움직임 검출 시스템에서 효율적인 카메라 패닝 보상 기법

김남효* · 김선우* · 김태훈* · 최연성*
*군산대학교 공과대학 전자정보공학부

Efficient Camera Panning Compensation Technique in Motion Detection System

Nam-hyo Kim* · Sun-woo Kim* · Tae-hun Kim* · Yeon-sung Choi*
Faculty of Electronic & Information Enginnering, *Kunsan National Univ
E-mail : namo@kunsan.ac.kr

요 약

동영상에서의 움직임 검출에 대한 기술은 다양한 응용분야를 가지고 있다. 특히 고정된 카메라에서 움직임 검출과 움직이는 카메라에 대응하여 움직임 검출하고 추적하는 기법들은 현재까지 많은 연구들이 진행되고 있고, 또 응용 제품들의 출시되어지고 있다. 그렇지만 일반적이고 보다 완전한 제품의 개발은 아직 한계가 있다. 본 논문에서는 움직임 벡터를 이용하여 카메라가 패닝(Panning) 여부를 판단하여 패닝 시에는 움직임 검출을 하지 않는 기법을 제안한다. 이 기법은 카메라 패닝 시 즉, 카메라 팬/틸트 드라이버의 고장이나 바람등의 상황에는 움직임 검출을 하지 않음으로써 보다 효과적인 움직임 검출을 할 수 있다. 또한 두 개의 임계치를 적응적으로 조절할 수 있어 기존의 고정된 카메라에서 사용하는 움직임 검출 기법들보다 효율적이라고 할 수 있다.

키워드
EPZS, 움직임벡터, Threshold, 카메라패닝

Ⅰ. 서 론

최근 들어 동영상으로부터 움직임 검출과 움직이는 물체에 대한 추적 등에 대한 중요성이 인식됨에 따라 많은 연구가 진행되어지고 있다.

현재까지 대부분의 연구나 응용 프로그램들이 실내에 설치되는 고정된 카메라를 중심으로 진행되어지고 개발되어졌다. 고정된 카메라에서 움직이는 물체를 검출해 내는 알고리즘은 카메라의 움직임(Pan/Tilt)을 고려한 시스템 보다는 비교적 쉽게 구현이 된다. 고정된 카메라를 이용하여 움직임을 검출하는 기법은 대부분 카메라로부터 입력되는 영상을 이용하여 배경 이미지를 제작하고, 이 배경 이미지와 현재 입력 영상과의 차를 이용하여 현재 영상에서의 움직임을 검출하는 방법과 입력되는 영상에서 특정 부분을 감시하여 이전 프레임과의 색상의 차이나 또는 휘도의 차이를 가지고 움직임을 검출하는 방법들이 있다.

카메라가 움직이지 않는다는 가정 하에서는 기존의 기법들은 특별하게 문제가 없는 움직임 검출 기법이라고 말할 수 있다. 그렇지만 실외의 경우를 놓고 판단하였을 때, 아주 많은 조건들이 카메라를 움직이게 한다. 이러한 조건들을 고려하였을 때 기존의 움직임 검출 기법들은 이러한 환경에는 적응을 못하는 단점을 가지고 있다.

그러다 보니 영상에서 카메라가 고정되어 있을 때 움직임 검출뿐만 아니라 카메라가 움직일 경우 움직임 검출과 같은 기법들에 대한 연구가 많이 이루어지고 있다.

본 논문에서는 이러한 단점을 보완하고자 EPZS(Enhanced Predictive Zonal Search) 움직임 추정 알고리즘을 이용하여 움직임 벡터를 추출하고 추출 된 움직임 벡터를 이용하여 카메라의 패닝 여부를 판단하는 알고리즘을 제안하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 움직임 기법에 대해서 알아보고, 3장에서는 본 논문에서 제안하는 카메라 패닝시 움직임 검출 기법에 대해서 기술하고, 4장에서는 실험 결과를 나타내었으며, 끝으로 5장에서는 결론을 맺는다.

Ⅱ. 기존의 움직임 검출 기법

고정된 카메라를 사용하여 움직임 검출을 하는 기본적인 기법은 기준 프레임을 저장하고 기준 프레임과 현재 프레임의 차이를 비교하는 기법, 기준 프레임을 이전 프레임으로 설정하고 이전 프레임과 현재 프레임의 차이를 비교하는 기법, 특정 영역을 몇 개 정하고 정해진 이 영역들의 색상 또는 휘도 값을 비교하여 차이가 나면 그때

움직임이 발생했다고 시스템에 알려주는 방법들이 대부분이다.

본 논문에서는 가장 많이 사용되는 이전 프레임과 현재 프레임의 차이를 이용한 기법에 대해서만 이야기 하겠다. 움직임 검출에 사용되어지는 일반적인 기법은 다음 식 1과 같다.

$$d_{ij}(x,y)=\begin{cases}1 & \text{if } |f(x,y,t_i)-f(x,y,t_j)|>THF \\ 0 & \text{otherwise}\end{cases}$$

$$Detect=\begin{cases}1 & \text{if } \sum_{x=1}^{M}\sum_{y=1}^{N} d_{ij}(x,y) > THA \\ 0 & \text{otherwise}\end{cases}$$

식 1. 일반적인 움직임 검출 기법

여기서 f는 두 영상의 각 픽셀 값이며, M과 N은 각각 영상의 세로 크기와 가로 크기를 의미하고, i와 j는 입력된 영상의 순서를 의미한다. 그리고 THP(픽셀 문턱치)는 각 픽셀의 변화에 대한 문턱치 이고, THA(면적 문턱치)는 변화가 있는 픽셀의 수에 대한 문턱치 이다.

이러한 움직임 검출 방법을 이용하여 개발되는 시스템의 대부분은 움직임 검출을 동영상인코딩 하기 전 과정에서 수행하게 만든다. 인코딩과 동시에 한다면, 움직임 검출이 되었을 경우에만 녹화를 하게 하는 기능 등을 수행할 수 없기 때문이다. 다음은 일반적인 움직임 검출 시스템 이다.

그림 1. 일반적인 움직임 검출 시스템

본 논문에서 사용한 움직임 검출 기법의 기본적인 알고리즘은 다음과 같다.

$$D(x,y) = \sum_{y=0}^{H}\sum_{x=0}^{W} \text{if}(ABS(lpData[y][x]-m\_lpData[y][x]) > ThA)$$

then dwDifference++;

$$Detect=\begin{cases}1 & \text{If (dwDifference > THB)} \\ 0 & \text{else}\end{cases}$$

식 2. 영상의 차이를 이용한 움직임 검출 기법

여기서, lpData[y][x]는 현재 프레임이고, m_lpData[y][x]는 이전 프레임이다. H와 W는 각각 영상의 세로 크기와 가로 크기를 의미한다. 그리고 임계치(ThA)는 각 픽셀의 변화에 대한 임계치이다. 즉, 블록 단위로 이전 영상과 현재 영상의 차이를 비교하면서 임계치 보다 클 때 dwDiffernce를 증가시킨다. 이 증가된 카운트 값인 dwDifference가 임계치(THB) 보다 크면 그때 움직임이 검출되었다고 판단한다.

III. 움직임 벡터를 이용한 카메라 패닝에 효율적인 움직임 검출 기법

먼저 카메라 패닝이라는 것은 카메라가 좌/우로 움직이는 것을 이야기 하는 것이다. 현재 움직임 검출 기법이 적용된 시스템들은 설정한 블록들의 색상의 변화 값을 이용한 검출 방법들이 주로 많이 사용되어지고 있다.

그러나 이러한 것들에서의 문제점은 카메라가 갑자기 좌/우로 움직여 버릴 경우 즉, 패닝이 발생 하였을 경우에 아무런 대책이 없다는 것이다. 현재 대부분의 움직임 검출 시스템들은 카메라 패닝 시 움직임이 발생했다고 시스템에 통보를 하고 있다.

본 논문에서는 이러한 오류를 바로잡고자 EPZS 탐색 기법을 사용하여 카메라 패닝 보정 기법에 대한 알고리즘을 보여준다.

아래 그림 2는 패닝 시 움직임 벡터의 방향성을 나타낸다.

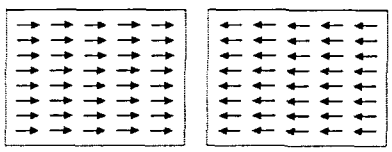


그림 2. 카메라 패닝 시 움직임 벡터의 흐름

본 논문에서 제시하는 움직임 추정을 이용한 카메라 패닝 보정 알고리즘은 다음과 같다.

EPZS 움직임 추정 알고리즘을 이용하여 움직임 벡터 값을 추출해 내고, 추출된 움직임 벡터(X, Y)값을 이용하여 카메라의 패닝 여부를 판단한다.

그리고 이 알고리즘에서 중요한 것은 f_Code의 선택과 두 개의 임계치와 두 개의 카운트 값이다. 이 f_Code와 두 개의 임계치와 두 개의 카운트 값을 적응적으로 사용할 수 있다는 것이 이 알고리즘의 특징이다. 여기서의 f_code는 1로 설정 하였다. 따라서 psMotionX가 가질 수 있는 움직임 벡터의 값의 범위는 [8, -8]이 된다.

일반 CCD 카메라에서 영상을 얻기 위해서 캡춰 보드를 이용하여 YUY420 형태의 352×240 사이즈로 영상 데이터를 얻어낸다. 얻어낸 영상에서 16×16의 블록으로 움직임 벡터를 구하고 구해진 움직임 벡터에 f_Code=1일 때, [8 or -8]인 값들을 카운트 한다. 이렇게 얻어진 첫 번째 카운트 값들에 첫 번째 임계치를 적절하게 적용하여 두 번째 카운트 값을 얻어낸다. 이 두 번째 카운트 값이 두 번째 임계치 이상이면 그때 카메라 패닝

이라고 인식을 하고 다른 움직임 검출을 하지 않고 스킵하면 된다.

그림 3은 제안한 카메라 패닝 추정의 순서를 간단히 나타낸 흐름도이다.

보다 상세한 과정은 다음과 같다.

과정 1) 입력받는 영상 데이터가 첫 번째 프레임인 경우는 참조 프레임으로 사용하기 위해서 움직임 벡터를 추출하지 않고 저장한다.

과정 2) 첫 번째 프레임은 참조 프레임이 되고 두 번째 프레임이 현재 프레임이 된다. 현재 프레임과 참조 프레임을 이용하여 16×16 데이터 블록에 대한 움직임 벡터 값을 추출한다. X와 Y축의 움직임 벡터는 22(X축), 15(Y축) 개의 곱인 총 330개의 벡터 값이 나온다. 이때 저장되는 변수를 psMotionX 라 하면 psMotionX[X][Y]의 이중 배열 형태로 나타낼 수 있다.

과정 3) 이렇게 얻어진 벡터 값들을 가지고 카메라의 패닝이 발생했는지 아니면 움직임이 발생했는지를 알 수 있다. psMotionX[X][Y]의 배열에서 Y가 0 일 때 X축의 값 중에서 [8, -8]의 값을 가지는 psMotionX의 숫자를 카운트 한다. 이 과정을 Y가 0에서 14가 될 때까지 수행한다. 여기에서의 8과 -8은 MPEG-4에서 f_code를 1로 했을 때의 최대 최소값이다.

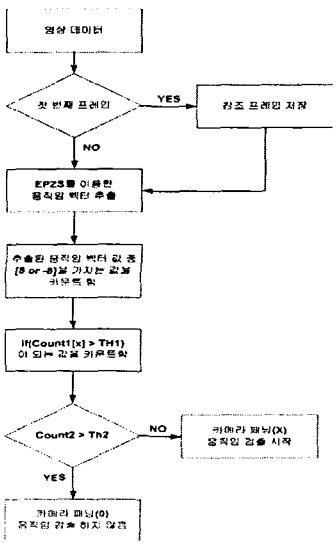


그림 3. 카메라 패닝 추정 흐름도

과정 4) 이렇게 저장된 CountX 값들은 22개가 될 것이다. 이 22개의 값들 중에서 첫 번째 임계치 TH1 이상의 값들은 가지는 두 번째 카운트 값을 찾아낸다.

과정 5) 두 번째 카운트 값이 두 번째 임계치 TH2 이상의 값을 가지면 그때 이것을 카메라 패닝이라 보고 움직임 검출을 하지 않고 스킵한다.

여기서 중요한 것은 f_code 값의 선택과 두 개의 임계치와 두 개의 카운트 값이다. 이 값들을 조절하게 조절하면 상황에 맞는 카메라 패닝 보정이 가능하게 된다.

아래 그림 4는 카메라 패닝 추정을 위해서 실제로 적용된 프로그램의 소스 코드의 일부분이다.

```
for(j = 0; j <15; j++)
{
        for(i = 0; i < 22; i++)
        {
                if(psMotionX[i][j] == 8 || psMotionX[i][j] == -8)
                        Count1++;
        }
        CountX[j] = Count1;
}

for(int k=0; k<22; k++ )
{
        if(CountX[k] >Threshold1 )
                F_Count++;
}

If(F_Count > Threshold2)
{
   CameraPanning으로 간주함
   return;
}
else
   MotionDetect를 수행
```

그림 4. 카메라 패닝 추정 코드

IV. 구현결과 및 실험

실험에 사용한 프로그래밍은 Visual C++6.0으로 하였으며, O/S는 winXP Pro에서 테스트하였다. 테스트에 사용되는 image 영상들은 동영상 캡춰 보드를 이용하여 YUY2데이터를 생성한 후 YUV420 데이터로 변환하여 실험에 사용 하였다.

YUV 데이터에서 Motion Estimation 과정에서 Motion Vector를 추출하기 위한 클래스의 구조체는 다음과 같다.

```
dmin = simpleme_epzs_motion_search(s, 0,
&mx, &my, P, pred_x, pred_y, rel_xmin,
rel_ymin, rel_xmax, rel_ymax, s->p_mv_
table,(1<<16)>>shift, mv_penalty);
```

그림 5. Motion Vector 추출을 위한 EPZS 함수

그림 5에서의 함수는 EPZS 알고리즘에서 움직임 벡터를 추출해내기 위해 실질적으로 사용되어지는 함수이다. 매개 변수 중 mx, my가 움직임 벡터 X와 Y의 값들이다.

앞서 보았듯이 f_Code와 2개의 임계치를 적응적으로 알고리즘에 적용한다고 하였다.

아래 그림 6은 실험 영상을 가지고 다양한 임계치를 적용한 실험 결과이다.

첫 번째 Threshold1는 값이 5 ~ 12 일때 까지 테스트를 해 보았는데 7 이하인 경우에는 값이 나오지 않았고, 12 이상이 면 모든 값이 다

TRUE로 나왔다.

또한 두 번째 Threshold2는 값이 4 ~ 11 일 때까지 테스트를 하였다.

결과적으로 Threshold1의 값이 11이고, Threshold2의 값이 7이 되었을 때가, 패닝 여부 판단에 적합한 Threshold 값이 추출되었다.

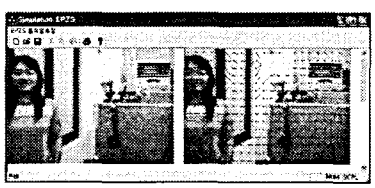
TH 1	TH 2	패닝 여부	TH 1	TH 2	패닝 여부
5	4	F	7	4	F
	6	F		6	F
	7	F		7	F
	8	F		8	F
	9	F		9	F
	11	F		11	F

TH 1	TH 2	패닝 여부	TH 1	TH 2	패닝 여부
8	4	F	9	4	F
	6	F		6	F
	7	F		7	F
	8	F		8	F
	9	F		9	T
	11	T		11	T

TH 1	TH 2	패닝 여부	TH 1	TH 2	패닝 여부
10	4	F	11	4	F
	6	F		6	T
	7	F		7	T
	8	T		8	T
	9	T		9	T
	11	T		11	T

그림 6. TH1 과 TH2 값에 따른 패닝 여부 결과

그림 7은 EPZS 움직임 추정 시 벡터 값의 변화량을 보여주기 위해서 실험 한 결과이다. 아래 실험들은 f_code = 1, TH1 = 11, TH2 = 7의 값을 가지고 실험 한 결과이다.



a) 움직임이 발생없이 카메라가 패닝 된 경우

b) 카메라는 고정되고 움직임이 발생한 경우

그림 7. EPZS 움직임 추정 시 벡터 변화량

구현된 EPZS 움직임 추정에서 왼쪽의 영상은 이전(참조) 영상(Reference Image)이고, 오른쪽의 영상은 현재 영상(Current Image)이 된다. 현재 영상에는 움직임 벡터들이 나타나고 있는데 빨간 선으로 나타난 것들이 움직임 벡터들의 양이다.

이때도 a)의 경우는 움직임은 발생하지 않고 카메라가 패닝 된 경우이다. 이 경우를 보면 움직임 벡터의 활동 양이 많아진 것을 볼 수 있다. 또한 움직임 벡터들도 일렬로 나란하게 나타나는 것을 볼 수 있다. 반대로 b)의 경우를 보면 카메라는 고정되고 움직임이 발생한 경우이다 이런 경우에는 적은 움직임 벡터의 활동 양을 볼 수 있다.

V. 결 론

본 논문에서는 EPZS 움직임 탐색 기법을 사용해서 움직임 벡터를 추출하고 이 추출된 움직임 벡터를 활용하여 카메라의 패닝을 보정하는 알고리즘을 제안하였다. 본 논문에서는 기존의 움직임 검출 기법에 추가적으로 카메라 패닝을 보정할 수 있는 알고리즘을 추가하여 보다 효율적인 움직임 검출을 방법을 제안하고 실험 하였다. 본 논문은 실제 DVR 프로그램에 적용하였으며 탁월한 성과를 얻고 있다.

논문에서의 실험 결과는 f_code는 1, Threshold1 = 11, Threshold2 = 7일 때 가장 좋은 결과를 얻을 수 있었다. 물론 이 값은 고정되는 값은 아니다. 다만, 실험을 통해서 가장 적합한 수치를 찾아낸 것이다. 이 값들은 각각 환경에 따라서 값들을 변화시켜 적응적으로 사용할 수 있다는 장점이 있다.

본 논문에서는 X값만을 가지고 하였지만 Y값을 가지고 할 경우 상/하의 카메라 움직임 경우도 보정이 가능하다. 즉, Panning 뿐만 아니라 Tilting에도 사용이 가능하다.

향후 계획은 Y값을 이용하여 Tilting 기법에도 적용을 할 것이고, 이 움직임 벡터 값들을 적용하여 상/하/좌/우/우상/우하/좌상/좌하 이렇게 8개의 값들을 추출해 내어서 보다 정확한 움직임 검출 기법을 제안하고자 한다.

참고문헌

[1] A.M. Tourapis, H.Y. Cheong, O.C. Au, and M.L. Liou, "N-Dimensional Zonal Algorithms. The Future of Block Based Motion Estimation", in proceedings of the 2001 IEEE International Conference on Image Processing (ICIP'01), Thessaloniki, Greece, October 2001.

[2] S. Zhu and K.K. Ma, "A new diamond search algorithm for fast block matching motion estimation," Proc. of Int. Conf. Information, Communications and Signal Processing, vol.1, pp.292-6, 1997.

[3] http://ffmpeg.sourceforge.net/