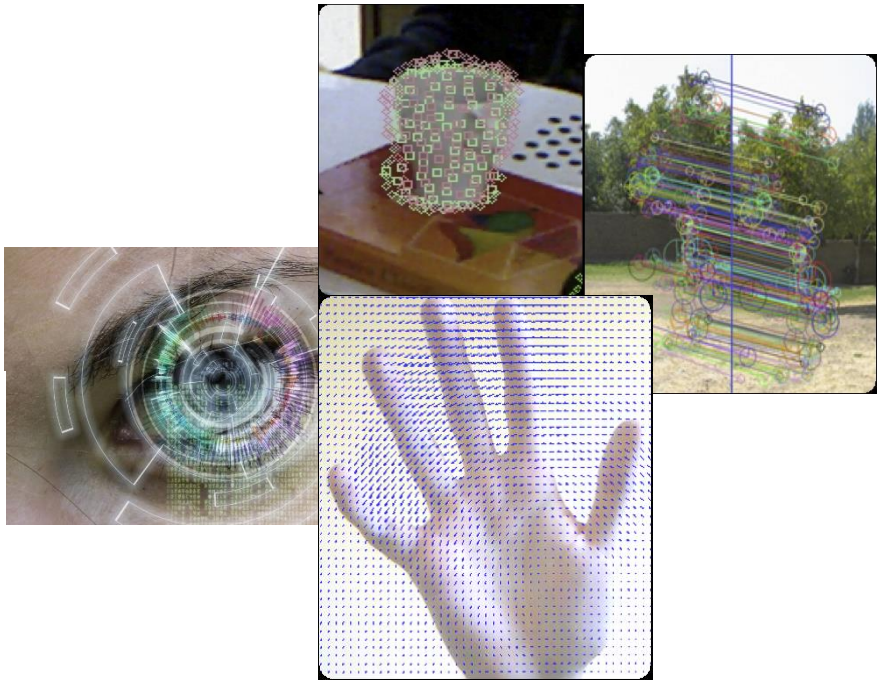
IVPL
Intelligent Vision Processing Lab

2023, 2-nd Semester

COMPUTER VISION 비젼 프로그래밍



Dept. of IT Engineering, Sookmyung Women's University
Prof. Byung-Gyu Kim

9장. 특징 기반 인식 기법
(Feature-based Detection and Recognition)

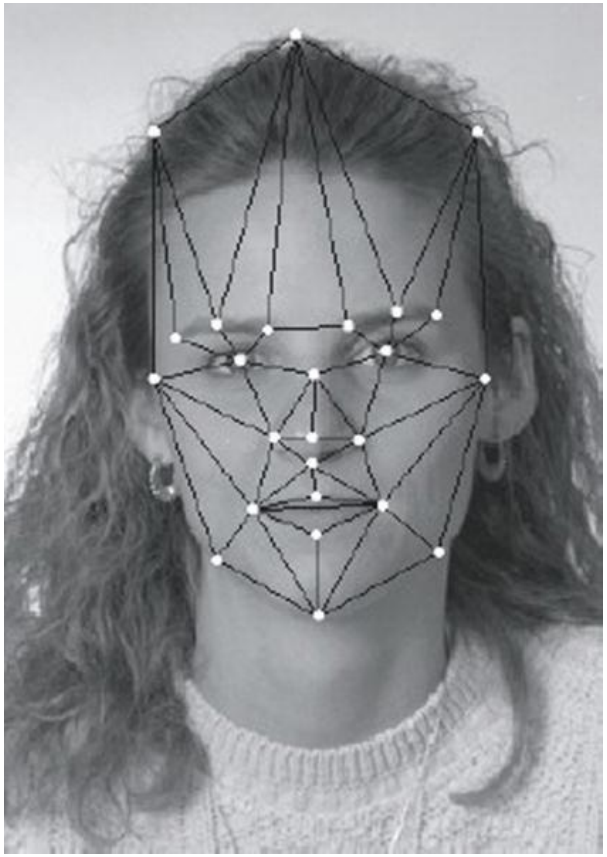
학습 목표

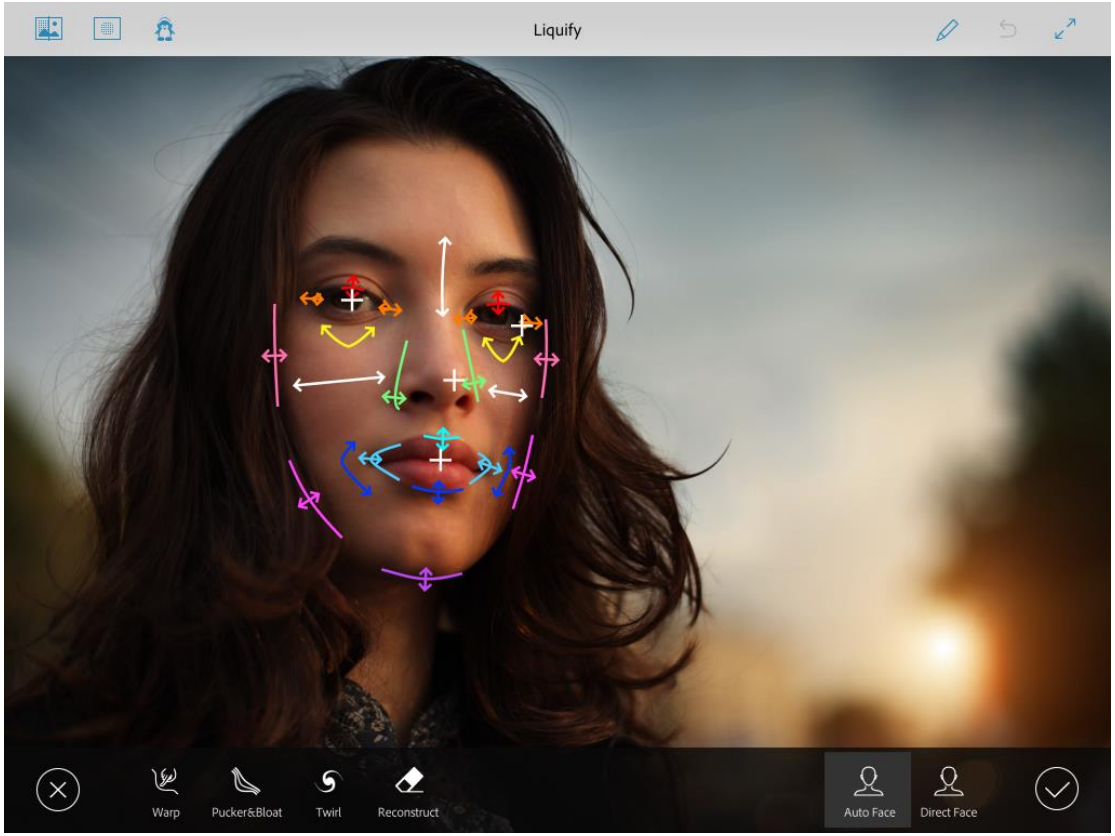
- 기타 국부적 특징(More local features)
  - The **histogram of oriented gradients (HOG)**
  - Harr feature
  - Local Binary Pattern (LBP)
- 특징 기반 검출 및 인식 기법
  - Open CV를 통한 실습 및 구현

PREVIEW

1장의 wide view 이미지로?

PREVIEW




Liquify
Warp
Pucker&Bloat
Twirl
Reconstruct
Auto Face
Direct Face

PREVIEW

- Correspondence detection (대응점 찾기)
  - **같은 장면을 다른 시점에서 찍은 두 영상에서 대응하는 점의 쌍을 찾는 문제**
  - 파노라마, 물체 인식/추적, 스테레오 등 컴퓨터 비전의 중요한 문제 해결의 단초

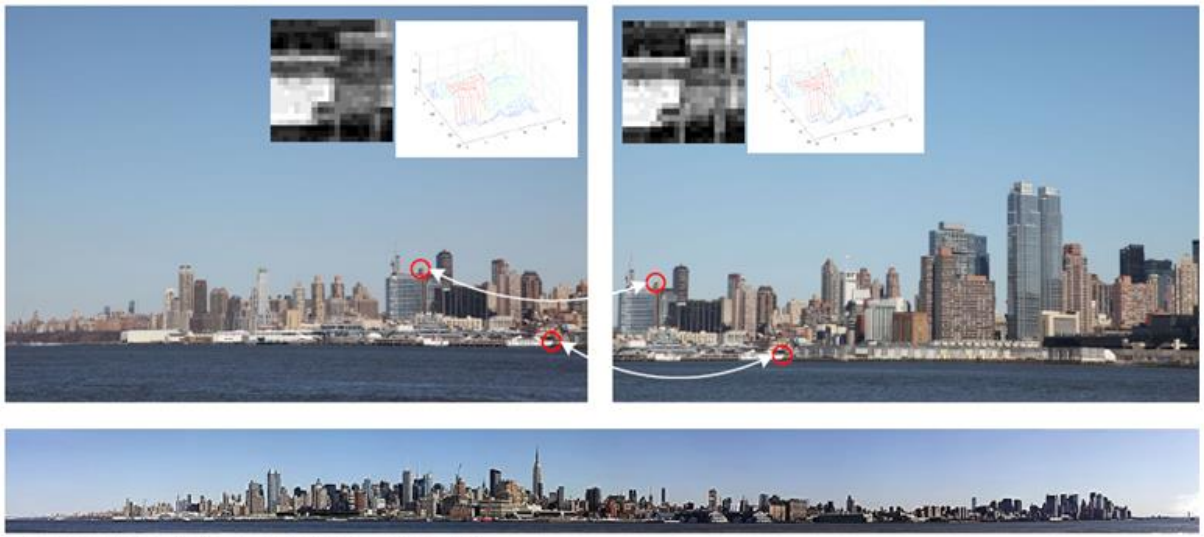


그림 4-1 대응점 찾기(확대 영상은 배의 꽁무니 부근)

  - 해결을 위해 세 단계 구성(일반적)

More local features : Histogram of Gradients (HoG)(1)

- a **feature descriptor** used in computer vision and image processing **for the purpose of object detection**
- A local object appearance and shape within an image can be described by **the distribution of intensity gradients or edge directions**.
- Basically,
  1) the image is divided into small connected regions called cells,
  2) for the pixels within each cell, a histogram of gradient directions is compiled.
  3) The descriptor is the concatenation of these histograms.

More local features : Histogram of Gradients (HoG)(2)

Object recognition

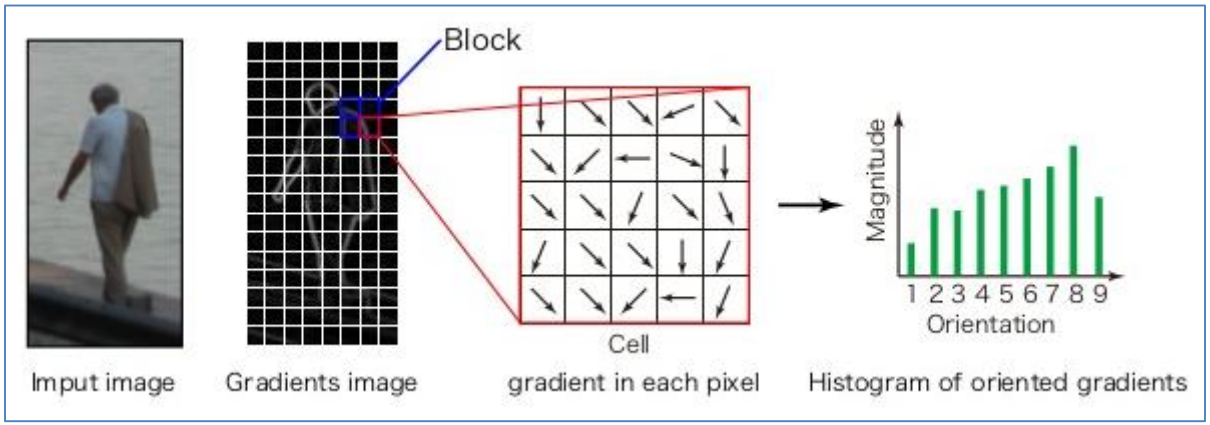
Gradient computation

Orientation binning

Block normalization

Learning

Object recognition



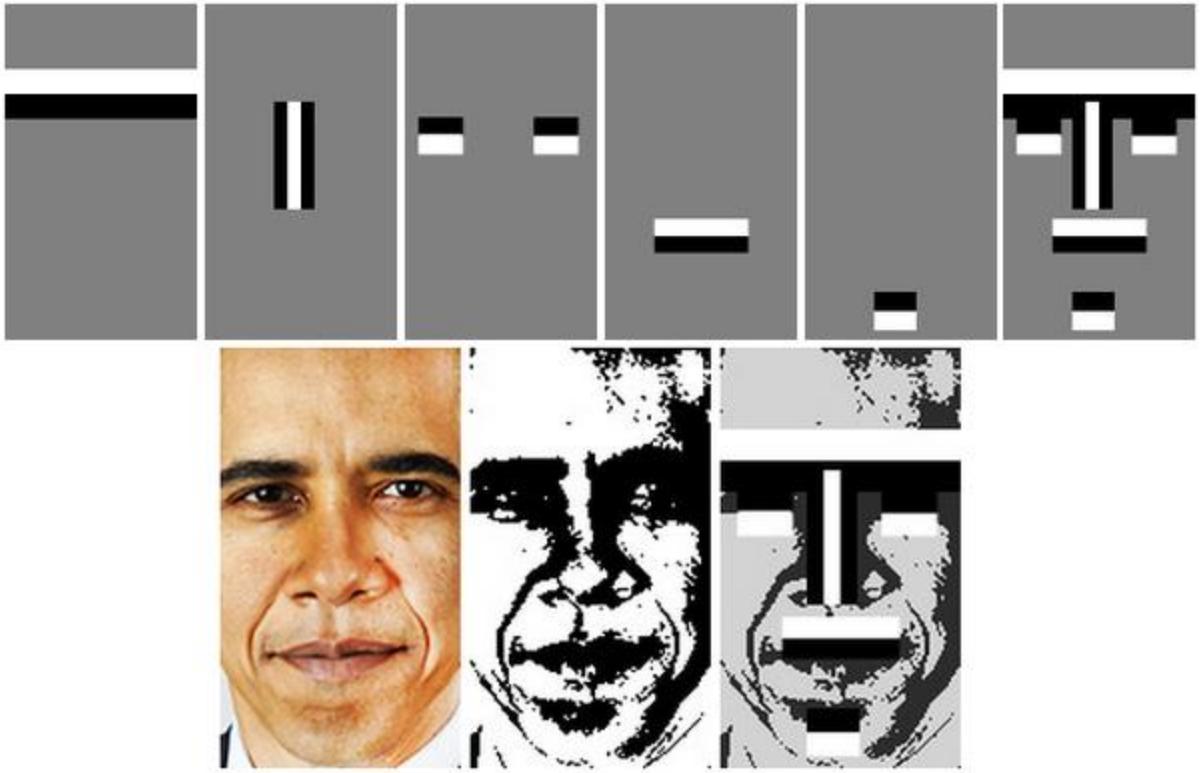
More local features : Haar (-like) feature (1)

- The name to their intuitive similarity with ***Haar wavelets*** and were used in the first real-time face detector [Viola and Jones].
- A **Haar-like feature** considers **adjacent rectangular regions at a specific location in a detection window**, **sums up the pixel intensities in each region** and **calculates the difference between these sums**. **This difference** (as threshold) is then used to categorize subsections of an image.
- **This difference is then compared to a learned threshold that separates non-objects from objects**

More local features : Haar (-like) feature (2)

Implementation Examples

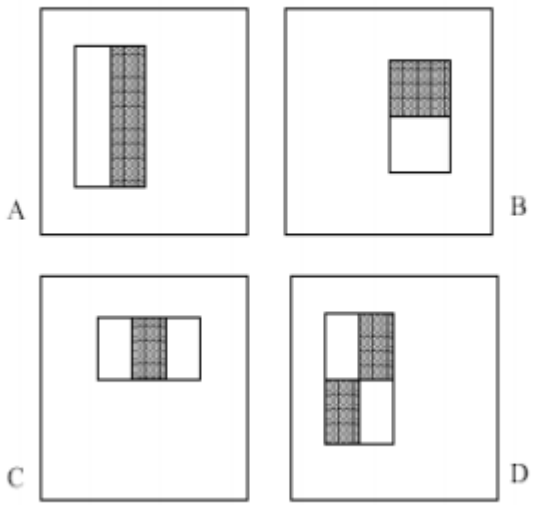
- Basic rectangular sets



More local features : Haar (-like) feature (3)

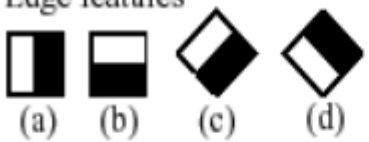
- **Extended rectangular sets**

- Original set of features:

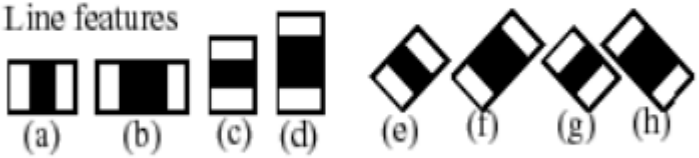


- Extended set of features:

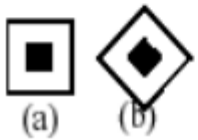
1. Edge features



2. Line features



3. Center-surround features

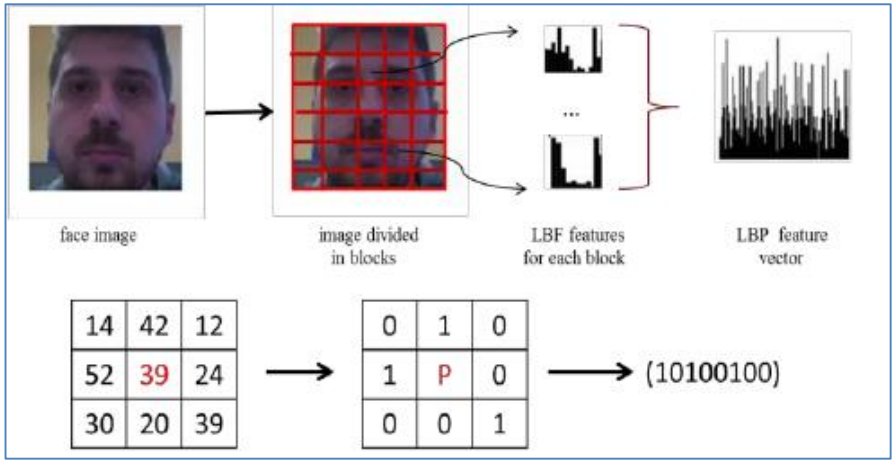


More local features : Local Binary Patterns (LBP) feature (1)

- Be a powerful feature for texture classification.
- LBP was first described in 1994.
- It has further been determined that when LBP is combined with the Histogram of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets
- Concept:
  - For each pixel, the intensity values of neighboring pixels are compared with the current pixel value. Then neighboring pixels are set 1 if it is greater than the current pixel, or 0 if it is less than the current pixel value.
  - With the obtained binary value are grouped to make a coded vector as feature.

More local features : Local Binary Patterns (LBP) feature (2)

- Implementation: The LBP feature vector
  - **Divide the examined window into cells** (e.g. 16x16 pixels for each cell).
  - **For each pixel in a cell**, compare the pixel to each of its 8 neighbors (on its left-top, left-middle, left-bottom, right-top, etc.). Follow the pixels along a circle, i.e. clockwise or counter-clockwise.
  - Where the center pixel's value is greater than the neighbor's value, write "0". Otherwise, write "1". This gives an 8-digit binary number (which is usually converted to decimal for convenience).
  - Compute the **histogram**, over the cell, of the frequency of each "number" occurring (i.e., each combination of which pixels are smaller and which are greater than the center). This histogram can be seen as a 256-dimensional feature vector.



Practice in Open CV: Haar(-like) feature-based Face Detection(1)

Face Detection

```
#include <opencv2₩core₩core.hpp>
#include <opencv2₩highgui₩highgui.hpp>
#include "opencv2₩objdetect₩objdetect.hpp"
#include "opencv2₩imgproc₩imgproc.hpp"

#include <iostream>
#include <stdio.h>

using namespace cv;
using namespace std;

/** Function Headers */

void detectAndDisplayII(Mat frame);

/** Global variables */
String face_cascade_name = "haarcascade_frontalface_alt.xml";
String eyes_cascade_name = "haarcascade_eye_tree_eyeglasses.xml";
CascadeClassifier face_cascade;
CascadeClassifier eyes_cascade;
string window_name = "Capture - Face detection";
RNG rng(12345);

//---- Face dectction(II) parameters ---//
int filenumber; // Number of file to be saved
string filename;


int main(int agrc, char** argv){

    (계 속)
```

Practice in Open CV: Haar(-like) feature-based Face Detection(2)

Face Detection

```
//-------- Face detection (II) ------------//
VideoCapture capture(0);

if (!capture.isOpened())
        return -1;

// Load the cascade
if (!face_cascade.load(face_cascade_name)){
        printf("--(!)Error loading₩n");
        return (-1);
};

// Read the video stream
Mat frame;
```

(계 속)

```
        for (;;){
            capture >> frame;

            // Apply the classifier to the frame
            if (!frame.empty()){

                detectAndDisplayII(frame);
            }
            else{
                printf(" --(!) No captured frame –
                                        - Break!");
                    break;
            }

            int c = waitKey(10);

            if (27 == char(c)){
                    break;
            }
        }
        //-------------------------------------//
        return 0;
}
```

Practice in Open CV: Haar(-like) feature-based Face Detection(3)

```
// Function detectAndDisplay
void detectAndDisplayII(Mat frame)
{
        std::vector<Rect> faces;
        Mat frame_gray;
        Mat crop;
        Mat res;
        Mat gray;
        string text;
        stringstream sstm;

        cvtColor(frame, frame_gray, COLOR_BGR2GRAY);
        equalizeHist(frame_gray, frame_gray);

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

        // Set Region of Interest
        cv::Rect roi_b;
        cv::Rect roi_c;

        size_t ic = 0; // ic is index of current element
        int ac = 0; // ac is area of current element

        size_t ib = 0; // ib is index of biggest element
        int ab = 0; // ab is area of biggest element
```

(계 속)

Practice in Open CV: Haar(-like) feature-based Face Detection(4)

```
for (ic = 0; ic < faces.size(); ic++) // Iterate through all current elements (detected faces)

{
		roi_c.x = faces[ic].x;
		roi_c.y = faces[ic].y;
		roi_c.width = (faces[ic].width);
		roi_c.height = (faces[ic].height);

		ac = roi_c.width * roi_c.height; // Get the area of current element (detected face)

		roi_b.x = faces[ib].x;
		roi_b.y = faces[ib].y;
		roi_b.width = (faces[ib].width);
		roi_b.height = (faces[ib].height);

		ab = roi_b.width * roi_b.height; // Get the area of biggest element, at beginning it is same as "curre

		if (ac > ab)
		{
			ib = ic;
			roi_b.x = faces[ib].x;
			roi_b.y = faces[ib].y;
			roi_b.width = (faces[ib].width);
			roi_b.height = (faces[ib].height);
		}
```

(계 속)

Practice in Open CV: Haar(-like) feature-based Face Detection(5)

```
            crop = frame(roi_b);
            resize(crop, res, Size(128, 128), 0, 0, INTER_LINEAR);
                                    // This will be needed later while saving images
            cvtColor(crop, gray, CV_BGR2GRAY); // to Grayscale

            // Form a filename
            filename = "";
            stringstream ssfn;
            ssfn << filenumber << ".png";
            filename = ssfn.str();
            filenumber++;

            imwrite(filename, gray);

            Point pt1(faces[ic].x, faces[ic].y);
                    // Display detected faces on main window - live stream from camera
            Point pt2((faces[ic].x + faces[ic].height), (faces[ic].y + faces[ic].width));
            rectangle(frame, pt1, pt2, Scalar(0, 255, 0), 2, 8, 0);
}
```

(계 속)

Practice in Open CV: Haar(-like) feature-based Face Detection(6)

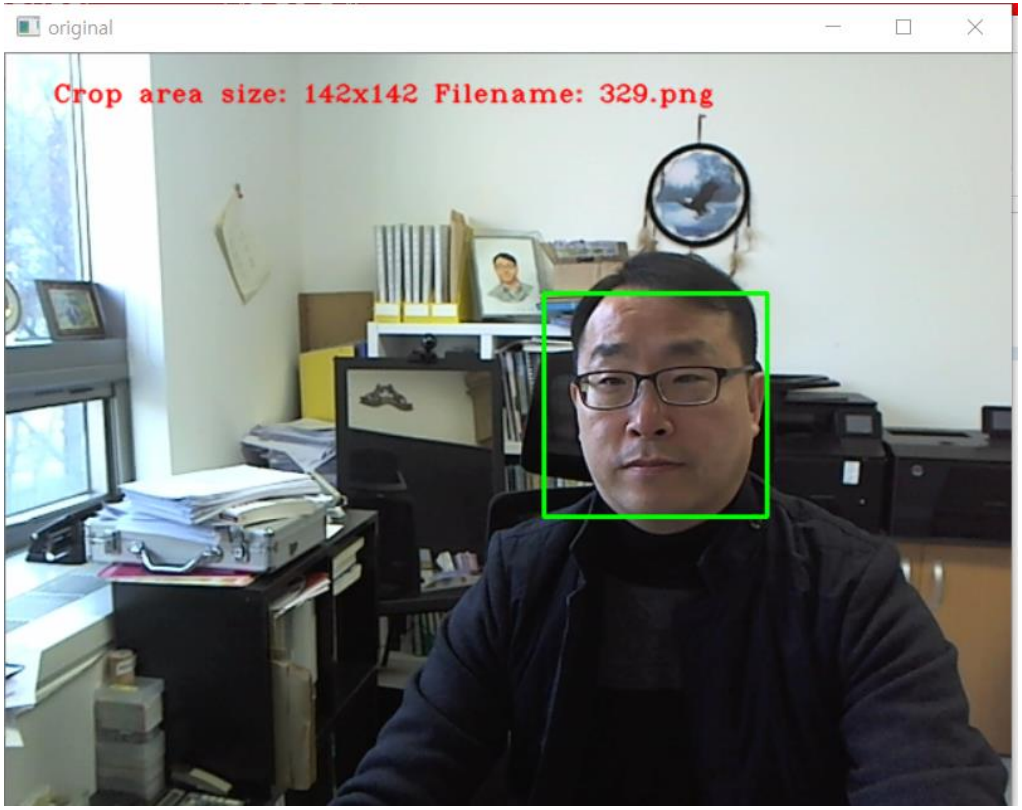
```
        // Show image
        sstm << "Crop area size: " << roi_b.width << "x" << roi_b.height << " Filename: " << filename;
        text = sstm.str();

        putText(frame, text, cvPoint(30, 30), FONT_HERSHEY_COMPLEX_SMALL, 0.8, cvScalar(0, 0, 255), 1, CV_AA);
        imshow("original", frame);

        if (!crop.empty())
        {
                imshow("detected", crop);
        }
        else
                destroyWindow("detected");
}
```

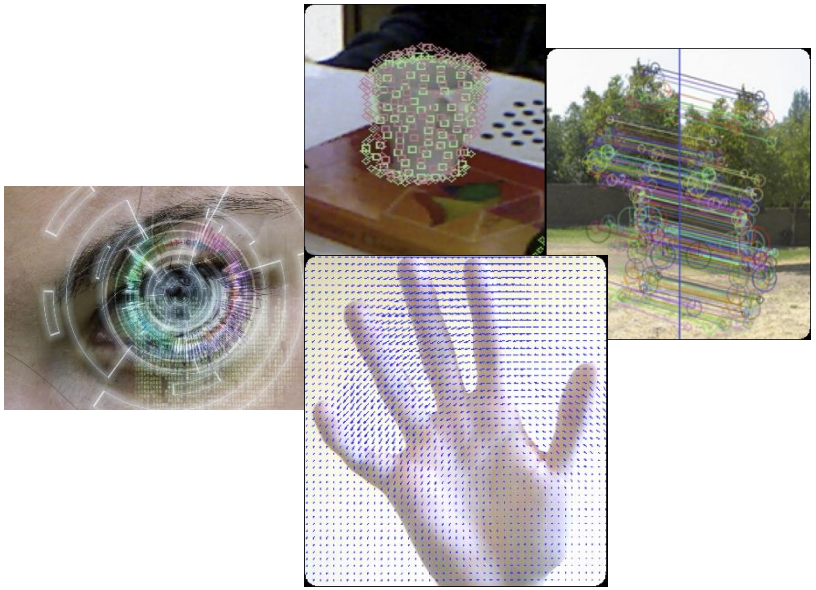
Practice in Open CV: Haar(-like) feature-based Face Detection(6)

- Result



COMPUTER VISION 비젼 프로그래밍

Thank you and Question?



IVPL
Intelligent Vision Processing Lab