韓國 컴퓨터情報學會 論文誌
第12卷 第2號, 2007. 5.

2007-12-2-1-16

다중해상도 알고리즘을 이용한 고속 움직임 정합

주 헌 식*

High Speed Motion Match Utilizing A Multi-Resolution Algorithm

Heon-Sik Joo *

요 약

본 논문에서는 다중해상도 알고리즘을 제안하여 탐색점과 복잡도를 블록정합 알고리즘과 비교하여 나타내었다. 또한 스피드 업을블록정합 알고리즘과 비교 하였다. 제안한 다중해상도 NTSS-3 Level 알고리즘을 비교대상인 TSS-3 Level 알고리즘과 NTSS 알고리즘에 비교하였다. 비교 결과 탐색점과 스피드업에서 제안한 NTSS-3 Level 알고리즘이 우수함을 나타내었다. 따라서 제안한 NTSS-3 Level 알고리즘이 비교 대상인 블록정합 알고리즘에 비해 탐색점에서 2~3배 우수한 성능을 나타내었고 복잡도 계산에서도 2~4배의 우수함을 나타내었다. 스피트업에서도 제안한 NTSS-3 Level 알고리즘이 2배 이상의 성능을 나타내었다. 따라서 제안한 다중해상도 NTSS-3 Level 알고리즘이 탐색점과 스피드 업 대비 PSNR 우수함을 나타내었다.

Abstract

This paper proposed a multi-resolution algorithm. Its search point and complexity were compared with those of block match algorithm. Also the speed up comparison was made with the block match algorithm. The proposed multi-resolution NTSS-3 Level algorithm was compared again with its targets, TSS-3 Level algorithm and NTSS algorithm. The comparison results showed that the NTSS-3 Level algorithm was superior in search point and speed up. Accordingly, the proposed NTSS-3 Level algorithm was two to three times better in search point and two to four times better in complexity calculation than those of the compared object, the block match algorithm. In speed up, the proposed NTSS-3 Level algorithm was two times better. Accordingly, the proposed multi-resolution NTSS-3 Level algorithm showed PSNR ration portion excellency in search point and speed up.

▸ Keyword : 다중해상도(Multi-Resolution), 블록정합알고리즘(Block Match Algorithm) 탐색점(Search Point), 복잡도(Complexity)

• 제1저자 : 주헌식
• 접수일 : 2007.5.8, 심사일 : 2007.5.12, 심사완료일 : 2007. 5.23.
* 삼육대학교 컴퓨터학부 부교수

Ⅰ. 서 론

최근 모바일 시대를 맞이하여 동영상을 다양한 분야에서 사용하고 응용한다. 동영상은 대용량이므로 압축 기술을 요구한다. 움직임 보상(Motion Compensation)을 비디오 압축기술[1, 2]로 사용하며 움직임 벡터(Motion Vector)를 구한다[3]. 블록 정합 알고리즘(Block Matching Algorithm: BMA)에서 움직임 벡터를 구하기 위해 탐색 영역 내의 탐색점 중에서 BDM(Block Distortion Mode)이 최소인 점을 움직임 벡터로 결정하여 사용한다. 움직임 벡터를 구하기 위해 영상의 한 프레임을 동일한 블록으로 나누어 현재 프레임과 이전 프레임에 매치시켜 두 블록 사이의 상대적인 화소의 값,

또는 위치 이동 좌표 값을 정합오차(SAD : Sum of Absolute Difference)로 구한다[4]. 그림 1의 (a)는 현재 프레임을 M× N의 겹쳐지지 않은 블록으로 분할한 후, 분할된 각각의 블록에 대해 이전 프레임에서 그 블록과 가장 유사한 블록을 찾아내어 움직임 벡터를 구하는 것을 나타낸다. 블록이 움직일 수 있는 최대 변위가 w 일 때, 이전 프레임에서 탐색어질 블록들은 $M(M+2w)(N+2\omega)$ 크기의 탐색 영역 안에 존재하며 블록 정합이 수행될 탐색 위치의 수는$(2\omega+1)^2$ 로 나타낸다.

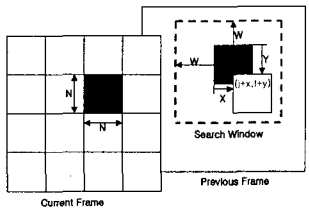

그림 1.(a)참조 프레임과 현재프레임
Fig 1. (a) Reference frame and current frame

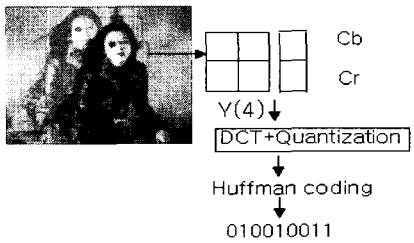

그림 1. (b)차분신호부호화
Fig 1. (b) Differential Signal Encoding

움직임 벡터는 참조 프레임의 Search Window에서 이전 프레임의 (x, y) 만큼 움직임을 현재 프레임의 위치로 움직임을 나타낸 것으로 참조 프레임과 현재 프레임의 움직임 벡터를 그림으로 나타낸 것이다. 그림1의 (b)는 동작 벡터에 의해 보상된 영상과 원영상과의 차분신호를 부호화로 처리한 것을 나타낸다. 고속정합알고리즘(FBMA : Fast Block Matching Algorithm)은 탐색 패턴을 이용하여 움직임 벡터를 찾는다. 탐색 패턴을 사용하면 계산량의 감소로 탐색 속도의 개선으로 성능이 향상되고 소프트웨어 구현이 용이하다. 고속 정합 알고리즘으로 TSS(Three Step Search)[5, 6], NTSS(New Three Step Search)[7], FSS(Four Step Search)[8], HS(Hexagon-based Search)[10],[9,] HMR (Hierarchical-Multi-resolution) [11, 12] DS(Diamond Search) 등을 사용한다. 본 논문에서는 다중해상도기법을 적용한 탐색 점과 스피드 업 그리고 PSNR를 비교하였다. 2장에서는 다중해상도기법 소개, 3장에서는 블록정합 알고리즘과 다중해상도 알고리즘 탐색점과 복잡도의 계산, 4장에서는 블록정합알고리즘과 제안한 알고리즘 설계 및 구현 5장에서는 실험 결과, 6장 결론을 나타내었다.

Ⅱ. 다중해상도(Multi-Resolution)기법

다중해상도 알고리즘은 상위 단계로 올라갈수록 공간 해상도(MR)가 낮아지는 Mean Pyramid을 생성한다. 상위 벡터에서 찾은 움직임 벡터를 중심으로 작은 범위의 움직임을 예측하여 하위 단계를 수행하여 하위 단계에서 움직임 벡터를 찾는다. 연산량을 크게 감소시키지만 국부 국소에 의한 성능 저하를 초래하는 우려가 있다. 이와 같은 특징들을 가지고 있어 Complexity를 줄 있 수 있다. 그림 2 는 다중해상도(MR : Multi-resolution)를 나타낸다[13]. 레벨에 따라 해상도가 다르다. 해상도가 낮은 순서대로 상위, 중간, 하위 단계를 나타낸다. 이들 레벨의 해상도는 이미지 크기를 나타낸다. 다중해상도 기법은 계층적으로 움직임을 예측하고 보상하는 다운샘플링(down sub sampling) 이용하여 저 해상도에서 최초움직임벡터를 예측하여 움직임을 예측하고 이때 얻어진 움직임 벡터를 적용하여 고 해상도 영역으로 스케일링(scaling)한다. 그림 2에서 상위 레벨인 레벨 2에서는 4×4의 해상도로 원영상의 1/4 축소된 크기에 해당한다. 이미지 크기가 1/4로 축소된 4×4의 블록에서 이전 프레임과 현재 프레임의 값을 SAD를 구하여 최소정합오차로서 움직임벡터를 찾는다.

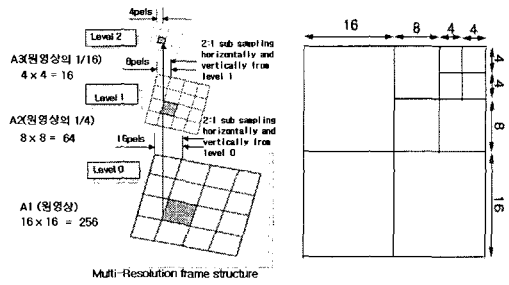

그림 2. 다중해상도 알고리즘
Fig 2. Multi-Resolution Algorithm

그리고 중간단계인 레벨1에서는 상위 레벨에 비해 이미지 해상도가 8×8 로 4×4에 비해 2배 확대된 것으로 SAD를 구하여 최소정합오차로서 움직임 벡터를 찾는다[19]. 그리고 다음 단계인 하위 단계에 해당하는 레벨 0로서 이미지 해상도가 처음 4×4 보다는 4배 확대 된 16×16 블록으로 원영상에 해당하며 움직임 벡터를 구하여 최종적인 해로 결정한다. 이렇게 해상도에 따라 단계별로 움직임 벡터를 구하는 것이 블록의 크기가 16×16에서 찾는 것보다 이미지의 크기가 1/4과 1/2로 줄어든 레벨에서 찾기 때문에 탐색하는 탐색점의 수가 적어짐으로 계산 복잡도가 줄어들게 된다. 따라서 레벨에 따른 해상도를 사용함으로써 계산 복잡도가 좋아진다는 이론에 근거하여 다중해상도 기법을 적용한 NTSS-3 Level 알고리즘을 제안하였고 이 알고리즘을 FS, TSS, NTSS 알고리즘과 다중해상도기법 적용한 TSS-3 Level 알고리즘과도 탐색점과 복잡도비교를 제안하였다.

III. 블록 정합 알고리즘과 다중해상도 알고리즘 탐색점 계산

본 절에서는 전역 탐색 알고리즘(FS : Full Search), TSS 알고리즘, NTSS알고리즘의 탐색점과 복잡도의 계산량을 나타낸다. 먼저 FS 알고리즘의 탐색점을 구하기 위해서 매크로 블록 16×16이고 탐색영역 W = ±7인 경우 매크로 블록과 탐색 영역에서 탐색점을 구하기 위해 식 (1)과 같이 계산식을 나타내었다.

$$(2W+1)^2 \times N^2 \quad \cdots\cdots (1)$$
$$= (2\times 7+1)^2 \times 16^2$$
$$= (15)^2 \times (16)^2$$

식(1)의 결과 57,600개의 탐색점이 계산되었다.

다음은 TSS 알로리즘의 탐색점을 16 x 16 Search Block에서 구하면 식 (2)과 같다.

1단계 256× 9(탐색점) = 2,304 ..... (2)
2단계 256× 8(탐색점) = 2,048
3단계 256× 8(탐색점) = 2,048

TSS 알고리즘 수행 결과 6,400개의 탐색점을 계산하였다. 다음은 NTSS 알고리즘의 탐색점을 구하기 위해 16×16 Search block에서 식(3)와 같이

1단계 256× 17(탐색점) = 4,352(최적의경우) ..... (3)
2단계 256× 8(탐색점) = 2,048
3단계 256× 8(탐색점) = 2,048(최악의경우)

계산식 수행 결과 최적의 경우는 1단계만 수행하여 4,352번을 나타내고 최악의 경우는 총 8,448번을 수행한다.

3.1 제안한 다중해상도 알고리즘 탐색점 계산

본 절에서는 제안한 다중해상도기법을 비교대상으로 적용한 TSS-3 레벨과 제안한 NTSS-3 레벨 알고리즘의 탐색점과 복잡도 계산량 나타낸다. 먼저 제안한 다중해상도기법을 비교대상으로 적용한 TSS-3 알고리즘을 식(4)와 같이 나타낸다.

Level 2 ; 16× 9= 144 ..... (4)
Level 1 ; 64× 9= 576
Level 0 ; 256× 9= 2,304

TSS-3 레벨 알고리즘은 기존 TSS에 다중해상도를 적용하여 레벨에 따라 다르게 수행한 계산 결과로서 3,024번을 나타내었다. 이 계산 결과는 FS알고리즘 보다는 19배의 계산을 적게 수행하는 높은 성능 향상을 나타내었다. 다음은 본 논문에서 제안한 NTSS-3레벨 알고리즘을 탐색점 구하기 위해 식 (5)와 같이 나타낸다.

Level 2 ; 16× 9= 144 ..... (5)
Level 0 ; 256× 9= 2,304(최적의경우)
Level 1 ; 64× 9= 576
Level 0 ; 256× 9= 2,304(최악의경우)

1단계를 수행한 계산 결과로서 최적의 경우는 2,448번을 나타내었는데 이 계산 결과는 FS 알고리즘 보다 23배의 높은 성능을 나타낸다. 그러나 최악의 경우에서도 수행한 계산 결과는 5,428번을 나타내었는데 이 계산 결과도 FS 알고리즘 57,600번 보다는 10배의 수행 결과로 높은 성능을 나타내었다.

3.2 블록 정합 알고리즘과 제안한 다중해상도 알고리즘의 복잡도 계산

3.2.1 블록 정합 알고리즘의 복잡도 계산

또한 FS, TSS, NTSS, TSS-3 레벨, NTSS-3 레벨 알고리즘의 탐색점 복잡도를 아래와 같이 정의하고 나타낸다.

- $[-w, w-1]$: 탐색영역
- 레벨 2, 레벨 1, 레벨 0 : 각각 4화소, 8화소, 16화소
- w : 16(정 방향 블록의 크기)
- $W \times H$: 영상 크기(352×288)
- M : 픽셀 당 SAD의 연산 수
- R_f : 프레임율(100×60)이라 할 때

FSBMA에 대한 계산 복잡도를 계산식 (6)과 같이 나타낸다.

$FSBMA$는 $C_{FSBMA} = (2W+1)^2 \times 16^2 \times (W \times H)/16^2 \times R_f$ $operations/s = 3,801$ ..... (6)

TSS 알고리즘에 대한 계산 복잡도를 계산식 (7)과 같이 나타내었다.

$$16^2 \times [3^2 + (3^2-1) + (3^2-1)] \times (352 \times 288)/16^2 \times 100 \times 60 = 422 \quad \cdots\cdots (7)$$

TSS 알고리즘은 탐색점 25개에 대한 복잡도는 422를 나타내었다. 다음은 NTSS에 대한 계산 복잡도를 계산식 (8)과 같다. NTSS 알고리즘은 최적일 경우 17개의 탐색점을 탐색하지만 최악인 경우 33개의 탐색점을 탐색하는 계산복잡도를 갖는다. 계산식 (8)은 NTSS 최적의 경우를 나타낸다.

$$16^2 \times [3^2 + (3^2-1)] \times (352 \times 288)/16^2 \times 100 \times 60 = 287 \quad \cdots\cdots (8)$$

NTSS 알고리즘의 최적의경우로 탐색점 17개에 대한 복잡 도는 287를 나타낸다.

NTSS 알고리즘의 최악의경우를 계산식 (9)로 나타낸다.

$$16^2 \times \{[3^2 + (3^2-1)] + (3^2-1) + (3^2-1)\} \times (352 \times 288)/16^2 \times 100 \times 60 = 558 \quad \cdots\cdots (9)$$

NTSS 알고리즘의 최악의 경우는 탐색점 33개에 대한 복잡도로 558를 나타낸다.

3.2.2 제안한 알고리즘의 복잡도 계산

제안한 NTSS-3 Level과 비교하기 위한 TSS-3 Level 알고리즘과 복잡도 비교한다.

1) TSS-3 레벨 알고리즘 복잡도

제안한 NTSS-3 Level과 비교하기 위한 TSS-3 Level 알고리즘의 복잡도는 (10)과 같다.

$$(4^2 + 8^2 + 16^2) \times 9 \times (352 \times 288)/16^2 \times 100 \times 60 = 200 \quad \cdots\cdots (10)$$

2) 제안한 NTSS-3 Level 알고리즘 복잡도

제안한 NTSS-3 Level 알고리즘의 복잡도는 (11), (12)와 같다.

$$(4^2 + 16^2) \times 9 \times (352 \times 288)/16^2 \times 100 \times 60 = 162 \quad \cdots\cdots (11)$$

NTSS-3 Level 알고리즘 최적의 경우 복잡도는 162를 나타낸다.

다음은 NTSS-3 Level 알고리즘의 최악의경우의 복잡도를 계산식 (12)로 나타낸다.

$$(4^2 + 16^2 + 8^2 + 16^2) \times 9 \times (352 \times 288)/16^2 \times 100 \times 60 = 337 \quad \cdots\cdots (12)$$

NTSS-3 Level 알고리즘의 최악의 경우 복잡도는 337를 나타내었다.

IV. 블록 정합 알고리즘과 제안하는 다중해상도 알고리즘 설계 및 구현

4.1 TSS (Three Step Search)알고리즘

움직임 벡터를 구하기 위한 TSS 알고리즘은 다음과 같다.

〔단계 1〕 탐색영역의 중심점으로부터 4픽셀 간격의 9개의 탐색 후보점에서 최소 정합 오차 값을 가지는 탐색 후보점을 구한다.

〔단계 2〕 1 단계 에서 구한 최소 정합오차 값을 갖는 탐색 후보점을 중심으로 2픽셀 간격의 8개의 탐색후보점을 추가하여 최소 BDM를 구한다.

〔단계 3〕에서는 〔단계 2〕에서 구한 최소 BDM 를 중심으로 1화소 간격에서 8개의 탐색 후보점을 추가하여 최소 BDM을 구하고 이때의 동작 벡터를 최종 움직임 벡터로 결정한다.

움직임 벡터를 구하기 위해 3단계 과정을 모두 수행하여 총 25개의 탐색 후보 점에 대해서 최소 정합 기준 오차 값을 구하여 최종 움직임 벡터로 결정하며 그림 3과 같다. TSS 알고리즘은 단순성과 효율성이 높아서 블록 정합 알고리즘으로 움직임 추정에 많이 사용 한다.

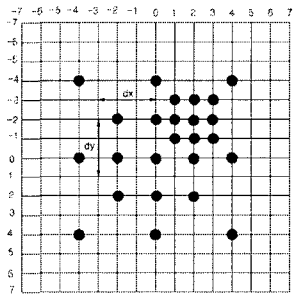

그림 3. TSS 알고리즘
Fig 3. TSS Algorithm

4.2 NTSS (New Three Step Search)알고리즘

NTSS 알고리즘을 Pseudocode로 다음과 같이 나타내었다.

초기값:

loc[17]={(0,0),(-1,0),(-1,-1),(0,-1),(1,-1),(1,0),
(1,1),(0,1),(-1,1),(-4,0),(-4,-4),(0,-4),
(4,-4),(4,0),(4,4),(0,4),(-4,4)}

Step 1:

p+loc[i](0 〈= i 〈 17)인 점에 대해 BDM을 계산한 후 BDM이 최소가 되는 i를 찾음

Step 2:

BDM이 최소가 되는 점이 중심점인 경우(i=0) 탐색 종료

Step 3:

BDM이 최소가 되는 점이 중심점 주위의 점인 경우(i〈8)모서리에 있는 4점인 경우(i가 짝수) 그 주위의 5점을 추가하여 탐색

그렇지 않은 경우(i가 홀수) 그 주위의 3점을 추가 탐색

Step 3-1:

BDM이 최소가 되는 점이 중심점에서 4만큼 떨어진 점인 경우(i〉=8) 그 점을 중심으로 2만큼 떨어진 8점을 추가 탐색 위의 결과의 점을 중심으로 1만큼 떨어진 8점을 추가 탐색

NTSS 알고리즘을 그림 4와 같이 Flow-Chart로 나타낸다.

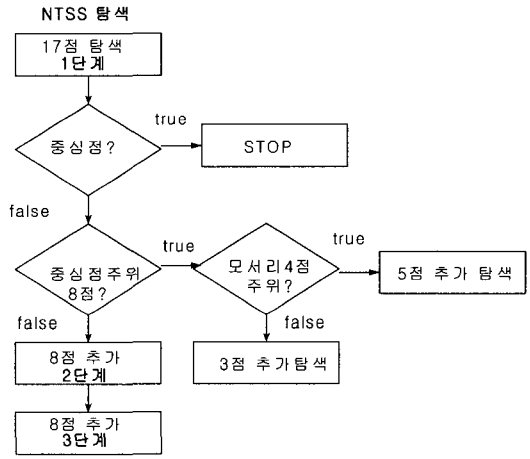

그림 4. NTSS 알고리즘 흐름도
Fig 4. NTSS Algorithm Flow-Chart

4.3 비교 대상인 TSS-3 레벨 알고리즘

4.3.1 TSS-3 Level 알고리즘

TSS-3 Level 알고리즘을 다음과 같이 나타내었다.

〔레벨 2〕 탐색 영역에서 중심점을 중심으로 1픽셀 간격의 9개의 탐색 후보 점에서 최소 BDM를 구한다.

〔레벨 1〕 레벨 2에서 구한 최소 BDM을 중심점으로 하여 이웃 하는 1픽셀 간격의 8개의 탐색 후보점을 추가 하여 최소 BDM을 구한다.

〔레벨 0〕 레벨 1에서 구한 최소 BDM을 중심점으로 하여 이웃 하는 1픽셀 간격의 8개의 탐색 후보점을 추가하여 최소 BDM을 구하고 최종 동작 벡터로 결정한다.

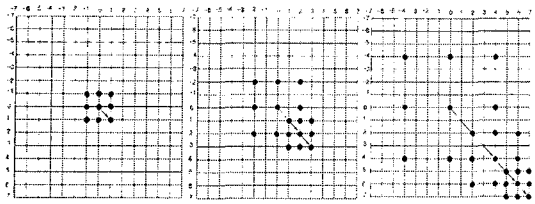

그림 5. TSS-3 Level 1단계 TSS-3 Level 2단계 TSS-3 Level 3단계
Fig 5. TSS-3 Level: Step 1 TSS-3 Level: Step 2 TSS-3 Level: Step 3

4.4 제안한 NTSS-3 Level 알고리즘 구현

제안한 NTSS-3 Level알고리즘을 Pseudocode로 나타내면 다음과 같다.

NTSS-3 Level 알고리즘 Pseudocode

초기값:

loclev2[9] =
{(0,0),(-1,0),(-1,-1),(0,-1),(1,-1),(1,0),
(1,1),(0,1),(-1,1)}
loclev0[8] =
{(-4,0),(-4,-4),(0,-4),(4,-4),(4,0),(4,4),
(0,4), (-4,4)}

Step 1:

p+loclev2[i](0<=i< 9)인 점에 대해 BDM을 계산한 후 BDM이 최소가 되는 i를 찾음(Level 2)

Step 2:

p+loclev0[j](0<=j< 8)인 점에 대해 BDM을 계산한 후 BDM이 최소가 되는 j를 찾음(Level 0)

Step 3:

BDM이 최소가 되는 점이 중심점인 경우(i=0) 탐색 종료

Step 4:

BDM이 최소가 되는 점이 중심점 주위의 점인 경우(Level 2에서 찾음)

모서리에 있는 4점인 경우(i가 짝수) 그 주위의 5점을 추가하여 탐색

그렇지 않은 경우(i가 홀수) 그 주위의 3점을 추가 탐색

Step 4-1:

BDM이 최소가 되는 점이 중심점에서 4만큼 떨어진 점인 경우(Level 0에서 찾음) 그 점을 중심으로 2만큼 떨어진 8점을 추가 탐색 위의 결과의 점을 중심으로 1만큼 떨어진 8점을 추가 탐색

제안한 NTSS-3 Level 알고리즘을 그림 6과 같이 Flow-Chart로 나타낸다.

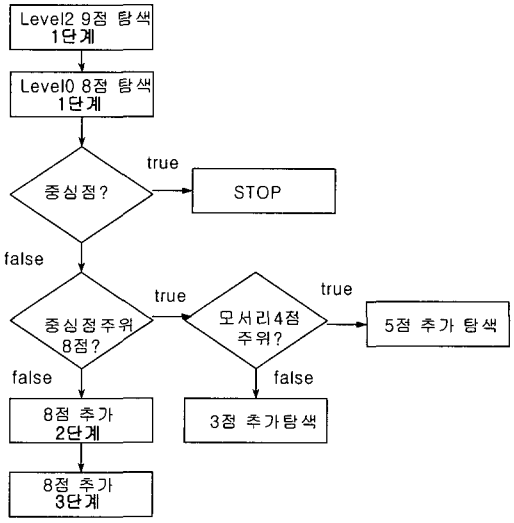

그림 6. NTSS-3 Level 흐름도
Fig 6. NTSS-3 Level Flow-Chart

따라서 본 논문에서 제안한 NTSS-3 Level 알고리즘과 기존의 알고리즘들과의 탐색점 계산량을 표 1로 나타내었다.

표 1. 기존 알고리즘과 제안한 알고리즘 계산량 비교
Table 1. Complexity Comparison of Existing Algorithm and Proposal Algorithm

알고리즘	계산량(최적)	계산량(최악)	NTSS-3레벨 (최적)으로 기준
FS	57,000		23 배
TSS	6,400		2.6배
NTSS	4,342	8,448	1.8배
TSS-3Level	3,024		1.2배
NTSS-3Level	2,448	5,428	NTSS-3Level 2,448기준

표 1의 탐색점 계산량 비교에서 제안한 NTSS-3 Level 알고리즘이 매우 높은 성능을 나타내었다.

또한 기존 알고리즘들과 제안한 NTSS-3 Level 알골즘의 탐색점 복잡도를 표 2로 나타내었다.

표 2. 알고리즘의 탐색점 복잡도 비교
Table 2. Complexity Comparison of Search points from Algorithms

알고리즘	탐색점 (최적/최악)	계산량 (최적)	계산량 (최악)	NTSS-3 Level(최적)으로 기준
FS	225	3,801		23 배
TSS	25	422		2.6배
NTSS	17 / 25	287	558	1.7배
TSS-3 Level	25	200		1.2배
NTSS-3 Level	17 / 35	162	337	NTSS-3 Level 162 를 기준

표 2에서 나타 낸 것과 같이 탐색점 복잡도가 NTSS-3 Level 〉TSS-3 Level 〉NTSS 〉TSS 〉FS의 순서대로 탐색점 복잡도의 성능이 높다.

V. 실험 결과

제안된 NTSS-3Level 알고리즘 성능 평가를 위하여 표 3의 조건에서 352 x 288 픽셀의 CIF의 동영상 2개를 실험대상으로 하여 실험하였다. 352 x 240 픽셀의 SIF 픽셀을 가지는 시퀀스 2개를 선택하여 100 프레임을 기준으로 실험하였다.

표 3. 구현 환경
Table 3. Experiment Environment

Development Software	Visual C++ 6.0
Coding Scheme	MPEG-2 Video Cording
Frame Size	CIF (352 x 288)
	SIF (352 x 240)
Block Size	100 frames
Number of Frames	10frames/sec.
Macro block Size	16 x 16 pixels
Search Area	$w = \pm 7$

블록 정합의 정도를 평가하기 위해 식 (13)의 평균 제곱 오차(MSE : Mean Squared Error)와 식 (14)의 평균 절대 값 오차(MAD : Mean Absolute Difference)

그리고 정합 오차 측정 함수로 식 (15)의 절대 값 오차의 합(SAD : Sum of Absolute Difference)을 이용하였다. 또한 탐색점과 스피드업을 비교하였다.

$$MSE(i,j) = \frac{1}{N^2}\sum_{k=1}^{N}\sum_{l=1}^{N}[I_t(k,l) - I_{t-1}(k+i,l+j)]^2 \quad \cdots (13)$$

$$MAD(i,j) = (\frac{1}{N^2})\sum_{k=1}^{N}\sum_{l=1}^{N}|I_t(k,l) - I_{t-1}(k+i,l+j)| \quad \cdots (14)$$

$$SAD(i,j) = \sum_{k=1}^{N}\sum_{l=1}^{N}|I_t(k,l) - I_{t-1}(k+i,l+j)| \quad \cdots\cdots (15)$$

여기서 N은 영상의 가로와 세로의 각각의 크기이며, $I_t(k,l)$은 원영상의 화면을 나타내고, $I_{t-1}(k+i,l+j)$은 움직임 예측 화면을 나타내며, 이들 정합 기준들은 최소 값을 가지는 위치를 움직임 벡터로 결정하였다. 그리고 화질의 평가를 위한 PSNR은 식 (16)과 같다.

$$PSNR = 10\log_{10}(\frac{255^2}{MSE}) \quad \cdots\cdots (16)$$

실험 영상에 대한 실험 결과는 〈표 4〉와 같다. CIF 영상과 SIF영상에 대해 FS, Original TSS, TSS-3 Level , Original NTSS, NTSS-3 Level 알고리즘을 적용하여 MAD, MSE, PSNR, 탐색점, 스피드업을 실험하였다. 실험결과 Akiyo 영상에 대한 MAD의 값이 FS는 0.61, Original NTSS가 0.61를 나타내었고, TSS-3 Level 는 0.73, NTSS-3 Level은 0.61로 제안한 NTSS-3Level이 TSS 3-Level보다는 좋은 결과를 나타냈다. 그리고 복원 영상의 화질을 비교한 PSNR을 보면 FS가 42.81를 나타내고, NTSS도 42.81, TSS는 42.68, NTSS-3 Level은 42.44, TSS-3Level은 40.49을 나타내어 제안한 NTSS-3 Level 알고리즘이 다소 떨어지나 탐색점 대비, 스피드 업 대비로 볼 때는 성능이 좋다고 볼 수 있다. 다음은 탐색점에서는 FS가 전역 탐색으로 264, TSS는 23, NTSS는 16, NTSS-3Level은 23, NTSS-3Level은 9를 나타내었다. NTSS-3레벨이 기존 NTSS에 비해 탐색점을 많이 적게 탐색하였다. 스피드업 비교결과 FS는 1, TSSSMS 11, NTSS는 16, TSS-3 LEVEL는 11, NTSS-3 Level은 29를 나타내었다. FS에 비해 무려 23배의 속도를 나타내었고 다른 알고리즘에 비해 좋은 성능을 나타내었다. 다른 영상에 대해서도 좋은 스피드업을 나타내었다. 본 논문에서 제안한 NTSS-3 Level 알고리즘은 동적인 영상보다는 정적인 영상에서 PSNR이 더 좋게 나타나는 것을 볼 수 있다.

Ⅵ. 결 론

본 논문에서는 기존의 블록정합 알고리즘인 FS 알고리즘과 TSS 알고리즘, NTSS 알고리즘 그리고 다중해상도 기법을 적용하여 제안한 NTSS-3 Level 알고리즘과 비교대상인 TSS-3 Level 알고리즘에 대해 탐색점, 복잡도, PSNR 등을 성능 평가하여 나타내었다. 먼저 탐색점과 성능평가로서 표 1에 나타난 것처럼 본 논문에서 제안한 NTSS-3 Level 알고리즘의 우수함이 FS 보다는 23배, NTSS 알고리즘보다는 평균 1.7 이상, 그리고 비교 대상인 TSS-3 Level 알고리즘과 비교해도 1.2배의 성능 향상을 나타내었다. 또한 기존 알고리즘들과 복잡도를 비교에서도 표 2와 같이 FS 보다는 23배, TSS 보다는 2.6배, NTSS 알고리즘보다는 1.7배, 그리고 비교대상인 TSS-3 Level 알고리즘보다는 1.2배 높은 성능을 나타내었다.

또한 영상 실험 결과로서 표4와 같이 Akiyo 영상에 대해서 탐색점을 비교해보면 FS 알고리즘은 264번 수행, TSS 알고리즘은 23번 수행, NTSS 알고리즘은 16번 수행, TSS-3 Level 알고리즘은 23번 수행하였고 제안한 NTSS-3 Level 알고리즘은 9번 수행함으로써 상당히 적은 탐색점을 구함으로써 높은 성능을 나타낸 것을 볼수 있다.

또한 스피업에서도 FS는 1, TSS는 11, NTSS는 16, TSS-3Level은 11, 제안한 NTSS-3 Level 은 29로 아주 높은 성능을 나타내었다. PSNR에서는 기존 알고리즘들에 비해 다소 낮은 복원력을 나타내고 있지만 탐색점 대비, 스피드 업 대비로 볼 때는 좋은 성능을 나타내었다고 볼수 있다.

따라서 제안한 NTSS-3 Level 알고리즘이 탐색점, 복잡도, 스피드업을 구하는데 있어 기존의 알고리즘보다 다중해상도 기법을 적용한 NTSS-3 Level 알고리즘의 우수함을 살펴보았다. 그러나 화질 복원에서는 복원력이 다소 떨어짐을 나타내었다. 하지만 탐색점 대비, 스피드 대비 관점으로 볼 때는 우수한 알고리즘이라고 사료되며 정적인 영상에서 화질의 복원인 PSNR이 더 좋게 나타나는 것을 볼 수 있었다. 앞으로 동적인 영상에서도 좋은 PSNR을 나타내기 위해 연구하겠다.

표4 실험 결과 (탐색점 과 스피드 업)
Table. 4 Experiment Results(Search Point and Speed Up)

	CIF										SIF									
	Akiyo					Foreman					Garden					Tennis				
Method	MAD	MSE	PSNR	Search P.	Speed Up	MAD	MSE	PSNR	Search P.	Speed Up	MAD	MSE	PSNR	Search P.	Speed Up	MAD	MSE	PSNR	Search P.	Speed Up
FS	0.61	3.94	42.81	264	1	2.88	32.67	33.27	264	1	7.51	299.77	24.60	264	1	4.32	111.09	29.98	264	1
TSS	0.61	4.08	42.68	23	11	3.14	38.26	32.59	23	11	8.37	273.55	23.83	23	11	4.95	142.77	28.87	23	11
NTSS	0.61	3.94	42.81	16	16	2.97	35.23	32.99	17	14	7.65	238.14	24.45	21	12	4.61	131.12	29.26	16	16
TSS-3 Level	0.73	8.65	40.49	23	11	4.90	97.66	28.57	23	11	19.17	1149.98	17.54	23	11	6.57	266.26	26.11	23	11
NTSS-3 Level	0.62	4.62	42.44	9	29	4.01	74.21	30.05	10	25	14.83	825.87	18.99	13	20	5.99	235.70	26.75	9	28

참고문헌

[1] A. K. Jain, "Image data compression : A review," Proc. IEEE, Vol.69, pp.349-389, Mar., 1981.

[2] A. N. Netravali and J. O. Limb, "Picture coding : A review," Proc. IEEE, Vol.68, pp.366-406, Mar., 1980.

[3] B. Liu and A. Zaccarin, "New fast algorithm forestimation of block motion vectors," IEEE Trans. Circuits and Systems for Video Tech., Vol.3, pp.148-157, Apr., 1993.

[4] Ya-Qin Zhang, Sohail Zafar "Motion Compensated Wavelet Transform Coding for Color Video Compression," IEEE Transactions

on Circuits and Systems for Video Technology, Vol.2, No.3, pp.285-296, September, 1992.

[5] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated Interframe Coding for Video Conferencing," in Proc. National Telecommunications Conf., New Orleans, LA, pp.G5.3.1-G5.3.5, Nov., 1981.

[6] J. Lu and M. L. Liou, "A simple efficient search algorithm for block matching motion estimation," IEEE Trans. Circuits Syst. Video Technol., Vol.7, pp.429-433, Apr., 1997.

[7] Reoxiang Li; Bing Zeng Liou, M.L, "A new three-step search algorithm for block motion estimation Circuits and Systems for Video Technology," IEEETransactions on, Vol.4, Issue.4, pp.438-442, Aug., 1994.

[8] L. M. Po, W. C. Ma, "A Novel Four-Step Search Algorithm for Fast Block-Matching Motion Estimation," IEEE Transactions on Circuits & System for Video Tech., Vol.6, No.3, pp.313-317, June, 1996.

[9] S. Zhu, K. K. Ma, "A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation," IEEE Transactions on Image Processing, Vol.9, No.2, pp.287-290, Feb., 2000.

[10] C. Zhu, X. Lin, L. P. Chau, "Hexagon-Based Search Pattern for Fast Block Motion Estimation," IEEE Transactions on Circuits & System for Video Tech., Vol.12, No.5, pp.349-355, May, 2002.

[11] Kwon Moon Nam; Joon-Seek Kim; Rae-Hong Park; Young Serk Shim; "A fast hierarchical motion vector estimation algorithm using mean pyramid" Circuits and Systems for Video Technology, IEEE Transactions on, Vol.5 , Issue.4, pp.344-351, Aug., 1995.

[12] M. Bierlig, "Displacement estimation by hierarchical block- matching," Proc. Visual Comm. And Image Proc., SPIE Vol.1001, pp.942-951, 1988.

[13] S. Zafar, Y.Q. Zhang, and B. Jabbari, "Multiscale video representation using multiresolution motion compensation and wavelet decomposition," IEEE J. Select. Areas Commun., Vol.11, pp.24-35, Jan., 1993.

저 자 소 개

주헌식

1992년 호서대학교 공과대학 컴퓨터공학과

1994년 호서대학교 대학원 전자계산학과(이학석사)

2005년 아주대학교 대학원 컴퓨터공학과(공학박사)

1997년 ~ 현재 삼육대학교 컴퓨터학부 부교수

관심분야 : 컴퓨터그래픽스, 영상처리, 멀티미디어