120200153 이성훈
2020/09/20

HW#1 Block matching algorithm

Full search

Block matching algorithm은 video의 압축 기법에 사용된다. Current frame과 previous frame이 있다고 가정해보면, previous frame에서 current frame을 복원하기 위해 motion vector를 이용하게 된다. Motion vector란 frame을 block단위로 나눈 상태에서 해당 block이 previous frame에서 가장 유사한 block을 그대로 가져올 수 있는 방법이다. 자세하게 설명하자면, current frame을 block 단위로 나눈 다음 해당 block들이 previous frame에서 가장 유사한 block을 찾는다. Previous block에서의 위치와 current block의 위치를 픽셀단위로 vector를 구하여 그 vector 정보를 저장한다. 이렇게 되면, 한 픽셀이 1의 정보량을 차지한다고 가정하면 block단위로 vector만 픽셀과 동일하게 저장하면 되기 때문에 $1/block_size^2$만큼으로 저장해야 하는 정보량이 줄어들게된다.

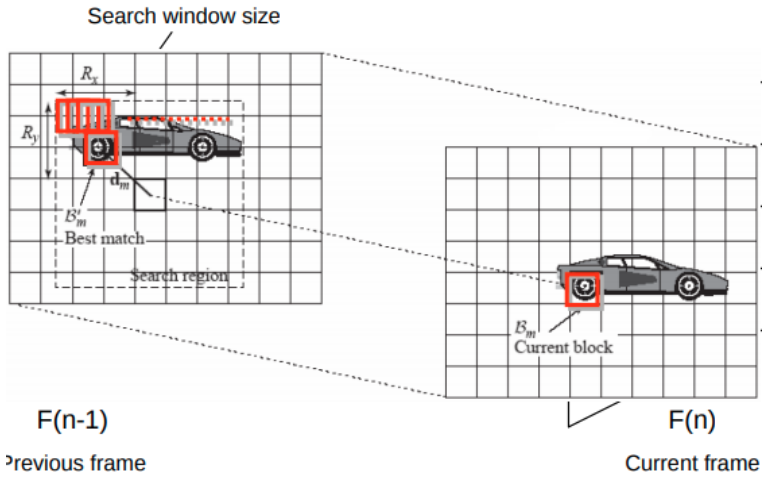

Figure 1: Block matching

Figure 1은 block matching algorithm을 잘 나타내주는 그림이다. Current block 부분인 바퀴부분을 previous frame에서 가장 잘 맞는 즉, MSE나 l1 error와 같은 error function을 가장작게하는 그림에서 best match에 해당하는 영역을 찾아낸다. 그런 다음 previous block과 current block의 상대적인 위치를 계산하여 vector로써 저장하게 된다. 이렇게 저장한 vector들로 current frame을 previous frame으로 estimation하게된다. 또한 previous frame을 통해 current frame을 estimation하기 때문에 current frame을 기준으로 계산하는 backward mapping이 필수적이다. Block matching에 사용되는 parameter들에 대해서 설명하면, search range, block size 두개로 나눌 수 있다. Block size는 말 그대로 block의 size를 말한다. Search range는 해당 block에 대해서 상하좌우로 얼마만큼의 range를 탐색할 것이지를 뜻한다. 이 두 parameter를 설정할 때 주의해야될 점은, block size의 경우 block이 커지면 커질수록 필요로 하는 vector가 적기 때문에 정보량 관점에서 이점이 있다. 하지만 block size가 크기 때문에 이전 frame에서 그대로 block을 가져올 때 세밀한 부분까지 신경쓰지 못하기 때문에 복원 성능이 좋지 않다. Search range가 커질 경우 더 많은 범위를 탐색하므로 좀 더 넓은 범위까지 탐색하므로 빠르게 이동하는 물체에 대해서도 잘 잡을 수 있다. 하지만 search range가 크기 때문에 비교해야하는 영역이 굉장히 많아진다. 즉, 계산량과 복원 성능에 trade off가 있는 parameter이기 때문에 적절한 search range와 block size를 정하는 것이 중요할 것이다. 이렇게 search range 안의 모든 영역을 pixel 마다 sliding하며 block matching하는 방식을 Full search라고 하며 이는 연산량이 너무 많기 때문에 간소화된 방법인 three step 방식을 설명하도록 하겠다.

Three step search

Three step search algorithm은 말 그대로 3번의 step만으로 block matching을 하는 방법을 얘기한다. 기본적으로 search range는 4로 정해져있고 이는 full search에서의 search range와는 약간 다른 역할을 한다. 첫번째로, block으로 current frame을 나누는 것은 동일하다. 이제 block에 대해서 8방향으로 search range만큼 떨어진 곳과 제자리, 총 9곳에서 error function에 따라 error를 구한다. 이때, 8방향은 euclidean 거리가 아닌 $\{(x,y)|x \in [-1,0,1], y \in [-1,0,1]\}$인 방향이다. 이렇게 구한 error중 가장 작은 error에 해당하는 곳에서 다시 search range를 반으로 줄여 위 과정을 반복한다. 이러한 과정을 search range가 1일때를 마지막으로 반복하면 총 3번만에 motion vector를 추정할 수 있다. 이 방법은 global minimum인 부분 근처도 역시 error값이 낮을 것이라고 가정하고 진행하는 방법이다. 하지만 꼭 그렇다는 보장이 없으므로 속도에 대해서는 굉장히 빠르겠지만 성능은 보장할 수 없는 방법이다.

Result

Figure 2: Reference, target, full, three step results

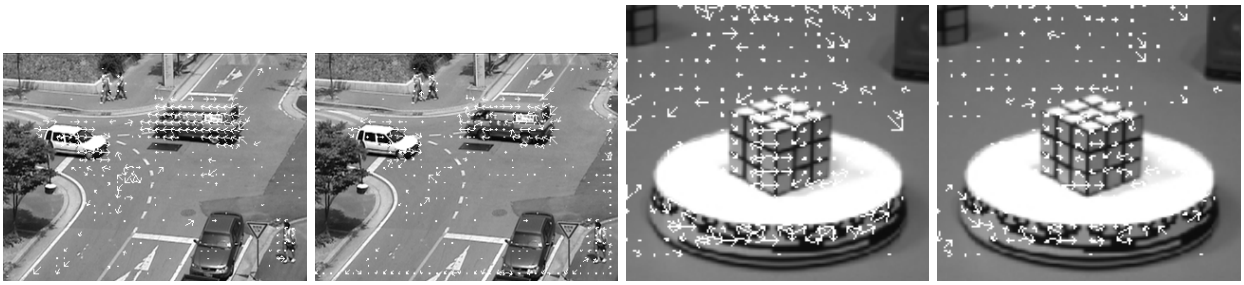

Figure 3: Full, three step motion vector results

결과를 살펴보면 2은 reference image에서 target image을 각각의 방법으로 복원한 것이다. 크기가 작아 잘 보이지는 않지만 three step의 경우 위쪽 차의 왼쪽 부분에 많이 일그러짐에 비해, full search의 경우 꽤 잘 복원된 것을 확인할 수 있다. Reference에서 흰차는 우측으로, 그 오른쪽의 차는 좌측으로 이동하는데 이를 Figure 3을 통해서도 확인할 수 있다. 추가적으로 cube에 대해서도 진행했는데 street과 마찬가지로 thresholding을 따로 처리해주지 않아 유사한 block끼리의 이동이 너무 많은 것을 확인할 수 있다. 또한 반시계방향으로 원판이 돌아가는 형태인데 큐브의 구조가 block마다 유사하기 때문에 오히려 반대 방향으로의 이동이 관찰되는 것을 확인할 수 있다.