Computer Vision #2

20 Jan 2021

자율주행시스템 개발팀
신 주 석

Driver Status Monitoring System

◆ DSM Summary

- 국내 및 세계 각 국의 교통사고 사망 원인으로 **졸음 운전 및 전방 주시 태만**이 큰 비중을 차지하는 것으로 분석.

- 이를 극복하기 위하여 자동차 제조사들은 운전 보조 시스템(Advanced Driver Assistance System, ADAS)에 운전자 상태 모니터링(Driver Status Monitoring) 시스템을 개발 및 제품으로 출시되고 있으며 ADAS 시장이 확대되는 것과 함께 고객 요구도 및 **시장이 확대 될 것으로 전망**.

- DSM 시스템은 **vision 기술을 이용**하여 얼굴 및 눈 인식, 객체 추적, 상태 판단 등의 알고리즘 처리를 통하여 구현.

- 차량 내 외부 조명 환경 및 야간에 강인하고, 신뢰할 수 있는 인식 성능을 가진 시스템 구현이 필요.

- 현재 기술 수준 및 차량 내 적용 사례는 vision 시스템을 중심으로 각종 센서를 이용하여 운전자의 상태를 모니터링 하고 이상 징후 발생 시 **알람 및 경고를 제공**해 주는 수준.

- 신뢰성 높은 제품을 개발하여 **운전자 인증**을 통한 편의 기능 제공 및 **ADAS 연계**를 통한 운전 환경 개선을 제공 함.

◆ DSM 개발 필요성

– Causes of Car Accidents

» 국내: 고속도로 교통사고 사망의 주요 원인 – 졸음운전

- 최근 5년간 고속도로 고속도로 교통사고 현황 – 1만2478건 사고 중 1473명 사망 (사망원인: **졸음운전(458명, 31%), 전방 주시 태만(425명, 28.8%)**, 과속(264명, 17.9%) 등 <2014.10.8, 아시아 경제>

» 일본: 교통사고 사망의 주요 원인 – 졸음운전 및 전방 주시 태만

- 2시간에 한명 꼴로 교통사고로 사망 – 2012년 년간 4500명 사망. 주요 사망원인: **졸음운전 또는 전방 주시 태만(40%)**, 속도 위반(18%) 등 <2013.12.19, Automotive Report>

» 북미 및 유럽

- OECD회원국 교통사고 비교 자료에 따르면 전 세계적으로 교통사고 빈도 수는 전반적으로 낮아지는 추세이며 그 중 북미가 교통사고가 가장 많이 발생하고 있다.(**졸음운전 사고 한해 약 6만 건 발생, 이 중 2만 건 사망사고**, NHTSA, 2002)
- 유럽의 국가들 중 독일의 경우 치명적 사고 중 25%가 운전자의 피로에 기인한 것이라는 통계자료가 있다.(독일 보험협회)

– Trend

» 주요 자동차 OEM은 ADAS 기능의 추가 및 개발에 투자를 하고 있으며, 졸음 감지 시스템 등의 운전자 모니터링 시스템에 기술 개발 투자 중

◆ DSM (주요 OEM 기술 현황)

- **BOSCH, DDD(Driver Drowsiness Detection) System (Vision system + sensors)**
  - » EPS 및 steering wheel angle sensor와 연동해 wheel의 움직임을 지속적으로 monitoring (warning signal or information 제공)

- **DENSO, Passenger Eye (Only Vision system)**
  - » 핸들 중앙에 설치된 카메라로 운전자 얼굴을 촬영해 얼굴의 방향이나 눈의 열린 상태를 감지하여 졸음운전이나 한눈을 팔면서 하는 운전 등이 벌어지면 운전자에게 경고

- **AISIN SEIKI (Vision system + sensors)**
  - » 핸들에 카메라 모듈을 설치하고 좌석에 압력 센서와 진동 장치를 내장한 뒤 운전자의 호흡이나 맥박 등을 감지. 카메라와 압력 센서를 통해 종합적인 판단으로 졸음 혹은 실신상태를 구별. 가벼운 졸음의 경우 음성 안내만으로 끝나지만, 깊은 졸음운전 및 실신 등 비상사태가 검출되면 음성 안내뿐만 아니라 좌석을 진동시켜 사고를 일으키기 전에 운전자를 깨우는 기능

<BOSCH>


<DENSO>


<AISIN SEIKI>

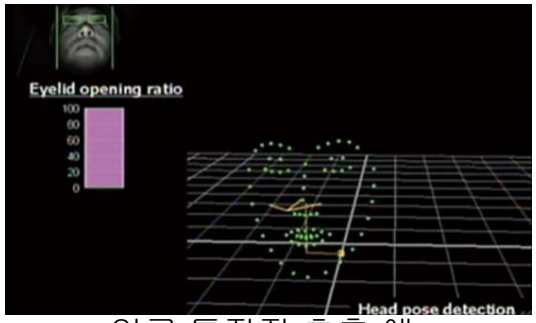
◆ DSM (주요 OEM 기술 현황)

– DENSO, Passenger Eye

» 17개의 얼굴 특징점을 이용하여 운전자의 상태 검출

» 6단계의 졸음 상태 정의 및 적용

<졸음 상태 정의 예>

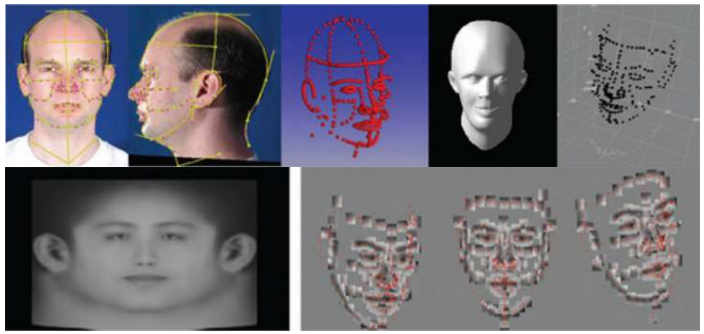


<얼굴 특징점 추출 예>

– TOYOTA & Image Science Division at the University of Manchester Biomedical Engineering

» 238개의 얼굴 특징점을 및 맨체스터 대학의 3D 텍스쳐 모델링 기법 이용

» 운전자 감정 인식을 이용한 서비스 계획



<3D 얼굴 모델링>

<3D 얼굴 특징점 매핑 예>

◆ DSM 소규모 Project 개요

– Concept

차량 주행 전

» 차량 탑승 시 운전자 인증을 통하여 운전 환경 자동 설정(핸들 위치 및 각도, 운전자 시트, 후사경 등)

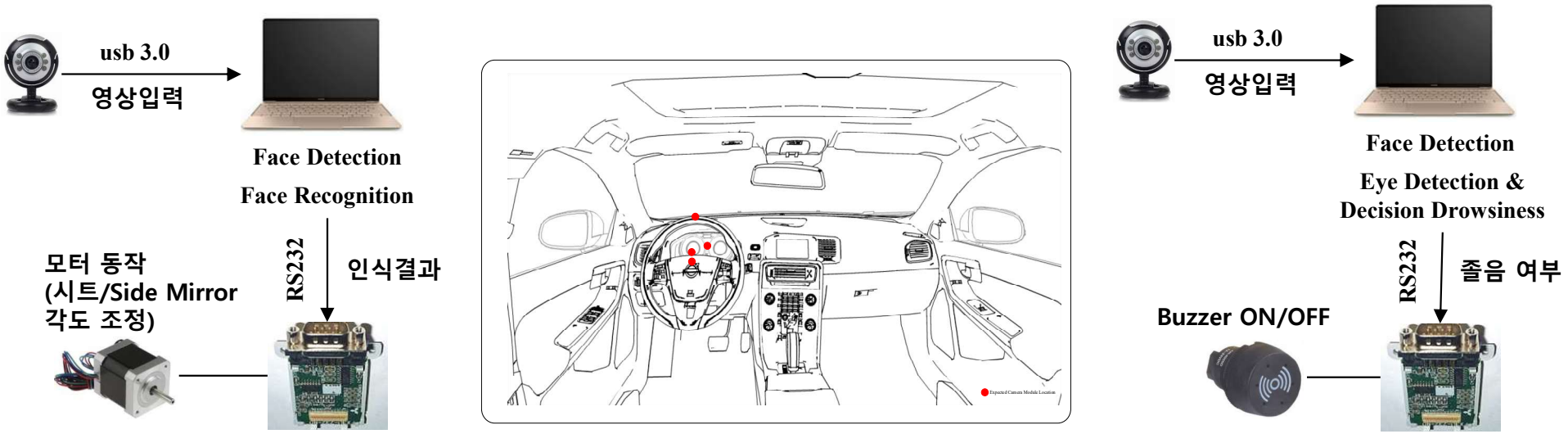
» 기존 시스템은 운전자가 바뀔 경우 수동으로 조작

차량 주행 중

» 차량 주행 중 운전자의 상태를 주기적으로 감시하여 운전자의 상태에 따른 편의 기능 혹은 경고 등을 제공

◆ DSM 소규모 Project 개요

– 시스템 구상도



차량 주행 전

» 차량 탑승 시 운전자 인증을 통하여 운전 환경 자동 설정(핸들 위치 및 각도, 운전자 시트, 후사경 등)

» 기존 시스템은 운전자가 바뀔 경우 수동으로 조작

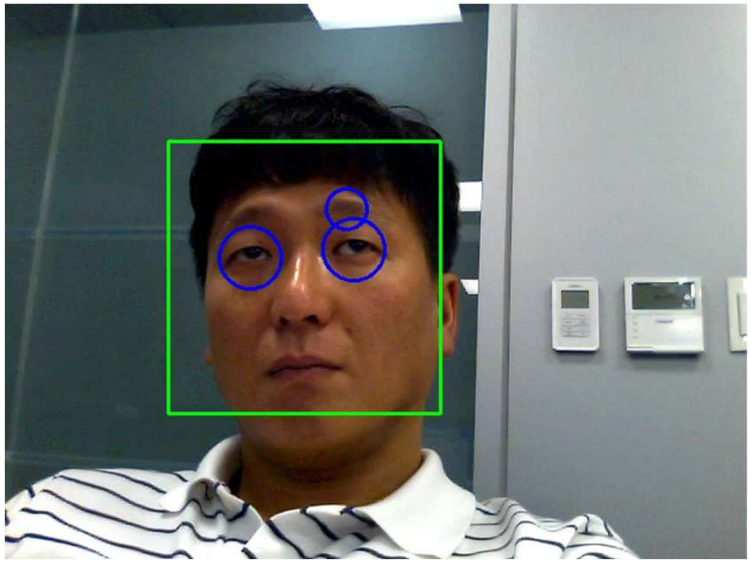
차량 주행 중

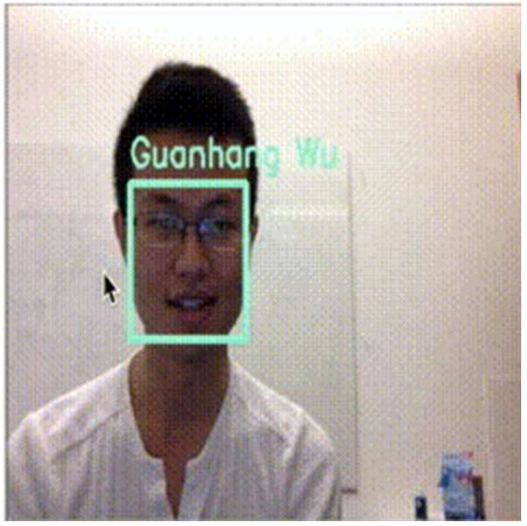
» 차량 주행 중 운전자의 상태를 주기적으로 감시하여 운전자의 상태에 따른 편의 기능 혹은 경고 등을 제공

+ SNOW App??

◆ DSM 소규모 Project 개요

– Computer Vision example





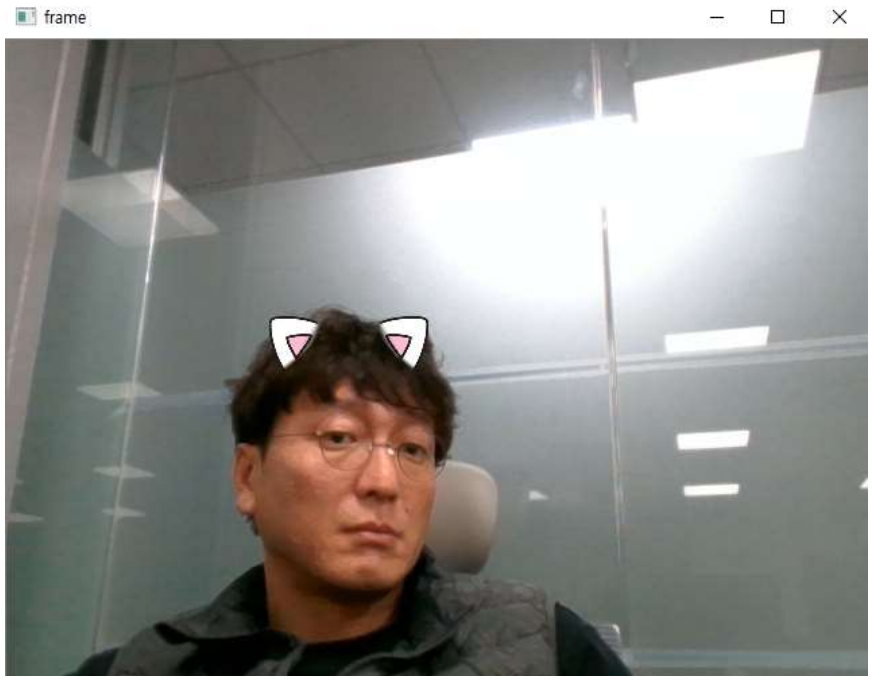
◆ DSM 소규모 Project 개요

- Computer Vision using Deep Learning

마스크 착용 여부....

◆ DSM 소규모 Project 개요

– Computer Vision example




```python
model = 'opencv_face_detector_uint8.pb'
config = 'opencv_face_detector.pbtxt'

cap = cv2.VideoCapture(0)

if not cap.isOpened():
    print('Camera open failed!')
    sys.exit()

net = cv2.dnn.readNet(model, config)

if net.empty():
    print('Net open failed!')
    sys.exit()

cat = cv2.imread('cat.png', cv2.IMREAD_UNCHANGED)
```

```python
blob = cv2.dnn.blobFromImage(frame, 1, (300, 300), (104, 177, 123))
net.setInput(blob)
detect = net.forward()

(h, w) = frame.shape[:2]
detect = detect[0, 0, :, :]

for i in range(detect.shape[0]):
    confidence = detect[i, 2]
    if confidence < 0.5:
        break

    x1 = int(detect[i, 3] * w + 0.5)
    y1 = int(detect[i, 4] * h + 0.5)
    x2 = int(detect[i, 5] * w + 0.5)
    y2 = int(detect[i, 6] * h + 0.5)

    fx = (x2 - x1) / cat.shape[1]
    cat2 = cv2.resize(cat, (0, 0), fx=fx, fy=fx)
    pos = (x1, y1 - (y2 - y1) // 4)
```

◆ DSM 소규모 Project 개요

– Computer Vision example

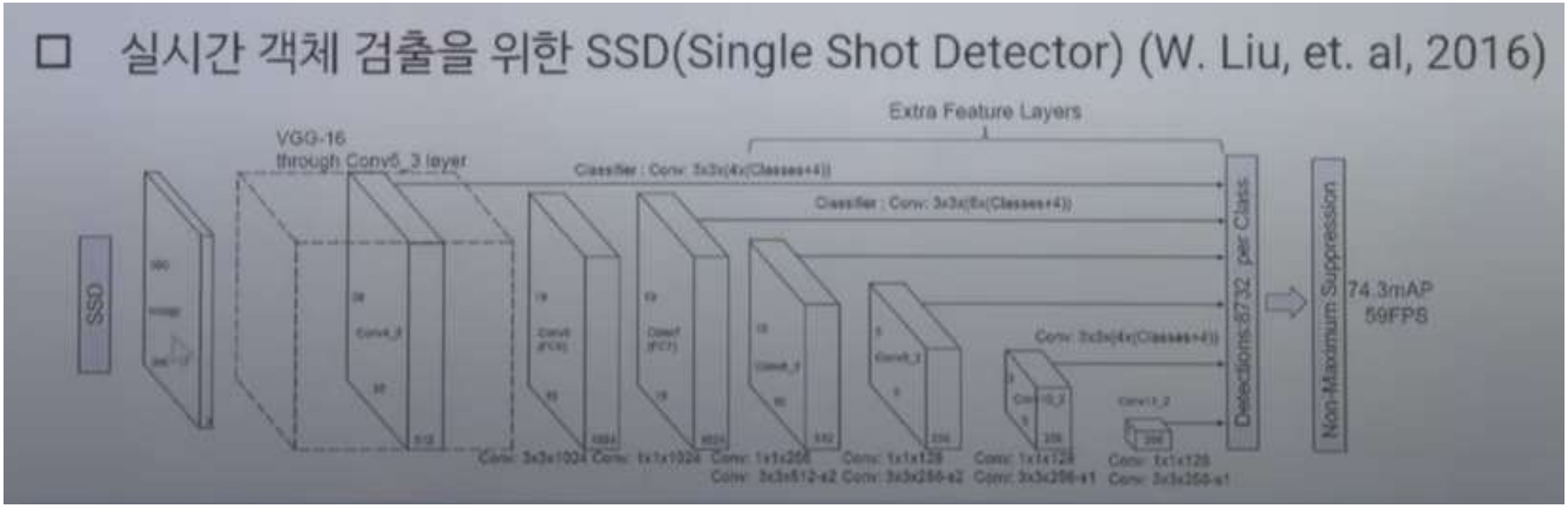
```
cv2.dnn.blobFromImage(image, scalefactor=None, size=None,
                      mean=None, swapRB=None, crop=None,
                      ddepth=None) -> retval
```

- image: 입력 영상
- scalefactor: 입력 영상 픽셀 값에 곱할 값. 기본값은 1.
- size: 출력 영상의 크기. 기본값은 (0, 0).
- mean: 입력 영상 각 채널에서 뺄 평균 값. 기본값은 (0, 0, 0, 0).
- swapRB: R과 B 채널을 서로 바꿀 것인지를 결정하는 플래그. 기본값은 False
- crop: 크롭(crop) 수행 여부. 기본값은 False.
- ddepth: 출력 블롭의 깊이. CV_32F 또는 CV_8U. 기본값은 CV_32F.
- retval: 영상으로부터 구한 블롭 객체. numpy.ndarray. shape=(N,C,H,W), dtype=float32

◆ DSM 소규모 Project 개요

– Computer Vision example

□ 실시간 객체 검출을 위한 SSD(Single Shot Detector) (W. Liu, et. al, 2016)



□ 입출력 형식

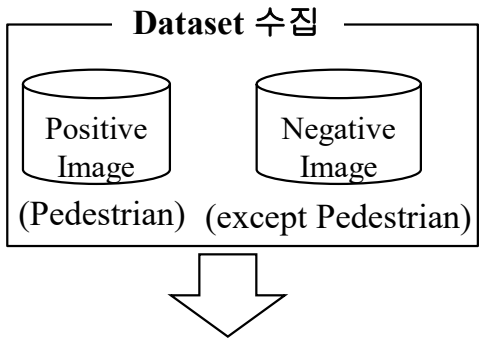
- 입력: 300x300, BGR, mean=(104, 177, 123)
- 출력: class, confidence, coordinate 등의 정보를 담고 있는 4차원 행렬
  shape=(1, 1, N, 7)

0	1	c	x1	y1	x2	y2
0	1	c	x1	y1	x2	y2
0	1	c	x1	y1	x2	y2
⋮	⋮	⋮	⋮	⋮	⋮	⋮

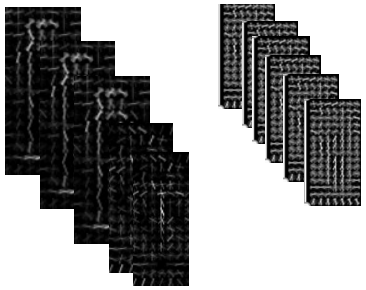
◆ Object Detection (e.g. Pedestrian)

– General & Conventional

off-line



Feature Extraction (HOG, MCT, etc.)



Classier (Adaboost, Random Forest, etc.)

PD Model

Pre-Processing

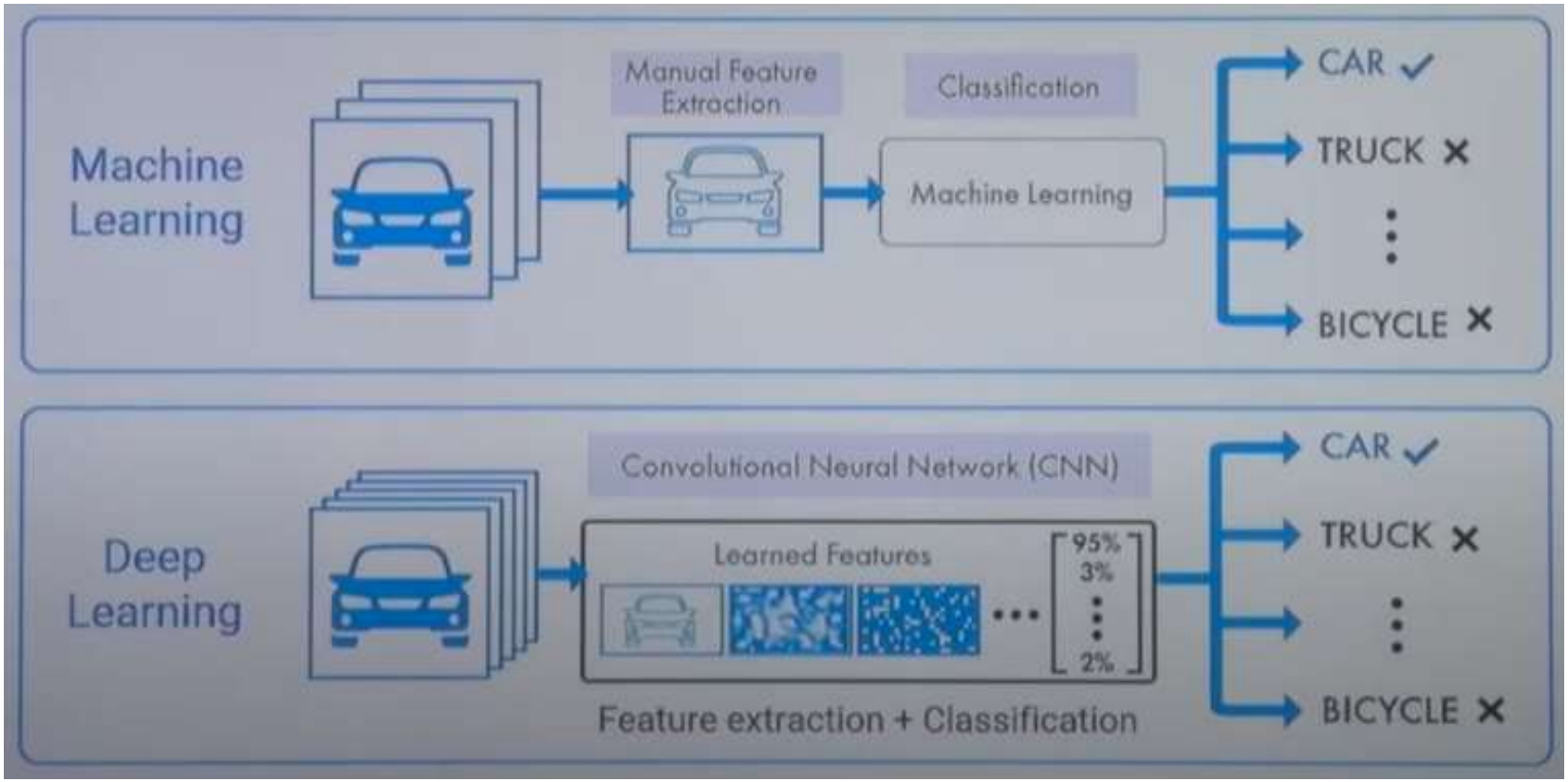
Gen. Pyramid Image

Gen. Feature Map (each scale)

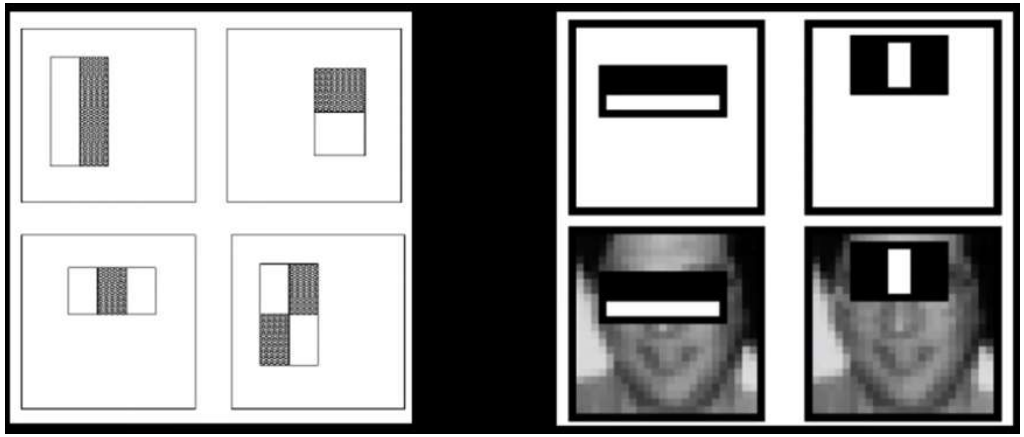
Scan Windows

NMS

- **Object Detection (e.g. Pedestrian)**
  - **General & Conventional**

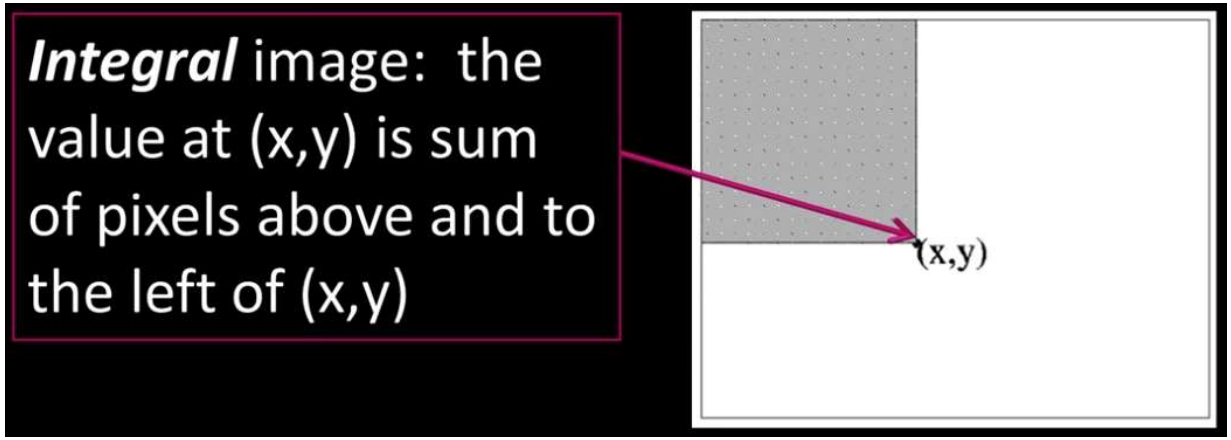


- **Face Detection (P. Viola & M. Jones)**
  - **Rapid Object Detection using Boosted Cascade of Simple Features**
    - » Objects: faces
  - **Main Idea**
    - » Features: “Rectangular” Filter (Haar-Like Feature)



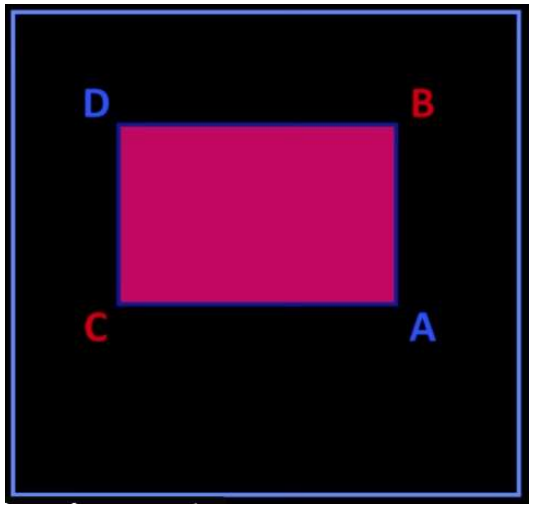
Difference in average intensity of adjacent regions

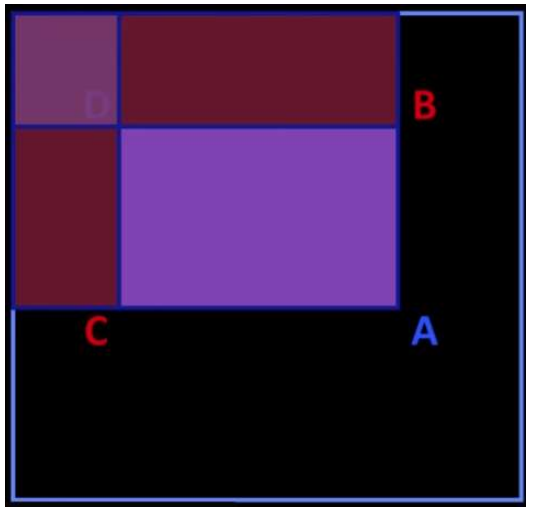
- » Integral Image



Face Detection (P. Viola & M. Jones)

- Rapid Object Detection using Boosted Cascade of Simple Features
- **Main Idea**
  - » Features: "Rectangular" Filter (Haar-Like Feature)
  - » Integral Image: Why this is very useful?
    - Only **2 minus(-) and 1 plus(+) operations are required for any size of rectangle**



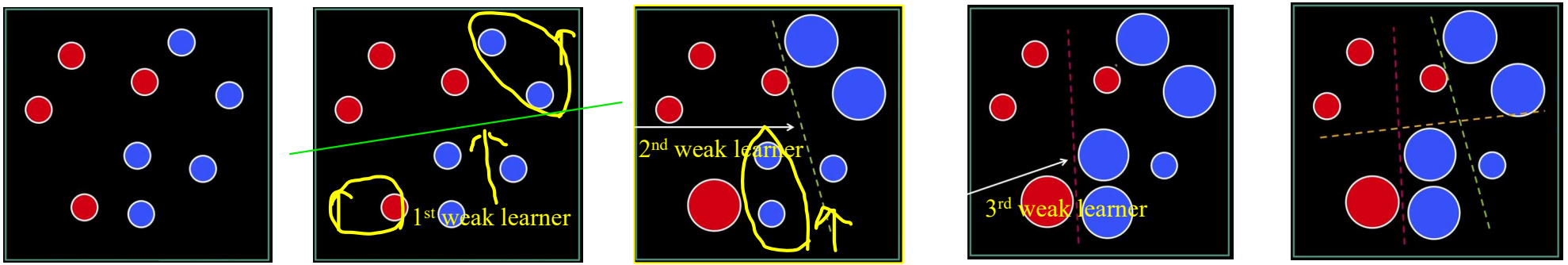


The Sum of original image values within the rectangle can be computed to integral image as:

$$sum = A - B - C + D$$

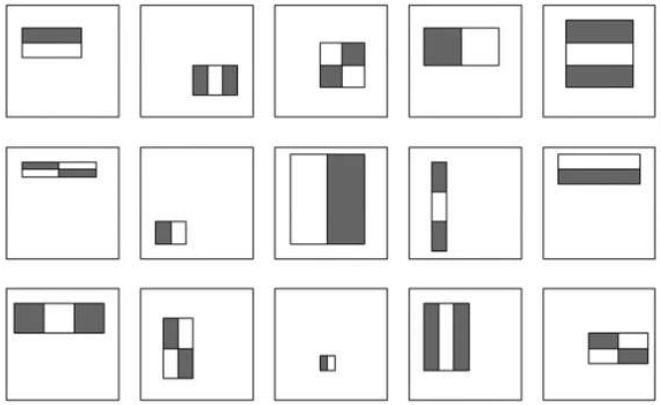
◆ Face Detection (P. Viola & M. Jones)

- Rapid Object Detection using Boosted Cascade of Simple Features
- **Main Idea**
  - » Features: "Rectangular" Filter (Haar-Like Feature)
  - » Integral Image
  - » Cascade Adaboost
    - ▪ Boosting: Iterative Learning Method
      - ✓ every iteration: calculates the weighted training error
      - ✓ Initially: weight each training example equally
      - ✓ In each boosting rounds
        - • **Find the weak Learner** that achieves the **lowest weighted training error**
        - • Raise weights of training examples misclassified by current weak learner
          - • weak learner: simply a function that partitions space
      - ✓ **Combines the weak learner:** Compute final Classifier as linear combination of all weak learners



Face Detection (P. Viola & M. Jones)

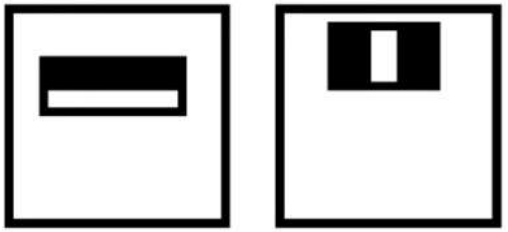
- Rapid Object Detection using Boosted Cascade of Simple Features
- **Main Idea**
  - » Features: "Rectangular" Filter (Haar-Like Feature)
  - » Integral Image
  - » Cascade Adaboost
    - Weak Learner(Classifier)

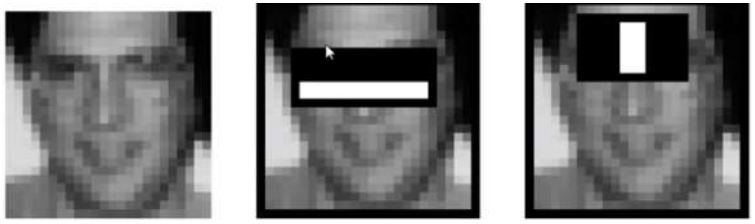


Considering all possible filter parameters – position, scale, and type:

180,000+ possible features associated with each 24 x 24 window

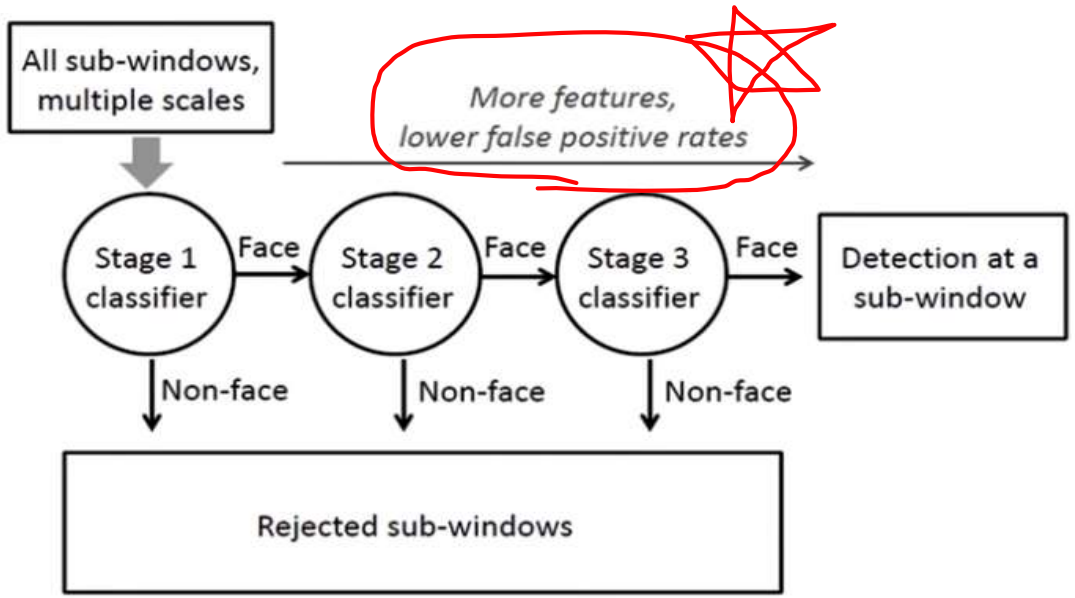
Choose discriminative features to be weak classifiers



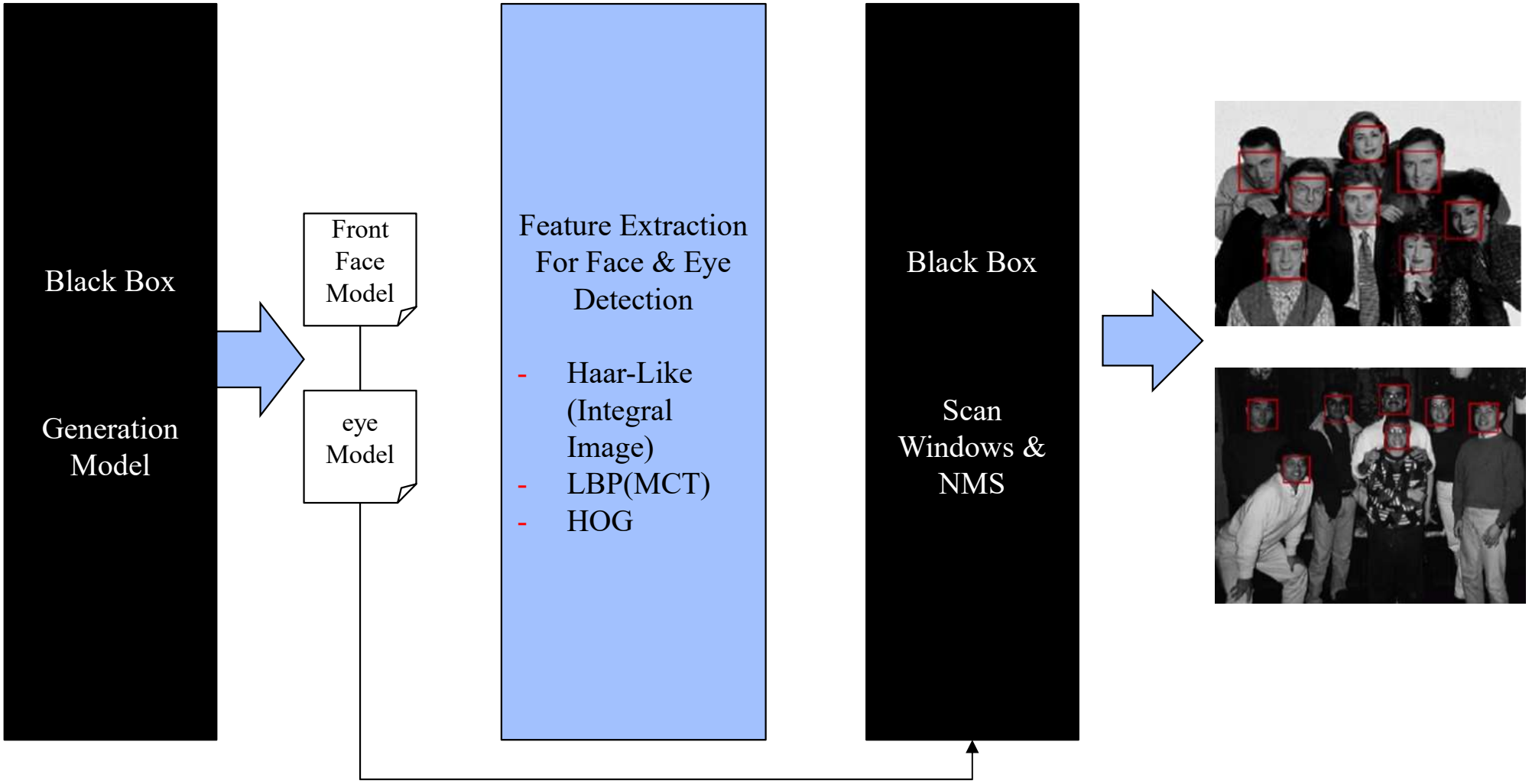


◆ Face Detection (P. Viola & M. Jones)

- Rapid Object Detection using Boosted Cascade of Simple Features
- **Main Idea**
  - » Features: "Rectangular" Filter (Haar-Like Feature)
  - » Integral Image
  - » Cascade Adaboost
    - ▪ **Rejecting clear negatives quickly**
      - ✓ **Key Idea: almost every where is a non-face**
        - **Detect non-faces more frequently than faces**
        - **If we can say it's not a face, be sure and move on at the window scanning**



◆ Face & Eye Detection



◆ Face & Eye Detection

– Using opencv library (detectMultiScale Library)

```
void cv::CascadeClassifier::detectMultiScale ( InputArray          image,
                                               std::vector< Rect > & objects,
                                               double              scaleFactor = 1.1,
                                               int                 minNeighbors = 3,
                                               int                 flags = 0,
                                               Size                minSize = Size(),
                                               Size                maxSize = Size()
                                             )
```

```
#define CV_HAAR_DO_CANNY_PRUNING    1
#define CV_HAAR_SCALE_IMAGE         2
#define CV_HAAR_FIND_BIGGEST_OBJECT 4
#define CV_HAAR_DO_ROUGH_SEARCH     8
```

Parameters

image	Matrix of the type CV_8U containing an image where objects are detected.
objects	Vector of rectangles where each rectangle contains the detected object, the rectangles may be partially outside the original image.
scaleFactor	Parameter specifying how much the image size is reduced at each image scale.
minNeighbors	Parameter specifying how many neighbors each candidate rectangle should have to retain it.
flags	Parameter with the same meaning for an old cascade as in the function cvHaarDetectObjects. It is not used for a new cascade.
minSize	Minimum possible object size. Objects smaller than that are ignored.
maxSize	Maximum possible object size. Objects larger than that are ignored. If `maxSize == minSize` model is evaluated on single scale.

```
//-- Detect faces
face_cascade.detectMultiScale( frame_gray, faces, 1.1, 2, 0|CV_HAAR_SCALE_IMAGE, Size(30, 30) );
```

◆ Face & Eye Detection

– Using opencv library (detectMultiScale Library)

```
#include "objdetect/objdetect.hpp"
String face_cascade_name = "haarcascade_frontalface_alt.xml"; //"
String eyes_cascade_name = "haarcascade_eye_tree_eyeglasses.xml";
CascadeClassifier face_cascade;
CascadeClassifier eyes_cascade;

int main()
{
    CvCapture* capture;
    Mat frame;

    //-- 1. Load the cascades
   if( !face_cascade.load( face_cascade_name ) ){ printf("--(!)Error loading\n"); return -1; };
   if( !eyes_cascade.load( eyes_cascade_name ) ){ printf("--(!)Error loading\n"); return -1; };

   //-- 2. Read the video stream
   capture = cvCaptureFromCAM( 0 );
   if( capture )
   {
     while( true )
     {
         frame = cvQueryFrame( capture );

     //-- 3. Apply the classifier to the frame
       if( !frame.empty() )
       { detectAndDisplay( frame ); }
       else
       { printf(" --(!) No captured frame -- Break!"); break; }

       int c = waitKey(10);
       if( (char)c == 'c' ) { break; }
      }
   }

    return 0;
}
```

◆ Face & Eye Detection

– Using opencv library (detectMultiScale Library)

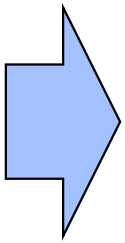
```
void detectAndDisplay( Mat frame )
{
  std::vector<Rect> faces;
  Mat frame_gray;

  cvtColor( frame, frame_gray, CV_BGR2GRAY );
  equalizeHist( frame_gray, frame_gray );

  //-- Detect faces
  face_cascade.detectMultiScale( frame_gray, faces, 1.1, 2, 0|CV_HAAR_SCALE_IMAGE, Size(30, 30) );

  Rect faceRect;
  for( size_t i = 0; i < faces.size(); i++ )
  {
      faceRect.x=faces[i].x;
      faceRect.y=faces[i].y;
      faceRect.width=faces[i].width;
      faceRect.height=faces[i].height;

      rectangle(frame, faceRect, Scalar( 0, 255, 0 ), 2, 8, 0);

    Mat faceROI = frame_gray( faces[i] );
    std::vector<Rect> eyes;

    //-- In each face, detect eyes
    eyes_cascade.detectMultiScale( faceROI, eyes, 1.1, 2, 0 |CV_HAAR_SCALE_IMAGE, Size(30, 30) );

    for( size_t j = 0; j < eyes.size(); j++ )
     {
       Point center( faces[i].x + eyes[j].x + eyes[j].width*0.5, faces[i].y + eyes[j].y + eyes[j].height*0.5 );
       int radius = cvRound( (eyes[j].width + eyes[j].height)*0.25 );
       circle( frame, center, radius, Scalar( 255, 0, 0 ), 2, 8, 0 );
     }
  }
   imshow( window_name, frame );
}
```

◆ Face & Eye Detection (Homework)

– **Using opencv library (detectMultiScale Library)**

off-line

Number	LBP Histogram	remark
1	tb_1	조**
2	tb_2	이**
3	tb_3	문**



Face Detection
(using Adaboost or Deep Learning)

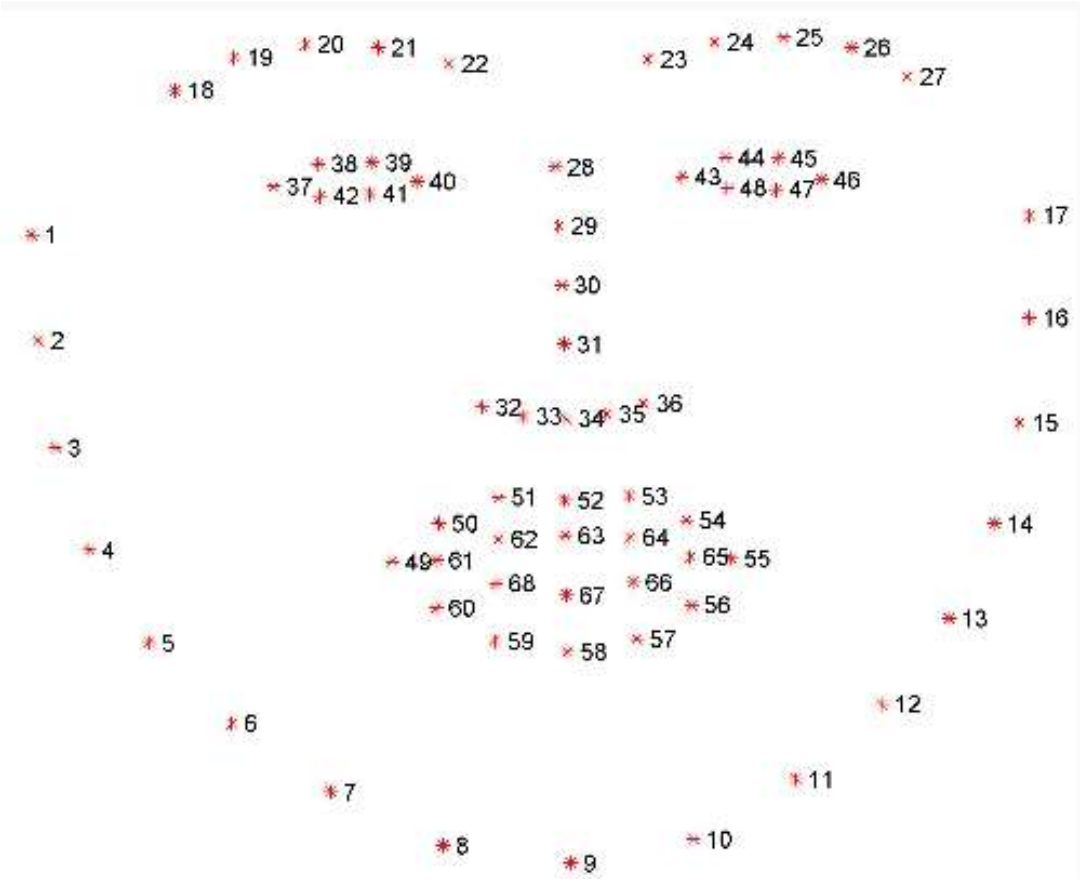
Face Recognition
(using Deep Learning)

Rs232

printf

◆ Face & Eye Detection

- Using Facial Landmark
- Using python: dlib & opencv
  - » **Face & Facial landmark detector** implemented inside **dlib**
  - » Produces **68 (x,y)-coordinates** (specific facial structures)
  - » These 68 points mappings were obtained by training a shape predictor on the labeled iBUG300-W dataset



◆ Face & Eye Detection

- Using Facial Landmark
- Using python: dlib & opencv

```
python facial_landmark_detect.py --shape-predictor shape_predictor_68_face_landmarks.dat
```

```
# initialize dlib's face detector (HOG-based) and then create
# the facial landmark predictor
print("[INFO] loading facial landmark predictor...")
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])
```

→ HOG + Linear SVM

```
while True:
    # grab the frame from the threaded video file stream, resize
    # it, and convert it to grayscale
    # channels)
    frame = vs.read()
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # detect faces in the grayscale frame
    rects = detector(gray, 0)

    # loop over the face detections
    for rect in rects:
        # determine the facial landmarks for the face region, then
        # convert the facial landmark (x, y)-coordinates to a NumPy
        # array
        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)

        # loop over the (x, y) coordinates for the facial landmaks
        # draw landmark on the image
        for(x,y) in shape:
            cv2.circle(frame, (x,y), 1, (0,255,0), -1)
    # show the frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
```

Detection Face

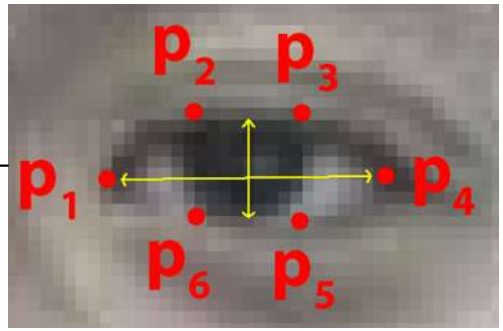
Detection Landmark

◆ Drowsiness Detection based on Face & Eye Detection using landmark

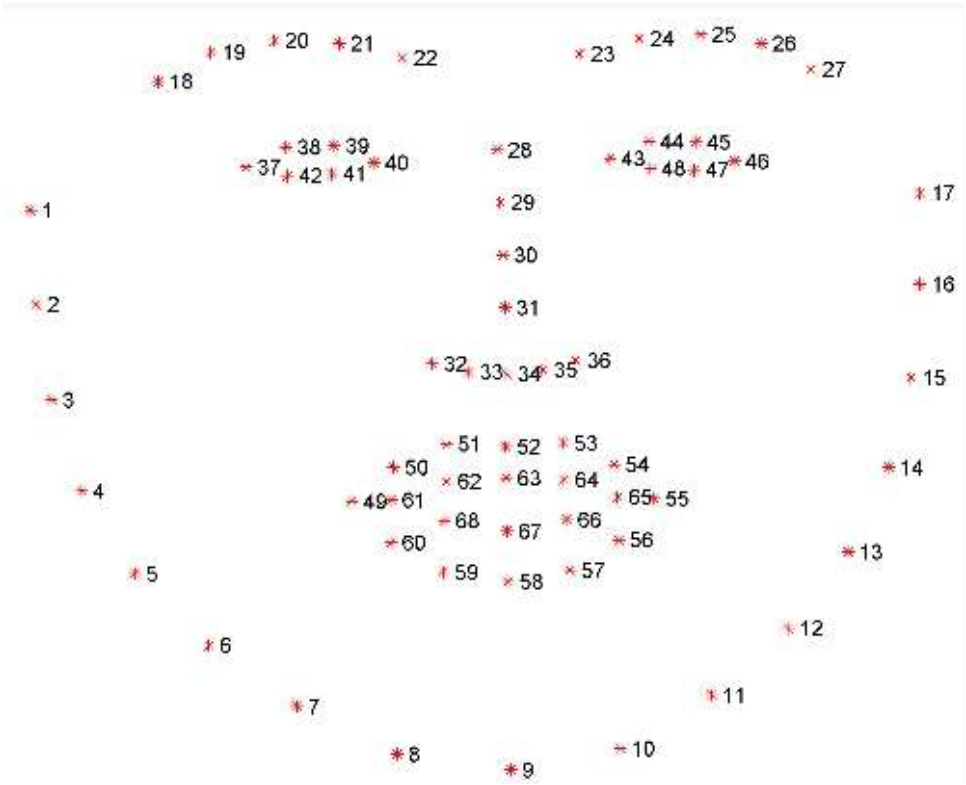
– Paper: Real-Time Eye Blink Detection using Facial Landmarks

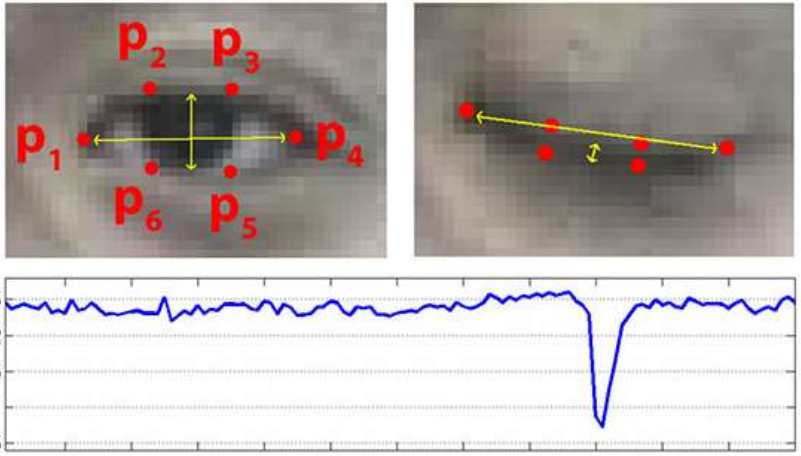
» Using EAR(Eye Aspect Ratio)

$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$



Using this simple equation, we can *avoid image processing techniques* and simply rely on the *ratio of eye landmark distances* to determine if a person is blinking.



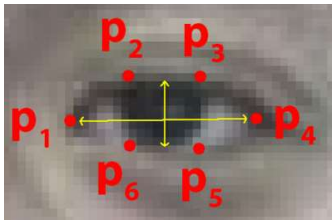


On the *top-left* we have an eye that is fully open — the eye aspect ratio here would be large and relatively constant over time. However, once the person blinks (*top-right*) the eye aspect ratio decreases dramatically, approaching zero.

◆ Drowsiness Detection based on Face & Eye Detection using landmark

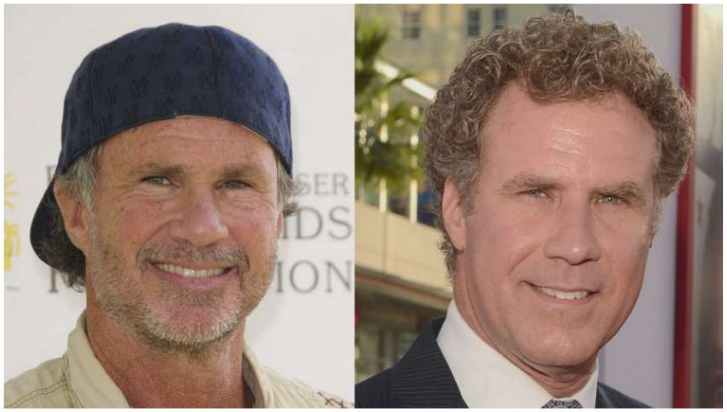
– Paper: Real-Time Eye Blink Detection using Facial Landmarks

```python
def eye_aspect_ratio(eye):
    # compute the euclidean distances between the two sets of
    # vertical eye landmarks (x, y)-coordinates
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])

    # compute the euclidean distance between the horizontal
    # eye landmark (x, y)-coordinates
    C = dist.euclidean(eye[0], eye[3])

    # compute the eye aspect ratio
    ear = (A + B) / (2.0 * C)

    # return the eye aspect ratio
    return ear

EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 48
```




```python
for rect in rects:
    # determine the facial landmarks for the face region, then
    # convert the facial landmark (x, y)-coordinates to a NumPy
    # array
    shape = predictor(gray, rect)
    shape = face_utils.shape_to_np(shape)

    # extract the left and right eye coordinates, then use the
    # coordinates to compute the eye aspect ratio for both eyes
    leftEye = shape[lStart:lEnd]
    rightEye = shape[rStart:rEnd]
    leftEAR = eye_aspect_ratio(leftEye)
    rightEAR = eye_aspect_ratio(rightEye)

    # average the eye aspect ratio together for both eyes
    ear = (leftEAR + rightEAR) / 2.0

    # compute the convex hull for the left and right eye, then
    # visualize each of the eyes
    leftEyeHull = cv2.convexHull(leftEye)
    rightEyeHull = cv2.convexHull(rightEye)
    cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
    cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)

    # check to see if the eye aspect ratio is below the blink
    # threshold, and if so, increment the blink frame counter
    if ear < EYE_AR_THRESH:
        COUNTER += 1

        # if the eyes were closed for a sufficient number of
        # then sound the alarm
        if COUNTER >= EYE_AR_CONSEC_FRAMES:
            # if the alarm is not on, turn it on
            if not ALARM_ON:
                ALARM_ON = True
```

◆ Face Recognition using dlib, openface



Chad Smith Will Ferrell

Is it easy to distinguish between these two person?

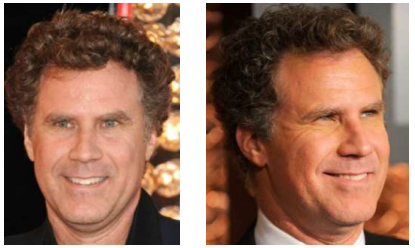
– **Face recognition pipeline**

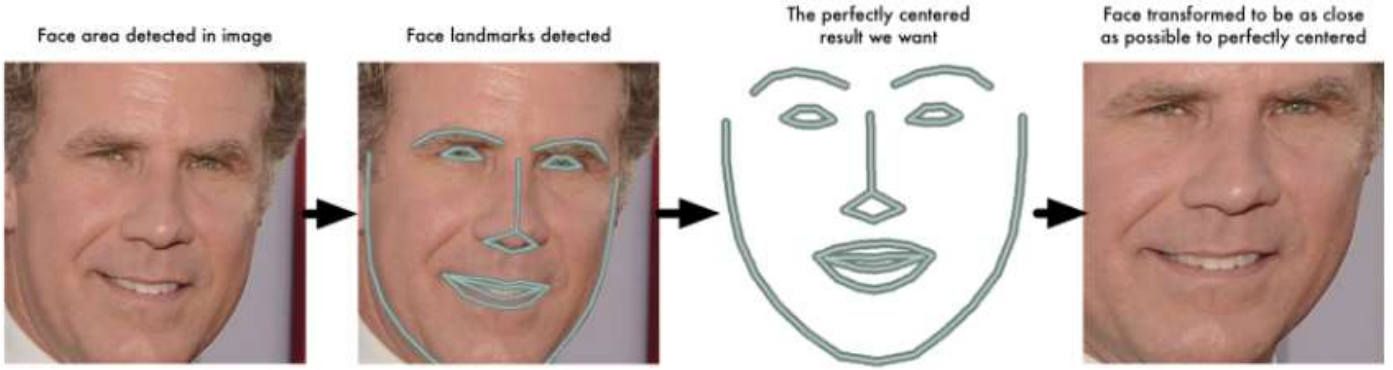
» 사진 또는 영상 내에 모든 얼굴 찾기 (dlib: HOG Feature + linear SVM)

» 얼굴의 위치 교정과 투영

▪ 얼굴이 다른 방향을 보면 전혀 다른 사람으로 인식하는 문제 해결

✓Dlib facial landmark & opencv affine transformations

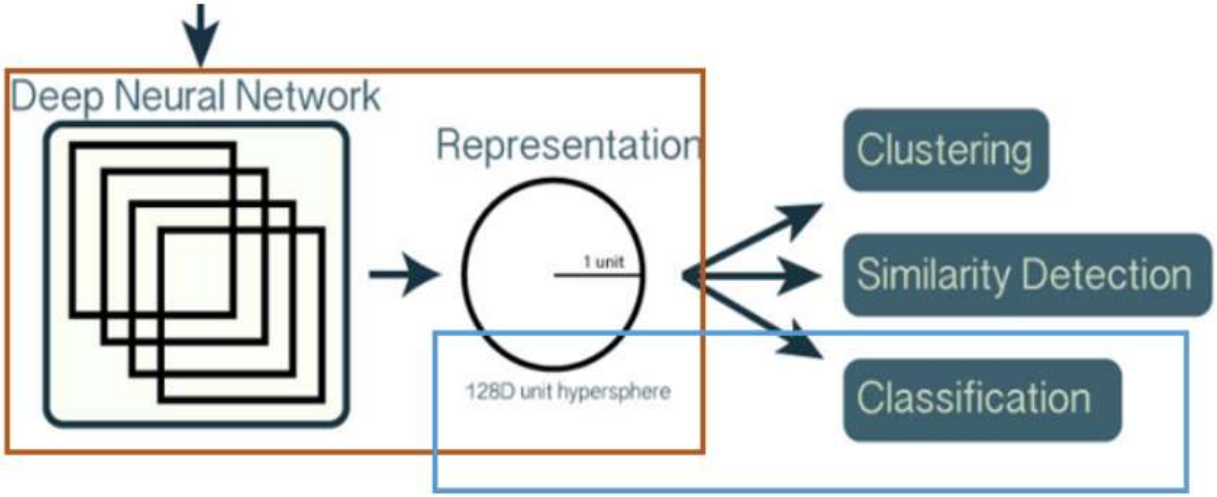


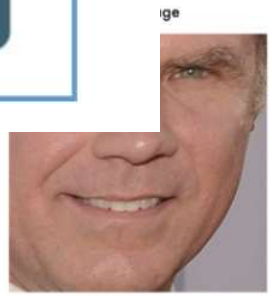


https://medium.com/@jongdae.lim/기계-학습-machine-learning-은-즐겁다-part-4-63ed781eee3c

◆ Face Recognition using dlib, openface

- **Face recognition pipeline**
  - » 사진 또는 영상 내에 모든 얼굴 찾기 (dlib: HOG Feature + linear SVM)
  - » 얼굴의 위치 교정과 투영 (Facial Landmark & affine transform)
  - » Embedding: 각 얼굴에 대해 128개의 측정 값 도출
    - Deep Learning을 통하여 사람 얼굴을 가장 잘 구분할 수 있는 128개의 측정값을 생성하는 것을 학습
      - ✓ OpenFace에서 이미 잘 학습된 모델 사용





128 Measurements Generated from Image

0.097496084868908	0.045223236083984	-0.1281466782093	0.032084941864014
0.12529824674129	0.060309179127216	0.17521631717682	0.020976085215807
0.030809439718723	-0.019814772531139	0.10801389068365	-0.0005216327845118
0.036050599068403	0.065554238855839	0.0731306001544	-0.1318951100111
-0.097486883401871	0.12262628972 53	-0.029626874253154	-0.0059557510539889
-0.006640171166509 4	0.036750309169292	-0.15958009660244	0.043374512344599
-0.14131525158882	0.14114324748516	-0.031351584941149	-0.053343612700701
-0.048540540039539	-0.061901587992907	-0.15042643249035	0.07819810509681 7
-0.12567175924778	-0.10568545013666	-0.12728653848171	-0.076289616525173
-0.061418771743774	-0.074287034571171	-0.065365232527256	0.12369467318058
0.046741496771574	0.0061761881224811	0.14746543765068	0.056418422609568
-0.12113650143147	-0.21055991947651	0.0041091227903962	0.089727647602558
0.061606746166945	0.11345765739679	0.021352224051952	-0.0085843298584223
0.061989940702915	0.19372203946114	-0.086726233363152	-0.022388197481632
0.10904195904732	0.084853030741215	0.09463594853878	0.020696049556136
-0.019414527341723	0.0064811296761036	0.21180312335491	-0.050584398210049
0.15245945751667	-0.16582328081131	-0.035577941685915	-0.072376452386379
-0.12216668576002	-0.0072777755558491	-0.036901291459799	-0.034365277737379
0.083934605121613	-0.059730969369411	-0.070026844739914	-0.045013956725597
0.087945111095905	0.11478432267904	-0.089621491730213	-0.013955107890069
-0.021407851949334	0.14841195940971	0.078333757817745	-0.17898085713387
-0.018298890441656	0.049525424838066	0.13227833807468	-0.072600327432156
-0.011014151386917	-0.051016297191381	-0.14132921397686	0.0050511928275228
0.0093679334966328	-0.062812767922878	-0.134074985698099	-0.014829395338893
0.058139257133007	0.0048638740554452	-0.039491076022387	-0.043765489012003
-0.024210374802351	-0.11443792283535	0.071997955441475	-0.012062266469002
-0.057223934680223	0.014683869667351	0.052281547337 77	0.012774495407939
0.023535015061498	-0.081752359867096	-0.031709920614958	0.069833360612392
-0.0098039731383324	0.037022035568953	0.11009479314089	0.116387888789 18
0.020220354199409	0.12788131833076	0.18632389605045	-0.015336792916059
0.0040337680839002	-0.094398014247417	-0.11768248677254	0.10281457751989
0.051597066223621	-0.10034311562777	-0.040977258235216	-0.082041338086128

◆ Face Recognition using dlib, openface

- **Face recognition pipeline**
  - » 사진 또는 영상 내에 모든 얼굴 찾기 (dlib: HOG Feature + linear SVM)
  - » 얼굴의 위치 교정과 투영 (Facial Landmark & affine transform)
  - » Embedding: 각 얼굴에 대해 128개의 측정 값 도출 (using Pre-Trained model)
  - » 인코딩에서 사람의 이름 찾기
    - 각각의 얼굴에 해당하는 128개의 값을 SVM으로 학습 => 모델 생성
  - » 분류기의 결과: 얼굴 인식
- **구동 방법**
  - » Step1: Make a subfolder for each person you want to recognize.

```
mkdir ./training-images/will-ferrell/
mkdir ./training-images/chad-smith/
mkdir ./training-images/jimmy-fallon/
```

  - » Step2: Copy all images of each person into sub-folders
  - » Step3: Run the openface scripts from inside the openface root directory

```
./util/align-dlib.py ./training-images/ align outerEyesAndNose ./aligned-images/ --size 96
```

    - 얼굴 검출 (한 장에 한 명만 존재할 것) + 얼굴 위치 교정 및 투영

◆ Face Recognition using dlib, openface

– 구동 방법

» Step1: Make a subfolder for each person you want to recognize.

» Step2: Copy all images of each person into sub-folders

» Step3: Run the openface scripts from inside the openface root directory

```
./util/align-dlib.py ./training-images/ align outerEyesAndNose ./aligned-images/ --size 96
```

```
./batch-represent/main.lua -outDir ./generated-embeddings/ -data ./aligned-images/
```

- 각각의 영상에서 Embedding 값 추출 (./generated-embeddings/ sub-folder will contain a csv file with the embeddings for each image)

```
./demos/classifier.py train ./generated-embeddings/
```

- 각각의 얼굴에 해당하는 128개의 값을 SVM으로 학습 => 모델 생성 (This will generate a new file called ./generated-embeddings/classifier.pkl. This file has the SVM model you will use to recognize new faces.

» Step4: Get a new picture with an unknown face. Pass it to the classifier script like below:

```
./demos/classifier.py infer ./generated-embeddings/classifier.pkl your_test_image.jpg
```

```
=== /test-images/will-ferrel-1.jpg ===
Predict will-ferrell with 0.73 confidence.
```

◆ Face Recognition using dlib, openface

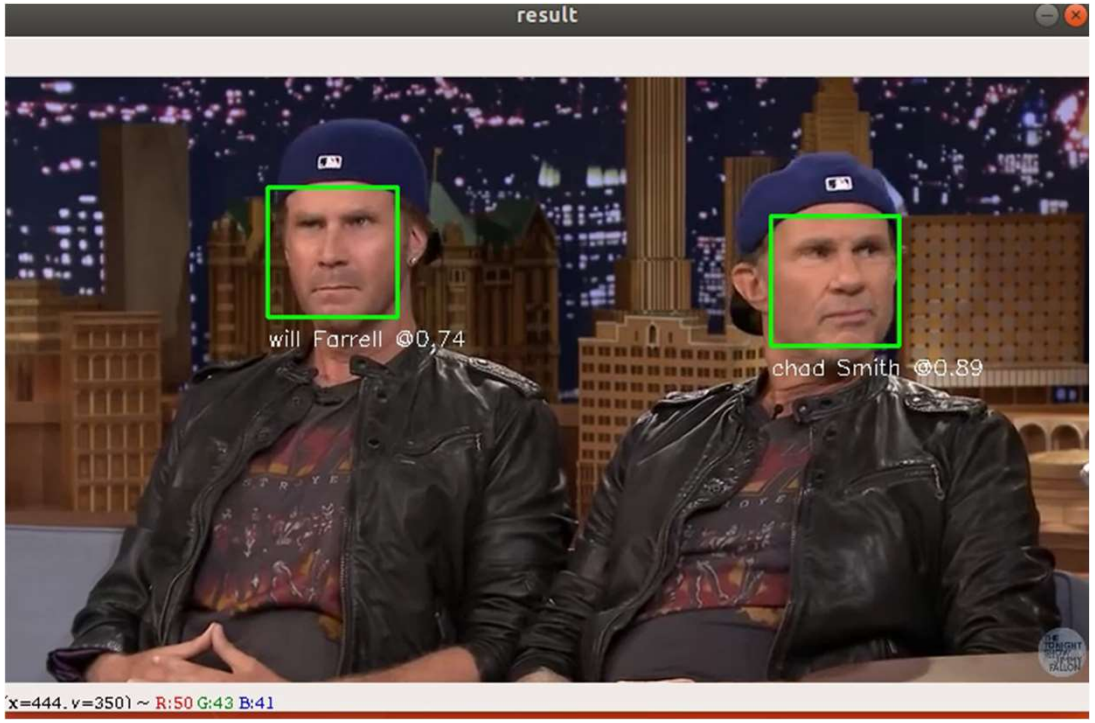
– 구동 방법

» Step1: Make a subfolder for each person you want to recognize.

» Step2: Copy all images of each person into sub-folders

» Step3: Run the openface scripts from inside the openface root directory

» Step4: Get a new picture with an unknown face. Pass it to the classifier script



Thank you & Good luck !