화소 평균 및 에지 정보를 이용한 움직임 검출기 구현

이승준[0]* 최철호* 권병헌** 최명렬*
*한양대학교 전자전기제어계측공학과
**유한대학 정보통신과
{lee30806[0], hbw, choimy}@asic.hanyang.ac.kr*
bhkwon@yuhan.ac.kr**

Implementation of Motion Detector using Edge Information & Average Pixel Difference

Seung-Joon Lee[0]* Chul-Ho Choi* Byung-Heon Kwon** Myung-Ryul Choi*
*Dept. of EECI, Hanyang University
**Dept. of Information and Telecommunication, Yuhan College

요 약

본 논문에서는 카메라 이동 및 피사체의 움직임 특성을 이용한 움직임 검출 방식을 제안한다. 카메라 이동 및 피사체의 움직임 특성을 파악하여 움직임을 검출하기 위해 이전 프레임과 현재 프레임 간의 화소 차이의 평균 및 화면 내의 물체의 에지 정보를 이용하여 현재 프레임의 움직임 정도를 판단한다. 그리고 움직임 검출의 정확도를 높이기 위해 화소 차이의 평균을 3단계로 나누어 연산한다.

제안된 움직임 검출 방식은 기존의 움직임 검출 방식에서 나타난 문제점을 보완하며 움직임 검출 범위를 높일 수 있음을 컴퓨터 시뮬레이션을 통해 확인할 수 있었으며, 아남 0.25μm 공정 라이브러리와 Synopsys 툴을 이용해 VHDL로 구현하였다.

1. 서 론

최근 컴퓨터 모니터와 TV 등 디스플레이 장치들의 화면이 커짐에 따라 그 크기 또한 대형화되고 있다. 이로 인해 LCD 모니터 및 PDP 등 적은 공간을 차지하고 저전력 구동이 가능한 디스플레이 장치들의 사용이 늘어나고 있는 실정이다. 이러한 평판형 디스플레이 장치들은 화면 확대를 위해 FIR 필터 혹은 가우시안 필터 등을 이용하여 일률적으로 화면을 확대하였다[1]. 하지만 입력 영상의 특성에 따라 화면 확대하여야만 이미지 열화 현상 등을 해결할 수 있다[2].

본 논문에서는 이러한 입력 영상의 특성에 있어 중요한 화면상의 움직임을 검출하는 알고리즘을 제안하였다. 이는 기존 알고리즘에서 나타난 비트 에러와 협소한 검출 범위로 인한 움직임 검출 오류 등을 해결함으로써 보다 정확한 움직임 검출 결과를 보여준다.

또한 제안된 움직임 검출기는 하드웨어 구조가 단순하여 실시간 처리가 가능하다.

2. 기존 움직임 검출 알고리즘

기존 움직임 검출 알고리즘은 (N-1)번째 프레임과 (N)번째 프레임, 그리고 (N+1)번째 프레임 사이의 움직임 검출 대상 화소 주위의 3×3 윈도우 내에 있는 화소들의 차이와 (N)번째 프레임의 검출 대상 화소의 에지 검출값을 이용하여 움직임을 판단한다[3]. 그림 1은 기존 움직임 검출 알고리즘의 흐름도를 보여준다. 그림 1에서와 같이 기존 움직임 검출 알고리즘은 각 프레임들 간의 화소들의 차이값과 (N)번째 프레임의 에지값의 논리 AND 연산에 의해 화면 내의 물체의 움직임을 검출하고 있으며 이 때 임계치들(CV_1, CV_2, CV_3)은 실험적으로 4와 2, 그리고 4로 정하였다.

그리고 식 (1)과 (2)는 각각의 화소 차이값과 에지 검출값, 그리고 움직임 판단 방법을 보여준다. 여기서 PD는 Pixel Difference이고 MV는 Motion Value, EV는 Edge Value이다.

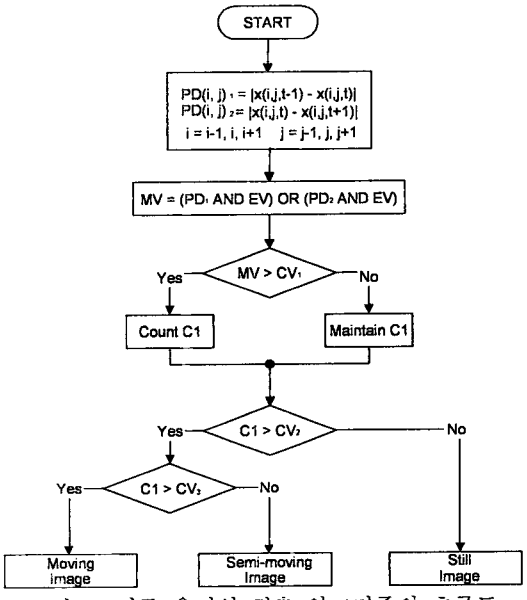

그림 1. 기존 움직임 검출 알고리즘의 흐름도

$$PD(i,j)_1 = |x(i,j,t-1) - x(i,j,t)|$$
$$PD(i,j)_2 = |x(i,j,t) - x(i,j,t+1)|$$
$$EV = \max[|x(i-1,j-1,t) - x(i,j,t)|] \quad (1)$$
$$MV = (PD_1 \ AND \ EV) \ OR \ (PD_2 \ AND \ EV)$$

if $MV > CV_1$ then $C1 = C1 + 1$
else $C1 = C1$

if $C1 > CV_2$ and $C1 > CV_3$
then Moving
else if $C1 > CV_2$ and $C1 \leq CV_3$ (2)
then Semi-moving
else
then Still

3. 제안한 움직임 검출 알고리즘

제안된 움직임 검출 알고리즘은 그림 2와 같이 (N-1)번째 프레임과 (N)번째 프레임에서 3×3 윈도우 내에 있는 화소들의 차이의 평균값을 계산하여 그 값이 10 미만일 경우에는 정지화 화소, 128 미만일 경우에는 준동화 화소, 128 이상일 경우에는 동화 화소로 판단하여 각각 0, 127, 255로 그 값을 대치한다. 이 때 대치된 값을 움직임 값 (Motion Value, MV)이라 한다.

판단된 움직임 값은 (N)번째 프레임의 에지 검출값과 논리 AND 연산을 수행하여 입력 영상 내의 물체의 움직임을 판단하게 된다. 이 때 AND 연산 결과값을 움직임 검출값 (Motion Detection Value, MDV)라 한다.

산출된 움직임 검출값에 대해 3×3 윈도우에서 움직임 검출값이 0 이상인 값이 2개 이하이면 윈도우의 중심 화소는 정지화 화소로, 3개 이상 5개 이하이면 준동화 화소로, 6개 이상이면 동화 화소로 판단된다.

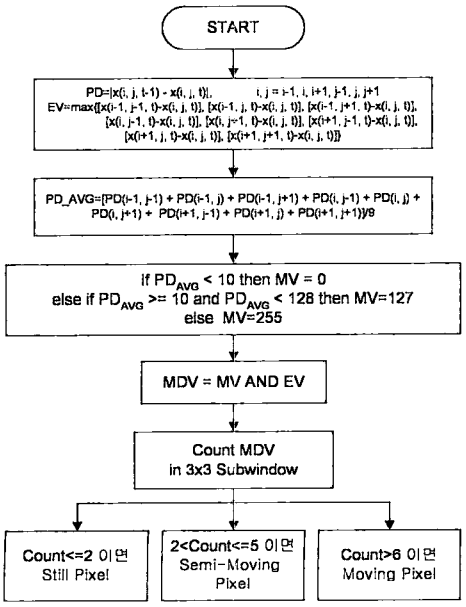

그림 2. 제안한 움직임 검출 알고리즘의 흐름도

식 (3)과 (4)는 화소에 대한 움직임 검출 과정을 나타낸다.

$$PD(i,j) = |x(i,j,t-1) - x(i,j,t)|$$
$$i = i-1, i, i+1 \quad j = j-1, j, j+1$$
$$EV = \max[|x(h,k,t), x(i,j,t)|]$$
$$h = i-1, i, i+1 \quad k = j-1, j, j+1 \quad (3)$$
$$PD_{AVG} = [PD(i-1,j-1) + PD(i-1,j) + PD(i-1,j+1) + PD(i,j-1) + PD(i,j) + PD(i,j+1) + PD(i+1,j-1) + PD(i+1,j) + PD(i+1,j+1)]/9$$

if $PD_{AVG} < 10$
then $MV = 0$
else if $PD_{AVG} \geq 10$ and $PD_{AVG} < 128$
then $MV = 127$ (4)
else
then $MV = 255$

$$MDV = MV \ AND \ EV$$

4. 하드웨어 구조

제안된 움직임 검출기의 하드웨어 구조는 그림 3과 같다. 제안된 움직임 검출기는 1개의 프레임 메모리와 화소 차이값을 계산하는 2개의 PD Block과 (N)번째 프레임의 에지 검출을 위한 ED Block, 화소 차이값들의 평균을 구하는 Average Block, 움직임값을 구하는 MV Block, 움직임 검출값을 계산하는 MDV Block, 그리고 마지막으로 화소의 움직임을 판단하는 Motion Determination Block으로 구성된다.

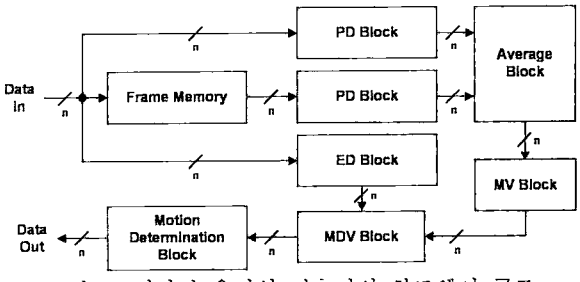

그림 3. 제안된 움직임 검출기의 하드웨어 구조

5. 시뮬레이션 결과

제안된 움직임 검출 알고리즘의 시뮬레이션은 준동영상 Salesman 30프레임과 Table tennis 30프레임, 동영상 Football 25프레임, 그리고 카메라가 이동하는 Flower Garden 30프레임으로 컴퓨터 시뮬레이션을 수행하였다. 움직임이 검출된 화소는 검은색으로 표시하였다.

그림 4는 각 이미지의 원본 이미지이며 그림 5는 기존 움직임 검출 알고리즘에 의해 처리된 이미지이다. 그림 6은 제안된 움직임 검출 알고리즘에 의해 처리된 이미지로 검출 정확도와 범위에 있어서 기존 알고리즘에 비해 향상된 것을 알 수 있다.


(a) Flower Garden 1st


(b) Flower Garden 2th


(c) Football 1st


(d) Football 2th


(e) Salesman 1st


(f) Salesman 2th


(g) Table Tennis 1st


(h) Table Tennis 2th

그림 4. 원본 이미지


(a) Flower Garden


(b) Football


(c) Salesman

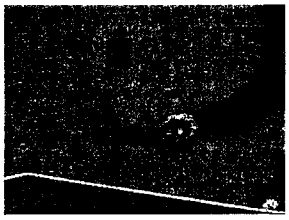
(d) Table Tennis

그림 5. 기존 움직임 검출 알고리즘에 의한 이미지


(a) Flower Garden


(b) Football


(c) Salesman


(d) Table Tennis

그림 6. 제안된 알고리즘에 의해 처리된 이미지

6. 합성 결과

제안된 움직임 검출기는 아남 0.25μm 공정 라이브러리와 Synopsys 툴을 사용하여 VHDL 구현하였다. 구현된 결과 2323개 게이트가 사용되었다. 그림 7은 합성 결과를 보여준다.

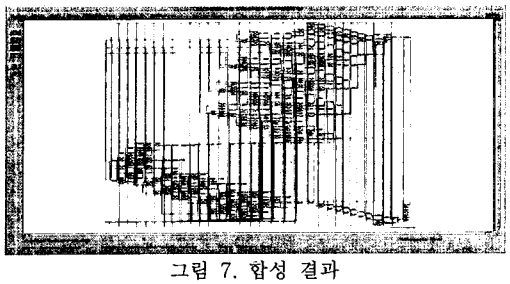

그림 7. 합성 결과

7. 결론

제안된 움직임 검출기는 기존 움직임 검출기에 비해 보다 정확한 움직임 검출이 가능하였으며, 검출 범위 역시 넓어짐을 알 수 있었다. 하지만 준동영상 이미지인 Salesman의 경우 각 프레임마다 검출 결과의 차이가 많이 났다. 이는 제안된 움직임 검출기는 아주 작은 움직임에 대해서는 기존 움직임 검출기에 비해 낮은 정확도를 가지는 것을 뜻한다. 하지만 정확도가 떨어지는 문제는 움직임값을 결정하는 임계치의 수정을 통하여 보완될 수 있다. 그리고 기존 움직임 검출기에 비해 적은 게이트 수를 가지고 있어서 실시간 처리에 더욱더 용이함을 알 수 있었다.

참고 문헌

[1] A. Murat Tekalp "Digital Video Processing", Prentice Hall, pp. 314-330, 1995

[2] Rafael C. Gonzalez, Richard E. Woods "Digital Image Processing", Edison Wesley, pp. 416-476, 1993

[3] S. J. Lee, et al "Design of a motion detector using edge information for a motion adaptive image scaler", IMID'01, Daegu, Korea, pp. 878-881, 2001