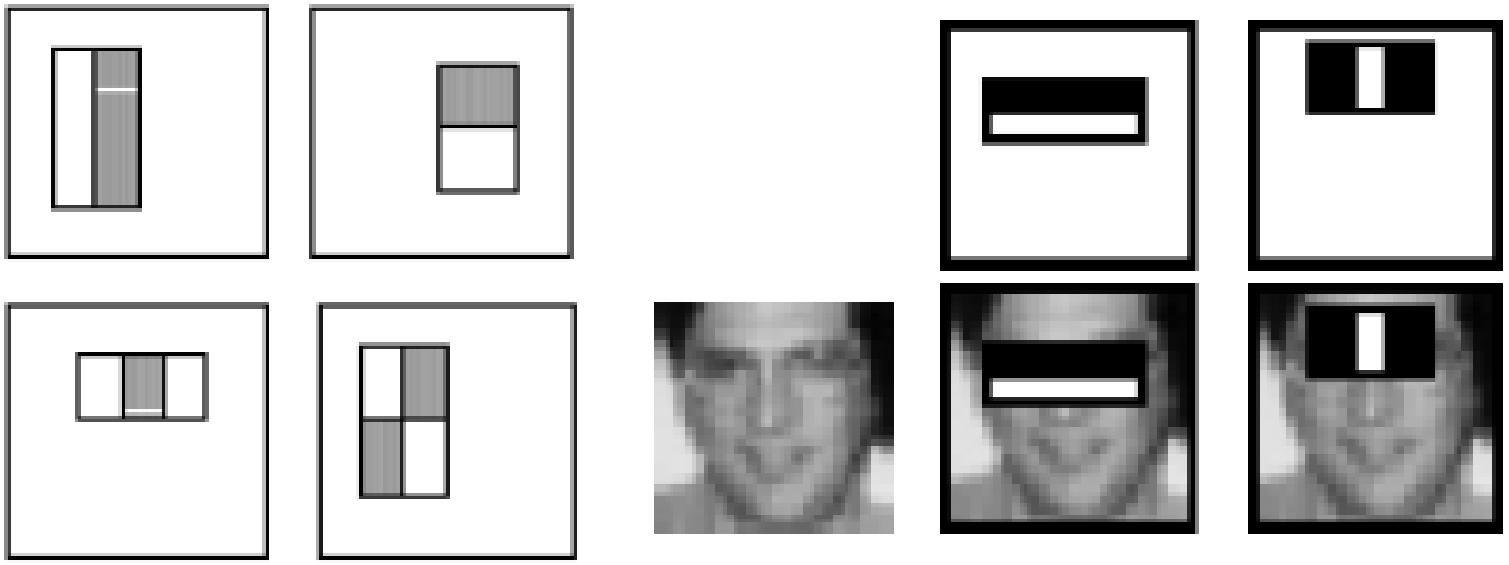
# Face Detection

Sung Soo Hwang

# Face Detection

- Feature
  - Harr-like feature is used in openCV
    - It can be defined as the difference of the sum of pixels of areas inside the rectangle
    - It can be at any position and scale within the original image

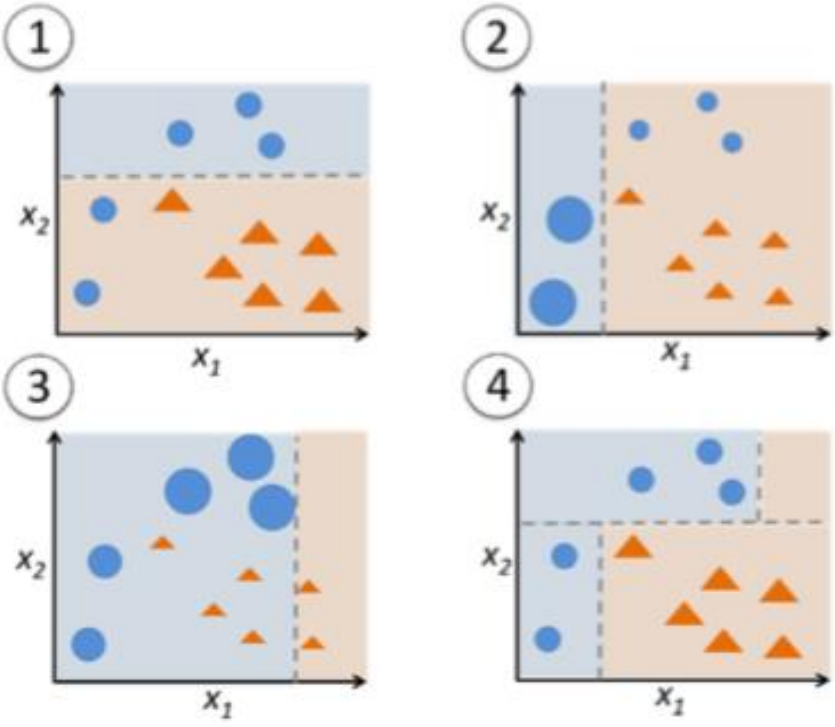


# Face Detection

- Training
  - By changing size and location, lots of features can be generated
  - Among them, choose features that classify human faces

# Face Detection

- Training
  - In openCV, Adaboost(Adaptive Boosting) is used
  - Boosting: A set of weak-learner generates a strong-learner
  - Adaptive: Weight of each sample is adjusted depending on the accuracy of already-trained weak learners

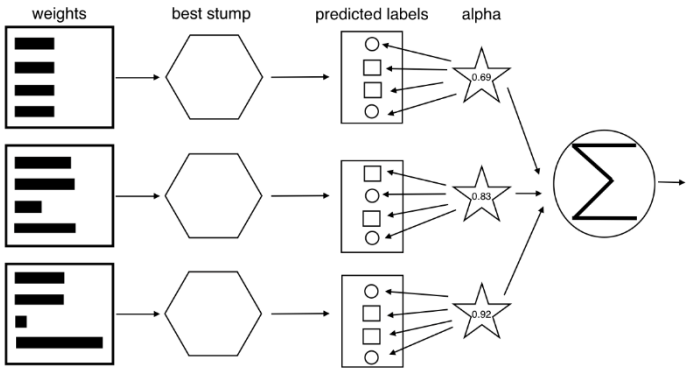


# Face Detection

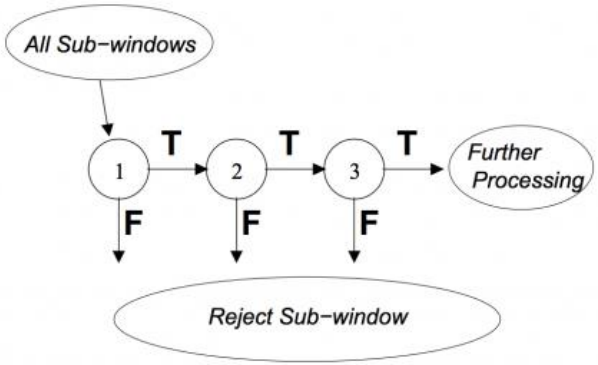
- Training
  - Face representation using Harr-like features

# Face Detection

- Cascade classifier
  - Generate a strong learner using multiple weak learners




  - Each strong learner is connected in cascade
    - Number of weak leaners in each strong learner: 3>2>1
    - Lots of non-face regions are easily eliminated



# Face Detection

- Fast computation of Harr-like features can be conducted by using integral image

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 4 | 1 |
| 3 | 4 | 1 | 5 | 2 |
| 2 | 3 | 3 | 2 | 4 |
| 4 | 1 | 5 | 4 | 6 |
| 6 | 3 | 2 | 1 | 3 |

input image

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 3 | 5 | 9 | 10 |
| 0 | 4 | 10 | 13 | 22 | 25 |
| 0 | 6 | 15 | 21 | 32 | 39 |
| 0 | 10 | 20 | 31 | 46 | 59 |
| 0 | 16 | 29 | 42 | 58 | 74 |

integral image

# Face Detection

- Example

```
CascadeClassifier face_classifier;

face_classifier.load("haarcascade_frontalface_alt.xml");
```

  - OPENCV provides various classifiers (eye, upper body, etc.) in "…₩opencv₩sources₩data₩haarcascades"

# Face Detection

- openCV function

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

- **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.

# Face Detection

- openCV function
  - **numDetections :** Vector of detection numbers for the corresponding objects. An object's number of detections is the number of neighboring positively classified rectangles that were joined together to form the object.
  - **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.

# Face Detection

- Example code

```
CascadeClassifier face_classifier;
Mat frame, grayframe;
vector<Rect> faces;
int i;

// open the webcam
VideoCapture cap(0);

// check if we succeeded
if (!cap.isOpened()) {
        cout << "Could not open camera" << endl;
        return -1;
}

// face detection configuration
face_classifier.load("haarcascade_frontalface_alt.xml");

while (true) {
        // get a new frame from webcam
        cap >> frame;

        // convert captured frame to gray scale
        cvtColor(frame, grayframe, COLOR_BGR2GRAY);
```

# Face Detection

- Example code

```
            face_classifier.detectMultiScale(
                        grayframe,
                        faces,
                        1.1, // increase search scale by 10% each pass
                        3,   // merge groups of three detections
                        0,   // not used for a new cascade
                        Size(30, 30) //min size
            );

            // draw the results
            for (i = 0; i < faces.size(); i++) {
                        Point lb(faces[i].x + faces[i].width, faces[i].y + faces[i].height);
                        Point tr(faces[i].x, faces[i].y);
                        rectangle(frame, lb, tr, Scalar(0, 255, 0), 3, 4, 0);
            }
            // print the output
            imshow("Face Detection", frame);
            if (waitKey(33) == 27) break;
}
```