



Researcher 김정현, 전자공학과 (kjh2250@ajou.ac.kr)
전지웅, 전자공학과 (ninezipjeongu@ajou.ac.kr)

Professor 선우명훈, 전자공학과

ABSTRACT

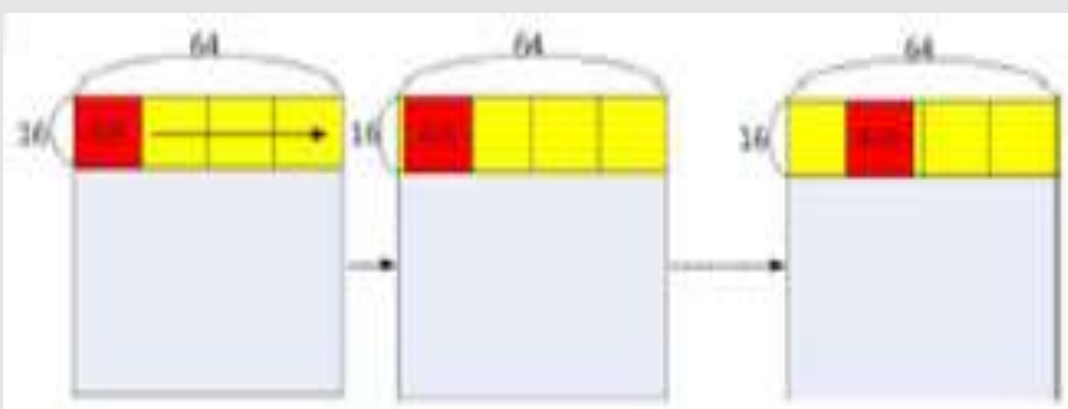
- 고화질 영상들을 더욱 빠르게 압축하고 처리하기 위해서는 움직임 추정의 연산을 보다 빠르게 처리할 수 있는 기술 혹은 부품이 필요하다. 움직임 추정에 있어서 SAD 연산량은 영상의 크기와 검색윈도우의 크기와 블록의 크기에 따라 증가한다. 연산량을 줄이기 위하여 기존에 제안된 연구 방법들은 연산시간과 메모리 접근을 줄이기 위하여 고속의 병렬 연산, 중간버퍼관리, 검색윈도우관리, 움직임추정을 위해 데이터 재사용을 이용한 하드웨어구조, 다중 참조프레임의 정보를 재사용하는 방법 등이 제안되고 있다. 여기서 움직임 추정의 목적은 되도록 많은 단위 블록 또는 블록을 이전 프레임에서 참조함으로써 시간적 중복성을 제거하여 압축률을 증가시키는데 그 목적을 둔다.

OBJECTIVES

- 이 연구에서는 논문을 바탕으로 하여 H.264 표준에서 사용되는 Block searching 방법을 구현하고 시뮬레이터를 작성해본다.
- 또한 고속처리를 위해서는 하드웨어 언어로의 작성에 필요해, 위한 Verilog 로 구현 후 FPGA타재가 최종 목적이다.

METHODOLOGY

1. Block searching



- 차이 값을 구하기 위해서는 블록 단위로 프레임 간의 차이를 비교해서 연산을 한다.

- 블록단위로 비교할 때는 블록 안의 값을 계산해서 나오게 되는 SAD값을 비교해서 가장 차이가 적은 주변 블록의 위치를 통해 모션벡터를 구한다.

$$SAD = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |C_{ij} - R_{ij}|$$

※SAD : sum of absolute difference 로 해당 영역의 휘도값의 차를 이용한다.

2. Test zone search

- 전역을 순서대로 전부 탐색하지 않고 효율적으로 탐색하기 위한 기법

(1) Motion Vector prediction

- 블록단위로 SAD가 가장 적은 벡터의 예측자를 선택. 다음 탐색의 시작점으로 사용됨

(2) Grid Search

- 정방향/다이아몬드형으로 탐색범위를 2배씩 늘리고 탐색거리를 저장한다.

(3) Raster Search

- 범위가 늘기만 하면 전역탐색과 다르지 않으므로 down-sampling 된 영역의 전역 탐색이다. 최소의 SAD값을 가지는 벡터 값이 탐색영역보다 크면 넘어 가는 단계.

(4) Star refinement

- Down sampling 된 grid search의 반복으로 최종 Motion Vector를 구하는 단계

3. Flow of Test zone search

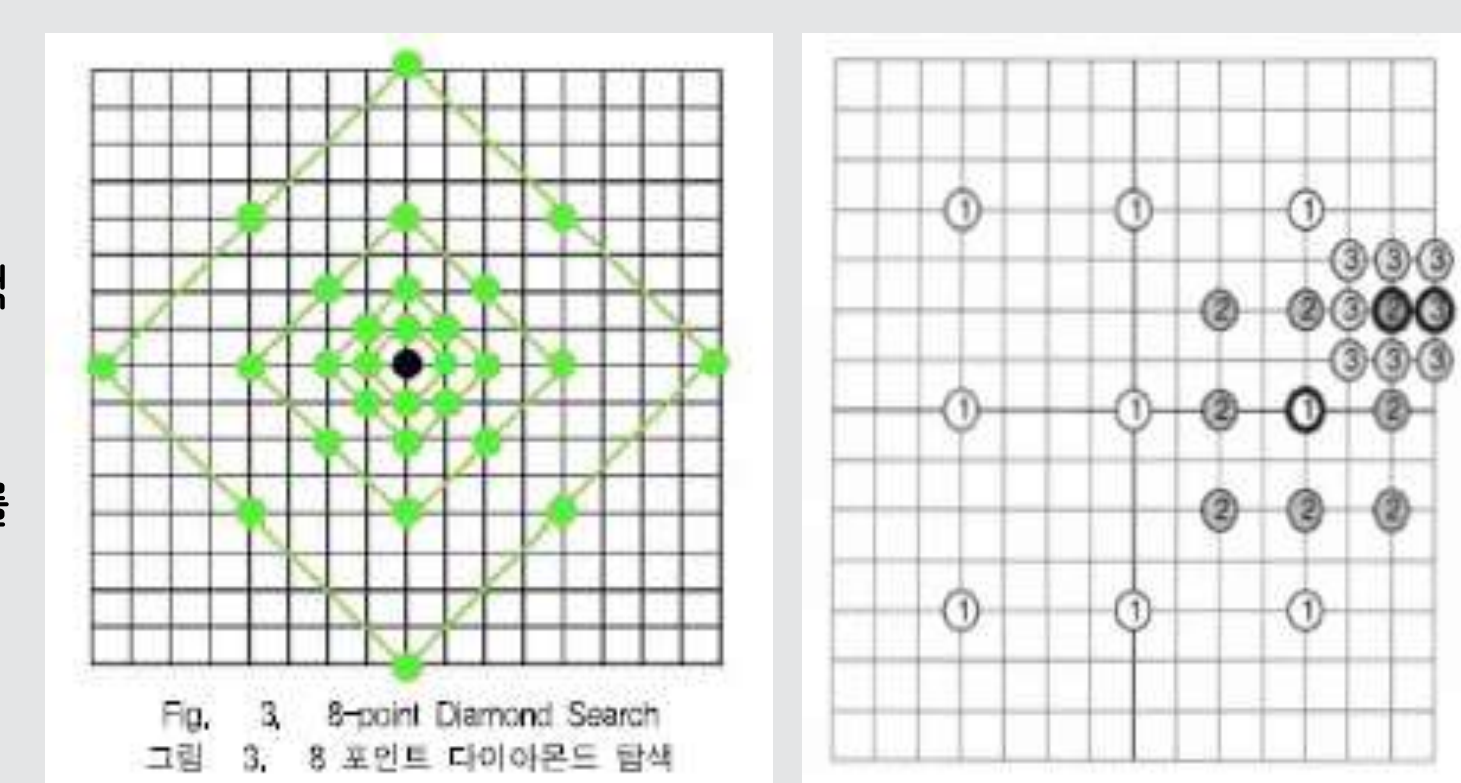
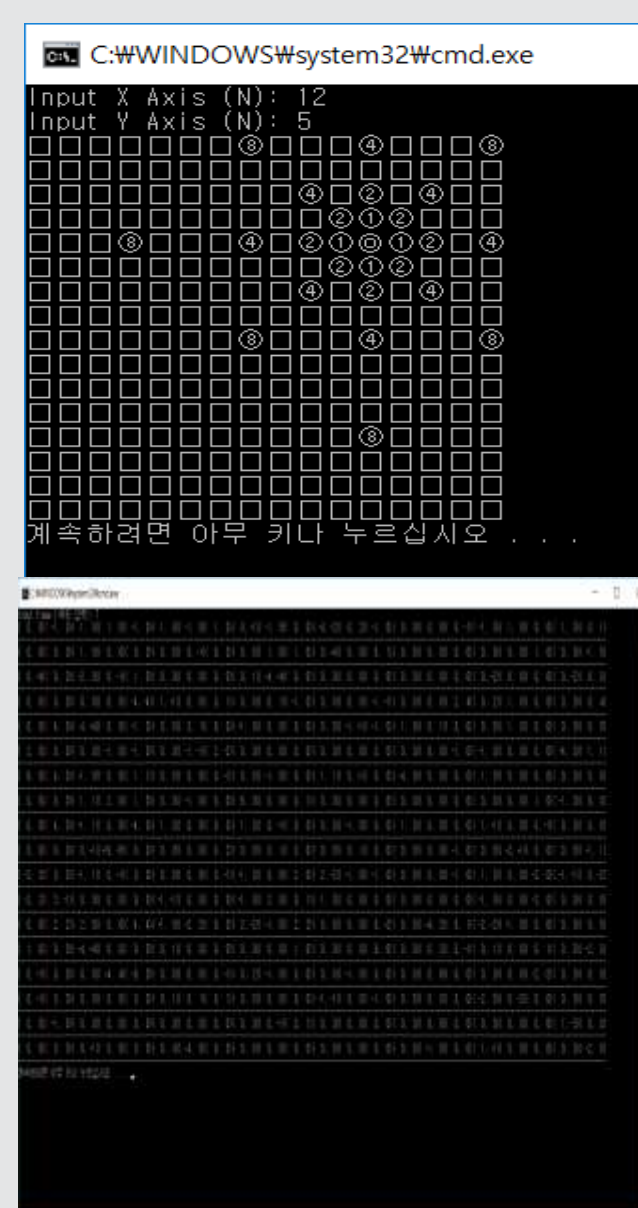


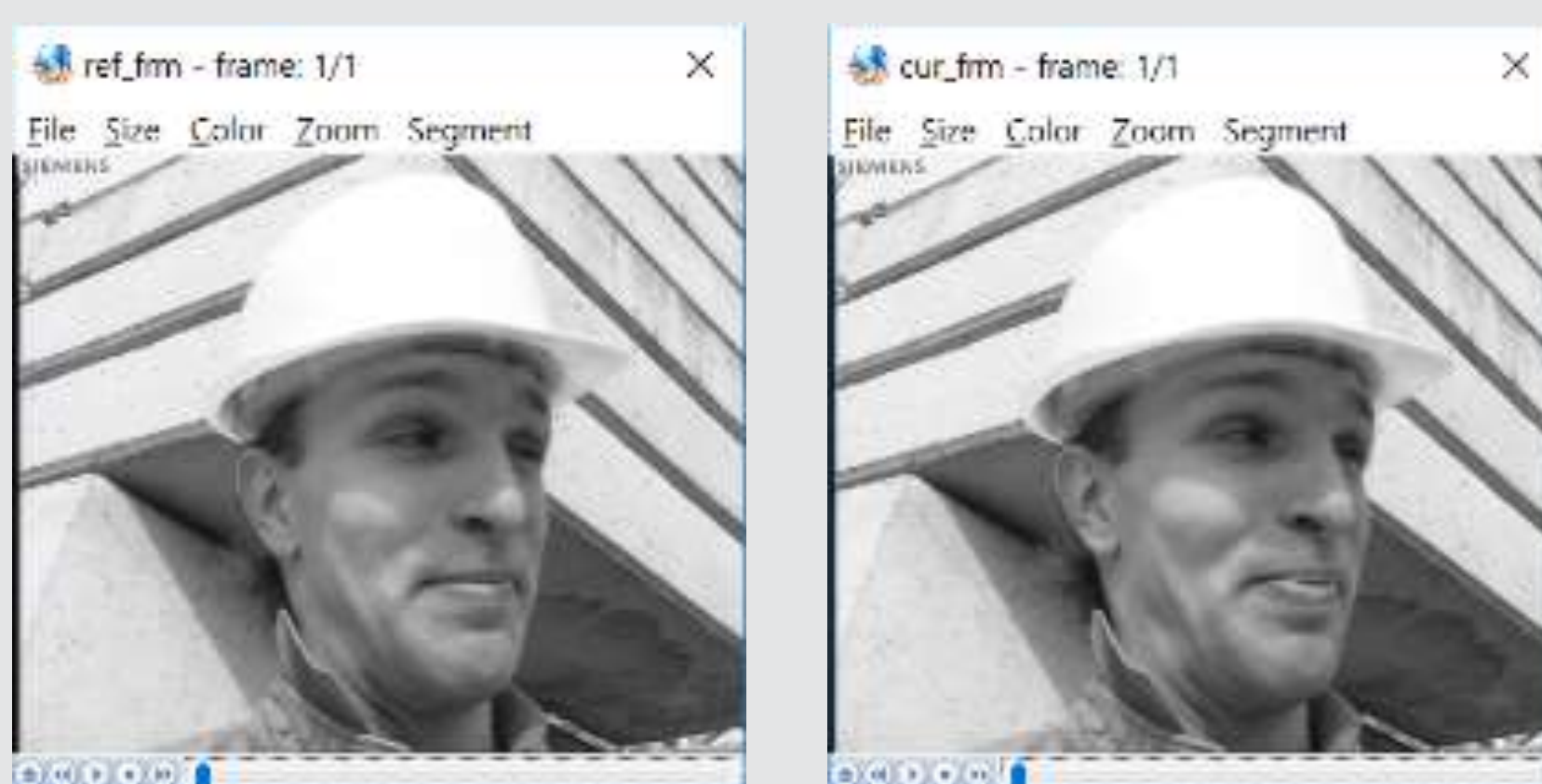
Fig 3. 탐색범위 확장(좌)
down-sampling된 영역의 탐색(우)

RESULTS

1. 시뮬레이터의 탐색순서 확인과 Motion Vector 출력

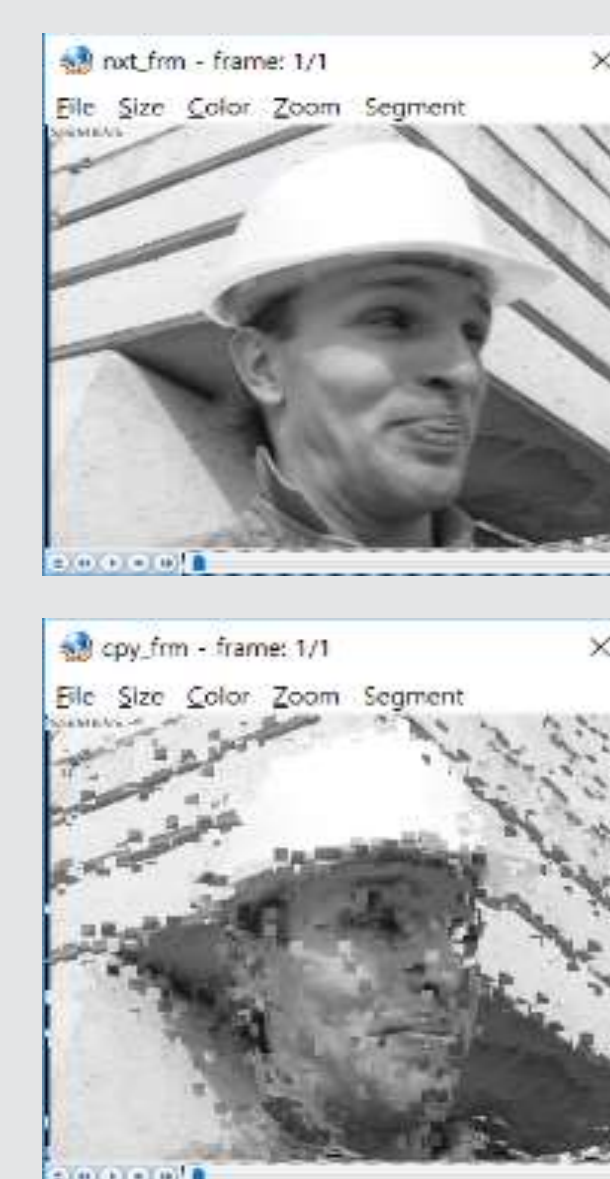


2. YUV파일의 입력 및 출력



(1) 원본 파일의 두 프레임

- 입력값으로 사용될 두 파일이며 실제로 입력 될 때는, 해당 블록이 갖고 있는 휘도값과 좌표관련 데이터로 입력이 된다.



(2) 실제로 올 다음 프레임

- 입력으로 사용된 파일의 뒷 부분이며, 앞 프레임과는 다소 차이가 있는 부분이 드러남

(3) Motion Compensation을 거친 값

- 입력으로 사용된 프레임을 통해 얻은 모션벡터값으로 움직임보상을 실행하고 그 값을 출력으로 내보낸 것.

=> 블록탐색의 정밀성 또는 움직임 보상 과정에서 원 프레임의 값을 가져오는 것에서 오차가 발생했었으므로 추정

CONCLUSIONS

- 이 연구에서는 논문을 바탕으로 Test Zone Search의 시뮬레이터를 구성할 수 있었다.
- 풀서치의 방법으로 Motion vector를 구한다음 Motion compensation을 적용한 프레임 값을 얻고 출력할 수 있었다.
- 이후에는 FPGA 탑재를 위해 TZS를 보완하여 C로 구현된 시뮬레이터의 내용을 Verilog로 구현할 것이다.
- 하드웨어 언어로 옮기는 만큼 더욱 빠른 연산을 위해서는 파이프 라이닝을 포함한 연산에 최적화 된 방식으로 고려해서 구현되어야 할 것이다.

- (4) FPGA 탑재를 위해서 RTL파일의 작성
 - PU array 의 경우 레지스터 값 저장을 위한 모듈
 - SAD selector 연산을 위해 CLA 를 활용
 - MV generator 모듈은 비교기를 사용

