이전 화면의 블록합을 이용한 효율적인 연속 제거 알고리즘

*정동진 **홍주성 ***정제창

한양대학교 전자컴퓨터통신공학과

*dlying@naver.com **hong2982@gmail.com ***jjeong@ece.hanyang.ac.kr

Efficient Successive Elimination Algorithm using Previous Frame's sumnorm

*Jung, Dong-Jin **Hong, Joo-Seong ***Jeong, Je-Chang

Electronics and Computer Eng. Dept., Hanyang University

요약

본 논문은 비디오 압축 알고리즘 중 움직임 예측에 해당하는 논문이다. 이와 관련하여 FS와 같은 PSNR을 유지하면서 계산량을 줄이는 SEA, MSEA 알고리즘이 제안되었다. 본 논문은 SEA 와 MSEA와 같은 알고리즘에서 이전블록의 sumnorm을 가지고 현재블록의 합과 차이를 구하여서 낮은 순으로 탐색 지점을 탐색하는 방법을 제안한다. 이 방법으로 SADmin을 빨리 찾게 되서 후보 탐색지점들을 높은 확률로 제거함으로써 계산량을 줄이는 알고리즘을 제안한다.

1. 서론

H.264/AVC, HEVC와 같은 비디오 압축 표준에서는 움직임 추정을 통하여 프레임 간의 시간적 중복성을 제거해 효율적으로 압축을 한다. 그러나 이와 관련하여 하드웨어적 설계가 용이하고 이상적인 PSNR을 가지는 FS(Full Search) 알고리즘은 모든 탐색지점에 대해 블록 정합 매칭을 시행을 함으로써 막대한 계산량을 가져오는 결과를 초래한다. 그래서 PSNR을 FS와 같이 유지하되, 계산량을 줄 일 수 있는 알고리즘이 필요하게 되었다.

SEA[1] 알고리즘은 FS와 같은 PSNR을 유지하면서 계산량을 줄이는 알고리즘이다. 탐색지점에서 정합 매칭을 하기 전에 탐색 지점이 일정한 조건을 만족을 시키지 않으면 정합 매칭을 하지 않고 다음 탐색 지점으로 이동함으로써 정합 매칭에 따른 계산량 이득을 얻을 수 있다.

MSEA[2] 알고리즘은 SEA 알고리즘에서 탐색지점 조건을 다단계로 분할함으로써 보다 많은 조건들이 이용되고 다단계의 조건들이 보다 강화된 형태로 이용되기 때문에 탐색지점의 수를 SEA보다 더 줄일 수 있다.

본 논문에서는 위 알고리즘을 사용하고 추가적으로 이전 블록의 sumnorm을 이용하여 보다 빨리 최소 정합 오차(SAD)를 찾아 나머지 탐색지점들의 건너뛰기(Skip)확률을 높여 계산량을 줄인다.

본 논문의 구성은 다음과 같다. 2장에서 SEA와 MSEA에 대하여 간략히 설명하고 3장에서는 제안하는 알고리즘을 설명한다. 4장에서는 실험결과에 대해서 분석하고 마지막으로 5장에서는 결론을 맺는다.

2. 기존 알고리즘 소개

가. 연속제거알고리즘(SEA)

식 1은 수학적인 부등식으로

$$\left|\sum_{i=1}^{N}\sum_{j=1}^{N} f(i,j,t)\right| - \left|f(i-x,j-y,t-1)\right| \le \sum_{i=1}^{N}\sum_{j=1}^{N}\left|f(i,j,t)-f(i-x,j-y,t-1)\right|. \quad (1)$$

명확하게 참이 되는 것을 알 수 있다. SEA 알고리즘에서는 식 1을 변형하여 다음과 같은 식 2와 3을 만들었다.

$$-M(x,y) \le SAD(x,y) \quad (2)$$

$$M(x,y) - R \le SAD(x,y)$$

$$R - SAD(m,n) \le M(x,y) \le R + SAD(m,n) \quad (3)$$

여기서, R은 현재블록 화소 값들의 합이고, M은 이전 프레임의 정합 후보 블록 화소 값들의 합이다. SAD(m,n)은 현재블록과 이전 프레임의 정합 후보 블록 화소 값들 차이의 합이다. SEA는 식 3의 조건을 사용해서 이 조건을 충족하면, 블록정합 오차를 구하는 과정을 거치고, 충족하지 않으면, 다음 탐색지점으로 넘어간다. 결과적으로 식 3 조건을 사용하여서 탐색 지점을 줄일 수 있고 무손실로 압축이 가능하다.

나. 다단계 연속제거알고리즘(MSEA)

연속제거 알고리즘을 사용할 경우, 탐색지점을 없애기 위한 조건이 느슨하고 조건이 하나이기 때문에 많은 수의 탐색지점을 없앨 수가 없다. 이를 해결하기 위해서 다양한 조건과 보다 강화 된 조건을 가지고 탐색지점을 없앤 다단계 연속제거 알고리즘이 제안되었다. 단계 0에서 R0과 M0(m,n)은 현재 블록의 합과 이전 화면의 후보블록의 합을 나타낸다.

$$R_0 = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|f(i,j,t)\right| \quad (4)$$

$$m,n) = \sum_{i=0}^{-1N-1} \sum_{j=0} |F(i-m,j-n,t-1)| \quad (5)$$

식 (4), (5)를 각각 16x16블록으로 가정할 때, 단계 1은 단계 0의 블록에 대해서 4개의 서브블록으로 분할한 블록들의 합이다.

$$R_1^{(u,v)} = \sum_{i=0}^{\frac{N}{2}-1} \sum_{j=0}^{\frac{N}{2}-1} f(i+u\frac{N}{2},j+v\frac{N}{2},t) \quad (6)$$

$$M_1^{(u,v)}(m,n) = \sum_{i=0}^{\frac{N}{2}-1} \sum_{j=0}^{\frac{N}{2}-1} \left| f(i+u\frac{N}{2}-m,j+v\frac{N}{2}-n,t-1) \right| \quad (7)$$

위의 식에서 (4),(5),(6),(7)은 다음과 같은 식으로 요약 할 수 있다.

$$SSAD_0 = |R_0 - M_0| \quad (8)$$

$$SSAD_1 = \sum_{u=0}^{1} \sum_{v=0}^{1} |R_1^{(u,v)} - M_1^{(u,v)}| \quad (9)$$

식 (8)과 (9)에서 보면 알 수 있듯이 다음 식 (10)을 유도 할 수 있다.

$$SSAD_l \le SSAD_{l+1} \le SAD(m,n) \quad (10)$$

다단계 연속제거 알고리즘은 다음과 같이 l=0부터 최소 서브 블록이 2x2가 될 때까지 블록을 분할을 한다. 연속제거 알고리즘에서는 하나의 경계조건을 정할 수 없었지만 다단계 연속 제거 알고리즘에서는 보다 강화된 조건을 여러 단계에 걸쳐 이용할 수 있어서 탐색지점의 제거확률을 높힌다.

3. 제안하는 알고리즘

위의 식(3)을 보면 연속제거 알고리즘에 속도를 올릴 수 있는 요소가 크게 두 가지가 있다. 첫 번재로 다단계 연속제거 알고리즘처럼 식(3)의 조건을 부분블록으로 분할하면서 조건을 강화 시키는 것이다. 두 번째 방법은 SADmin을 최대한 빨리 찾아서 나머지 탐색 지점들을 제거하는 방법이 있다. 본 논문에서 제안하는 방법은 현재블록의 합과 이전 블록 sumnorm블록과의 차이를 이용해서 기존의 나선형 탐색이 아닌 R-M이 낮은 블록부터 탐색을 시작해서 보다 빨리 SADmin을 빨리 찾는 방법이다.

4. 실험결과

실험은 CIF(352x288)영상을 가지고 하였으며 Off codec에서 실험을 하였다.

ANSP	FOREMAN	STEFAN	DANCER	BUS	FOOTBALL
FS	1089	1089	1089	1089	1089
SEA	258	447	422	428	597
Proposed	237	403	375	399	528

표 1 제안하는 알고리즘과 기존 알고리즘들의 실험 결과

여기서 ASNP는 움직임 예측을 할 때, 탐색지점을 말한다. FS는 33x33 탐색을 기준으로 1089 지점을 탐색을 하고 SEA 알고리즘과 제안하는 알고리즘은 식 (3)의 조건을 만족하지 않으면, 현재 탐색지점을 넘어가 다음 지점을 검사한다. 위 실험결과를 보면 SEA 알고리즘은 FS와 비교하여 2 4배까지 탐색지점이 줄어 드는 것을 확인 할 수 있다. 그와 비교하여 제안하는 알고리즘은 SEA 알고리즘에 비해서 8~12%까지 성능이 향상되었다. 이렇게 성능이 향상된 이유는 현재블록과 주변블록의 움직임 벡터에 유사성이 많아서 탐색지점을 효율적으로 없앤 것에 원인이 있다.

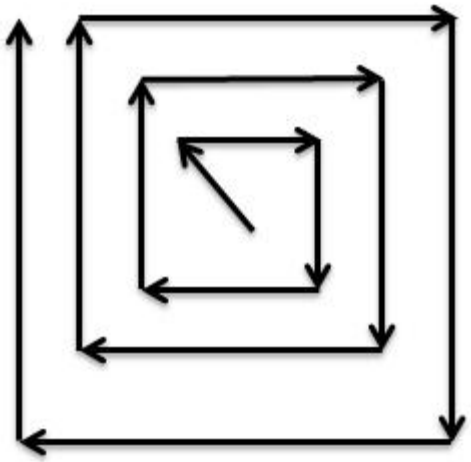

그림 1. 기존의 나선형 탐색 방법