9월 1주차

STUDY REPORT

김영민

진 행 상 황

02. 9.2~9.3
OpenCV

04. 9.9~
차후 계획

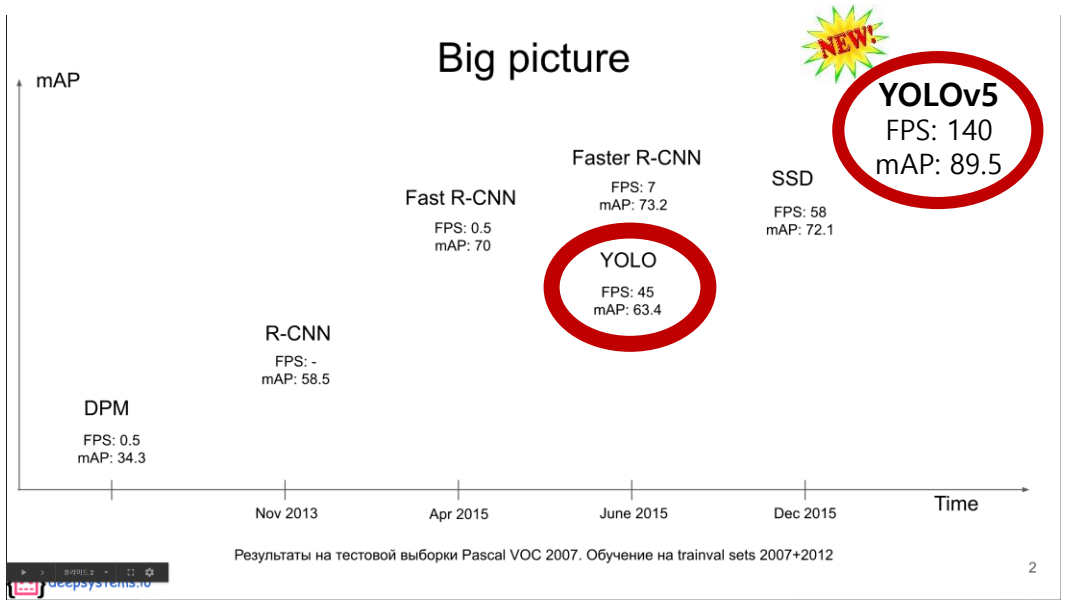
01. 9.1
YOLO Algorithm

03. 9.4~9.8
YOLO MARK

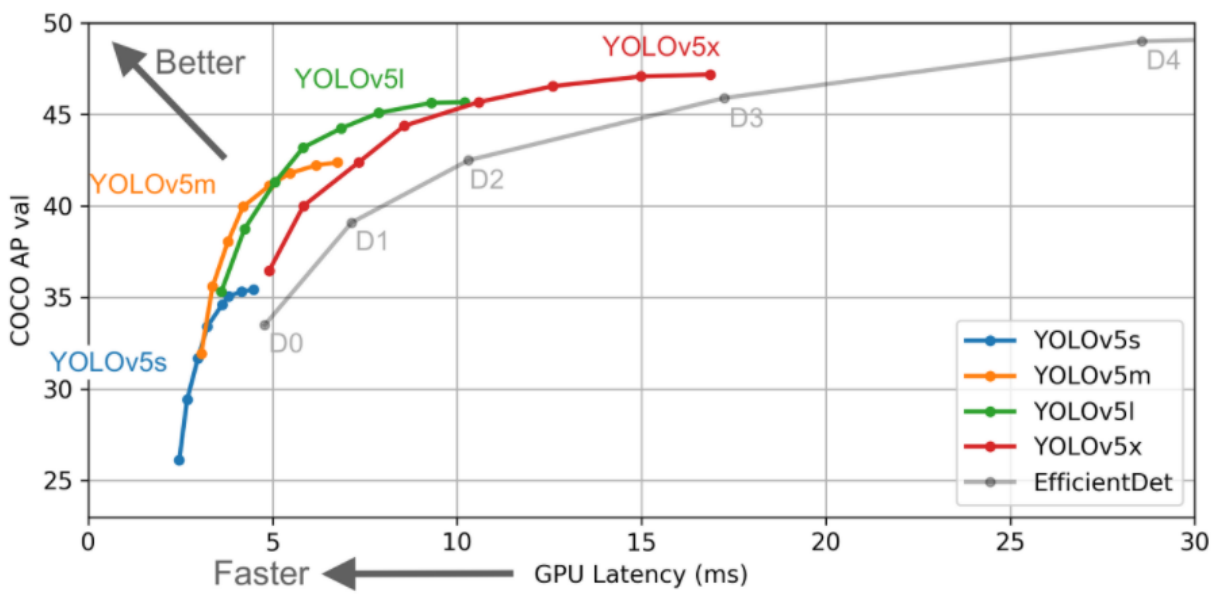
1 YOLO

9.1

YOLO란? You Only Look Once



YOLOv3

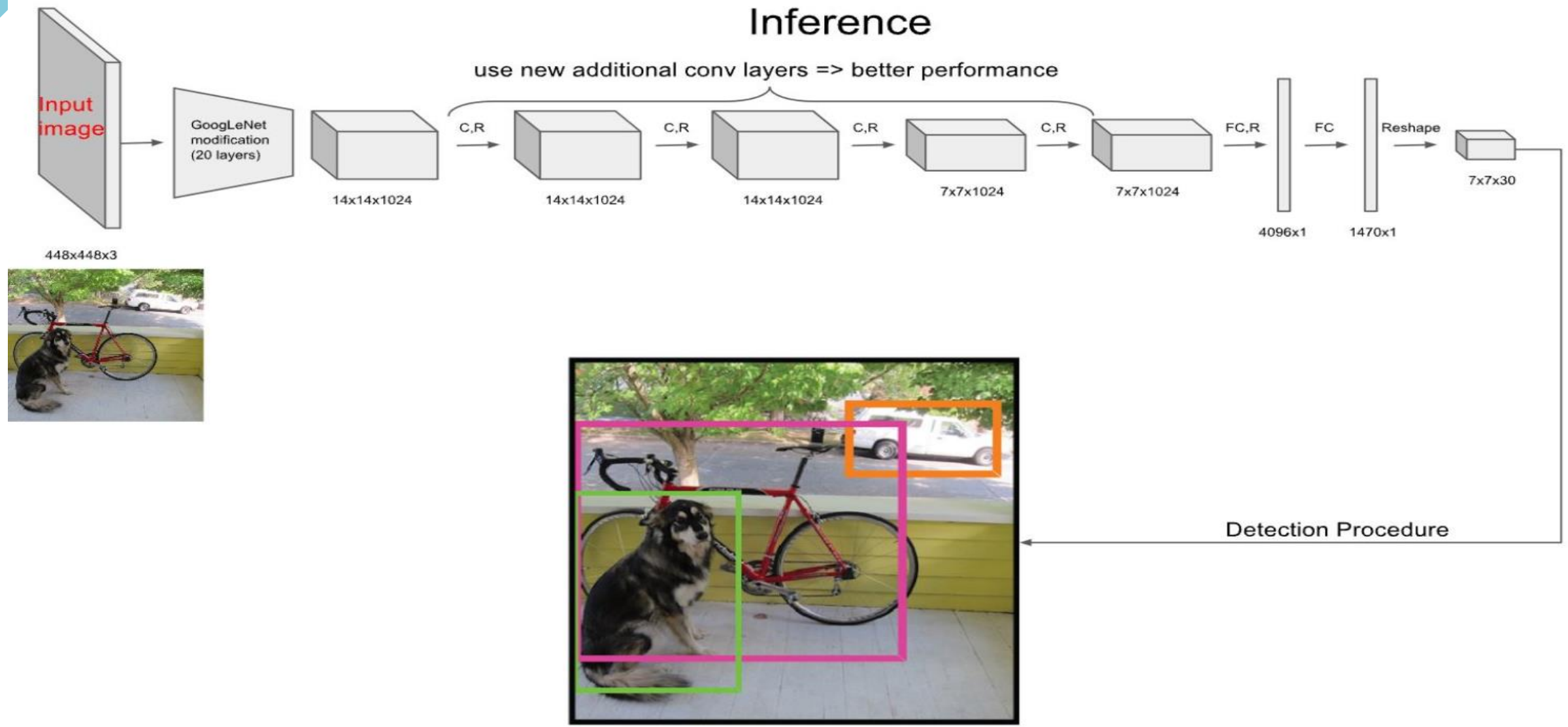


YOLOv5

https://docs.google.com/presentation/d/1aeRvtKG21KHdD5lg6Hgyhx5rPq_ZOsGjG5rJ1HP7BbA/pub?start=false&loop=false&delayms=3000&slide=id.p

https://medium.com/swlh/one-stop-for-object-detectors-2c99daa08c50

YOLO란? You Only Look Once



https://curt-park.github.io/images/yolo/DeepSystems-NetworkArchitecture.JPG

YOLO의 동작원리

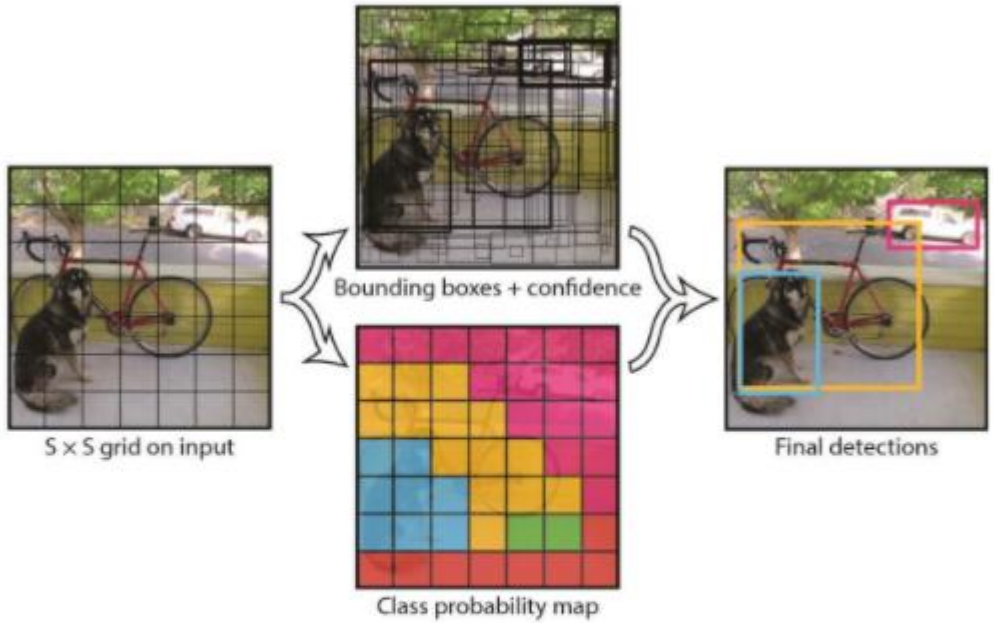


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

각 이미지를 SXS개의 Grid로 분할

신뢰도 계산 (경계 상자의 위치 조정)

객체 클래스 점수를 계산 (그리드에 객체 포함 여부 계산 목적)

S x S x N 객체가 예측

But 대부분 낮은 신뢰도 가짐

신뢰도를 높이기 위해 주변의 그리드 합침

임계값 설정해 불필요한 부분 제거 가능

2 OpenCV

9.2~9.3

>>>

Study Flow

Geometric Transformation

Edge Detection

Tracking

Geometric Transformation

- Color Transformation

Original

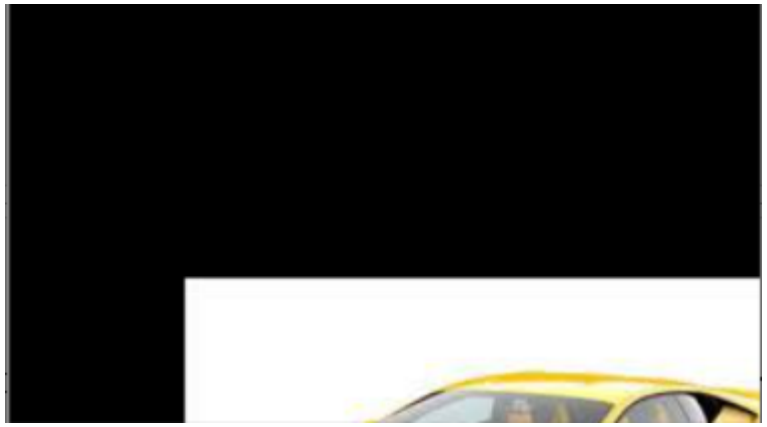
IMREAD_GRAYSCALE

COLOR_BGR2YUV

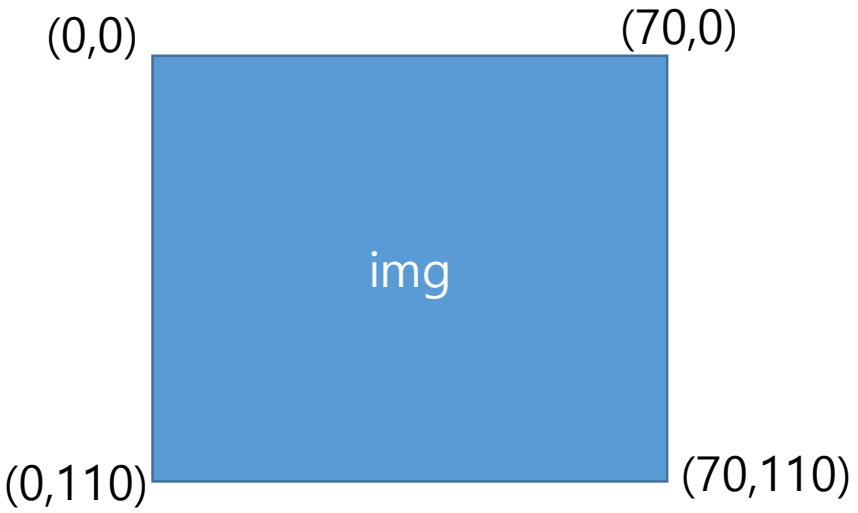
Geometric Transformation

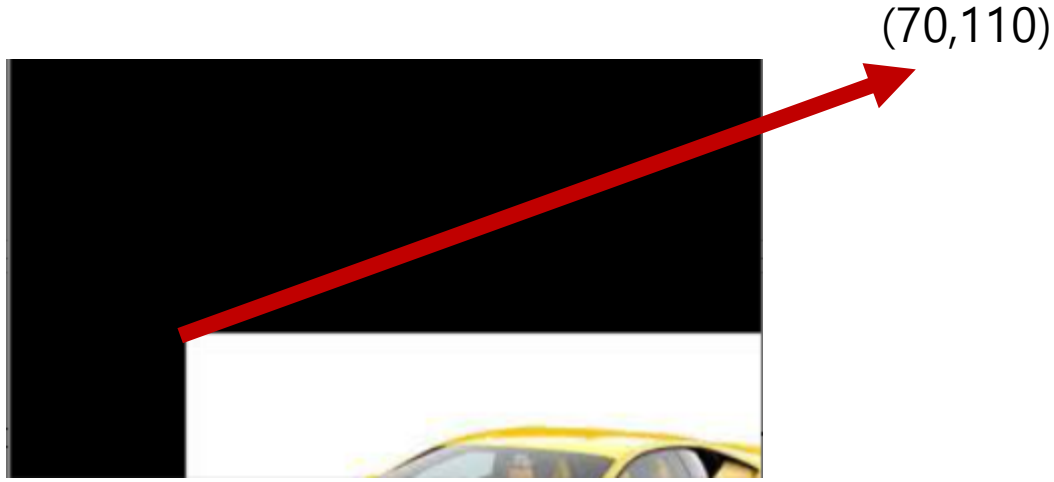
- Location Transformation

Original



warpAffine ->
[[1,0,70],[0,1,110]]





Geometric Transformation

- Angle Transformation

Original

Rotation

getRotationMatrix2D
Angle = 30 , scale = 0.7

Rotation
+ Move

getRotationMatrix2D
+ wrapAffine(row/2,col/2)

Geometric Transformation

- Interpolation

INTER_LINEAR

쌍선형 보간법

INTER_CUBIC

바이큐빅 보간법

INTER_AREA

영역 보간법

이미지

확대

이미지

축소

Geometric Transformation

- Image Warping

Vertical Wave

Horizontal Wave

Both Vertical and Horizontal

Concave effect

Edge Detection & Filtering

- Blurring

3x3 Blurring

5x5 Blurring

Motion Blurring

수평방향

Kernel의 크기가 높을 수록 더 넓은 지역을 평균화 하기 때문에

더 흐릿하게 보임

Edge Detection & Filtering

- Sharpening : Edge 향상

Original

Sharpening (Normalization O)

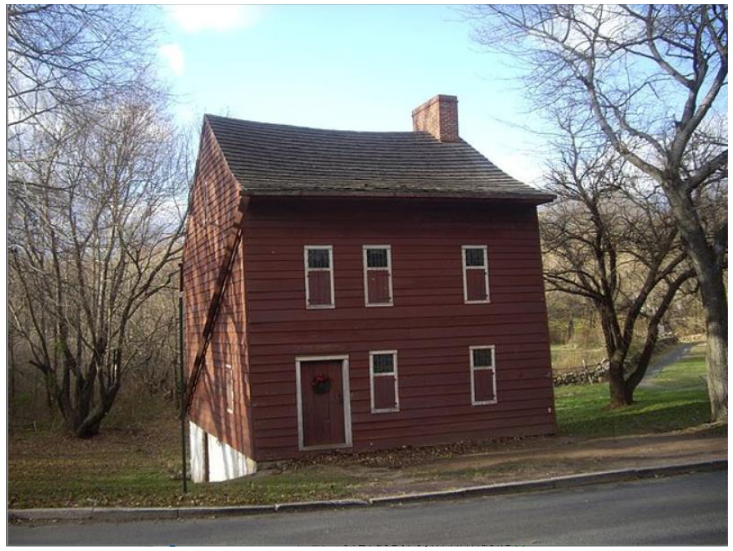
Excessive Sharpening (Normalization X)

Edge Enhancement (Normalization O)

Edge Detection & Filtering

- Embossing

Embossing 이란? 한쪽 면을 눌러 일정한 형태의 무늬가 도드라지게 만듦.



Original

Highlight

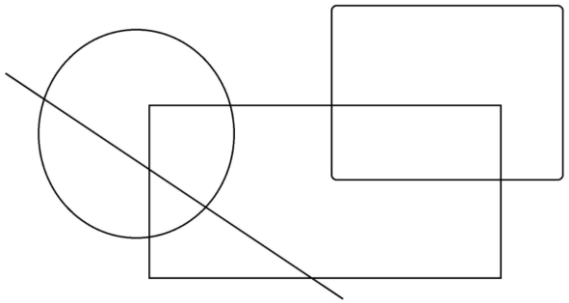
Shadow

||

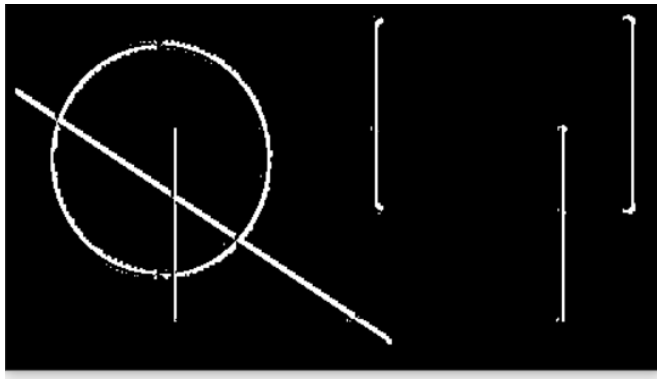
Highlight + 128

Edge Detection & Filtering

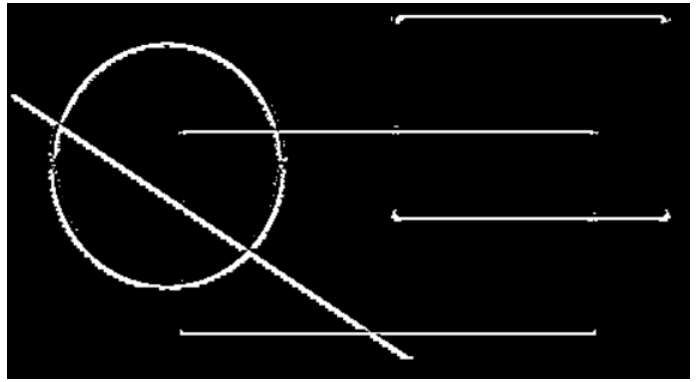
- Edge Detection



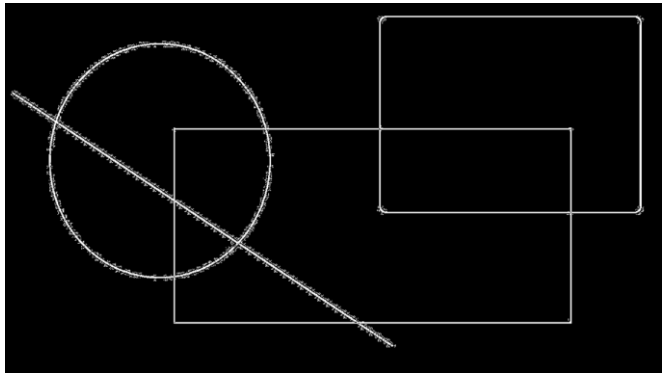
Original



Sobel_vertical



Sobel_horizontal

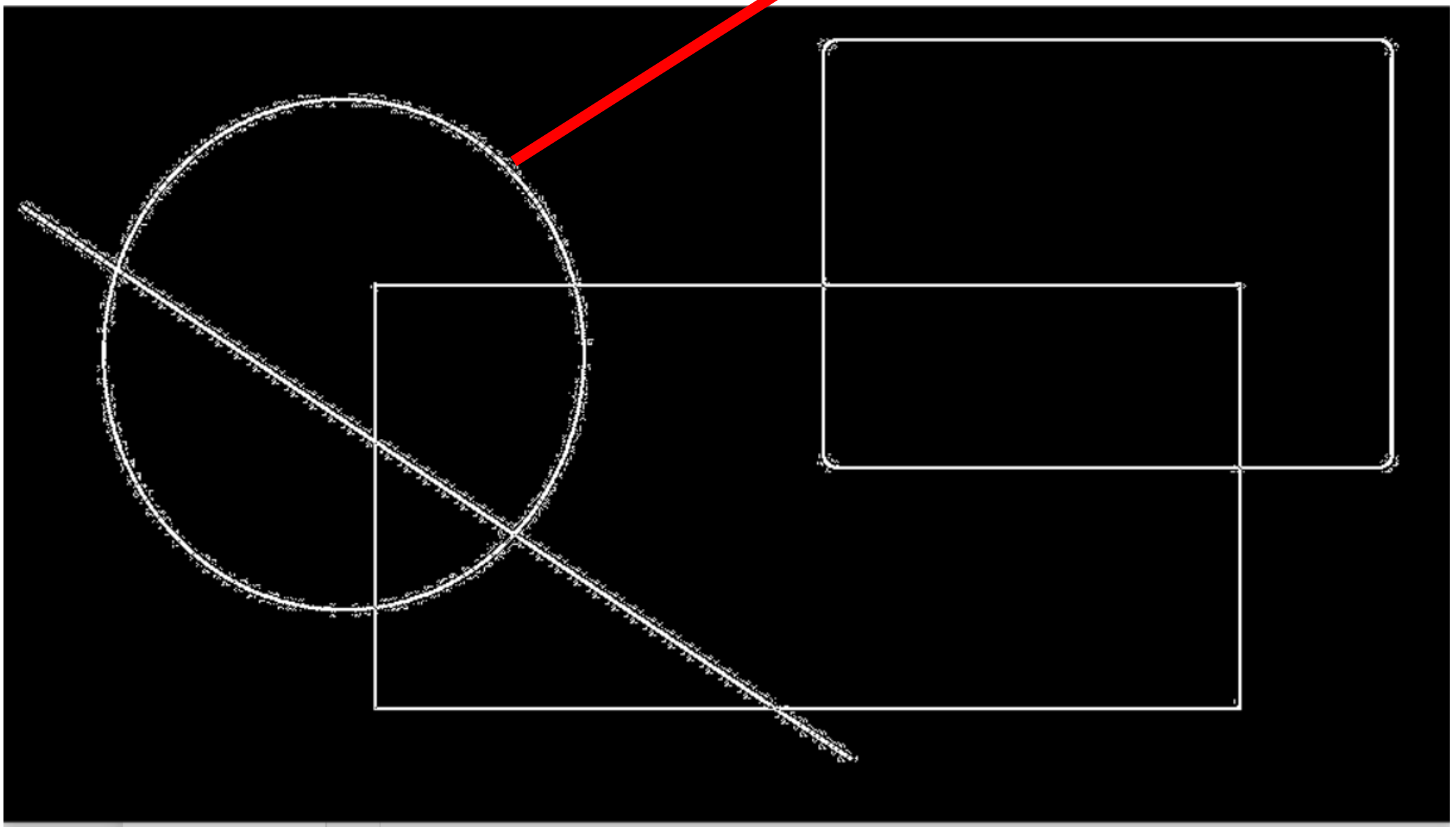


Laplacian

Canny

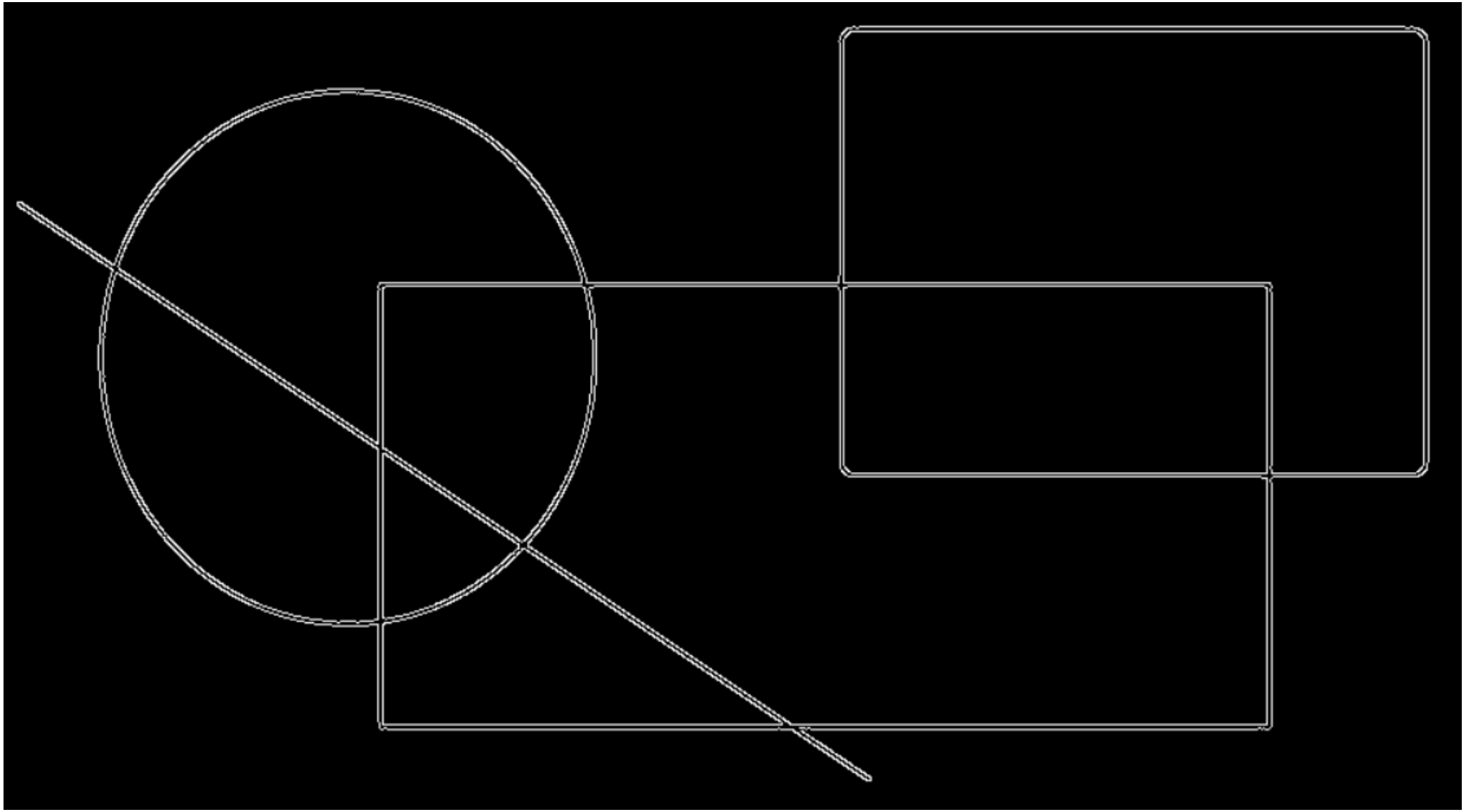
Edge Detection & Filtering

- Edge Detection - Laplacian



Edge Detection & Filtering

- Edge Detection - Canny



Edge Detection & Filtering

- Erode & Dilation

Erosion(침식)

가장 외부 화소를 벗겨냄

Dilation(팽창)

화소의 외부층을 더함

Iteration으로 침식과 팽창 정도 조절이 가능

Edge Detection & Filtering

- Vignette Filter & Brightness contrast

Vignette Filter(GaussianKernel)

가우시안의 표준편차는
밝기 영역의 반지름

Brightness contrast(Equalize Histogram)

단지 밝기값만 조절
비선형처리

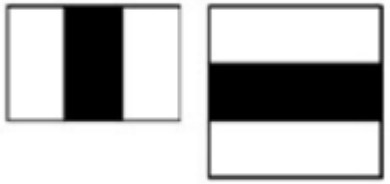
Tracking

- Haar Cascades

※ Haar Cascades : · ML 기반의 object 검출 알고리즘

· 특징 벡터를 만들기 위해 Haar 특징 사용

(a)Edge Features ⇒ 검은색 영역 합 – 흰색 영역 합



(b)Line Features ⇒ 중앙 검은색 사각형 영역 합 – 흰색 영역 합

(c) Four-Rectangle Features ⇒ 대각선 위치한 영역간의 차이

Tracking

- Face Detection

Tracking

- Eye Detection

In [*]:

```
face_cascade = cv2.CascadeClassifier('cascade_files/haarcascade_frontalface_alt.xml')
eye_cascade = cv2.CascadeClassifier('cascade_files/haarcascade_eye.xml')
if face_cascade.empty():
    raise IOError('Unable to load the face cascade classifier xml file')
if eye_cascade.empty():
    raise IOError('Unable to load the eye cascade classifier xml file')
cap = cv2.VideoCapture('img/bruno.avi')
ds_factor = 0.5
size = (int(cap.get(3)),int(cap.get(4)))
result = cv2.VideoWriter('avi_output/eye_bruno.avi',cv2.VideoWriter_fourcc(*'MJPG'),10,size) # save code
while True:
    ret,frame = cap.read()
    frame = cv2.resize(frame,None,fx=ds_factor,fy=ds_factor,interpolation=cv2.INTER_AREA)
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray,scaleFactor = 1.3,minNeighbors = 1)

    for (x,y,w,h) in faces:
        roi_gray = gray[y:y+h,x:x+w]
        roi_color = frame[y:y+h,x:x+w]
        eyes = eye_cascade.detectMultiScale(roi_gray)
        for (x_eye,y_eye,w_eye,h_eye) in eyes:
            center = (int(x_eye + 0.5*w_eye),int(y_eye+0.5*h_eye))
            radius = int(0.3*(w_eye + h_eye))
            color = (0,255,0)
            thickness = 3
            cv2.circle(roi_color,center,radius,color,thickness)
    cv2.imshow('Eye detector',frame)
    result.write(frame)
    c = cv2.waitKey(1)
    if c == 27:
        break
result.release()
cap.release()
cv2.destroyAllWindows()
```

Ear Detection

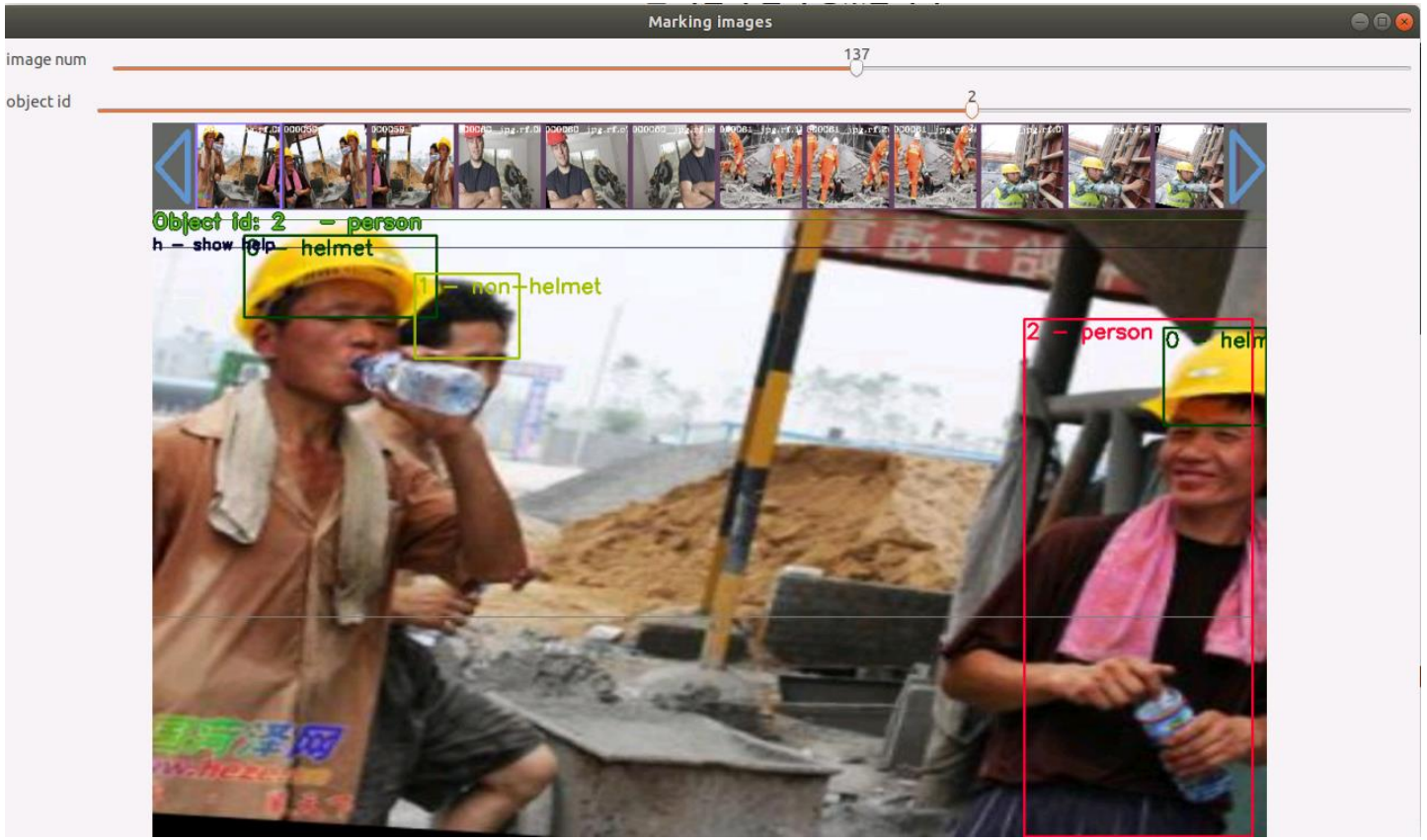
3 YOLO MARK

9.4 - 9.8

>>>

YOLO MRAK

- Definition



이미지파일에 직접 Bounding Box를 그리면서
Box의 좌표를 알 수 있다.

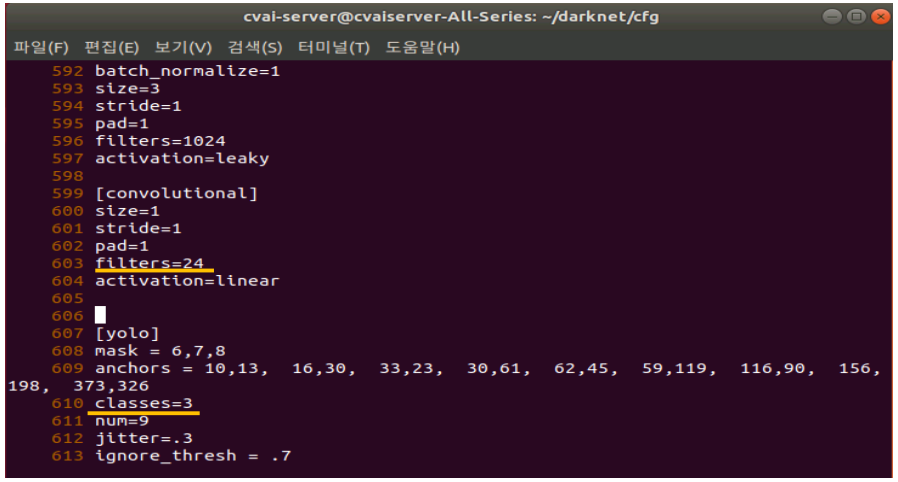
- Number of Images : 250
- Class 0 : Helmet
- Class 1 : Non-Helmet
- Class 2 : Person

Class	x	y	w	h
Class 0	0.168750	0.106250	0.173438	0.131944
Class 0	0.953516	0.265972	0.092969	0.156944
Class 1	0.282422	0.169444	0.094531	0.136111
Class 2	0.884766	0.586806	0.205469	0.826389

(x,y) : 중심점의 좌표, w : 넓이, h : 높이

YOLO MRAK

- Train

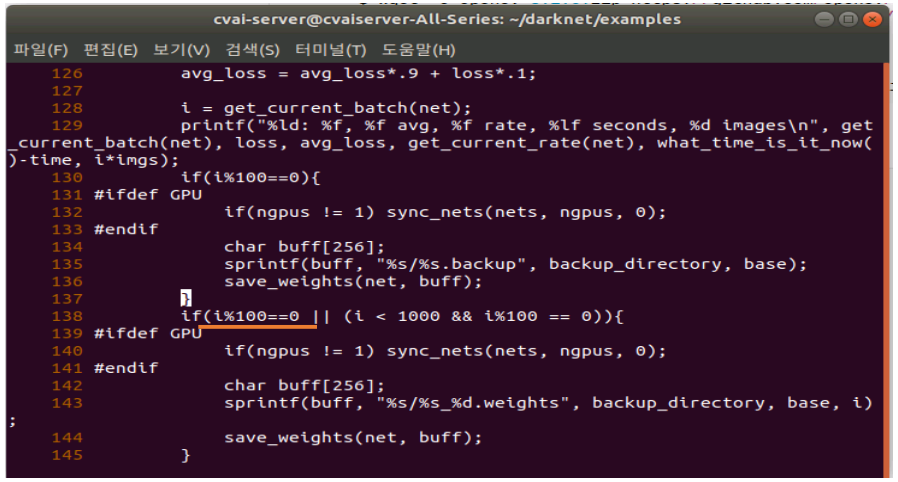


Classes = (helmet, non-helmet, person)

➔ **3**

Yolov3 : filters=(classes + 5) x 3

➔ **(3 + 5) x 3 = 24**



가중치 데이터가 저장되는 단위 변경

10000 ➔ 100

YOLO MRAK

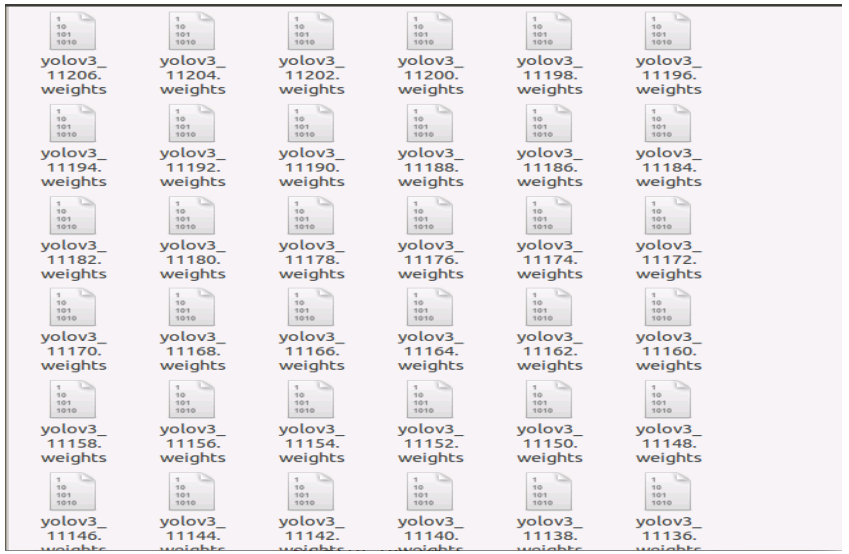
- Train

```
Region 94 Avg IOU: 0.171117, Class: 0.417521, Obj: 0.382767, No Obj: 0.458448, .5R: 0.000000, .75R: 0.000000,  count: 7
Region 106 Avg IOU: 0.133513, Class: 0.464519, Obj: 0.593565, No Obj: 0.467663, .5R: 0.000000, .75R: 0.000000,  count: 3
Region 82 Avg IOU: 0.149160, Class: 0.636569, Obj: 0.364883, No Obj: 0.494071, .5R: 0.000000, .75R: 0.000000,  count: 6
Region 94 Avg IOU: 0.383809, Class: 0.550680, Obj: 0.503506, No Obj: 0.459526, .5R: 0.000000, .75R: 0.000000,  count: 3
Region 106 Avg IOU: 0.151328, Class: 0.439671, Obj: 0.185648, No Obj: 0.467221, .5R: 0.000000, .75R: 0.000000,  count: 2
Region 82 Avg IOU: 0.327951, Class: 0.622584, Obj: 0.387336, No Obj: 0.493664, .5R: 0.285714, .75R: 0.000000,  count: 7
Region 94 Avg IOU: 0.291128, Class: 0.513116, Obj: 0.504167, No Obj: 0.460110, .5R: 0.166667, .75R: 0.000000,  count: 12
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.466911, .5R: -nan, .75R: -nan,  count: 0

8: 1086.300293, 1085.050659 avg, 0.000000 rate, 1601.097689 seconds, 512 image
Loaded: 0.000045 seconds
Region 82 Avg IOU: 0.308526, Class: 0.407153, Obj: 0.402713, No Obj: 0.492674, .5R: 0.250000, .75R: 0.000000,  count: 4
Region 94 Avg IOU: 0.274241, Class: 0.643794, Obj: 0.350882, No Obj: 0.459426, .5R: 0.230769, .75R: 0.000000,  count: 13
Region 106 Avg IOU: 0.125086, Class: 0.374554, Obj: 0.690089, No Obj: 0.466526, .5R: 0.000000, .75R: 0.000000,  count: 4
Region 82 Avg IOU: 0.161344, Class: 0.533201, Obj: 0.654204, No Obj: 0.492181, .5R: 0.000000, .75R: 0.000000,  count: 6
Region 94 Avg IOU: 0.297576, Class: 0.495495, Obj: 0.425217, No Obj: 0.459240, .5R: 0.222222, .75R: 0.000000,  count: 9
```

8번의 학습 진행중

– Average Loss rate 1085.050659



최종결과

- Average loss rate: 약 0.3

Weight 11206까지 저장

YOLO MRAK

- Test

4 차후 계획

9.9 ~

차후 계획

- YOLOv5의 개념 이해
- 3D 객체 탐지 모델인 Google Objectron에 대한 이해
- YOLOv5를 활용한 3D 객체 탐지 연구

차후 참고 자료

https://venturebeat.com/2020/03/11/googles-objectron-uses-ai-to-track-3d-objects-in-2d-video/

https://github.com/ultralytics/yolov5

THANKS

STUDY REPORT