2019-1 디지털 영상처리

Block Matching 알고리즘을 이용한 Depth 추정 및 실제 거리 추정

소프트웨어학과
201511173 강인한
201511211 윤지호

1. 배경 및 동기

수업에서 영상 내에 존재하는 어떤 특정 부분이 다음 프레임에 어디로 이동했는지 측정하는 기술을 배웠다. 이를 모션 벡터라고 하고, 블록 매칭 알고리즘을 이용해 특정 블록의 모션 벡터를 측정할 수 있다는 사실을 알게 되었다.
이 때, 카메라를 평행하게 이동하면 상대적으로 가까운 물체는 빠르게 움직이고, (영상 내에서 빠르게 움직이는 것처럼 보이고) 먼 곳의 물체는 느리게 움직인다. (느리게 움직이는 것 처럼 보인다)
여기에서 아이디어를 얻어, 영상을 특정 크기의 블록으로 나누고 해당 블록이 몇 픽셀만큼 움직였는지와 카메라가 몇 cm만큼 움직였는지, 그리고 카메라 시야각의 정보를 이용해서 해당 물체까지의 깊이를 추정하고, 나아가 특정 물체의 길이를 계산할 수 있겠다고 생각했다.

2. 문제 정의

입력받은 영상에 대하여 한 블록이 이동한 픽셀 수를 측정한 뒤 각도 변화를 이용하여 depth를 구하고 모든 블록들에 대하여 상대적 depth를 추정한다. 이를 이용하여 블록들의 실제 depth, 블록들끼리의 거리를 추정하는 것을 목적으로 한다.

3. 개발 내용

3-1) 환경 및 입력 영상

가까이 있는 물체와 멀리 있는 물체가 명확하게 구분되고, 블록 매칭이 잘 수행되도록 특징이 있는 부분이 많은 영상으로 제한한다.
카메라는 영상은 찍는 도중 x,y 좌표가 동시에 달라지면 계산과 결과 도출 과정이 복잡해지므로 왼쪽에서 오른쪽으로 평행하게 이동하면서 찍고, 물체는 반대로 왼쪽으로 이동하는 것처럼 보여야 한다.
거리를 측정하기 위해 카메라의 시야 각도와 얼만큼 이동했는지에 대한 정보가 필요하다. 핸드폰과 연결된다면 핸드폰의 센서를 이용해 얼마만큼 움직였는지, 시야 각은 얼마나 되는지 정보를 얻을 수 있겠지만 현재 아이디어를 제안하는 단계에서는 핸드폰의 시야 각을 50도로 가정하고, 3프레임에 0.8cm 정도 이동했다고 가정한다.
블록 매칭에 사용되는 두 입력 프레임은 영상 시작 후 10번째 frame과 13번째 frame으로 임의로 정해 사용하였다.

3-2) 전처리

먼저 입력된 영상을 480x270의 size로 resizing 한다. 그 다음 잡음을 제거하기 위해 미디안 필터를 3x3 사이즈의 커널로 수행한다. 그렇게 잡음이 제거된 영상에 대해서 엣지 디텍팅 후 원 영상과 더해서 하이부스트 필터 효과를

적용한다. 이는 블록 매칭이 더 잘 수행되는 효과를 얻기 위해서 특징을 부각시키기 위함이다.
엣지 디텍팅에는 sobel 필터링을 사용하였다. 원 영상에 x축에 대해 sobel filtering한 영상과 y축에 대해 sobel filtering한 영상을 각각 더하는 방식으로 구현하였다.

3-3) 주 알고리즘

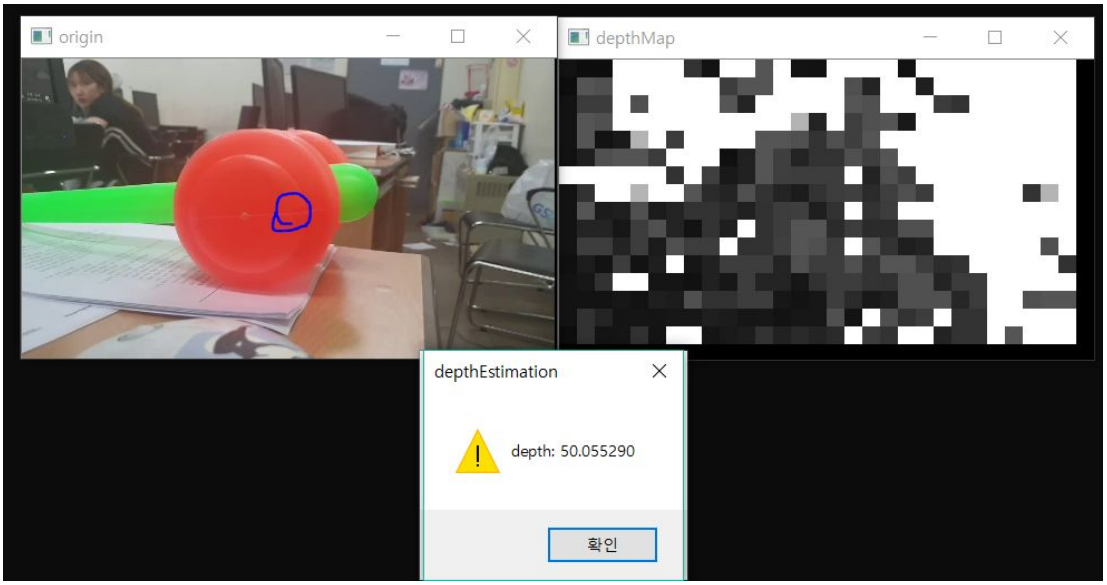
depth 정보와 두 점 사이의 거리를 추정하기 위해 주로 사용되는 알고리즘은 Block Matching 알고리즘이다. 전처리된 영상을 블록 사이즈(16으로 고정) 간격으로 나누어 해당 블록에 대해서 각각 블록 매칭을 수행한다.
블록 매칭 알고리즘으로는 3SS알고리즘을 채택하였다. 이 때, 블록이 현재 블록보다 오른쪽에 있는 블록과 매칭되면 추정되는 depth 값이 음수가 나오는 오류가 있었다. 해결책으로는 무조건 영상은 왼쪽으로만 이동하므로, 블록 매칭 시에 현재 블록보다 오른쪽 블록은 보지 않도록 하는 방법으로 해결했고 오차를 줄였다. 따라서 현재 블록과 같은 x값 또는 적은 x값을 가지는 블록에 대해서만 후보로 하는 수정된 3ss 알고리즘을 사용하였다.
한 블록의 depth 값은 매칭되는 블록 간의 x,y 픽셀 좌표 값을 이용하여 해당 좌표와 카메라 초점간의 각도를 계산 하고 그 각도 차이와 카메라가 움직인 실제 거리를 이용하여 실제 거리를 계산한다. 또한 depth 계산이 끝난 뒤 두 지점사이의 거리는 각 지점의 depth값과 그 지점이 이루는 각을 구하고, 코사인 법칙을 이용하여 계산한다.

4. 구현

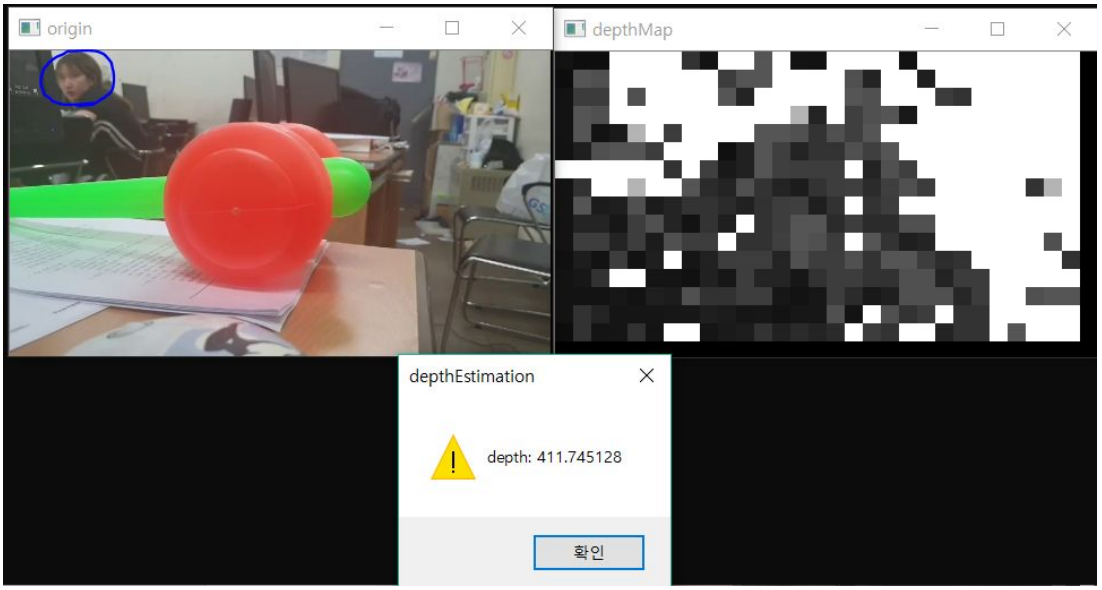
5. 결과

입력 영상으로 아래의 3가지 샘플 영상을 촬영하였다.
왼쪽 영상은 원본 영상, 오른쪽 영상은 각 블록에 대해 측정된 depth를 0~255사이의 값으로 정규화하여 표현한 depthMap이다.(검은색에 가까울 수록 가깝고, 흰색에 가까울 수록 멀다.)
거리를 추정할 두 점을 왼쪽 마우스 클릭하면 두 점 사이의 거리를 추정해 메세지로 띄워 주고, 한 점을 오른쪽 마우스 클릭하면 해당 점 까지의 depth를 추정해 메세지로 띄워 준다.

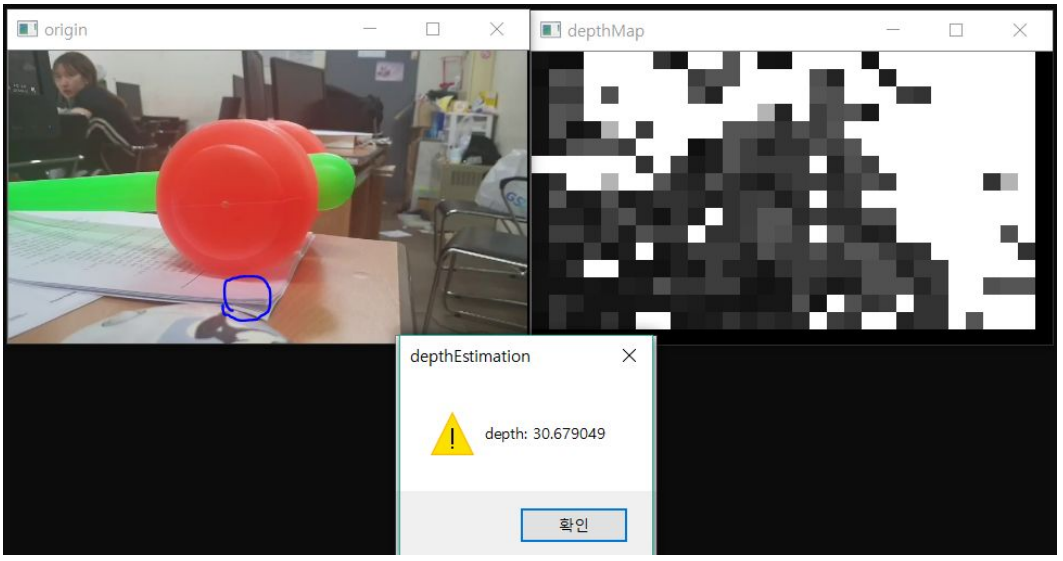
<영상 1>



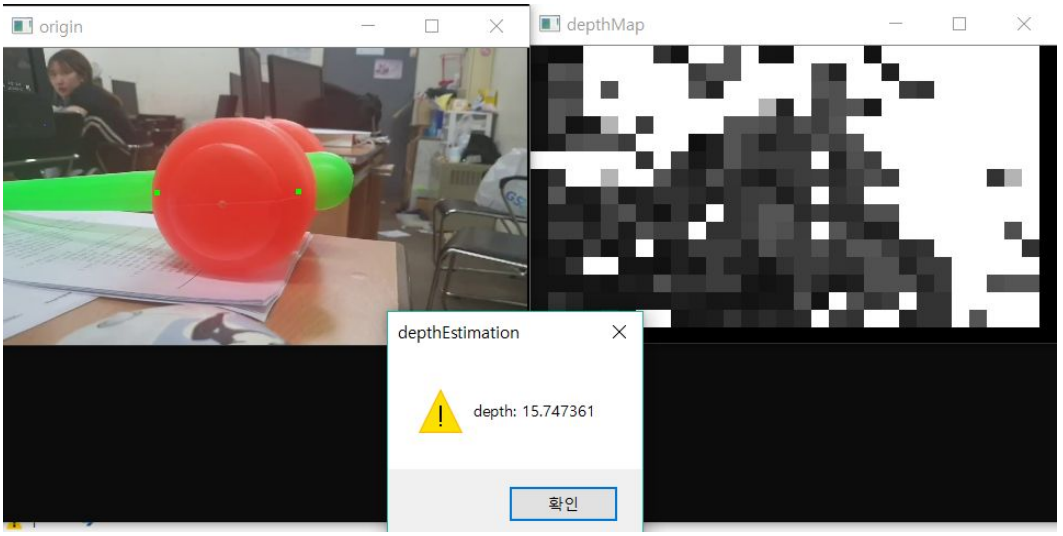
(뽕망치까지의 거리를 측정한 결과)



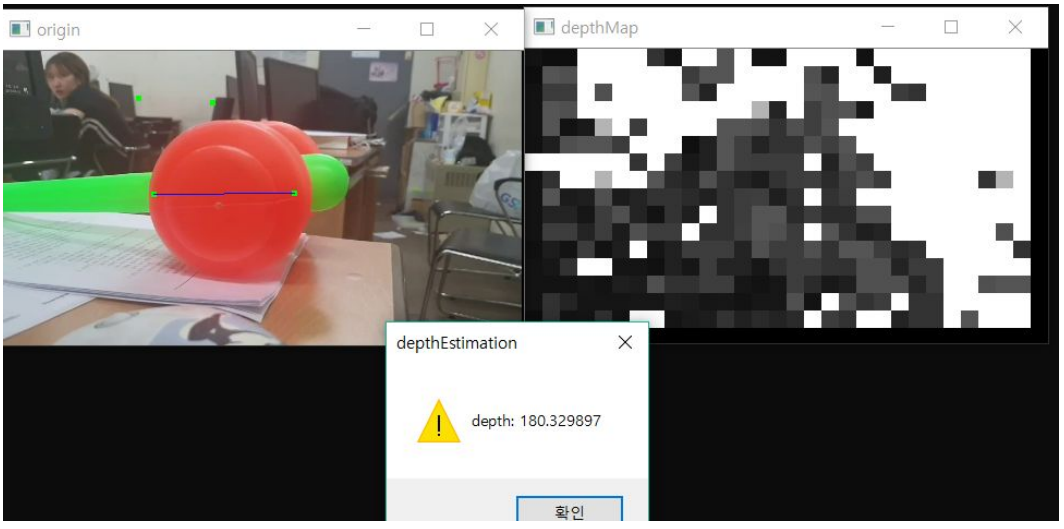
(뒤의 학생까지의 거리)



(앞에 있는 책상까지의 거리)

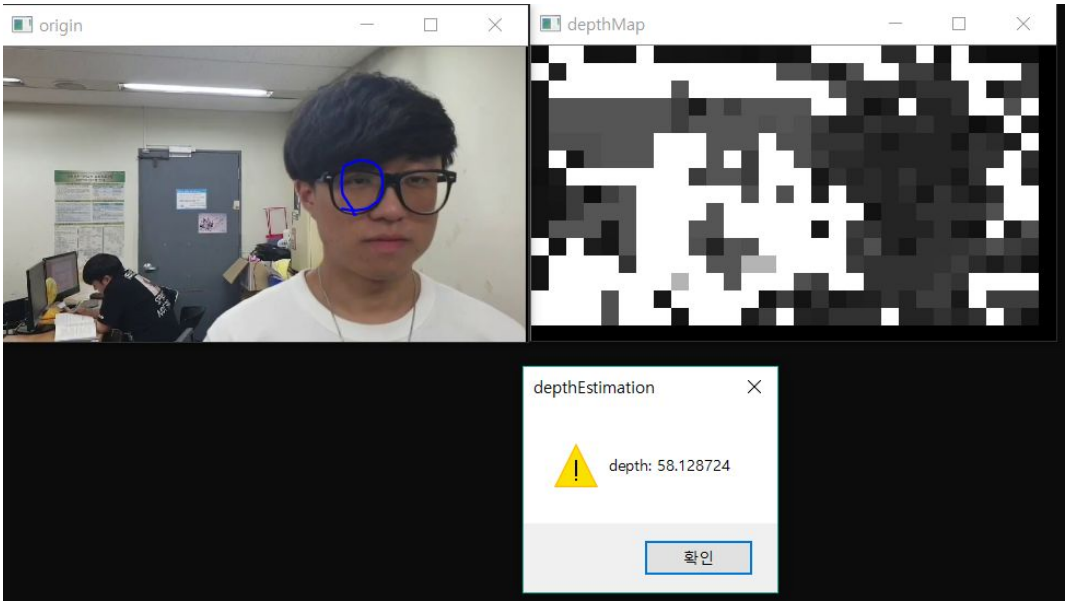


(뽕망치의 망치 머리 부분 직경)

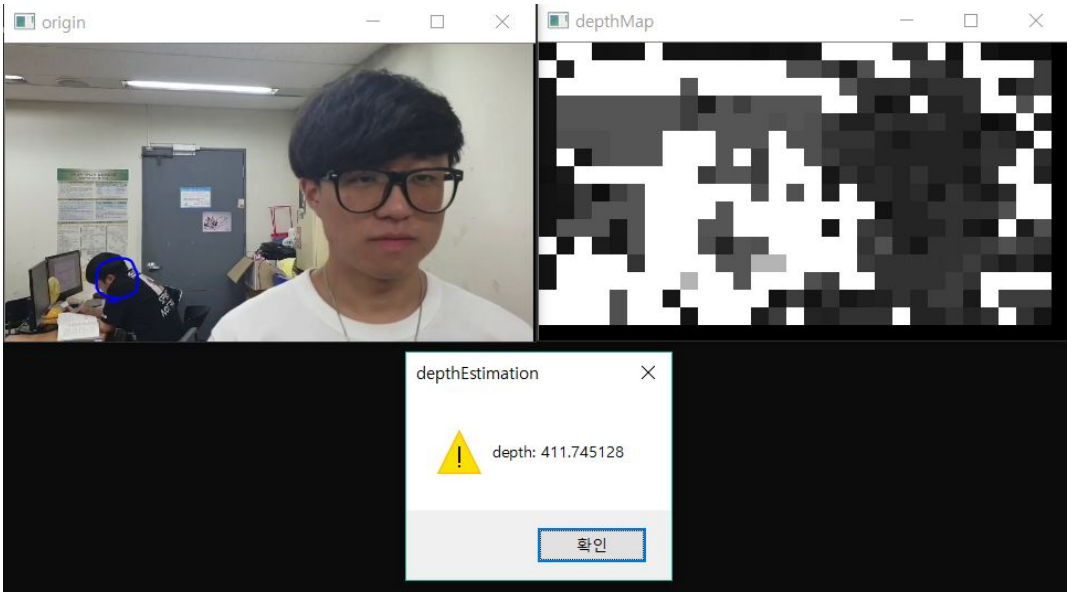


(뒤의 벽에서 좌석간 간격)

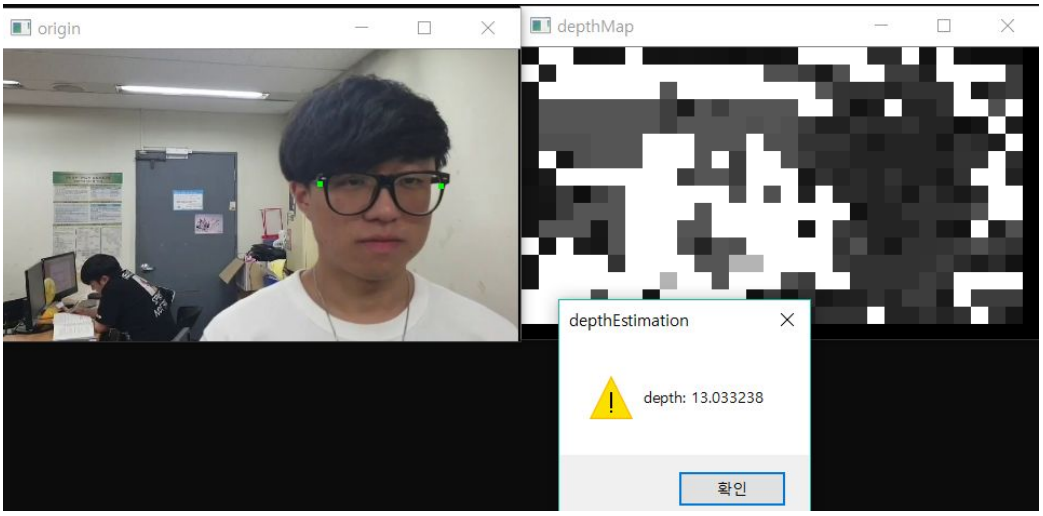
<영상 2>



(학생의 얼굴까지의 거리)

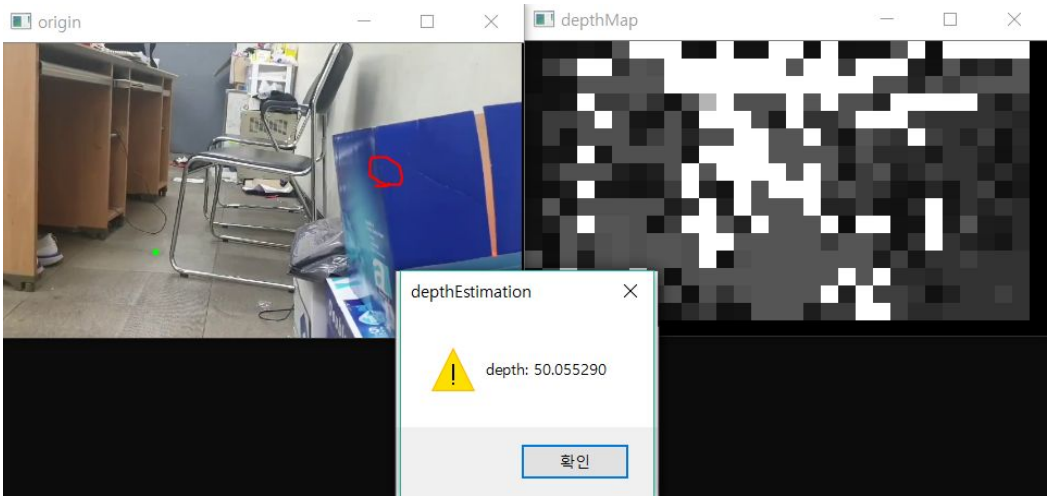


(뒤에 있는 학생까지의 거리)

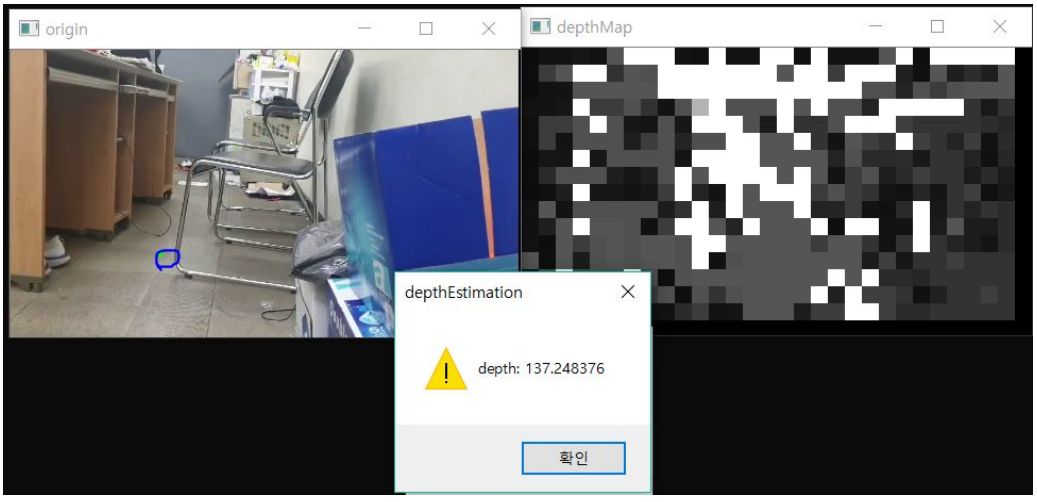


(학생의 얼굴 가로 길이)

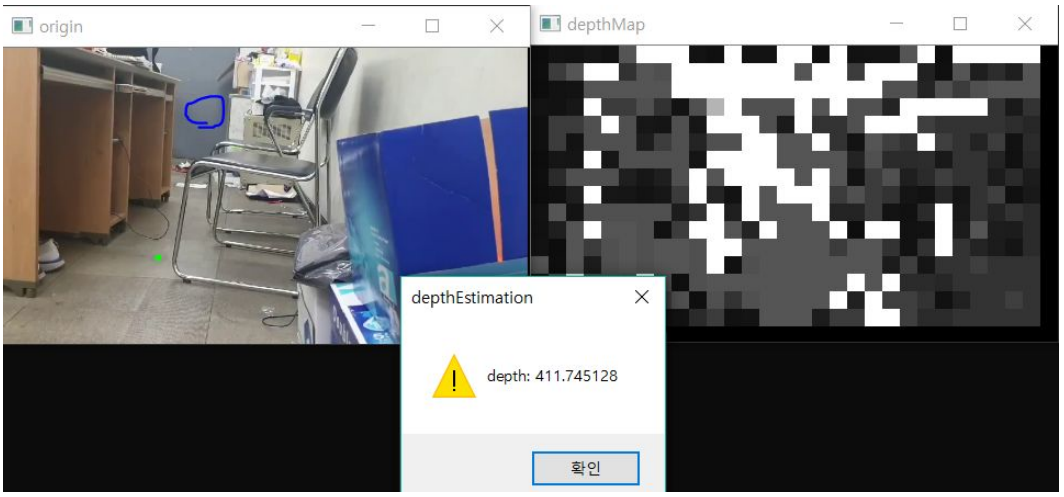
<영상 3>



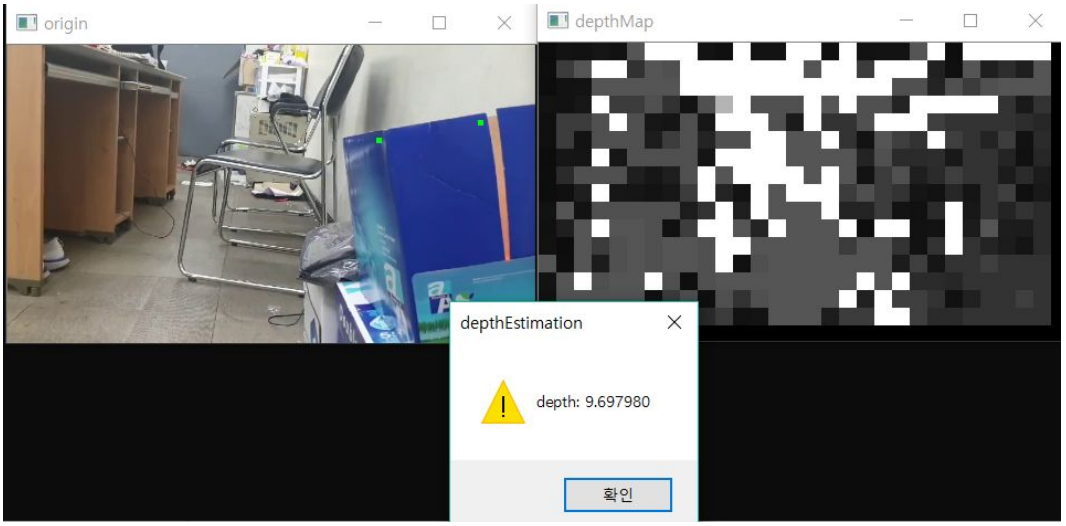
(오른쪽 박스까지의 거리)



(바닥까지의 거리)



(문까지의 거리)



(A4용지 가로 길이의 절반)

5. 한계 및 개선점

1) 우리는 핸드폰의 시야각과 핸드폰이 실제 촬영하며 이동한 거리를 임의로 가정하여 계산했다. 따라서 정확한 정보를 얻을 수 있다면 좀 더 정확한 추정이 가능할 것이라 생각된다. 또한 가로 각도에 대해서만 계산했기 때문에 가로 길이만 비교적 정확하게 추정되고, 세로 각도에 대한 계산식을 추가하면 더 정확한 결과와 세로 길이도 추정할 수 있을 것이라 생각된다.
2) 블록 매칭 알고리즘을 풀 서치가 아닌 3SS로 선택했기 때문에 프레임 간격이 멀어질 수록 윈도우 사이즈를 늘리더라도 정확도가 심하게 낮아지는 문제가 생겼다. 따라서 프레임 간격을 좁힐 수 밖에 없었고, 이에 따라 어느 정도 거리 이상 먼 물체는 픽셀이 움직이지 않아 정해진 최대값으로 측정될 수 밖에 없었다. 또한 상대적으로 가까이 있는 물체에 대해서는 비교적 잘 추정하는 반면, 멀리 있을 수록 부정확해지는 경향이 있었다.

3) 특징이 없는 물체 (예를 들면 흰 바탕의 벽)에 대한 블록 매칭은 정확하게 수행되지 않아서 오차가 심하다. 이를 개선할 수 있다면 정확도가 크게 향상될 것이라고 생각된다.