이동 거리 측정 프로젝트

이동거리 측정을 위해 많은 방법들을 적용해본 결과 다크프로그래머 블로그의 “영상의 지면투영” 방법이 가장 좋은 결과를 나타내어 이 방법을 사용함.

$$u=(x-cx)/fx$$
$$v=(y-cy)/fy \quad \text{--- (1)}$$

구해진 정규좌표 (u, v)와 카메라 높이 h, 틸트 θ_{tilt}를 이용한 이후의 계산과정은 아래와 같다 (앞서 두 글에 대한 이해와 상상력을 믿고 자세한 설명은 생략한다). 정규 이미지 평면에서는 주점의 좌표가 c(0, 0)이 되고 이미지 평면과 카메라 원점과의 초점거리가 1이 됨에 주의한다.

$$CC' = h$$

$$C'P' = CC' * \tan(\frac{\pi}{2} + \theta_{tilt} - \text{atan}(v))$$

$$CP' = \sqrt{(CC')^2 + (C'P')^2}$$

$$Cp' = \sqrt{1 + v^2}$$

$$PP' = u * CP'/Cp' \text{ (닮음비 이용)}$$

$$d = \sqrt{(C'P')^2 + (PP')^2}$$

$$\theta = -\text{atan2}(PP', C'P')$$

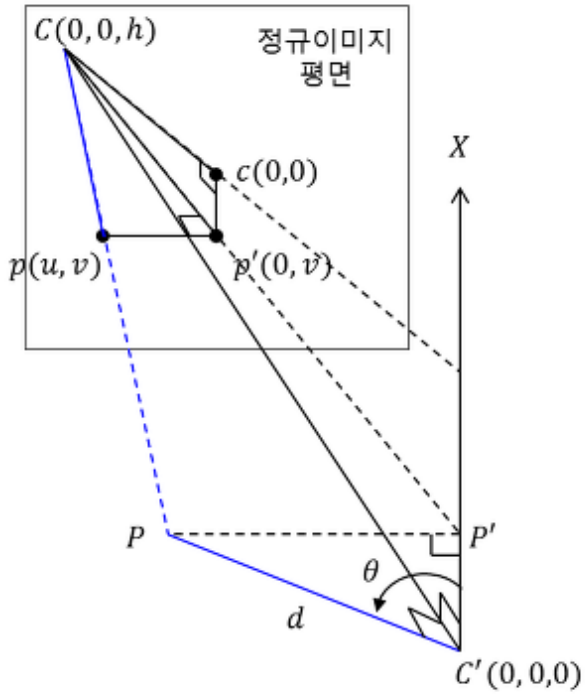


그림 4. 기하학적 계산 과정

위 사진과 같은 방법을 코드로 구현하였음.

거리와 각도를 재는 부분은 함수로 구현해놓았으며

```python
def distance_angle_measure(x,y):
    fx = 1464.04236
    fy = 1459.47648
    h = 2300
    theta_tilt = -49
    width = 2560
    height = 1440
    cx = width/2
    cy = height/2
    u = (x-cx)/fx
    v = (y-cy)/fy
    c_p_ = h * math.tan(math.pi/2+math.radians(theta_tilt) - math.atan(v))
    cp_ = math.sqrt(h*h+c_p_*c_p_)
    cp = math.sqrt(1+v*v)
    pp_ = u*cp_/cp
    d = math.sqrt(c_p_*c_p_ + pp_*pp_)
    theta = -math.atan2(pp_,c_p_)
    theta = math.degrees(theta)
    return d,theta
```

위 사진과 같음

사용자의 상황에 맞게끔 코드를 사용하기 위해서는 빨간색 동그라미 안의 fx, fy, h, theta_tilt, width, height의 값을 현재 카메라 상황에 맞는 초점거리 x,y 카메라가 설치된 높이, 카메라 렌즈의 각도, 카메라가 불러들이는 이미지의 폭과 넓이를 입력해 주어야 함.

위 함수를 거쳐가면 d, theta 즉 거리와 각도의 값 2가지가 반환 됨.

코드를 돌리는 방법이 두 가지가 있음 첫 번째는 openpose를 사용하는 것이고 두 번째는 YOLO를 사용하는 것.

openpose를 사용할 시 GPU 환경이 갖춰져 있지 않으면 속도가 매우 느리고 co-lab에서 돌리는 것을 추천함. 코랩 환경에서 model(딥러닝 가중치)은 다운받게 되어 있으므로 2페이지에 나와 있듯이 카메라 정보와 캘리브레이션 값을 바꾸면 됨.

YOLO 사용 시 현재 YOLOv4를 사용하고 있으며 가중치, cfg, classes.txt 파일들이 로컬에 있어야 하며 이를 코드상에서 현재 로컬 위치로 변경해야 함.

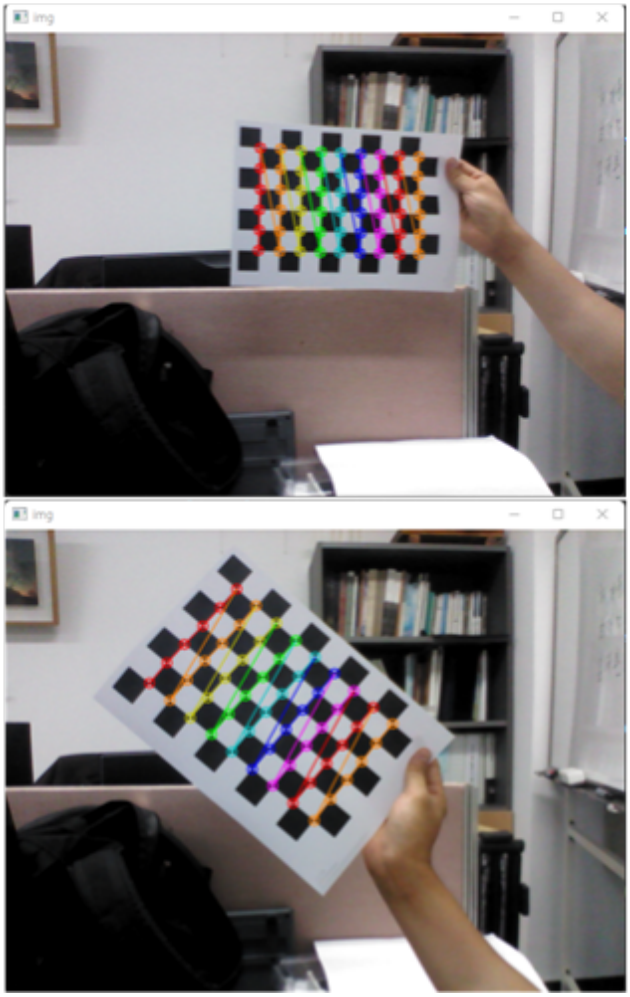
```
with open("C:\\Users\\BRIGHTEN3\\Downloads\\Yolov4-Detector-and-Distance-Estimator-master\\classes.txt", "r") as f:
    class names = [cname.strip() for cname in f.readlines()]
yoloNet = cv.dnn.readNet("C:\\internship\\weights\\yolov4.weights", "C:\\internship\\Yolov4-Detector-and-Distance-Estimator-master\\yolov4.cfg")
```

YOLO, openpose 마찬가지로

```
cap = cv.VideoCapture("C:\\internship\\output_triangle.avi")
```

위 코드에서와 같이 현재 돌릴 동영상 혹은
웹캠에 관한 정보를 사용자에게 맞게 변경해야 함.

2페이지의 fx, fy의 값은 카메라 캘리 브레이션으로 얻어지며



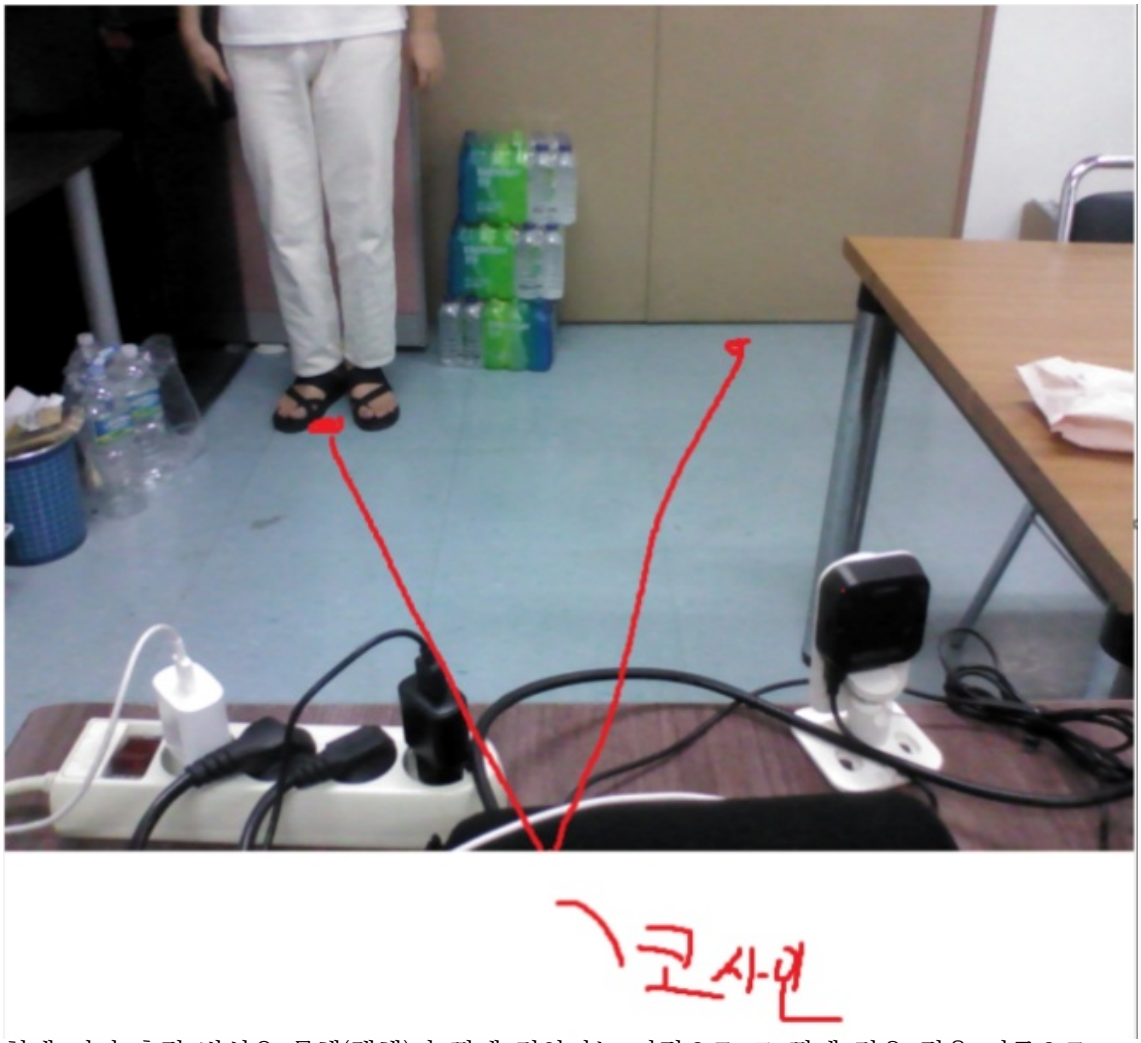
위 사진과 같이 카메라 앞에서 체스 보드를 여러 각도와 여러 위치에서 찍고 저장하여 calibration.py 코드를 실행하면 fx,fy 값이 나오게 됨.
*카메라 캘리 브레이션 진행 시 체스보드가 카메라에 정방향이 아닌 사선으로 휘어진 사진도 많아야 함.

```
Camera matrix :

[[2.59482916e+03 0.00000000e+00 1.32886124e+03]
 [0.00000000e+00 2.80956772e+03 9.65609964e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
dist :

[[ 0.18460266 -2.23712147  0.08112175 -0.01432855  4.93480703]]
```

calibration.py 실행 시 빨간색 친 왼쪽 순서대로 fx, fy 값임.



현재 거리 측정 방식은 물체(객체)가 땅에 닿았다는 가정으로 그 땅에 닿은 점을 기준으로 거리와 각도를 재며 거리 측정 방식은 발목 점 혹은 객체 인식의 마지막 점을 프레임별로 찍어 이 점이 2개가 되면 코사인 제 2법칙을 이용해 거리를 잼.

코드에 관한 설명과 위의 calibration.py, chessboard 등등은 zip 파일로 남겨 놓음.