다중해상도를 이용한 새로운 3단계 블록정합 알고리즘

주 헌 식†

요 약

본 논문에서는 기존 NTSS 알고리즘을 다중해상도(MR : Multiple Resolution)기법을 이용하여 NTSS-3 레벨 알고리즘으로 제안하였다. 고속 블록정합 알고리즘은 패턴 방식에 따라 속도에 많은 영향을 미치는데 본 논문에서는 기존 NTSS의 패턴 방식과 다른 다중해상도 기법을 이용한 레벨에 따른 블록정합 알고리즘을 제안하였다. 블록 정합알고리즘에서 국부최소화 문제(Local minima problem)로 발생하는 화질 저하를 개선하기 위해 MC(Multiple Candidate)라는 다중후보를 이용하였다. 실험 결과 제안한 기법을 FS와 비교하면 16배의 탐색 속도를 나타내었고 기존 고속 블록 정합 알고리즘인 NTSS 방식에 비교 할 때 PSNR값에 있어서는 0.11~0.12[dB] 화질이 개선되었으며 속도 면에서도 0.1배 향상되었고 탐색점 대비 화질개선이 우수함을 나타내었다.

A NTSS of 3 Levels Block Matching Algorithm using Multi-Resolution

Heon-Sik Joo†

ABSTRACT

In this paper, we notice that the original NTSS algorithm can be proposed as the NTSS-3 Level algorithm by the multi-resolution technique. The fast block matching algorithm affects the speed by the patten combination and this paper proposes the block matching algorithm in different levels by multi-resolution technique, quite different from the original NTSS patten. The block matching algorithm requires the multi-candidate to reduce the occurrence of low-image quality by the local minima problem. The simulation result compared to FS shows search speed 16 times quicker, and the PSNR 0.11~0.12[dB] gets improved image quality compared to the original fast block matching algorithm NTSS, and the speed is improved up to 0.1 times for improved image by the search point portion.

키워드 : 다중해상도(Multi-Resolution), TSS-3 레벨(TSS-3 Level), NTSS-3 레벨(NTSS-3 Level)

1. 서 론

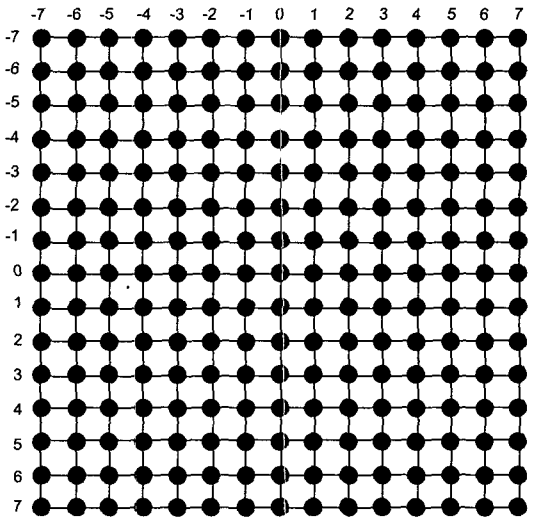
영상 압축은 디지털 비디오 데이터를 전송 및 저장 하는데 중요하다. 화상 회의와 같은 방대한 실시간 정보 제공을 위해 압축, 복원을 국제 표준으로 제정하고 있다. 동영상에서 움직임 보상(Motion Compensation)은 비디오 데이터를 압축[1, 2]하는데 적합한 기술이며, 움직임 보상을 하기 위해 움직임 벡터(Motion Vector)를 구하는 것은 매우 중요하다[3]. 움직임 벡터를 찾기 위해 블록 정합 알고리즘(Block Matching Algorithm)을 가장 많이 사용하고 있으며, BMA 방식은 탐색 점 중에서 BDM(Block Distortion Mode)이 최소인 점을 움직임 벡터로 결정하여 사용한다. 움직임 벡터를 찾기위해 블록 내의 화소들이 움직임을 수평, 수직으로만 움직인다는 전제하에 영상의 한 프레임을 동일한 블록으로 나누어 현재 프레임과 이전 프레임에 매치시켜 두 블록 사이의 상대적인 화소의 값, 또는 위치 이동 좌표 값을 정합 오차(SAD : Sum of Absolute Difference)로써 움직임 벡터(motion vector)로 나타낸다[4]. 움직임 예측(ME : Motion Estimation)을 함으로써 중복된 데이터를 최소화하여 저장 공간과 시간적 낭비를 줄인다. 전역 탐색 알고리즘(FS : Full Search)은 프레임 내의 모든 블록들을 탐색하기 때문에 성능이 가장 높지만 계산량이 많아 실시간 비디오 코딩 및 하드웨어 구현이 어렵다. 그래서 고속 정합 알고리즘 **(BMA : Block Matching Algorithm)**들을 제안하게 되었다. 고속 정합 알고리즘은 탐색 패턴을 이용하여 속도를 개선한 한 것으로 탐색 영역 일부에서 움직임 벡터를 찾는 방법이다. 탐색 패턴을 이용하면 계산량의 감소와 소프트웨어 구현이 용이하다. 하지만 적절한 탐색영역과 블록의 크기 결정에 어려움도 있다. 고속정합알고리즘(FBMA : Fast Block Matching Algorithm)으로 TSS(Three Step Search)[5, 6], NTSS(New Three Step Search) [7], FSS(Four Step Search) [8], DS(Diamond Search)[9, 23, 24] HS(Hexagon-based Search)[10, 21, 22], HMR (Hierarchical-Multi-resolution)[11, 12]등이 있으며, 이들은 모두 속도 개선을 위한 알고리즘들

† 정 회 원 : 삼육의명대학 컴퓨터정보과 교수
논문접수 : 2004년 7월 8일, 심사완료 : 2004년 9월 30일

로 주로 탐색 영역 내에서 탐색할 위치의 포인터 개수를 감소시켜 계산량을 유도하는 탐색 패턴을 사용한다. 이들 고속 블록 정합 알고리즘은 움직임 탐색 방법[25, 26]에 사용되는 탐색 패턴의 모양과 크기에 따라 탐색 속도와 성능을 좌우한다. 이러한 고속 정합 알고리즘은 계산량을 줄이기 위해 탐색 영역에 포함되는 특정한 패턴들의 몇몇 탐색 점들만 조사하여 움직임 벡터를 찾기 때문에 국부(Local minima Problem)적인 탐색을 하게 되거나, 일부 탐색 점들을 블록 정합 대상에서 제외시키므로 복원된 영상의 화질 저하를 초래한다. 기존 TSS 알고리즘이나 NTSS 알고리즘은 패턴에 의해 원 영상에서 단계별로 BDM을 찾아 움직임벡터를 찾는 방법으로 결과는 <표 1>, <표 2>로서 탐색 패턴에 대한 복잡도와 복원시의 화질저하의 문제를 나타낸다. 본 논문에서는 이러한 단점을 보완하기 위해서 기존 패턴에 대한 알고리즘을 연구하여 이미지를 해상도에 따라 레벨 별로 나누어 움직임 벡터를 찾는 방법을 제시하고 화질 저하에 대한 방안으로 다중후보 사용을 제안한다. 움직임 벡터를 구하기 위하여 이미지 해상도에 따른 레벨별 알고리즘을 적용하여 탐색 점을 적게 탐색하면서도 탐색 점 대비 좋은 화질을 얻기 위한 방법을 나타내었다. 제안한 방법은 탐색 패턴에 대한 알고리즘으로 기존 TSS 알고리즘을 TSS-3 레벨 알고리즘으로 NTSS 알고리즘을 NTSS-3 레벨 알고리즘으로 설계하여 이미지 해상도에 따라 레벨별로 적용하여 탐색점을 적게 탐색하면서도 탐색 점 대비 화질 개선(PSNR)을 구현하는 것이다. 상위 레벨은 이미지의 해상도가 가장 낮으며 하위로 내려올수록 이미지의 해상도가 높다. 상위레벨에서 이미지의 해상도가 낮기 때문에 탐색할 점을 수가 많이 줄어 들게 된다. 해상도는 상위부터 레벨 2, 레벨 1, 레벨 0 로 계층적인 피라미드 구조를 갖고 있으며 원본 이미지는 레벨 0이다. 레벨 0가 16화소일 때 레벨 2는 4화소로써 원본 이미지의 1/4 해상도로 나타내게 된다. 따라서 1/4로 축소된 해상도 상태에서 이미지를 탐색함으로 탐색 점이 1/4로 적은 상태에서 움직임 벡터를 찾게 된다. 레벨 1은 8화소로 써 원본 이미지의1/2로 해상도를 나타낸다. 따라서 1/2로 축소된 해상도 상태에서 이미지를 탐색함으로 탐색점이 1/2로 적은 상태에서 움직임 벡터를 찾게 된다. 따라서 본 논문에서는 고속 블록정합 알고리즘인 TSS, NTSS 알고리즘을 다중해상도(MR : Multi Resolution)기법[13, 14]을 적용하여 TSS 알고리즘을 TSS-3 알고리즘으로 그리고 NTSS 알고리즘을 NTSS-3 레벨 알고리즘으로 설계하고 구현하여 탐색점을 적게 탐색하면서도 탐색 점 대비 화질 개선을 제안하였다. 또한 제안한 방법의 성능평가를 비교하기 위해 기존 TSS 알고리즘과 NTSS 알고리즘 그리고 TSS-3 레벨 알고리즘과 제안한 NTSS-3 레벨 알고리즘에 대한 계산량과 복잡도를 계산식으로 나타내었다. 계산식에 의한 결과로서 제안한 결과가 기존 결과보다 계산량과 복잡도에서 향상된 성능을 나타내었다. 또한 이미지 복원시 국부 최소화 문제(Local Minima Problem)로 발생되는 화질 저하를 개선하기 위하여 MC (Multiple Candidate)[15-17]라는 다중 후보를 사용하여 국부 최소화(local minimum)를 개선하였다. 본 논문의 구성은 서론에서는 영상 압축의 필요성과 기본 알고리즘의 문제점을 제시하고 2장에서는 기존 고속알고리즘인 TSS 알고리즘과 NTSS 알고리즘을 TSS-3 레벨 알고리즘과 제안한 NTSS-3 레벨 알고리즘에 대해서 계산량과 복잡도에 대해서 나타내었고 3장에서는 다중해상도 기법과 적용에 대해서 4장에서는 TSS-3 레벨 알고리즘과 제안한 NTSS-3 레벨 알고리즘 설계 및 구현에 대하서 5장에서는 실험 결과와 성능 평가를 제시하고 6장에서는 결론을 나타내었다.

2. 기존 FS, TSS, NTSS 알고리즘과 TSS-3 레벨 NTSS-3 레벨 알고리즘의 계산량과 복잡도비교

본 장에서는 기존의 FS 알고리즘, TSS 알고리즘, NTSS 알고리즘 그리고 제안한 NTSS-3 레벨 알고리즘에 대한 계산량과 복잡도를 알아보고자 한다.



(그림 1) FS 알고리즘

(그림 1)은 전역탐색(FS : Full Search) 알고리즘으로 전역탐색에 대한 계산량을 나타내고자 한다. FS 알고리즘으로서 매크로 블록 16×16이고 탐색영역 W = ± 7인 경우 매크로 블록과 탐색 영역에 대한 계산식 (1)과 같이 나타낸다.

$$\begin{aligned} & (2W+1)^2 \times N^2 \\ = \; & (2\times 7+1)^2 \times 16^2 \\ = \; & (15)^2 \times (16)^2 \end{aligned} \tag{1}$$

계산식을 수행한 결과 57,600번을 나타내었다. 계산 결과

에서 나타낸 것처럼 계산량이 많아 실시간 코딩 및 소프트웨어 구현이 어렵다. 따라서 FS 알고리즘을 개선한 여러 고속 정합 알고리즘(FBMA : Fast Block Matching Algorithm)들이 제안되었으며 본 장에서는 기존 3단계 탐색 알고리즘(TSS : Three Step Search)과 새로운 3단계 탐색 알고리즘(NTSS : New Three Step Search)을 TSS-3레벨 알고리즘과 NTSS-3레벨 알고리즘으로 설계하여 계산량과 복잡도를 나타내고자 한다.

먼저 기존 TSS 알고리즘에 대한 계산량을 구한다. 16×16 search block에서 TSS 알고리즘을 구하기 위해 단계별 탐색에 대한 계산식 (2)와 같이 나타낸다.

$$\begin{aligned} &1\text{단계 } 256\times 9(\text{탐색점}) = 2,304 \\ &2\text{단계 } 256\times 8(\text{탐색점}) = 2,048 \\ &3\text{단계 } 256\times 8(\text{탐색점}) = 2,048 \end{aligned} \tag{2}$$

TSS 알고리즘은 3단계에 걸쳐 계산식을 수행한 결과 6,400번을 나타내었다. 계산 결과를 전역 탐색(FS) 알고리즘 57,600번에 비교해 보면 계산량이 상당히 감소하여 7배나 되는 높은 성능 개선을 나타내었다.

다음은 NTSS 알고리즘에 대한 계산량을 나타내고자 한다. 16×16 Search block에서 NTSS 알고리즘에 대한 계산량을 나타내기 위해 먼저 NTSS 알고리즘에 대해서 알아본다. NTSS 알고리즘은 TSS 알고리즘을 개선한 알고리즘으로 동작 벡터가 탐색영역의 중심에 분포한다는 사실을 이용하여 3단계 탐색기법을 보완한 방식으로 정적블록과 동적블록의 빠른 식별을 위해 중간-정지 기술을 이용하였다. **[1단계]**에서 TSS 알고리즘과 같이 중심점을 중심으로 4픽셀 간격의 9개의 탐색점을 탐색함과 동시에 중심점을 중심으로 1픽셀 간격의 8개의 탐색점도 동시에 탐색하여 총 17개의 탐색점을 단계별로 탐색하며 계산식 (3)과 같이 나타내었다.

$$\begin{aligned} &1\text{단계 } 256\times 17(\text{탐색점}) = 4,352(\text{최적의경우}) \\ &2\text{단계 } 256\times\ 8(\text{탐색점}) = 2,048 \\ &3\text{단계 } 256\times\ 8(\text{탐색점}) = 2,048(\text{최악의경우}) \end{aligned} \tag{3}$$

계산식 수행 결과 최적의 경우는 1단계만 수행하여 4,352번을 나타내었다. 이 계산 결과는 FS보다 13배가 넘는 계산량의 성능을 나타낸다. 그러나 최악의 경우에서는 단계1, 단계2, 단계3을 모두 수행하여 총 8,448번을 나타내었는데 이 계산 결과는 FS보다는 6배의 성능을 나타낸다.

다음은 TSS-3 레벨 알고리즘에 대한 계산량을 나타낸다. TSS-3 레벨 알고리즘은 기존 TSS 알고리즘에 해상도를 레벨에 따라 다르게 적용된 것으로 레벨 2, 레벨 1, 레벨 0로 해상도를 다르게 설계하여 계산식 (4)와 같이 나타내었다.

$$\begin{aligned} &Level2;\ 16\times 9 =\ \ 144 \\ &Level1;\ 64\times 9 =\ \ 576 \\ &Level0;\ 256\times 9 = 2,304 \end{aligned} \tag{4}$$

TSS-3 레벨 알고리즘은 기존 TSS에 다중해상도를 적용하여 레벨에 따라 다르게 수행한 계산 결과로서 3,024번을 나타내었다. 이 계산 결과는 FS알고리즘 보다는 19배의 성능 향상을 나타낸다.

다음은 본 논문에서 제안한 NTSS-3레벨 알고리즘을 설계하여 계산식으로 나타내었다. NTSS-3레벨 알고리즘은 앞에서 나타낸 기존 NTSS 알고리즘에 다중해상도 기법을 적용하여 설계한 방법으로 최적의 경우는 1단계에서 17개의 탐색점을 수행하는데 1단계에서 레벨 2와 레벨 0를 동시에 탐색하는 것으로 설계하여 하였고 1단계에서 만족하지 않을 경우 2단계, 3단계를 수행하는 단계별 탐색에 대한 계산식을 식 (5)와 같이 나타내었다.

$$\begin{aligned} &Level2;\ 16\times 9 =\ \ 144 \\ &Level0;\ 256\times 9 = 2,304(\text{최적의경우}) \\ &Level1;\ 64\times 9 =\ \ 576 \\ &Level0;\ 256\times 9 = 2,304(\text{최악의경우}) \end{aligned} \tag{5}$$

1단계를 수행한 계산 결과로서 최적의 경우는 2,448번을 나타내었다. 이 계산 결과는 FS 알고리즘 보다 23배의 높은 계산 결과의 성능을 나타낸다. 그러나 최악의 경우에서도 수행한 계산 결과가 5,428번을 나타내었다. 이 결과도 FS 알고리즘 57,600번 보다는 10배의 수행 결과로 높은 성능을 나타낸 것이다. 따라서 기존 알고리즘인 FS 알고리즘 계산량과 TSS 알고리즘 계산량과 그리고 NTSS 알고리즘 계산량과 TSS-3 레벨 알고리즘과 제안한 NTSS-3 Level의 계산량을 비교해보면 기존 FS 알고리즘은 57,000번의 계산결과를 나타내었고, TSS 알고리즘은 6,400번의 계산결과를 나타내었으며, 그리고 NTSS 알고리즘은 최적의 경우 4,352번을 나타내었고 최악의 경우는 8,448번을 나타내었다. TSS 알고리즘을 TSS-3 레벨 알고리즘으로 설계한 TSS-3 레벨 알고리즘은 3,024번을 나타냈으며, 제안한 NTSS-3 레벨 알고리즘은 최적의 경우 2,448번을 나타내었고 최악의 경우는 5,428번을 나타내었다. 따라서 제안한 NTSS-3 레벨 알고리즘이 최적의 경우 기존 FS 알고리즘에 비해 23배의 성능 향상을 나타내었고, TSS 알고리즘과 비교하면 2.6배의 성능 향상을 나타었으며 NTSS 알고리즘보다는 1.7배의 성능 향상을 나타내고, TSS-3 레벨 알고리즘 보다는 1.2배의 성능 향상을 나타내었다. 결과적으로 제안한 NTSS-3 레벨 알고리즘이 계산량에 있어서 이론적으로 우수함을 나타내었다.

다음은 본 논문에서 제안한 각각의 알고리즘에 대한 계산

복잡도를 계산식으로 나타낸다. 전역탐색 알고리즘인 FS, TSS, NTSS, TSS-3 레벨, NTSS-3 레벨 알고리즘에 대해 아래와 같이 나타내었다.

- $[-w, w-1]$: 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.

$$FSBMA는\ C_{FSBMA} = (2w)^2 M \times 16^2 \times (W \times H)/16^2 \times R_f$$
$$operations/s = 17,301 \quad (6)$$

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

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

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.

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

NTSS 알고리즘의 최적의경우로 탐색점 17개에 대한 복잡도는 287를 나타낸다.

NTSS 알고리즘의 최악의경우를 계산식 (9)로 나타낸다.

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

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

다음은 TSS-3 레벨에 대한 계산복잡도를 계산식 (10)으로 나타낸다.

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

다음은 제안한 NTSS-3 레벨 알고리즘에 대한 계산 복잡도로 최적의 경우 계산식 (11)로 나타낸다.

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

NTSS-3 레벨 알고리즘 최적의 경우 복잡도는 162를 나타낸다.

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

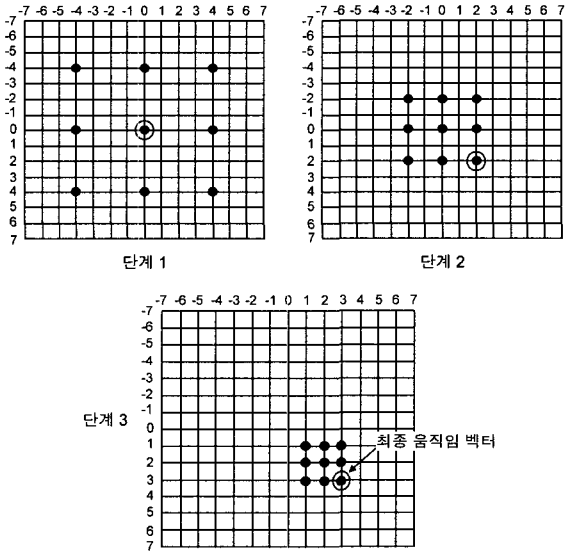
$$(4^2 + 16^2 + 8^2 + 16^2) \times 9 \times (352 \times 288)/16^2 \times 100 \times 60 = 337 \quad (12)$$

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

FSBMA의 복잡도 계산식 결과 17,301을 나타내었고, TSS의 복잡도는 422를 나타냈으며, NTSS 알고리즘의 최적의 복잡도는 287를 나타냈고, 최악의 경우는 558의 복잡도를 나타냈으며, TSS-3 레벨의 복잡도는 200를 나타내었고, 제안한 NTSS-3 레벨 알고리즘 최적의 경우 복잡도는 162를 나타냈으며, 최악의 경우의 복잡도는 337를 나타내었다. 복잡도가 높은 순서대로 나타내면 NTSS-3 레벨 최적 > TSS-3 레벨 > NTSS > NTSS-3 레벨 최악 > TSS > FS의 순서대로 복잡도가 높다. 제안한 NTSS-3 레벨의 최적의 복잡도는 FSBMA에 비해 106배의 높은 성능을 나타내었고, 최악의 경우도 51배의 성능을 나타내었고 TSS와 비교해 보면 2.6배의 성능을 나타내며, NTSS 최적과 비교해보면 1.7배의 성능을 나타낸다. 제안한 NTSS-3 레벨 알고리즘이 우수함을 나타낸다.

2.1 TSS(Three Step Search) 기법

3단계 탐색 알고리즘(TSS)은 단순성과 효율성이 높아서 블록 정합 알고리즘으로 움직임 추정에 폭 넓게 사용 되어 왔다. (그림 2)는 고속 알고리즘인 TSS를 나타낸다.



(그림 2) 3단계 탐색 기법(TSS)

TSS 알고리즘은 단계별로 움직임 벡터를 구하기 위하여 3단계 과정을 수행하면서 최종적인 움직임 벡터를 구하는

알고리즘이다. 움직임 벡터를 찾기 위해 탐색 영역의 중심으로부터 4 화소 간격의 9×9의 화소 간격의 9개의 탐색 후보점을 시작하여 점차로 범위를 좁혀가면서 단계별에서 움직임 벡터를 구하여 최종적인 움직임 벡터를 구하기 위해 3단계 과정을 모두 수행하여 총 25개의 탐색 후보 점에 대해서 최소 정합 기준 오차 값을 구하여 최종 움직임 벡터로 결정하며 TSS 알고리즘은 다음과 같다.

[단계 1] 탐색영역의 중심으로 부터 4화소 간격의 9개 탐색 후보 점에 대해 최소 BDM를 구한다.

[단계 2] 단계 1에서 구한 최소 BDM을 갖는 탐색 후보점을 중심으로 2화소 간격으로 위치한 8개의 탐색 후보점에 대해 최소 BDM를 구한다.

[단계 3] 단계 2에서 구한 최소 BDM를 갖는 탐색 후보점을 중심으로 1화소 간격으로 위치한 8개의 탐색 후보점에 대해 최소 BDM를 구하고 최종 동작 벡터로 결정 한다.

(그림 2)는 TSS 알고리즘으로 탐색 패턴에 따라 움직임 벡터를 구하는 것을 나타내고 있다. 움직임 벡터를 찾기 위한 3단계 알고리즘을 간략하게 설명하면 다음과 같다.

[단계 1]에서 탐색영역의 중심점에서 4화소 간격의 9개의 탐색후보점에서 최소 BDM를 구한다(0, 0).

[단계 2]에서는 [단계 1]에서 구한 최소 BDM(0, 0)를 중심으로 2화소 간격의 8개의 탐색후보점을 추가하여 최소 BDM를 구한다(2, 2).

[단계 3]에서는 [단계 2]에서 구한 최소 BDM (2, 2)를 중심으로 1화소 간격에서 8개의 탐색 후보점을 추가하여 최소 BDM을 구하고 이때의 동작 벡터(3, 3)를 최종 움직임 벡터로 결정한다.

2.2 새로운 3단계 탐색(NTSS : New Three Step Search)

NTSS 알고리즘은 TSS 알고리즘을 개선한 알고리즘으로 동작벡터가 탐색영역의 중심에 분포한다는 사실을 이용하여 3단계 탐색기법을 보완한 방식으로 정적블록과 동적블록의 빠른 식별을 위해 중간-정지 기술을 이용하였다. 3단계 탐색 기법을 보안한 방식으로 움직임 벡터가 중심점을 중심으로 중심점에서 발견되면 탐색을 중지하고, 만일 최소 블록 정합 오차가 탐색 블록 중심점의 이웃에 위치하면 8개의 탐색 후보점을 추가하여 정합오차를 계산하고 탐색을 중지한다. 또한 중심점의 이웃에도 위치하지 않으면 단계 2와 단계 3을 수행한다. 따라서 1단계에서는 총 17개의 탐색 후보점을 탐색하며, 그렇지 않으면 단계별로 2단계, 3단계를 수행하며 (그림 3)은 NTSS 알고리즘을 나타낸다.

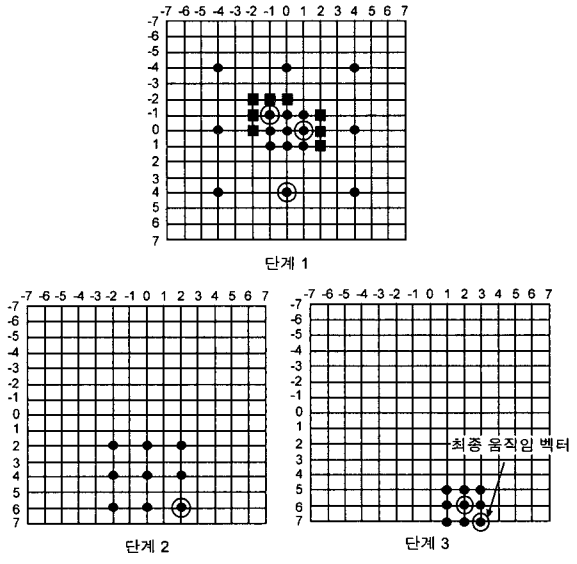
[단계 1] 17개의 탐색 후보 점에 대해서 최소 BDM을 계산하여 최소 BDM이 탐색 영역의 중심점에서 발견되면 그 점을 동작 벡터로 결정하고 탐색을 중지하고, 만일 최소 BDM이 탐색 영역의 중심점의 8개의 이웃점에 있으면 탐색 점 3혹은 5을 추가 하여 최소 BDM 값을 구하고 그 점을 동작 벡터로 결정하고 탐색을 중지한다.

[단계 2] 단계 1에서 구한 최소 BDM을 갖는 탐색 후보점을 중심점의 중심으로 2 화소 간격으로 위치한 8개의 탐색 후보점에 대해 최소 BDM를 구한다.

[단계 3] 단계 2에서 구한 최소 BDM를 갖는 탐색 후보점을 중심점의 중심으로 1화소 간격으로 위치한 8개의 탐색 후보점에서 최소 BDM 구하고 최종 동작 벡터로 결정하고 탐색을 중지한다.

(그림 3)은 움직임 벡터를 찾기 위한 탐색으로 간략하게 설명하면 다음과 같다.

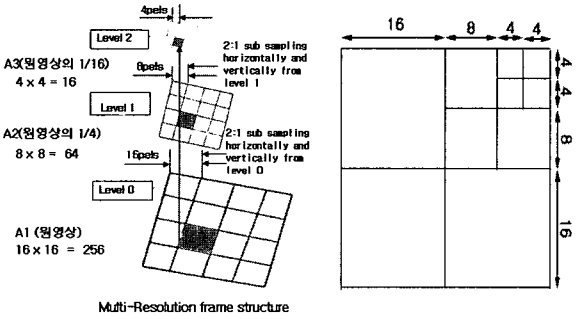
- **[단계 1]**에서 탐색영역의 중심점에서 최소 BDM를 구한 경우 움직임벡터가 (0, 0)에 있을 경우는 움직임 벡터로 최종움직임 벡터로 결정하고 중지한다. 그렇지 않은 경우 탐색점이 탐색영역의 중심으로 중심점의 8개의 이웃 모서리에 위치(-1, -1)한 경우 5개의 탐색점을 추가하여 움직임 벡터로 결정하고 중지한다. 그렇지 않은 경우 중심점을 중심으로 (그림 3)에서 보는 것처럼 오른쪽(0, 1)에 위치한 경우로 중심점에 이웃한 3개의 탐색점을 추가하여 탐색한다. 그리고 중심점으로부터 (0, 4)의 위치에서 최소 BDM이 발견될 경우 최소 BDM을 구한다.
- **[단계 2]**에서는(0, 4)를 중심으로 2화소 간격에서 최소BDM (2, 6)를 구한다. 단계 3에서는 단계 2에서 구한 최소 BDM (2, 6)를 중심으로 1화소 간격에서 최소 BDM(3, 7)을 구하고 최종 동작벡터로 결정한다.



(그림 3) 새로운 3단계 탐색 기법(NTSS)

3. 다중해상도 알고리즘(MR : Multi-Resolution)

Zafer와 **Zhang**이 제한한 **Multiresolution** 알고리즘으로 해상도가 낮은 상위 단계, 중간 단계, 하위 단계 구분하여 움직임 벡터를 구하고 움직임을 예측하여 보상하는 기법을 제안하였다[4]. 다중해상도 **MR**(Multi-Resolution)알고리즘은 몇 가지 특징을 가지고 있다. 첫째 상위 단계로 올라갈수록 공간 해상도(MR)가 낮아지는 **Mean Pyramid**을 생성한다. 둘째 상위벡터에서 찾은 움직임 벡터를 중심으로 작은 범위의 움직임을 예측하여 하위 단계를 수행하여 하위 단계에서 움직임 벡터를 찾는다. 셋째 연산량을 크게 감소시키지만 국부 국소에 의한 성능 저하를 초래하는 우려가 있다. 이와 같은 특징들을 가지고 있으며, 이미지를 해상도에 탐색을 함으로써 Complexity를 줄 있 수 있다. (그림 4)는 다중해상도(MR : Multi-resolution)를 나타낸 것으로 레벨에 따라 해상도가 다르며, 해상도가 낮은 순서대로 상위, 중간, 하위 단계로 각각의 해상도를 나타낸다. 이들 각각의 해상도는 이미지 크기를 나타낸다. 이들 각각의 이미지 사이즈는 레벨에 따라 다르게 이미지 처리를 한다[18].



(그림 4) 다중해상도(MR : Multi Resolution)

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 4)는 다중해상도를 나타낸 것으로 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 최소 정합오차로서 움직임벡터를 찾는다. 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 이미지해상도가 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 좋아진다.

4. TSS-3 레벨 알고리즘과 제안하는 NTSS-3 레벨 알고리즘의 설계 및 구현

4.1 TSS-3 Level 알고리즘 설계 및 구현

TSS-3 레벨 알고리즘은 기존 TSS 알고리즘을 다음과 같이 설계하였다. 기존 TSS 알고리즘의 단계 1을 레벨 2에 적용시키고, 단계 2는 레벨 1에 적용시고, 단계 3은 레벨 0에 적용시키면서 패턴에 따른 알고리즘으로 다중해상도 3레벨을 적용하여 움직임 벡터를 구하는 것이다. 따라서 레벨 2에서는 1/4로 축소된 1픽셀 간격의 9개의 탐색점에서 움직임 벡터를 찾고, 레벨 1에서는 레벨 2에서 찾은 움직임 벡터를 중심으로 1/2 축소된 1픽셀 간격의 8개의 탐색점을 추가하여 움직임 벡터를 찾는다. 레벨 0에서는 레벨 1에서 찾은 움직임 벡터를 중심으로 8개의 탐색점을 추가하여 움직임 벡터를 찾고 최종 움직임 벡터로 결정한다. 해상도에 따른 레벨을 적용하기 때문에 각각의 레벨에서 탐색점의 수가 달라짐으로 이미지 처리가 달라지고 따라서 움직임 벡터를 찾기 위해 탐색하는 탐색점의 수도 레벨에 따라 각각 다르게 수행된다. 따라서 수행되는 연산량의 처리도 각각 다르게 된다. 그리하여 기존 알고리즘과 다른 연산량으로 처리 된다. 이미 2장에서 이론적인 계산량과 계산 복잡도를 계산식으로 나타내어 TSS 알고리즘을 TSS-3 레벨로 설계한 계산 결과를 나타내었고 그 결과를 비교하여 제시하였다. 또한 3장에서 다중해상도 알고리즘에 대한 각 레벨에서의 해상도에 따른 이미지 사이즈를 (그림 4)로 나타내었고 다음과 같이 TSS-3 레벨 알고리즘 설계를 나타내었다.

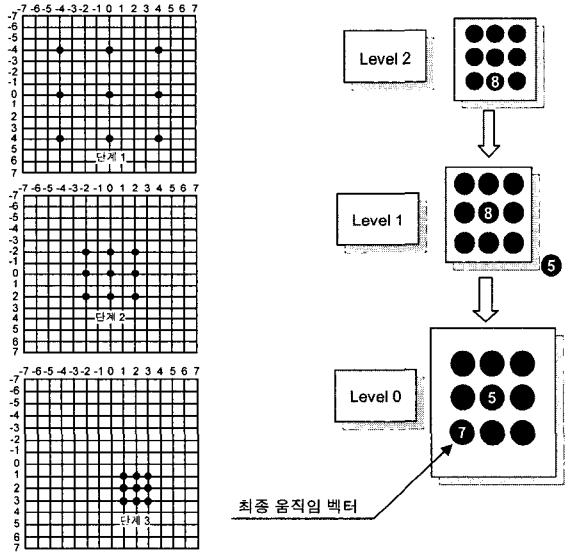
[레벨 2] 탐색 영역에서 중심점을 중심으로 1픽셀 간격의 9개의 탐색 후보 점에서 최소 BDM를 구한다.

[레벨 1] 레벨 2에서 구한 최소 BDM을 중심점으로 하여 이웃하는 1픽셀 간격의 8개의 탐색 후보점을 추가 하여 최소 BDM을 구한다.

[레벨 0] 레벨 1에서 구한 최소 BDM을 중심점으로 하여 이웃하는 1픽셀 간격의 8개의 탐색 후보점을 추가하여 최소 BDM을 구하고 최종 동작 벡터로 결정한다.

(그림 5)는 TSS 알고리즘을 TSS-3 레벨로 설계한 알고리즘에 대한 설명으로 왼쪽 그림은 기존 TSS 알고리즘으로서 단계별 탐색점에서 움직임 벡터를 구하는 것이고, 오른쪽의 그림은 TSS-3 벨 알고리즘으로 이미지의 해상도에 따라 레벨별로 설계한 것이다. **[레벨 2]**는 탐색 영역에서 중심점

을 중심으로 최소 BDM❽를 구한 것이다. 기존 TSS 알고리즘의 1단계를 적용한 것으로 TSS의 4픽셀 간격의 9개의 탐색점의 크기가 1/4 축소된 것으로 1픽셀 간격의 9개의 탐색점에서 최소 움직임 벡터를 구하는 것이다. **[레벨 1]**은 기존 TSS 알고리즘의 2단계를 적용한 것으로 TSS의 2픽셀 간격의 9개의 탐색점의 크기가 1/2 축소된 상태에서 1픽셀 간격의 9개의 탐색점에서 최소 움직임 벡터를 구하는 것으로 **[레벨 2]**에서 구한 최소 BDM❽를 중심으로 1화소간격의 8개의 탐색후보점을 추가하여 최소 BDM❺를 구한다. **[레벨 0]**는 기존 TSS 알고리즘의 3단계를 적용한 것으로 TSS의 1픽셀 간격의 9개의 탐색점의 크기가 원영상인 1픽셀 간격의 9개의 탐색점에서 최소 움직임 벡터를 찾는 것으로 **[레벨 1]**에서 구한 최소 BDM❺를 중심점으로 하여 이웃하는 1픽셀 간격의 8개의 탐색 후보점을 추가하여 최소 BDM❼를 구하고 최종 움직임 벡터로 결정한다. TSS-3 레벨은 이미지 해상도에 따라 레벨을 적용한 것으로 각각의 레벨에서 움직임 벡터를 구하여 다음 레벨에서 적용한 것으로 레벨에 따라 이미지의 크기를 달리함으로 탐색하는 탐색점 수를 각각 레벨에 따라 다르게 처리하고 연산량의 결과도 달라짐으로 계산 복잡도를 줄일 수 있다.



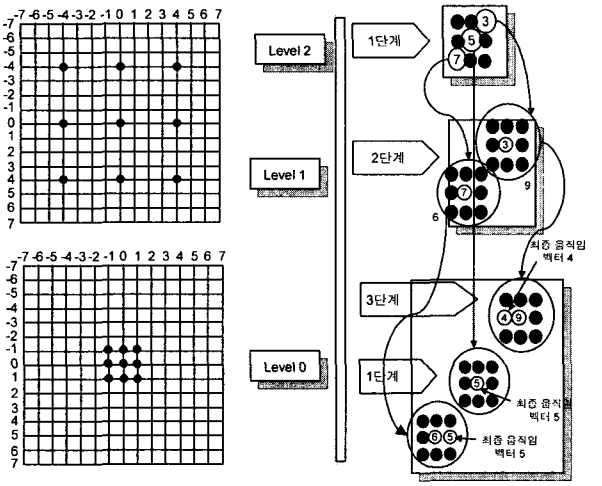
(그림 5) TSS 알고리즘을 TSS-3 알고리즘으로 설계

4.2 제안한 NTSS-3 Level 알고리즘의 설계 및 구현

본 장에서는 기존 NTSS 알고리즘을 NTSS-3 레벨 알고리즘으로 다음과 같이 설계하여 제안하였다. NTSS-3 레벨 알고리즘은 기존 NTSS 알고리즘의 특성을 이용하여 레벨 2에 기존 NTSS 알고리즘의 단계 1를 적용시켜 레벨에 따른 해상도를 적용한 것으로 중간-정지 기술을 이용하였다. 움직임 벡터가 중심점을 중심으로 중심점에서 발견되면 탐색을 중지하고, 만일 최소 블록 정합 오차가 탐색 블록 중심점의 이웃점이면 최소 블록 정합 오차 8개의 이웃점에 대해서 정합오차를 계산하고 탐색을 중지한다. 따라서 레벨 2에서는 1단계로서 총 17개의 탐색 후보점을 탐색하여 움직임 벡터를 구하는 것으로 탐색점의 중심점에 움직임벡터가 위치하면 최종 움직임 벡터로 결정하고 중지한다. 레벨 2로서 1단계를 만족하지 않으면 레벨 1의 2단계를 수행하여 움직임벡터를 구하고 다음 단계인 3단계를 수행하여 최종적인 움직임 벡터를 구하게 된다. NTSS-3 레벨은 1단계에서 17개의 탐색점을 탐색하는데 기존 NTSS의 4화소 간격의 9개 탐색점과 1화소 간격의 8개의 탐색점을 동시에 탐색하여 움직임 벡터를 구하는 것인데 이들 탐색점에서 움직임 벡터를 구하기 위해 먼저 레벨 2에서 4화소 간격의 9개의 탐색점에서 움직임 벡터를 구한다. 레벨 2는 4화소 간격이 1/4로 축소된 1화소 간격의 9개의 탐색점으로 된 것으로 레벨 2에서 움직임 벡터를 구하게 된다. 또한 1화소 간격의 8개의 탐색점은 레벨 2에서 더 이상 축소할 수 없으므로 원영상인 레벨 0에서 1화소 간격의 8개의 탐색점을 탐색하게 된다. 이렇게 4 화소 간격이 1/4로 축소된 1화소 간격의 9개의 탐색점과 1화소 간격의 8개의 탐색점을 레벨 2와 레벨 0에서 1단계로서 동시에 17개의 탐색점을 탐색하여 움직임 벡터를 찾도록 설계하였다. 1단계에서 중심점에서 움직임 벡터를 찾게 되면 최종 움직임 벡터로 결정하고 중지한다. 그러나 1단계에서 움직임벡터가 중심점에 위치하지 않으면 다음 2단계인 레벨 2에서 1/2 축소된 1픽셀 간격의 8개의 탐색점을 추가하여 움직임 벡터를 구한다. 그리고 3단계인 레벨 0에서 8개의 탐색점을 추가하여 움직임 벡터를 구하고 최종 움직임 벡터로 결정한다. NTSS-3 레벨은 해상도에 따라 이미지를 적용하지만 중간-정지 기술을 이용하기 때문에 최적의 경우 탐색하는 탐색점이 적어 연산량이 기존의 알고리즘에 비해 훨씬 적게 탐색 하게 된다. 따라서 최적의 경우 계산량과 복잡도가 매우 우수하며 다음과 같이 제안한 NTSS-3 레벨 알고리즘 설계를 나타내었다.

① 레벨 2에서 중심점을 중심으로 9개 탐색 후보 점에서 최소 BDM을 구한다.
② 레벨 0에서 중심점을 중심으로 8개의 탐색 후보점에서 최소 BDM을 구한다.
③ ①, ②에서 구한 최소 BDM이 중심점에서 발견되면 그 점을 동작벡터로 결정하고 탐색을 중지한다.
④ 만일 최소 BDM이 중심점의 8개의 이웃에 있으면 3 혹은 5의 탐색 후보 점을 추가하여 탐색하고 그 점을 동작 벡터로 결정하여 중지한다.
⑤ ③과 ④을 만족하지 않으면 2배 확대한 레벨 1에 8개의 탐색 후보 점을 추가하여 최소 BDM을 구한다.
⑥ ⑤에서 구한 최소 BDM에 2배 확대한 레벨 0에 8개의 탐색 후보 점을 추가하여 최소 BDM을 구하고 최종 움직임 벡터로서 결정한다.

(그림 6)은 NTSS-3 Level 알고리즘 설계하여 나타낸 것으로 **[1단계]**에서 레벨 2와 레벨 0를 동시에 수행하여 17개의 탐색점에서 최소 BDM을 구하는 것으로 레벨2에서 최소 BDM이 중심점의 중심에 있으면 그 점을 동작벡터로 결정하고 중지한다. (그림 6)은 레벨 2에서 **[1단계]** 화살표가 가리키는 9개의 탐색점에서 최소 BDM⑤가 레벨 0의 8개의 중심점에 위치하면 최소 BDM으로 움직임 벡터로 결정하여 중지한다.



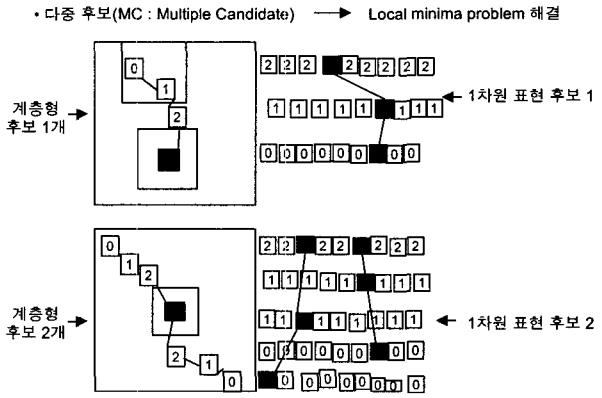
1단계 17개 탐색점 수행

(그림 6) NTSS-3 Level 알고리즘 설계

그렇지 않고 ③과 ⑦인 최소 BDM인 경우는 이것을 최소 BDM으로 결정하고 [2단계]을 수행한다. [2단계]에서는 1단계에서 구한 최소 BDM(③,⑦)인 경우 2단계에서 ③을 중심점을 중심으로하여 8개의 탐색점을 추가하여 최소 BDM를 구한다. 마찬가지로 ⑦를 중심점을 중심으로 8개의 탐색점을 추가하여 최소 BDM를 구한다. 다음 단계인 [3단계]에서 2단계에서 최소 BDM 구한 값(⑨, ⑥)을 중심으로 하여 8개의 탐색점을 추가하여 다시 최소 BDM을 구한다. 여기서는 ④, ⑤가 최소 BDM으로 최종 움직임 벡터로 결정하고 중지하는 것을 나타낸 것이다. 설계한 NTSS-3 레벨 알고리즘은 1단계에서 레벨 2와 레벨 0을 동시에 수행하여 17개의 탐색점에서 움직임 벡터를 찾는 것으로 움직임 벡터가 중심점에 위치하면 최종적인 움직벡터로 결정하고 정지한다. 이것은 NTSS 알고리즘의 중간-정지 기술을 이용한 것이며 다중해상도의 레벨에 따른 이미지 사이즈를 적용하여 최적의 탐색을 하게한다. 따라서 NTSS-3 레벨 알고리즘은 1단계에서 움직임 벡터를 구하는 경우 최적의 경우로서 연산량과 계산 복잡도에 있어서 매우 좋은 성능을 나타내었다.

4.3 제안한 NTSS-3 Level 알고리즘에 다중후보(MC)를 사용하여 화질 개선

고속 정합 알고리즘은 계산량을 줄이기 위해 탐색 영역에 포함되는 특정한 패턴들의 몇몇 탐색 점들만 조사하여 움직임 벡터를 찾기 때문에 국부(Local minima Problem)적인 탐색을 하게 되거나, 일부 탐색 점들을 블록 정합 대상에서 제외시키므로 복원된 영상의 화질 저하를 초래 한다.따라서 블록 정합 알고리즘에서 이미지 복원 시 발생하는 화질 저하 문제를 제시하고 개선 하고자 한다. 화질 저하의 첫 번째 문제는 국부적 최소화를 제안한다. 블록정합 알고리즘에서 움직임 벡터를 구하기 위하여 탐색 패턴에서 일부 탐색점들을 제외시키므로 국부적 최소(local minimum)문제가 발생한다. 또 다른 문제로서 국부 최소화에서 일부 탐색점들을 탐색하여 최소 BDM을 구하였을 경우 다음 단계에서도 최소 BDM으로 결정하여 움직임 벡터를 구할 때 발생한다. 반드시 항상 최소 BMD이라고 할 수 없다. 탐색 영역의 위치에 따라 다른 탐색점도 최소 BDM이 될 수도 있다. 이들 제시한 문제를 해결하기 위하여 다중후보(MC : Multiple Candidates) 사용을 제안한다.



(그림 7) 다중 후보(MC : Multiple Candidate)

(그림 7)은 다중 후보(MC : Multiple Candidate) 사용을 나타낸 것으로 단일 후보의 경우와 다중 후보의 경우로 1차원 구조와 계층형구조로 경우로 나타낸것이다. 또한 다중후보 사용의 일례로 (그림 6)을 제시한다. (그림 6)에서 1단계에서 최소 움직임 벡터를 찾기 위해 탐색 후보로서 ③, ⑦을 최소 BDM을 갖는 다중후보(MC : Multiple Candidates)로 선택한다. 이렇게 레벨에서 다중 후보를 선택하여 최소 BDM를 구한다. 다중 후보를 사용하여 최소 BDM을 구하기 때문에 가장 최선의 BDM을 구할 수 있게 된다. 본 논문에서 NTSS-3레벨 알고리즘을 설계하고 구현하여 그 결과를 나타내었는데 제안한 NTSS-3 레벨 알고리즘이 이론적으로는 계산량이나 복잡도에서 우수함을 나타내었고 설계하고 구현한 결과 MAD, MSE, SearchPoint는 기존 TSS나 NTSS보다 좋은 결과를 나타냈지만 PSNR에서는 다소 화질저하를 나타내었다[20]. 그래서 이 문제를 해결하기 위해 다중의 후보를 사용하여 결과를 구현했는데 <표 2>와 같이 각각 이미지에 대한 다중 후보(MC)결과를 PSNR로 나타내었다.

5. 실험 결과

본 논문에서의 구현 방안은 기존 TSS 알고리즘을 TSS-3 레벨 알고리즘으로 그리고 NTSS 알고리즘을 NTSS-3 레벨알고리즘으로 다중해상도 기법을 적용하여 탐색점을 적게 탐색하면서도 탐색 점 대비 화질 개선을 설계하고 구현하였다. 기존 TSS 알고리즘은 원영상인 16×16 블록에서 단계별로 탐색하여 탐색점의 수와 계산 복잡도가 높았던 반면 TSS-3 레벨 알고리즘, 혹은 NTSS-3 레벨 알고리즘은 레벨에 따라 해상도를 달리함으로 연산량의 처리에 좋은 결과를 나타내었다. 레벨 2에서는 해상도가 원영상의 1/4로 축소된 이미지사이즈 4×4 화소에서 이미지을 탐색함으로 탐색점의 수를 적게 탐색하여 계산 복잡도를 1/4로 줄이는 방안으로 설계하였다. 레벨1에서는 이미지 사이즈를 원영상의 1/2 크기에서 탐색함으로 탐색점과 계산복잡도를 줄일 수 있었다. 2장에서 계산식을 이론적으로 나타낸 결과에서 제안한 NTSS-3 레벨이 탐색점의 수를 적게 탐색함으로 계산 복잡도를 줄일 수 있었으나 복원시 화질 저하의 문제가 발생함으로 이 문제를 해결하기 위한 방안으로 <표 2>와 같이 다중 후보를 사용하여 문제를 해결하였다. 블록 정합 알고리즘의 단계별 탐색 알고리즘을 다중 후보를 사용하여 복원시 화질 개선을 나타내는 탐색점 대비 성능 향상 방안으로 설계하고 구현하였다. 구현 방안으로 다중후보를 사용하여 4장에서 알고리즘 설계와 구현으로 기존 TSS 알고리즘을 TSS-3 레벨 알고리즘으로 NTSS 알고리즘을 NTSS-3 레벨 알고리즘보다 개선된 탐색점의 수와 PSNR를 <표 1>과 <표 2>로 나타내었다. 제안한 NTSS-3 레벨 기법의 성능을 평가하기 위하여 CIF(Common Intermediate Format : 352×288 pixels)의 akiyo, foreman, news, stefan 등의 4개 영상에 대해 각각 100프레임씩을 대상으로 실험하였고, 비교 탐색 기법으로는 FS와 TSS, NTSS, TSS-3 레벨과 제안한 NTSS-3 레벨로 <표 1>로 나타내었고, 다중 후보를 사용하여 국부최소화 문제를 다중후보(MC)를 사용하여 NTSS MC로서 <표 2>로 나타내었다. 그리고 움직임 예측에 사용된 매크로블록의 크기는 16×16 화소이며, 탐색 영역의 변위는 ±8를 적용하여 Pentium Ⅳ 1.6GHz와 256MB 메모리가 장착된 컴퓨터상에서 실험을 수행하였다. 블록 정합의 정도를 평가하기 위해 대표적인 정합 기준인 평가 함수(cost function)로 영상 화질의 품질을 평가하기 위한 식 (13)의 평균 제곱 오차(MSE : Mean Squared Error), 식 (14)의 평균 절대 값 오차(MAD : Mean Absolute Difference)와 정합 오차 측정 함수로는 식 (15)의 절대 값 오차의 합(SAD : Sum of Absolute Difference)을 이용하였다. 또한 제안하는 기법의 성능 향상을 측정하기 위해 블록 당 탐색점의 개수를 기존 방법들과 비교하였다.

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

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

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

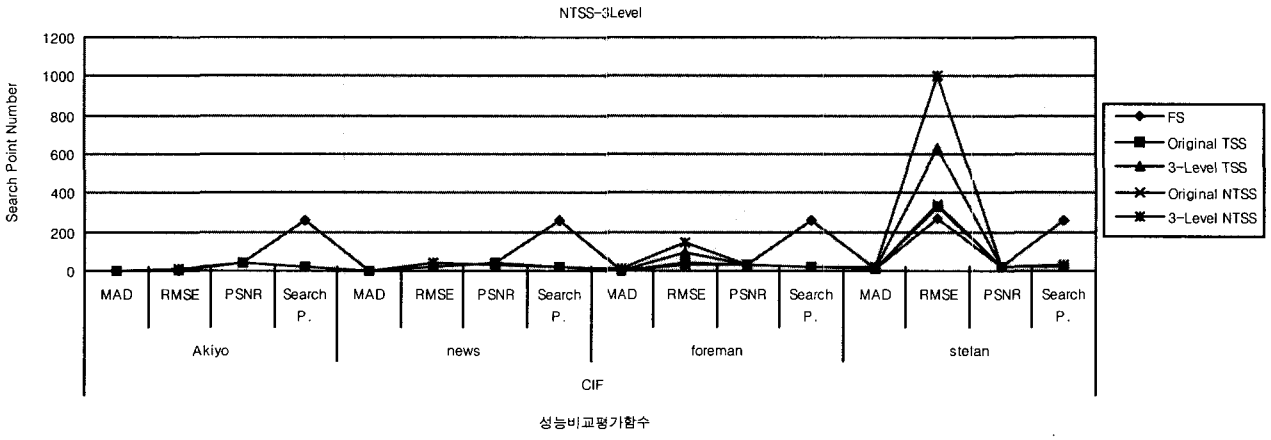
여기서 N은 영상의 가로와 세로의 각각의 크기이며, $I_t(k,l)$은 원영상의 화면을 나타내고, $I_{t-1}(k+i,l+j)$은 움직임 예측 화면을 나타내며, 이들 정합 기준들은 최소 값을 가지는 위치를 움직임 벡터로 결정하였다. 그리고 화질의 평가를 위한 PSNR은 식 (16)과 같다.

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

실험 영상에 대한 실험 결과는 <표 1>로 CIF 영상에 대해 FS, Original TSS, 3-Level TSS, Original NTSS, 3-Level NTSS 알고리즘을 적용하여 MAD, RMSE, PSNR, Search Point를 비교하였다. 이 결과에서 Akiyo 이미지에 대한 분석 결과 MAD에서는 FS가 0.65를 나타내는데 Original NTSS가 0.605를 나타내었고, 3-Level TSS는 0.732, 3-Level NTSS는 0.704로 NTSS-3 Level이 TSS 3-Level보다는 좋은 결과를 나타낸 것을 볼 수 있다. 그리고 RMSE에서는 FS가 3.943을 나타내고 다음으로는 NTSS, TSS, NTSS-3 Level, TSS-3 Level 순으로 결과를 나타냈다. 다음은 PSNR을 분석해 보면 FS가 42.806를 나타내고, 다음으로 좋은 것은 NTSS, TSS, NTSS-3 Level, TSS-3 Level 순으로 나타내었다. 다음은 Search Point에서는 FS가 262를 나타냈고, 제일 좋은 것은 NTSS, NTSS-3 Level, TSS-3 Level, TSS순으로 나타났다. 다른 이미지에서도 NTSS-3Level이 비교적 좋은 결과를 나타냈다. 다음은 <표 2>에 대한 실험 결과 분석으로 제안한 NTSS-3 Level에 MC를 적용한 결과이다. 먼저 akiyo 이미지에 대한 분석을 적용한 결과를 나타낸다. FS의 MAD는 0.65이며, NTSS도 0.605를 나타내었으며 다중후보를 적용한 NTSS-3 Level MC에서는 0.61를 후보 1부터 6까지 모두 같았다. 그리고 RMSE에서도 FS가 3.943일때 NTSS, NTSS-3 Level MC 모두도 3.943를 나타내었다. 다음은 PSNR에 있어서 FS가 42.806일 때 NTSS는 42.805인데 비해서 NTSS-3 Level MC(3)는 42.806를 나타내어 더 좋은 결과를 나타내었다. 다음은 Search Point에서 FS는 262를 나타내고, NTSS, NTSS-3 Level도 16을 나타내었다.

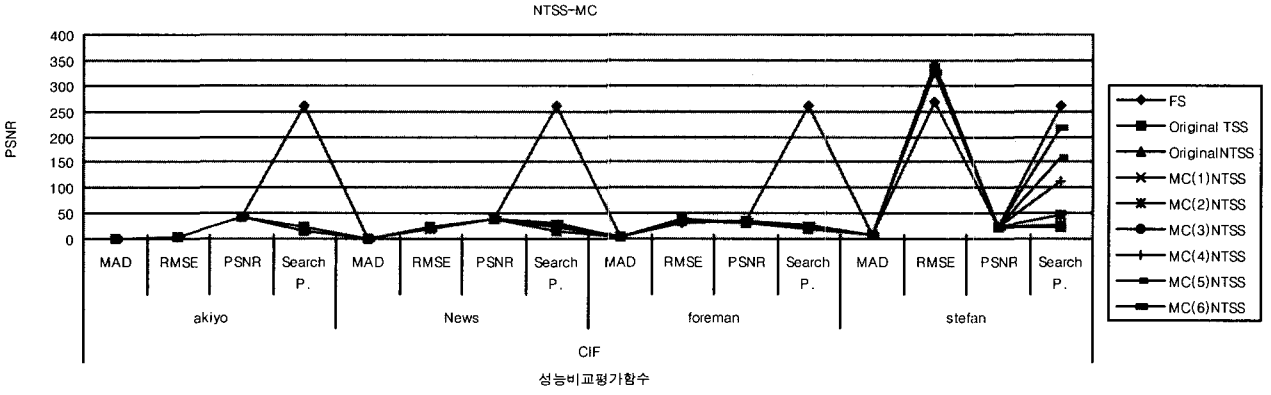
〈표 1〉 각 실험 영상에 대한 성능비교평가 함수 결과 비교값 NTSS-3 Level

Sequence	CIF															
	Akiyo				news				foreman				stefan			
Method	MAD	RMSE	PSNR	Search P.	MAD	RMSE	PSNR	Search P.	MAD	RMSE	PSNR	Search P.	MAD	RMSE	PSNR	Search P.
FS	0.65	3.943	42.806	262	1.212	20.335	37.311	262	2.885	32.667	33.27	262	7.841	268.154	24.941	262
Original TSS	0.61	4.078	42.683	23	1.229	21.599	37.16	23	3.138	38.255	32.591	23	8.689	332.104	24.295	23
3-Level TSS	0.732	8.65	40.49	21	1.542	36.85	34.736	23	4.9	97.658	28.572	23	12.88	627.337	21.454	23
Original NTSS	0.605	3.943	42.805	16	1.223	21.563	37.178	17	2.97	35.233	32.988	18	8.778	340.126	24.238	26
3-Level NTSS	0.704	7.933	40.63	17	1.529	41.368	34.701	19	5.531	140.85	27.069	20	16.78	1003.67	19.456	28



〈표 2〉 각 실험 영상에 대한 성능비교평가 함수 결과 비교값 NTSS-MC

Sequence	CIF															
	akiyo				News				foreman				stefan			
Method	MAD	RMSE	PSNR	Search P.	MAD	RMSE	PSNR	Search P.	MAD	RMSE	PSNR	Search P.	MAD	RMSE	PSNR	Search P.
FS	0.61	3.94	42.81	262	1.21	20.34	37.31	262	2.89	32.67	33.27	262	7.84	268.15	24.94	262
Original TSS	0.61	4.08	42.68	23	1.23	21.60	37.16	23	3.14	38.26	32.59	23	8.69	332.10	24.30	23
OriginalNTSS	0.61	3.94	42.81	16	1.22	21.56	37.18	17	2.97	35.23	32.99	18	8.78	340.13	24.24	26
MC(1)NTSS	0.61	3.94	42.81	16	1.22	21.56	37.18	16	2.97	35.23	32.99	18	8.78	340.13	24.24	26
MC(2)NTSS	0.61	3.94	42.81	16	1.22	21.34	37.19	16	2.95	34.88	33.02	19	8.67	331.52	24.31	45
MC(3)NTSS	0.61	3.94	42.81	16	1.22	21.32	37.19	16	2.95	34.77	33.04	20	8.63	328.38	24.34	45
MC(4)NTSS	0.61	3.94	42.81	16	1.22	21.15	37.20	23	2.95	34.75	33.04	22	8.61	326.42	24.36	112
MC(5)NTSS	0.61	3.94	42.81	16	1.22	21.15	37.20	27	2.95	34.75	33.04	25	8.59	325.46	24.36	160
MC(6)NTSS	0.61	3.94	42.81	16	1.22	21.15	37.20	31	2.95	34.73	33.04	28	8.59	324.91	24.37	218



다른 이미지에서는 News에서는 제안한 NTSS-3 Level MC(3)에서 PSNR이 더 높고, MAD를 제안된 기법을 FS을 비교해 보면 akiyo, news, RMSE의 수치가 더 낮게 나와 더 좋은 것을 나타내었고, 탐색점의 개수에서도 더 적은 결과를 나타내었다. foreman, stefan 모두 탐색점이 262점인데 비해 제안한 기법의 탐색수는 16~26점으로 약 16배가 빠른 탐색을 함을 볼 수 있다. 영상 화질을 평가하기 위한 비교 평가 함수로서 사용된 MAD에서는 0.020.11, PSNR값에서는 0.11~0.12[dB] 화질이 개선되었고 속도 면에서도 0.1배 향상되었다. 결론적으로 제안한 NTSS-3 Level이 탐색점을 낮추면서 좋은 화질을 얻을 수 있는 것을 볼 수 있다.

그래서 Local Minima Problem를 해결함으로서 블록 정합의 문제점인 계산 복잡도 를 좀 더 빠르게 하였고, 탐색점은 적게 찾으면서 탐색점 대비 높은 화질을 유지하는 개선을 나타내는 알고리즘을 제안하게 되었다. 실험 결과 이론적으로 제안한 NTSS-3Level에 대한 계산 복잡도가 탐색점 대비 우수함을 나타내었고, PSNR에 있어서도 다중후보를 사용하여 좋은 결과를 나타내었다.

6. 결 론

본 논문에서는 기존 3단계 탐색 알고리즘(TSS : Three Step Search)과 새로운 3단계 탐색 알고리즘(NTSS : New Three Step Search)을 TSS-3 레벨 알고리즘과 NTSS-3 레벨 알고리즘으로 다중해상도 알고리즘으로 설계하고 구현하여 적은 탐색점을 탐색하면서도 탐색점 대비 화질 개선을 제안하였다. 결과로서 이론적인 계산식으로 나타냈으며 NTSS-3 레벨 알고리즘으로 설계하여 구현하였다. 또한 블록정합 알고리즘에서 국부 최소화 문제로 화질 저하를 초래하는 문제를 개선하기 위해 MC라는 다중후보를 사용하여 Local minima problem를 해결하여 화질을 개선을 제안하였다. 이론적으로 NTSS-3 레벨 알고리즘이 우수하였으나 복원시 화질저하가 나타남을 결과로 알 수 있었고 다중 후보 사용으로 화질 개선을 나타내었다. 실험 결과 제안한 기법을 FS와 비교하면 16배의 탐색 속도를 나타내었고 기존 고속 블록 정합 알고리즘인 NTSS 방식에 비교 할 때 PSNR 값에 있어서는 0.11~0.12[dB] 화질이 개선되었으며 속도 면에서도 0.1배 향상되었고 탐색점 대비 화질개선이 우수함을 나타내었고 탐색수와 화질 개선을 위해 더 많은 연구의 보강이 필요된다.

참 고 문 헌

[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 for estimation 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.

[14] Jae Hun Lee, Kyoung Won Lim, Byung Cheol Song, Jong Beom Ra, "A Fast Multi-Resolution Block Matching Algorithm and its LIS Architecture for Low Bit-Rate Video Coding" IEEE Transactions on Circuits and Systems for Video Technology, Vol.1, No.12, pp.1289-1301, December, 2001.

[15] K. M. Chun and J. B. Ra, "An improved block matching algorithm based on successive refinements of motion vector candidates," Signal Processing : Image Commun., Vol.6, pp.115-122, 1993.

[16] J. C.-H. Ju, Y.-K. Chen, and S. Y. Kung, "A fast rate-optimized motion estimation algorithm for low-bit-rate video coding," IEEE Trans. Circuits Syst. Video Technol, Vol.9, pp.994-1002, Oct., 1999.

[17] X. Song, T. Chiang, X. Lee, and Y.-Q. Zhang, "New fast binary pyramid motion estimation for MPEG2 and HDTV encoding," IEEE Trans. Circuits Syst. Video Telenol., Vol. 10. pp.1015-1028, Oct., 2000.

[18] ie Wei, Ze-Nian Li, "An Enhancement MRMC Sheme in Video Compression," IEEE Transactions on Circuits and Systems for Video Technology, Vol.7, No.3, pp.564-568, June, 1997.

[19] Jinwen Zan, M.Omair Ahmad, M.N.Swany, "New Techniques for Multi-Resolution Motion Estimation," IEEE Transactions on Circuits and Systems for Video Technology Vol.12, No.10, pp.934-947, October, 2002.

[20] Christopoulos, V., Cornelis, J, "A center-biased adaptive search algorithm for block motion stimation," IEEE Transactions on Circuits and Systems for Video Technology, Vol.10, Issue.3, pp.423-426, April, 2000.

[21] 남현우, 위영철, 김하진, "고속정합을 위한 납작한 육각패턴 기반탐색 알고리즘", 정보과학회 2003년 추계학술대회, Vol. 30 No.2-2, pp.215-216, Nov., 2001.

[22] 박인영, 남현우, 위영철, 김하진, "십자와 육각패턴을 이용한 고속 블록정합 동작예측기법", 정보처리학회논문지B, Vol. 10-B, No.07, pp.811-814, Dec., 2003.

[23] 남현우, 위영철, 김하진, "단위 다이아몬드와 납작한 육각패턴을 이용한 고속 블록 정합 알고리즘", 한국정보과학회논문지C, Vol.10-C, No.01, pp.0057-0065, Feb., 2004.

[24] 곽성근, 위영철, 김하진, "고속블록정합을 위한 수정된 다이아몬드 지역탐색 알고리즘", 정보과학회2004년 춘계학술대회 Vol.31, No.1, pp.886-888, Apr., 2004.

[25] 곽성근, 위영철, 김하진, "이전 프레임의 움직임 정보와 탐색 구간별 예측 후보점을 이용하는 블록정합", 한국정보과학회 논문지C, Vol.10, No.3, pp.273-281, Jun., 2004.

[26] 곽성근, 위영철, 김하진, "움직임 벡터 예측 후보들과 적응적인 패턴 탐색을 이용하는 블록정합 알고리즘", 정보처리학회 논문지B, Vol.11-B, No.3, pp.247-256, Jun., 2004.

주 헌 식

e-mail : hsjoo21014@syu.ac.kr

1992년 호서대학교 컴퓨터공학과(학사)

1994년 호서대학교 대학원 전자계산학과 (이학석사)

1998년 아주대학교 대학원 컴퓨터공학과 박사수료

1997년~현재 삼육의명대학 컴퓨터정보과 조교수

관심분야 : 컴퓨터그래픽스, 컴퓨터비젼, 영상처리