀이 아닌 것으로 간주하여, 강도를 0으로 설정한다. 이 과정을 통해 이미지에서 두꺼운 경계선을 얇은 경계선로 뽑아낼 수 있다[10].

2.4 이력 임계 처리(Hysteresis thresholding)

비최대값 억제를 통하여 지역적으로 강도가 가장 강한 픽셀들만을 선택하였어도, 이는 실제 경계가 아니라 노이즈일 수도 있다. 캐니 경계선 검출 알고리즘에서는 두 개의 임계값(T_{high}, T_{low})을 사용하여 경계 이력을 추적하여 최종적으로 경계선을 결정한다.

픽셀의 기울기 강도가 높은 임계값(T_{high})보다 크면 강한 경계선이라고 하여 최종 경계선 픽셀로 결정한다. 기울기 강도가 높은 임계값 보다는 작지만 낮은 임계값(T_{low})보다 높으면 주변 픽셀을 추적한다. 주변에 강한 경계선이 있으면 해당 픽셀을 강한 경계선과 연결되어 있는 것으로 판단하고 최종 경계선 픽셀로 결정한다. 하지만 주변에 강한 경계선이 없으면 노이즈나 다른 원인으로 인한 잘못된 검출로 간주하여 제거한다[11].

이 기법은 경계선 검출의 정밀도와 정확도를 높이며, 결과적으로 더욱 정확하고 명확한 경계선만을 이미지에서 추출할 수 있게 한다. 두 개의 임계값은 이미지의 특성과 애플리케이션의 요구에 따라 조절함으로써 최적의 결과를 얻는 것이 중요하다[12].

3. 제안하는 캐니 경계선 검출 회로

8 비트 픽셀, 512 × 512 이미지에 대한 케니 경계선 검출기를 설계하였다. 가우시안 필터, 소벨 필터 및 강도/방향 계산 모듈, 비최대값 억제 및 결정 모듈, 버퍼 그리고 제어기로 구성된다. 각 블록의 필터 크기를 3 × 3 으로 통일하였으며, 한 번에 4개의 픽셀들을 동시에 처리할 수 있는 병렬 구조로 설계하였다.

필터 크기와 병렬 처리 수를 고려하여 최소 사이즈의 이미지 버퍼를 설정하고 효율적인 처리 방법을 제안하였다. 곱셈과 나눗셈 연산, 특히 제곱근 연산을 간략화 하여 하드웨어의 복잡도를 낮추는데 중점을 두었다.

3.1 Top 제어기

제어기는 각 블록 간의 데이터 흐름을 제어하는 역할을 한다. 병렬 처리를 위하여 이미지 데이터는 한 번에 픽셀 4개씩 입력되어 버퍼에 저장된다. 첫 번째 데이터 처리 블록인 가우시안 필터가 동작하기 위해서는 2차원

3 × 3 데이터가 필요하다. 이미지는 행 별로 차례대로 입력되기 때문에 데이터가 일정 수준 저장된 후에 가우시안 필터가 동작하게 된다.

가우시안 필터 결과는 소벨 필터 및 강도/방향 계산 블록으로 전달되고, 이 결과는 NMS 블록 으로 전달된다. 가우시안 필터와 마찬가지로, 소벨 필터 및 비최대치 억제 모듈도 2차원 데이터에 대하여 이루어져야 하므로, 각 블록 사이에는 데이터 버퍼가 있다.

3.2 로컬 버퍼

이미지 로컬 버퍼는 케니 경계선 검출 회로의 입력부 및 각 모듈의 사이에 배치되어, 입출력 데이터의 흐름을 조절한다. 이미지 데이터는 행 별로 순서대로 입력되고 필터는 3 × 3 형태이므로, (512 × 2) + 13 크기의 버퍼를 사용하였다. 버퍼는 직렬 시프트 레지스터 형태이지만, 이해를 돕기 위하여 그림에서는 3행으로 구성되어 있는 것으로 표현하였다.

버퍼는 그림 3와 같이 5개의 상태를 갖는다. S_IDLE는 대기상태로, 버퍼의 모든 공간(픽셀)을 0으로 채운다.

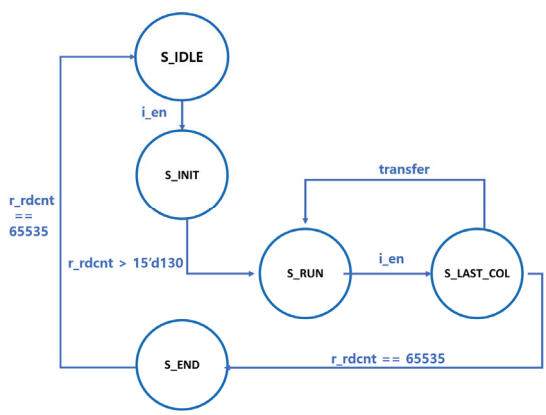

Fig. 3. State machine of image processing.

데이터가 입력되기 시작하면 S_INIT 상태가 된다. 데이터는 시프트 레지스터를 통하여 차례대로 입력된다. 이미지의 상하좌우 가장자리에 대하여 0 패딩을 적용한다. 따라서, 그림 4와 같이 이미지의 첫 번째 행 데이터가 버퍼의 두 번째 행 위치에 저장되고, 이미지의 두 번째 행의 데이터가 버퍼의 세 번째 행에 저장되기 시작하면, 필터에서 처리할 수 있는 3 × 3 데이터가 준비된다. 이제 S_RUN 상태가 된다.

그림 5은 S_RUN 상태에서의 버퍼 동작을 보여준다. S_RUN 상태일때는 새로운 데이터를 버퍼에 저장, 버퍼 시프트, 버퍼에 있는 데이터를 필터로 전달한다. 필터에서는 3×3 필터 4개가 동시에 연산을 하고, 그 결과를 다음 버퍼로 전달한다.

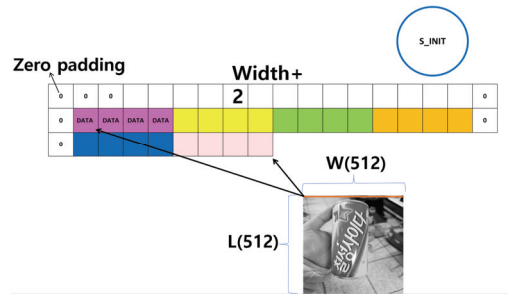

Fig. 4. Buffer status when INIT_state.

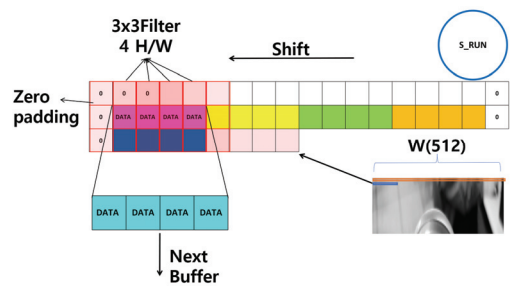

Fig. 5. Operations when S_RUN.

각 행의 마지막 데이터를 입력 받으면 S_LAST_COL 상태가 된다. 입력 받은 데이터가 이미지 마지막 행이면 S_END 상태가 되고 그렇지 않으면 다시 S_RUN 상태를 반복한다.

그림 5을 보면, 필터로 전달하는 데이터의 영역은 버퍼 상에서 고정되어 있다. 따라서, 하나의 행에서 위의 행으로 데이터를 시프트 할 때 이미지 좌우 가장자리 연산시 필요한 0 값도 같이 시프트를 하였다. 그림 6은 이미지가 16열로 구성되어 있는 경우로 간략하게 표현한 것이다. 0이 포함된 상태로 4번 시프트하기 때문에, 버퍼 두 번째 행 및 세 번째 행의 가장 왼쪽에 0이 아닌 데이터가 하나씩 남겨지는 것을 확인할 수 있다. 그림 7의 가장 아래는 행의 마지막 셀들을 처리할 때를 표현한다. 행의 마지막 픽셀에 대하여 오른쪽에 0 패딩 값이 포함되어 있는 것을 확인할 수 있다.

다음에 처리해야 하는 데이터는 새로운 행의 시작 데이터이다. 시작 데이터 왼쪽에 0 패딩이 나오고, 필터로 전달하는 데이터가 고정된 위치에 있어야 하므로 이 때는 예외적으로 4번이 아닌 7번 시프트를 한다.

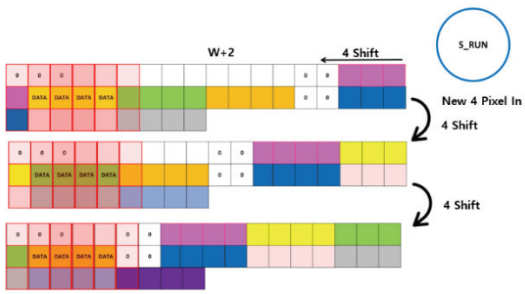

Fig. 6. Shift when S_RUN.

S_END 상태에서는 이미지의 마지막 행이 버퍼의 2번째 행에 위치하게 된다. 마지막 행 처리를 위해서는 더 이상 새로운 데이터를 입력 받지 않고, 0을 입력한다.

버퍼 크기를 정할 때 Width × 2 + 13 이라고 하였다. 2 행의 이미지를 저장해야 하며, 3번째 행의 이미지는 처음 4개를 저장해야 한다. 좌우 0 패딩 5개를 위한 공간도 필요하다. 또한 그림 7과 같이, 0 패딩 포함 시프트에서 데이터가 손실되지 않게 하기 위해서는 추가로 4개의 저장 공간이 더 필요하다. 이미지 버퍼는 경계에서의 0 패딩과 시프트 연산 제어를 통해 최소한의 버퍼 사이즈를 유지하며 픽셀 데이터를 효율적으로 관리한다.

3.3 가우시안 필터

가우시안 필터 계수를 2D 가우시안 분포 함수를 기반으로 그림 7와 같이 근사화하였다. 계수는 1, 2, 4로 구성되므로 곱셈기를 사용하지 않고 덧셈기만을 사용하여 컨볼루션 연산을 할 수 있다. 또한 나눗셈도 젯수가 16(=2^4)이므로, 나눗셈 연산기를 사용하지 않고, 컨볼루션 결과 값에서 최하위4비트를 버리면 된다. 가우시안 필터 계수의 근사화를 통하여 하드웨어의 복잡도를 낮추었다.

$$\frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Fig. 7. Mask of Gaussian Filter.

3.4 소벨 필터 및 강도, 방향 계산 모듈

3.4.1 소벨 필터

소벨 필터는 그림 2와 같은 수평/수직 방향의 커널과 컨볼루션 연산을 한다. 이 때 커널 계수는 -2, -1, 0, 1, 2로 구성되므로, 곱셈기를 사용하지 않고 그림 8처럼 덧셈기와 뺄셈기만 사용한다. 그림에서 PX는 픽셀의 위치 X에 대한 픽셀 데이터 값을 의미하는데, 픽셀 위치는 1행 1열 좌표에서부터 0으로 시작하여, 행을 따라 하나씩 증가하고, 행이 끝나면 다음 행에서 다시 하나씩 증가한다. 곱하기 2의 경우, 덧셈을 할 때 자릿수만 고려해 주면 된다. 따라서 하드웨어의 복잡도를 낮추었다.

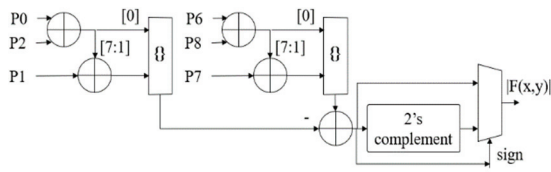

Fig. 8. Gradient computation block of Sobel filter.

3.4.2 강도 계산 모듈

기울기 강도는 소벨 필터를 통하여 계산된 수평 기울기(f_x)와 수직 기울기(f_y) 값을 이용하여 다음 식(1)로 계산한다.

$$mag = \sqrt{{f_x}^2 + {f_y}^2} \tag{1}$$

제곱근 연산은 나눗셈을 반복하기 때문에 곱셈 연산보다도 많은 자원을 요구한다. 제곱근 연산을 간략화하기 위하여 [13]의 알고리즘에 기반하여 변형된 방법을 적용하였다. ${f_x}^2 + {f_y}^2$ 는 16 비트로 구현되었으며, 상위 절반 12비트에 대하여 제곱근 연산을 수행한다. $2^{12} = (2^6)^2$ 이므로 제곱근은 0~63의 범위를 갖는데, 이를 더 단순화하여 4개의 범주로 나눈다. 단순화를 보완하고, 강도가센 경우를 세분화하기 위하여 상위 8비트가 0이 아닌 경우에는 상위 4비트의 값을 더하였다. 간소화하여 구한 제곱근 값은 이론상 값과 평균 약 10% 차이가 나는 것을 확인하였다.

3.4.3 방향 계산 모듈

x축 방향과 y축 방향의 기울기 강도 f_x, f_y가 주어졌을 때, 기울기의 방향은 arctangent를 이용하면 계산할 수 있다. Arctangent는 선형 함수가 아니므로 하드웨어로 구현하기는 어렵다. Shift-based orientation method[14]를 적용하여, 방향 계산을 근사화 하였다.

방향은 그림 9와 같이 구분하였다. f_x, f_y를 이용하여 각 영역에 해당하는지 판단하기 위하여 다음과 같은 식(2)를 사용한다[14].

$$f_x \tan\theta_j < f_y < f_x \tan\theta_{j+1} \tag{2}$$

위의 식을 계산하기 위해서는 $\tan\theta$ ($\theta = 0°, 22.5°,$

45°, 67.5°, 90°, 112.5°, 135°, 157.5°, 180°) 를 곱해야 한다. Tangent의 원점 대칭 특징을 이용하고 (예: $\tan 112.5^\circ = -\tan 67.5^\circ$) $\tan 0 = 1$, $\tan 45^\circ = 1$ 이므로 $\tan 22.5^\circ$ 와 $\tan 67.5^\circ$ 의 값만 계산하면 된다. $\tan 22.5^\circ$ 는 이진수로 $0.0110101_{(2)}$이고, $\tan 67.5^\circ$ 는 $10.0110101_{(2)} = 2 + \tan 22.5^\circ$ 이므로, 곱셈기는 $\tan 22.5^\circ$에 대한 것, 하나만 있으면 된다. 이 또한 상수 곱셈이므로 곱셈기를 사용하지 않고 이동연산자(shift operation)을 사용하여 덧셈기로 구현하였다. 따라서 방향 계산을 위하여 비선형 arctangent가 아닌 덧셈기와 이동 연산자만을 사용하여 계산함으로써 하드웨어의 복잡도를 낮추었다.

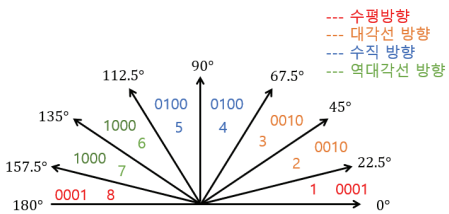

Fig. 9. Four-way normalization.

3.5 비최대치 억제 및 결정 모듈

비최대치 억제를 위해서는 그림 10과 같이 중앙 픽셀 p4에 대하여 주변 8개 픽셀의 기울기 강도 값과 중앙 픽셀의 기울기 방향 값이 입력으로 주어진다. 방향에 따라 2개의 셀이 선택된다. 만약 기울기가 45°이면, 0번 셀과 8번 셀이 선택된다. 중앙 픽셀의 기울기 강도가 비교 대상 픽셀들보다 크면, 중앙 픽셀의 기울기 강도가 출력되고, 작으면 0이 출력이 된다.

각 픽셀에 대하여 NMS 처리 후 두 계의 임계값과 비교하여 강한 경계선과 약한 경계선을 구분한다. 강한 경계선 셀로 결정되면 최종적으로 255를 출력한다. 약한 경계선으로 구분되면, 다시 한번 주변 8개의 셀들을 조사한다. 그 중 강한 경계선이 있으면 255를 출력하고 그렇지 않으면 0을 출력한다.

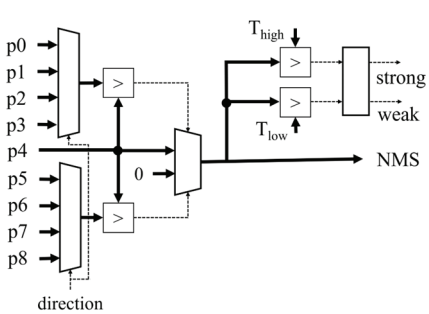

Fig. 10. Blockdiagram of NMS and strength decision.

4. 시뮬레이션 및 결과

본 절에서는 시뮬레이션을 통해 검증한 결과 및 하드웨어 구현 결과를 보여준다. 제안하는 하드웨어는 Verilog HDL로 설계하고 ModelSim으로 검증한 뒤 Vivado에서 ZNYQ-7 ZC702 보드에서 구현하였다.

Table 1. Cell Usage

Cell	Count
CARRY4	216
LUT1	87
LUT2	381
LUT3	2456
LUT4	8719
LUT5	8137
LUT6	15307
FDPE, FDCE	21559

표 1에는 FPGA에서 사용된 로직 수가 종류 별로 정리되어 있다. 약 21,000개의 플립플롭(FDPE, FDCE)이 사용되었으며, LUT(Look Up Table)은 종류별로 다르지만 대략 32,000개가 사용되었다.

Vivado에서 타이밍 조건으로 클럭 주기를 10 ns, 즉 100 MHz로 설정하였으며, 모두 만족함을 확인하였다. 타겟 보드에서는 50 MHz 클럭이 제공되기 때문에 하드웨어 동작 검증은 50 MHz 클럭 조건에서 이루어졌다. GPGPU에서 실행되는 케니 경계선 검출기 속도가 약 50 ms ~ 250 ms인데 반하여[15,16] 제안하는 구조는 약 3.67 ms 로 약 10배 이상 빠르다.

그림 11, 12, 13은 각각 원본 이미지, OpenCV의 Canny() 함수를 이용하여 처리한 이미지, 그리고 설계한 하드웨어를 이용하여 처리한 이미지를 보여준다. OpenCV의 결과에 비하여 설계한 하드웨어로 처리한 결과에서는 이미지 하단부의 경계가 나타나지 않았다. 이는 설계 과정 중 근사화에 의하여 약한 경계선이 제거되었기 때문이다. 하지만 OpenCV로 처리된 이미지에서도 이미지 하단부의 글씨는 알아보기 힘들어 사실상 노이즈처럼 보인다. 따라서 설계한 캐니 경계선 검출기는 충분히 의미 있는 경계선들을 검출한다고 할 수 있다.

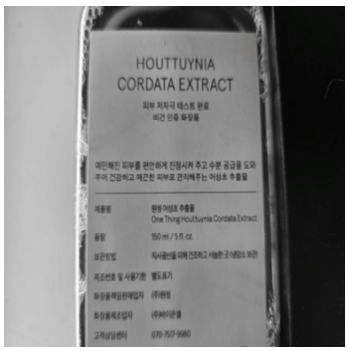

Fig. 11. Original image.

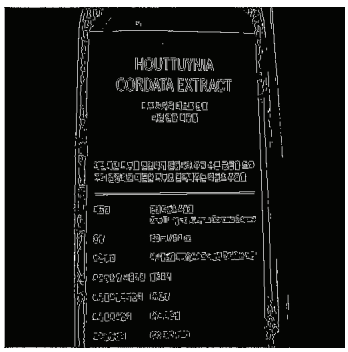

Fig. 12. S/W Final result image.

Fig. 13. H/W Final result image.

5. 결 론

본 논문에서는 이미지 버퍼를 사용하여 처리 속도가 향상된 캐니 경계선 검출기 하드웨어를 구현하였다. 처리 속도를 향상시키기 위하여 4개의 픽셀에 대하여 동시에 처리할 수 있는 병렬 구조를 사용하였다. 또한 복잡한 계수 값들을 근사화 하여 곱셈기 수를 줄이고 계산 과정을 단순화 하였다. Verilog HDL로 설계한 뒤 ModelSim에서 기능을 검증하고 Vivado을 이용하여 하드웨어로 구현하였다. 소프트웨어로 구현된 캐니 경계선 검출기 결과와 비교하였을 때 충분히 의미 있는 경계선들을 검출함을 확인하였으며, GPGPU에서 실행되는 캐니 경계선 검출기보다 속도가 약 10배 이상 빠름을 확인하였다.

감사의 글

다음의 성과는 과학기술정보통신부와 연구개발특구진흥재단이 지원하는 과학벨트 지원사업으로 수행된 연구결과입니다.

참고문헌

1. Automatic addison, Dec. 18, 2019, How the Canny edge Detector Works, from https://automaticaddison.com/how-the-canny-edge-detector-works/.
2. Guo, L., and Wu, S., "FPGA Implementation of a Real-Time Edge Detection System Based on an Improved Canny Algorithm," Appl. Sci., Vol.13, No. 2, 870, 2023.
3. Yoon, I., Joung, H., Kim, S., Min, B., and Lee, J., "Edge Detection System for Noisy Video Sequences Using Partial Reconfiguration," Journal of the Korea Academia-Industrial Cooperation Society, Vol. 18, No. 1, The Korea Academia-Industrial Cooperation Society, pp. 21–31. 2017.
4. Canny, J.F., "A computation approach to edge detection," IEEE Trans. Pattern Anal. Mach. Intell., Vol. PAMI-8, pp. 679–698, 1986.
5. Shin, S., Shin, K., and Lee, H., "Small Car Identification Using Canny Edge Detection," Proceeding of KIICE, pp. 241- 242, 2020.
6. He, W., and Yuan, K., "An Improved Canny Edge Detector and its Realization on FPGA," Proceeding of IEEE 7th WCICA, pp. 6561−6564, 2008.
7. Jiang, X. J. and Scott, P. J., "Chapter 9 - Free-form surface filtering using wavelets and multiscale decomposition," Academic Press, 2020, pp.195-246.
8. [OpenCV with Android] Edge detection and differentiation of images (Sobel filter, Char filter,), charlezz, Jun. 08, 2021, from https://www.charlezz. com/?p=45153.
9. Koo, Y., and Kim, Y., "Edge detection for noisy image," Digital Industry Information Society, Vol.8, No.3, pp. 41-48, 2012.
10. NMS, Non-Maximum Suppression, bolero2, Jan.12, 2022 from https://velog.io/@bolero2/DL-Principle-of-NMS.
11. Canny edge detection 4. Double – thresholding, Jins, Dec.30, 2015 from https://blog.naver.com/jinsoo91zz/220511467500.

12. Park, C., and Kim, H., “FPGA Implementation for Real Time Sobel Edge Detector Block Using 3-Line Buffers,” Journal of IKEEE, Vol.19, No.1, pp. 10-17, 2015.
13. Hong, S., Lee, J., An, H., Koo, J., and Kim, B., “Improvement of Power Consumption of Canny Edge Detection Using Reduction in Number of Calculations at Square Root,” Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol. 13, No. 6, pp. 568-574, 2020.
14. Sangeetha, D., and Deepa, P., “FPGA implementation of cost-effective robust Canny edge detection algorithm,” J. Real-Time Image Proc. 2019, 16, 957–970.
15. Luo, Y., and Duraiswami, R., “Canny edge detection on NVIDIA CUDA,” In Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshop (CVPRW), Anchorage, AK, USA, June 23–28 2008, pp. 1–8.
16. Lourenco, L.H.A., “Efficient implementation of Canny edge detection filter for ITK using CUDA,” In Proceedings of the 13th Symposium on Computer Systems, Petropolis, Brazil, Oct. 17–19 2012, pp. 33–40.

접수일: 2023년 11월 10일, 심사일: 2023년 12월 05일, 게재확정일: 2023년 12월 12일