논문 2010-5-36

광센서용 움직임 벡터 검출 알고리즘 구현

The Implementation of Motion Vector Detection Algorithm for the Optical-Sensor

박노경*, 박상봉**, 박민형***

Park Nho-Kyung, Park Sang-Bong and Park Min-Hyeong

요 약 본 논문에서는 광센서에서 화면상의 픽셀에 대한 변형된 움직임 벡터 추출 알고리즘을 제안한다. 기존 블록 매칭 알고리즘보다 움직임 추출의 정확성을 높이고 연산량을 줄이도록 설계하였다. 제안된 알고리즘은 Cyclone과 삼성 0.35um CMOS 1-poly-4-metal 공정을 이용하여 칩으로 제작하였다. CARTESIAN ROBOT을 이용하여 제작된 칩의 테스트 결과 원하는 성능을 만족하였다.

Abstract In this paper, we propose modified algorithm of motion vector detection for the pixel of image in the optical sensor. It is designed to reduce the amount of operation and have more accuracy in the motion detection than previous block matching algorithm. The proposed algorithm is implemented with Cyclone and fabricated using SEC 0.35um CMOS 1-poly-4-metal technology. The result of test with CARTESIAN ROBOT meets the desired performance.

Key Words : 움직임 벡터 검출 알고리즘, 광 센서, 블록 매칭 알고리즘

Ⅰ. 서 론

현대 사회에서는 광 마우스, 디지털 카메라, 지문인식 스캐너 등 일상생활의 많은 부분에서 광센서용 매체들을 사용하고 있으며, 기술이 발전함에 따라 더 높은 정밀도와 성능을 요구하고 있다. 정밀도를 좀 더 정확하게 구현하기 위해서는 움직임 벡터 검출의 성능을 최대화 할 필요성이 있다. 일반적으로 광센서는 붉은 빛을 통해 광학 센서 위로 반사되고, 센서의 신호는 다시 각 영상을 디지털 신호 처리 장치로 보낸 후 각 영상의 패턴을 감지하고 이전 영상과 비교해 움직임 벡터를 검출한다. 광센서 동작원리는 센서 어레이가 표면으로부터 반사되는 빛의 빔을 읽는다. 센서 어레이는 작은 디지털 그레이 스케일 카메라와 같은 기능으로, 16x16 픽셀에서 최대 30x30 픽셀로 구성된다. 초당 6000회 이상 연속적인 프레임을 사용하여 정확한 위치와 방향과 속도를 구하게 된다. 광센서를 이용한 움직임 벡터 검출 알고리즘은 광센서에서 받아들이는 표면의 질감과 명암, 채도에 따라 움직임 벡터 검출에 많은 차이가 난다. 표면의 영향을 덜 받도록 구현하고자 개발되고 있다.[1]

움직임 벡터를 검출하는 방법으로는 크게 화소 순환 알고리즘과 블록 매칭 알고리즘이 있다. 화소 순환 알고리즘은 프레임 내 모든 화소에 대하여 움직임을 추정하는 방식으로 움직임 추정이 완료된 화소의 움직임 벡터를 인접화소의 움직임 추정을 위해 사용하게 된다. 화소 순환 알고리즘은 정확한 움직임 벡터 검출이 가능하지만 과다한 연산량이 소요되므로 실제 사용되는 경우는 별로 없다. 블록 매칭 알고리즘은 움직임을 추정할 프레임을

*정회원, 호서대학교
**정회원, 세명대학교 정보통신학부
***정회원, @lab(주) 로직설계팀
접수일자 2010.8.26 수정일자 2010.9.30
게재확정일자 2010.10.15

임의의 작은 블록으로 나누고, 각 블록 내의 모든 화소는 동일한 움직임을 갖는다고 가정하여 블록 당 한 개의 움직임 벡터를 부여하는 방식이다. 움직임에 의한 변형이 크지 않아서 단지 물체의 이동으로 모든 움직임을 근사화 할 수 있다는 전제하에 움직임 추정을 행하게 된다. 따라서 움직임 벡터는 각 블록 당 1개의 (x,y) 좌표로서 표현된다.[1][2]

본 논문은 블록 매칭 알고리즘을 이용하여 광센서에서 보다 정확한 움직임 벡터를 찾는 알고리즘 개발과 구현을 연구한다. 정확한 움직임 벡터를 검출하기 위해서 광센서 어레이의 디지털 값의 비트 수를 늘리고, 움직임 벡터를 찾는 방법을 기존 보다 세분화시킨 알고리즘을 개발하고, 삼성 0.35um 1-poly 4-metal CMOS Standard Cell Library 공정을 통해 설계하여 구현 후 테스트를 통하여 검증하였다.

II. 기존 움직임 벡터 검출 알고리즘

1. 기존 움직임 벡터 검출

기존 움직임 벡터 검출은 18x18 이미지의 4비트 픽셀 데이터를 입력을 받아서 17x17 이미지의 1비트의 픽셀 데이터로 변환을 시킨다. 17x17 이미지로 변환된 1비트 값을 가지고 17x17 현재 프레임과 기준 프레임을 만든다. 현재 프레임과 기준 프레임을 11x11 현재 윈도우와 기존 윈도우를 만들어서 지그재그 탐색방법으로 49개의 상관관계의 합을 구한다. 49개의 상관관계 합 중에서 가장 작은 상관관계가 있는 픽셀의 좌표를 출력으로 내보낸다.

가. 기존 움직임 벡터 검출 블록도

움직임 벡터 검출은 크게 2가지로 나눈다. 18x18의 4비트 픽셀 이미지를 17x17의 1비트 픽셀로 해주는 pre-filter 블록과 움직임 벡터 검출하는 블록으로 이루어진다. 그림 1은 기존 움직임 벡터 검출의 전체 블록도를 나타내고 있다.[2][3]

나. 움직임 벡터 검출의 Pre-Filter

Pre-filter는 18x18 이미지 센서의 4비트의 ADC 데이터를 순차적으로 1 픽셀씩 받아들인다. [x,y]로 순서를 정하여 나타내면, [0,0], [0,1] ...[1,0], [1,1].....[17,17]의 순서로 ADC 데이터를 받아들인다.

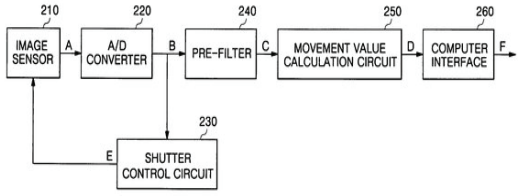

그림 1. 기존 움직임 벡터 검출 블록도
Fig. 1. Block Diagram of Original Motion Vector Detection

4비트 데이터를 순차적으로 받아들이면서 1비트로 줄이기 때문에 게이트의 수를 줄일 수 있다는 장점이 있다. 1픽셀 당 4비트씩 데이터를 저장하고 있는 18x18 이미지를 1픽셀 당 1비트씩 데이터를 저장하고 있는 17x17 이미지로 변화 저장시킨다.

(1) Pre-Filter 동작

Pre-filter는 18x18의 [x,y]로 표시한 ADC 데이터 배열을 17x17의 [x,y]로 바꾸어 변화한다. 아래의 식(1)은 17x17 이미지로 변화하는 방법이다.

```
for ( X = 1 : X=<17 :++)
 for ( Y=1: Y=< 17 : ++)
  Case1.  (X,Y) 에서 X 좌표가 1 인 경우는
    (X,Y) = if ( [X+1, Y+1] < [X,Y] )
  Case2.  case1 의 경우를 제외한 모든 경우는
    (X,Y) = if ( [X+2, Y+1] < [X,Y]) .....식 (1)
```

Case1은 위쪽 방향으로 1, 왼쪽 방향으로 1만큼 비교 연산한 식을 나타낸다. (1,1) 좌표를 가질 때 (2,2) 좌표와 비교를 해서 (2,2) 좌표 값이 작으면 (1,1)의 좌표 값에 '1'의 값이 할당된다. (2,2) 좌표 값이 크면 '0'의 값이 할당된다. Case2는 위쪽 방향으로 1, 왼쪽 방향으로 2만큼 비교 연산한 식을 나타낸다. (2,1) 좌표를 가질 때 (4,2) 좌표와 비교를 해서 (4,2) 좌표 값이 작으면 (2,1) 좌표 값에 '1'의 값이 할당되고, (4,2) 좌표 값이 크면 '0'의 값이 할당된다.

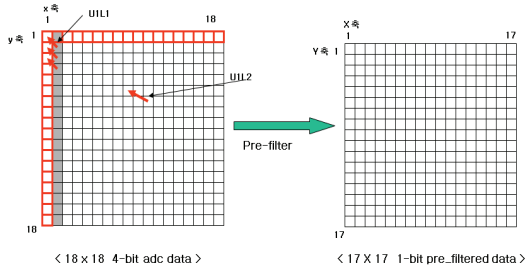

그림 2. Pre-Filter 동작 과정
Fig. 2. Pre-Filter Action Process

그림 2는 18x18 4비트 ADC 데이터에서 보여 지는 바와 같이 한 픽셀의 4비트 ADC 데이터를 위치상으로 위쪽 방향으로 1, 왼쪽 방향으로 2 만큼 위치한 픽셀과 비교하여 Pre-filter를 수행한다. 위쪽으로 방향으로 1, 왼쪽 방향으로 1 만큼 위치한 픽셀과 비교한 값들은 다른 위치의 픽셀들과 계산된 값으로 잘못된 입력 값이 된다.

다. 움직임 벡터 검출 과정

움직임 벡터 검출 과정은 두 가지로 방법으로 나뉜다. 첫 번째 방법은 기준 프레임과 현재 프레임을 사용하여 지그재그 탐색 방법으로 움직임 벡터를 검출한다. 두 번째 방법으로는 두 개의 윈도우를 사용하여 지그재그 탐색 방법으로 움직임 벡터를 검출한다. 그림 3은 기준 프레임과 현재 프레임을 가지고 지그재그로 탐색한 방법을 나타내고, 그림 4는 2개의 윈도우를 사용하여 지그재그 탐색을 하는 방법을 나타낸다.

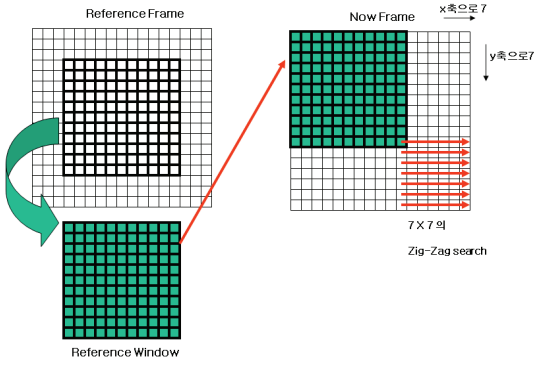

그림 3 기준 프레임과 현재 프레임의 탐색 과정
Fig. 3. Detection Process of Reference Frame and Now Frame

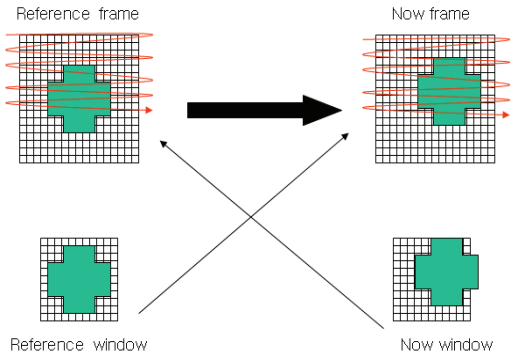

그림 4. 2개의 윈도우를 사용한 탐색 과정
Fig. 4. Detection Process using Two Windows

기준 프레임과 현재 프레임을 사용하여 지그재그 탐색 방법을 사용할 경우는 기준 17x17프레임의 중앙인 11x11을 기준 윈도우로 정한 후 각각을 XOR 연산을 수행하여 121개의 출력 값을 구한 후 모두 더하여 상관관계 값을 구한다. X축으로 7, Y축으로 7을 움직여 49개의 상관관계를 구해 가장 작은 값을 저장하고, 그 때의 위치를 찾게 된다. 그림 4는 2개의 윈도우를 이용하여 앞선 방법과 마찬가지로 상관관계 값을 구한 후 합산하여 새로운 상관관계 값을 만든 후, 49개의 값 중에서 가장 작은 값과 위치를 찾게 된다.

Ⅲ. 제안된 움직임 벡터 검출 알고리즘

1. 제안한 움직임 벡터 검출 알고리즘 블록도

기존 알고리즘은 상호관계_1과 상호관계_2의 좌표를 각각 구해서 (x_1,y_1)/2, (x_2,y_2)/2한 좌표 값을 더한다. 2로 나눈 값의 좌표의 소수점은 소수점끼리 더해서 MSB가 1이 되는 시점에 2개의 좌표 더한 값에 1을 더해서 최종 움직임 벡터를 출력하고, 상호관계_1과 상호관계_2를 비교를 해서 더 작은 상호관계 값의 좌표를 출력한다. 다른 방법은 상호관계_1과 상호관계_2를 비교해서 같으면 상호관계_1의 좌표를 출력하고 같지 않으면, 이전의 좌표 값을 출력한다. 기존 알고리즘들보다 테스트 결과 더 정확한 움직임 벡터를 검출하는 알고리즘을 본 논문에서 제안한다.

그림 5는 제안한 움직임 벡터 검출의 전체 블록도를 나타내고 있다.

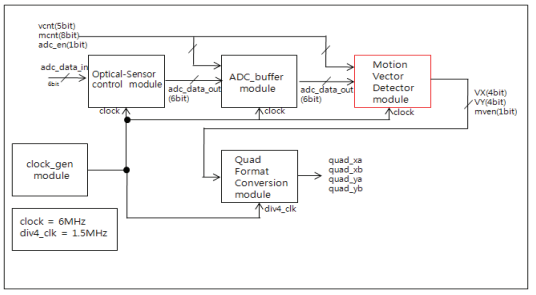

그림 5. 제안된 움직임 벡터 검출 블록도
Fig. 5. Block Diagram of proposed Motion Vector Detection

Optical-Sensor control에서는 광센서에서 들어오는 아날로그 데이터를 디지털 데이터 6비트로 변환 후 출력한다. ADC_buffer에서는 Optical-Sensor control에서 출력되는 디지털 데이터를 필요한 타이밍에 맞게 출력한다. Motion Vector Detector에서는 18x18 이미지를 17x17 이미지로 변경한 후 움직임 벡터 검출을 한다. Quad Format Conversion에서는 Motion Vector Detector에서 출력된 벡터를 최종 quad_xa, quad_xb, quad_ya, quad_yb를 통해 출력한다.

가. Optical-Sensor Control and ADC-buffer Module

광센서의 센서 인터페이스를 제어하고, 센서에서 출력이 되는 아날로그의 신호들을 제어하여, 18x18 이미지에 적합하게 디지털 데이터를 출력한다. ADC-buffer 모듈에서는 Optical-Sensor control module에서 출력되는 디지털 데이터 6비트를 입력으로 받아 타이밍에 맞춰 동기화시켜 데이터를 출력한다.

나. Motion Vector Detector Module

Motion Vector Detector 모듈은 그림 6과 같이 구현하였다. Image Conversion에서는 ADC의 디지털 6비트를 입력을 받아서 17x17 이미지를 만든다. Image Register에서는 17x17 이미지를 현재 프레임 과 어떤 시점에서 현재 프레임을 저장하는 기준 프레임을 만들고, 지그재그 탐색을 하기 위하여 11x11의 기준 윈도우와 현재 윈도우를 만든다. Correlation Detection에서는 상관관계 값을 구하고, Vector Detection에서는 상관관계값을 가지고 움직임 벡터를 찾아 최종 좌표를 출력한다.

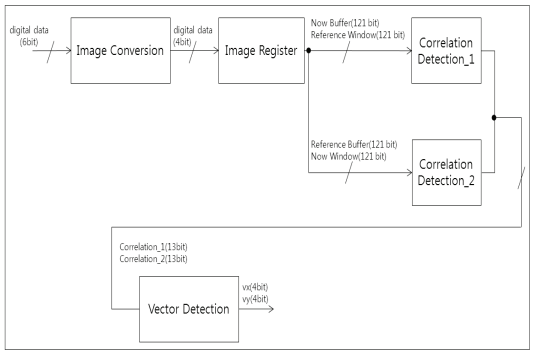

그림 6. Motion Vector Detector 하위 모듈
Fig. 6. Motion Vector Detector Sub-Module

(1) Image Conversion Module

18x18 이미지의 디지털 6비트 값을 MSB의 상위 4비트를 이용하여, 17x17 이미지의 디지털 4비트 값으로 출력한다. 기존 연구에서는 4비트를 1비트로 변환을 하였지만, 본 연구에서는 6비트를 입력을 받아 4비트로 변환을 하였기 때문에 광센서에서 출력이 되는 디지털 데이터를 더 정확하게 표현 할 수 있다. 18x18 6비트 이미지에서 17x17 4비트 이미지로 변환을 할 때는, 6비트 이미지가 입력으로 들어올 때 18x18 이미지 시작을 알려주는 신호인 edge_pipe 신호를 늦춰 불필요한 맨 위 픽셀 과 맨 왼쪽의 픽셀들을 제거하는 Pre-Filter를 구현하였다.

(2) Image Register Module

Image conversion 모듈에서 Pre-filter되어 출력된 4비트를 입력을 받는다. edge_pipe 신호에 동기 되어 '1'일 때 pre-filter된 4비트 데이터는 17x17 현재 이미지의 맨 하위 (17,17)부터 입력이 된다. (17,17)부터 입력이 된 4비트 데이터는 시프트 되면서 현재 이미지의 맨 상단 (1,1)까지 채워진다. (1,1)까지 데이터가 채워지면 하나의 현재 이미지가 완성 된다. 일정한 값 이상이면 현재 이미지는 기준 이미지로 업데이트 된다. 처음에 현재 이미지가 기준 이미지로 업데이트 된 후 일정한 값 이상이 된 시점에 다시 업데이트가 되지만 일정한 값 이하이거나 움직임이 없을 경우에는 업데이트 되지 않고 계속 기준 이미지를 사용한다. 2개의 윈도우를 가지고 지그재그 탐색을 하기 위해 17x17 이미지 현재 이미지의 중앙을 11x11 이미지인 현재 버퍼로 저장한다. 17x17 이미지 기준 이미지의 중앙을 11x11 이미지인 기준 버퍼로 저장한다. 현재 이미지가 기준 이미지로 업데이트되는 시점에 11x11 이

미지인 현재 윈도우가 기준 윈도우로 저장이 된다. 기준 이미지로 현재 이미지가 현재 버퍼로 저장이 되는 시점에 현재 버퍼 데이터는 11x11 이미지 현재 윈도우로 저장이 된다. 지그재그 탐색을 하기 위해서 11x11 이미지(121-비트)가 현재 버퍼, 현재 윈도우, 기준 버퍼, 기준 윈도우가 출력이 된다.

(3) Correlation Detection Module

Correlation Detection 모듈에서는 상관관계_1과 상관관계_2 모듈로 구분하여 연산한다.

상관관계_1 모듈에서는 현재 버퍼와 기준 윈도우를 입력을 받는다. 본 논문에서는 4비트 데이터를 갖는 현재 버퍼와 기준 윈도우를 1 픽셀 당 일대일 대응을 하여 현재 버퍼에서 기준 윈도우 값을 뺀다. 뺀 값을 4비트 데이터 용량을 가진 1 픽셀에 저장을 한다. 11x11 이미지에서 맨 위 열부터 맨 아래 열 까지 순차적으로 픽셀 별로 값을 더한다. 121개의 픽셀을 더한 값을 49개의 상관관계_1로 출력을 한다. 4비트 데이터를 연산하기 때문에 1비트 데이터를 연산하는 것 보다 더 정확한 상관관계를 구한다.

상관관계_2 모듈에서는 기준 버퍼와 현재 윈도우를 입력을 받는다. 상관관계_1 모듈의 연산과 같은 방법을 사용하여 상관관계_2로 출력한다.

그림 7은 상관관계_1과 상관관계_2의 연산과정을 나타낸다.

(4) Vector Detection Module

Vector Detection module은 49개의 상관관계_1과 상관관계_2를 더해서 49개의 상관관계 합을 구한다. 49개의 상관관계 합중에 가장 작은 값을 가졌을 때의 좌표를 구한다.

각각의 좌표 (x_1,y_1), (x_2,y_2)를 비교했을 때 같으면 최종 출력은 (x_1, y_1)이 출력이 된다. 그러나 각각의 좌표가 같지 않을 경우 (x_1, y_1)좌표가 (2,2)를 기준으로 주변의 (x_2, y_2)좌표가 (1,1) 또는 (2,1) 또는 (3,1) 또는 (1,2) 또는 (3,2) 또는 (1,3) 또는 (2,3) 또는 (3,3)중에 하나의 좌표가 최종 출력이 되고, 주변에 위치하지 않는 상관관계_2의 좌표가 올 경우에는 기존의 좌표가 최종 출력이 된다.

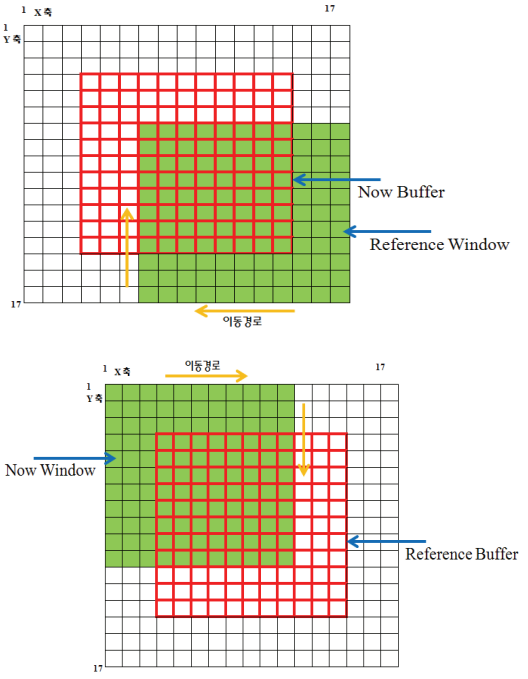

그림 7. 상관관계_1과 상관관계_2의 연산 과정
Fig. 7. Operation Process of Correlation_1 anc Correlation_2

다. Quad Format Conversion Module

Motion Vector Detector 모듈에서 최종 출력되는 움직임 벡터 좌표 4비트 데이터 vx, vy를 입력으로 받는다. Timing generation에서 vx, vy를 최종 출력에 타이밍을 맞추기 위하여 카운터를 만든다. vx, vy의 MSB가 '1', '0'일 때의 조건을 나누어 양수 좌표, 음수 좌표 일 때를 구분하여 최종 quad_xa, quad_xb, quad_ya, quad_yb에 출력을 내보낸다.

Ⅳ. 시뮬레이션 검증 및 테스트 결과

RTL(Register Transfer Level) 단계에서 하드웨어 설계 언어인 Verilog HDL을 사용하여 설계된 움직임 벡터 검출의 디지털 블록을 구현하고, 각 블록에 대한 시뮬레이션을 Cadence의 Verilog 논리 시뮬레이터를 이용하여 수행하였다.

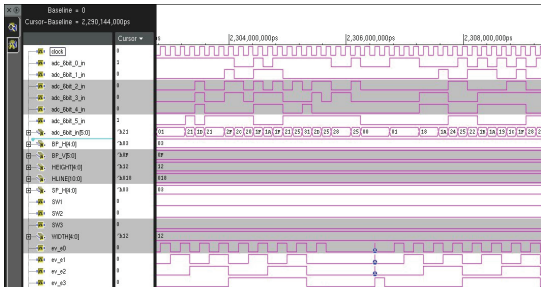

그림 8. 시뮬레이션 결과
Fig. 8. Simulation Result

또한, Synopsys의 Design Compiler를 사용하여 합성 및 검증을 한 후 칩으로 제작하였다.

그림 8은 시뮬레이션의 일부 결과이며, 그림 9는 제작된 칩 사진이다. 표 1은 제작된 칩에 대한 사양이다.

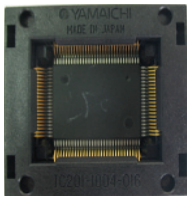

그림 9. 칩 사진
Fig. 9. Chip Image

표 1. 칩 사양
Table 1. Chip Specification

공정	삼성 0.35um 2-poly 4-metal 공정
칩 면적(Core)	3mm x 3mm
게이트 수	52,824
전원 전압	3.3 V
패키지	MQFP 144핀
클럭 주파수	6MHz

설계된 전체 움직임 벡터 검출 알고리즘 블록을 FPGA 보드인 Cyclone(Altera EP1C120240C8 칩 탑재)에 다운로드하고, 광센서가 부착된 마우스를 부착하여 테스트 하였다. 전체 module 테스트 후 시뮬레이션은 CARTESIAN ROBOT을 사용하여 검증 하였다.

그림 10은 CARTESIAN ROBOT을 이용하여 기존 구조와 제안하는 구조의 움직임 벡터 검출 시뮬레이션결과이다. 원, 리본, 빗살 모양의 좌표를 입력하여 테스트를 하였다. 원의 움직임을 비교하면 기존 구조보다 작은 원을 보았을 때 제안한 구조가 원의 간격 오차가 작은 것을 볼 수 있다. 리본의 움직임을 비교하면 기존 구조는 좌상우하의 움직임이 매끄럽지 못한 것을 볼 수 있는데 제안한 구조는 좌상우하의 움직임이 깨끗한 것을 볼 수 있다. 빗살 움직임을 비교하면 기존 구조는 맨 우측 직선이 겹치는 것을 볼 수 있는데 제안한 구조는 겹치는 직선이 없이 시뮬레이션 그림과 동일한 것을 보여준다. 세 가지 실험에서 제안된 구조는 기존 구조보다 정확한 움직임 벡터를 검출함을 확인할 수 있다.

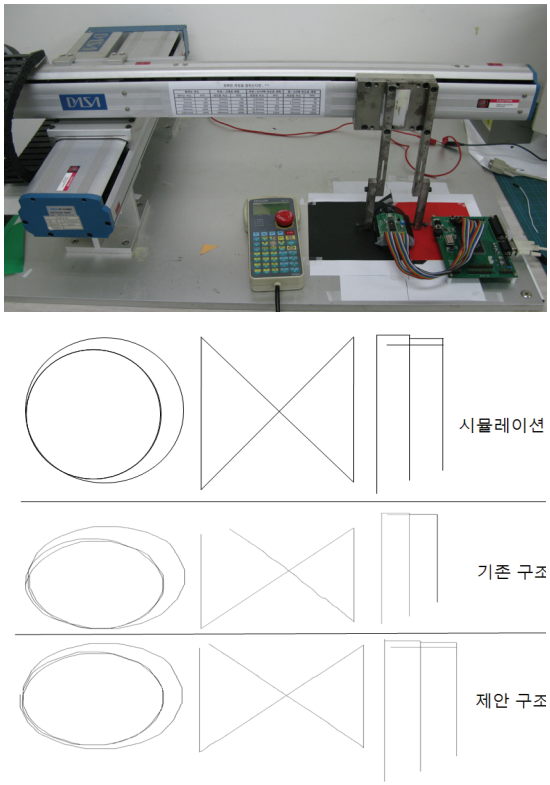

그림 10. 기존구조와 제안된 구조의 움직임 벡터 검출 시뮬레이션
Fig. 10. Motion Vector Detection Simulation of Original Structure and Proposed Structure

V. 결 론

본 논문에서 제안한 움직임 벡터 검출 알고리즘은 Motion Vector Detector 모듈에서 사용되는 비트 수를 1비트에서 4비트로 확장하였기 때문에 더 정확한 픽셀에

대한 데이터를 가지고 상관관계와 움직임 벡터를 검출하여 정밀한 움직임을 가지는 것을 검증하였다. 상호관계를 가지고 움직임 벡터 검출 하는 방법의 경우의 수를 증가시켜 기존 연구 보다 더 정밀한 움직임 벡터를 검출하는 내용을 기술하였다. 삼성 0.35um CMOS 1-poly-4-metal 공정을 이용하여 칩을 제작하였으며, 테스트 결과 원하는 클럭 주파수인 6MHz에서 정상 동작함을 확인할 수 있었다.

향후, 게이트 수 감소 및 칩 면적을 고려하여 최적의 저 전력 설계를 하고자한다.

참 고 문 헌

[1] 박상봉, IDEC Newsletter, "입력 디바이스 광학 센서 기술", pp 6-7, 2005년 5월

[2] Publication Number: 04799055 United States Patent and Trademark Office, "Optical Mouse", August 26, 2008

[3] Publication Number: 07327351 United States Patent and Trademark Office, "Optical pointing system, signal transmission method of the same, and signal processing method of computing apparatus used in the same", February 5, 2008

[4] Publication Number: 06697053 United States Patent and Trademark Office, "Image sensor mouse", February 24, 2004

[5] R.W. Corrigan, et al. "Calibration of a Scanned Line ar Grating Light Valve Projection System" May 18, 1999, pp. 1-4

※ 본 연구는 2007년도 호서대학교의 재원으로 학술연구비를 지원받아 수행된 연구임
(과제번호:20070106)

저자 소개

박 노 경(정회원)

- 1990년 2월 : 고려대 전자공학과 공학박사
- 1985년 3월 ~1989년 2월 : 삼성반도체 반도체설계연구소 연구원
- 1998년 ~ 2000년 : Oregon State Uni 교환교수
- 1985년 3월 ~ 현재 : 호서대학교 정보통신공학과 교수
- 2002~현재 R .I. C 중소기업 산학협력 센터장
- 2005~현재 차세대 반도체 설계 연구소장
- 2006~현재 TIC 부소장

<주관심분야 : SOC 설계, 임베디드 시스템 설계, IPTV & WIRELESS BROADBAND INTERNET>

박 민 형(정회원)

- 2009년 2월 : 세명대 전산정보 이학석사
- 2009년 3월 ~현재 : @lab(주) 로직설계팀 주임

<주관심분야 : ASIC 설계, 터치 센서, 이동통신 무선>

박 상 봉(정회원)

- 1992년 2월 : 고려대 전자공학과 공학박사
- 1992년 3월 ~1992년 2월 : 삼성전자 선임 연구원
- 1999년 3월 ~ 현재 : 세명대학교 정보통신학부 부교수
- 2000년 7월 ~ 현재 : @lab(주) Digital 설계팀 기술고문

<주관심분야 : RFID/USN 기술, ASIC 설계, 광센서, 멀티터치>