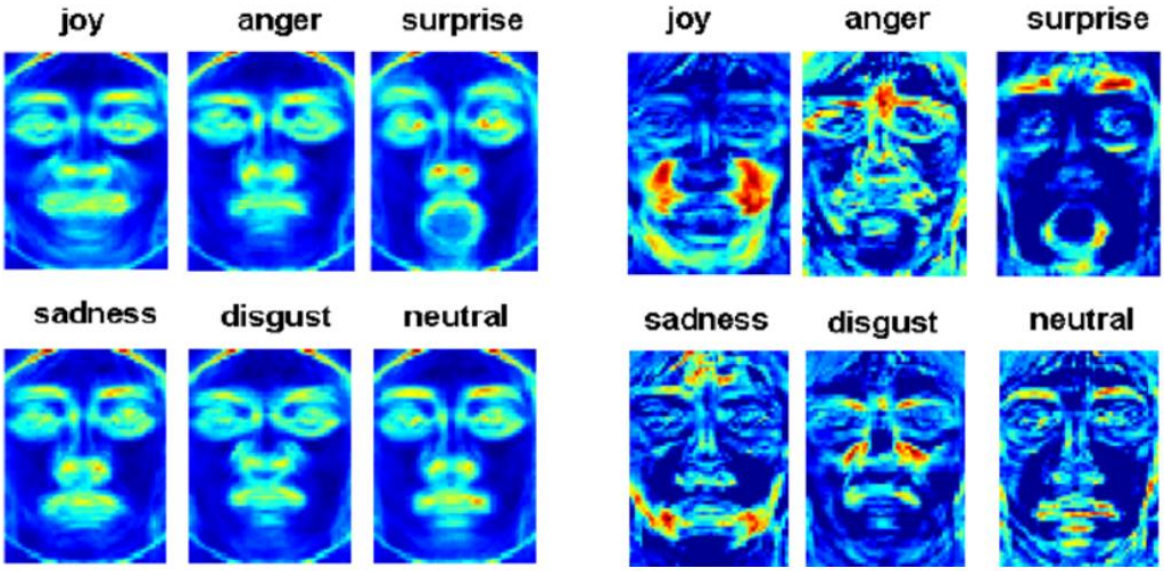
Exploring Feature Extraction Methods for Facial Expression Recognition Systems

Introduction:

Facial expression recognition is a problem that has gained significant popularity with the increased focus on understanding human emotions. Facial expression recognition systems can be divided into five steps, which are image acquisition, face detection and pre-processing, feature extraction and classification. Many different systems for FER are proposed, and there exist many different use combinations of various feature extraction methods with various machine learning models. Some commonly used feature detectors and descriptors that we've also studied are Histogram of Oriented Gradients (HOG), MSER features, Features from Accelerated segment test (FAST), Oriented FAST and Rotated BRIEF Feature Detector (ORB), Local Binary Patterns (LBP) and Gabor Filter. The most used machine learning classification technique we came across was Support Vector-Machines (SVMs).



Mean gradient image (left) and log-likelihood ratios (right) for the 6 facial expressions. Hot colors indicate the discriminative areas for a given emotion.

(Gregory Lopez, 2009, https://www.researchgate.net/profile/Gregory-Rogez/publication/221258894/figure/fig1/AS:305604913582080@1449873297257/Mean-gradient-image-left-and-log-likelihood-ratios-right-for-the-6-facial.png)

Can Korkmaz
Miray Dicle

Face Detection

First, face detection algorithm with built-in functions applied to a chosen emotion recognition video from CK+ database. It basically selects a video frame from the read video randomly and creates a bounding box after detecting the face with face detector object. Then, it imposes the generated shape into the video frame by referring to the coordinates of the bounding box. The code for this algorithm and the result are as below;


```
% Create a cascade detector object.

faceDetector = vision.CascadeObjectDetector();



% Read a video frame and run the face detector.

videoReader = VideoReader('neutral.mp4');

videoFrame      = readFrame(videoReader);

bbox            = step(faceDetector, videoFrame);



% Draw the returned bounding box around the detected face.

videoFrame = insertShape(videoFrame, 'Rectangle', bbox);
```

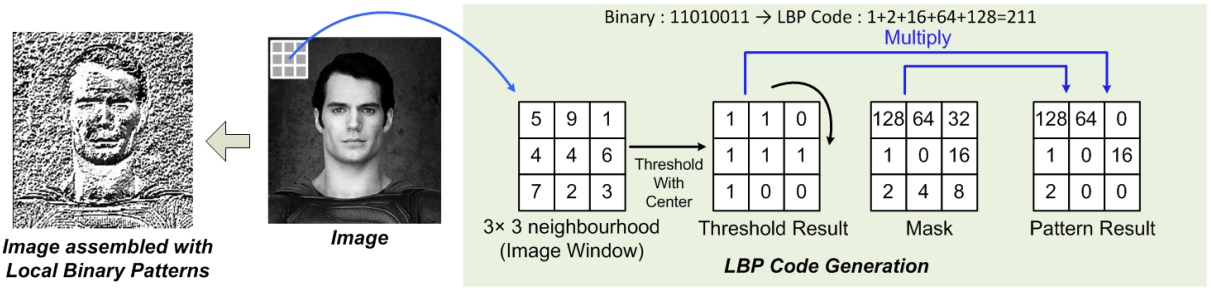
```
figure; imshow(videoFrame); title('Detected face');
```

After completing this section, next step is the feature extraction with some useful algorithms.

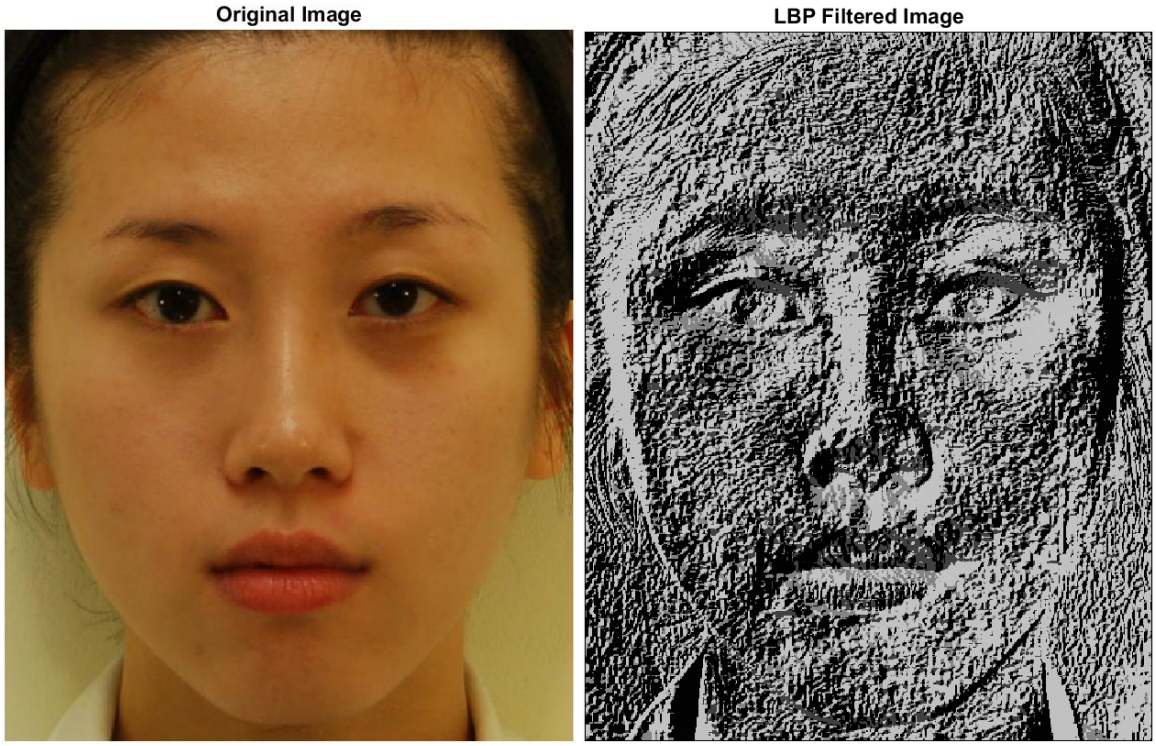
Local Binary Pattern (LBP):

Local binary pattern is a very efficient way of analysing the texture properties of an image. In this algorithm, the idea is to create a 3x3 window and comparing each center pixel with its neighbourhood pixels. By setting the center pixel as a threshold, smaller values become zero and larger or equal values become 1. After convolution operation with a mask that consists of weights for translating a binary number into a decimal type, this new value replaces the center pixel of the image as the window scans the whole image. This algorithm is commonly used for face detection and analysis to get more information about face similarities by collecting LBP codes into a histogram distribution (Kyrkou, 2018).

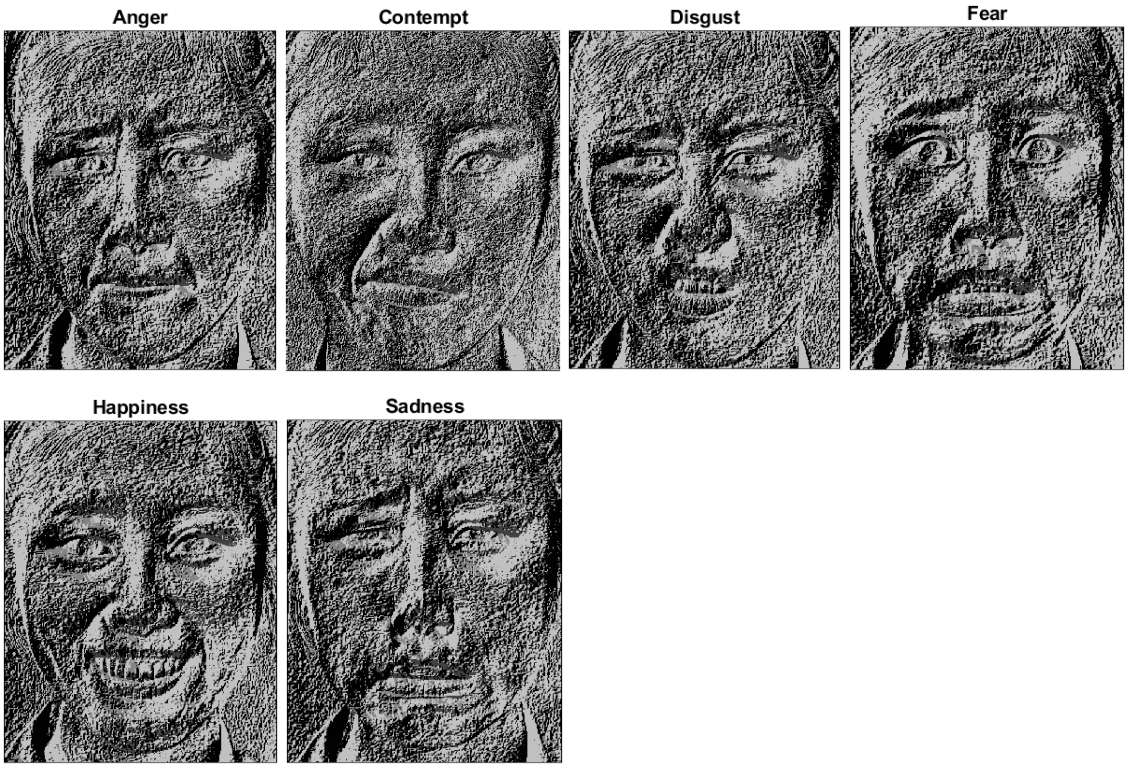
The process for this algorithm is described as below;



For a reliable analysis for face features purposes, neutral face is selected. Result is as below;



Results for other emotions are as below;

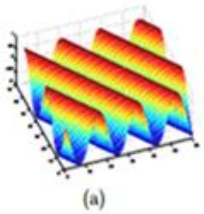



```
function [ImLBP] = lbp(Im)



Im = rgb2gray(Im);

Im = double(Im);

Im = imgaussfilt(Im);

[row,col] = size(Im);

ImLBP = zeros(row,col);



Lbp = [128 1 2; 64 0 4; 32 16 8];

Lbp = reshape(Lbp,[1,9]);



for i = 2:1:row-1

    for j = 2:1:col-1

        w = Im(i-1:i+1, j-1:j+1);
```

```
        for k = -1:1:1

            for l = -1:1:1

                if w(2,2)<w(2+k,2+l)

                    w(2+k,2+l) = 0;

                else

                    w(2+k,2+l) = 1;

                end

            end

        end

        w = reshape(w, [9,1]);

        value = Lbp*w;

        ImLBP(i,j) = value;

    end

end



ImLBP = uint8(ImLBP);



End
```

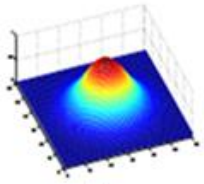
Gabor Filter

Another useful filter for feature extraction and texture analysis is Gabor filter which has been found by Dennis Gabor. It is a special case of a linear bandpass filter and can be viewed as a spatially localized sinusoidal wave with a particular frequency and orientation that is modulated with a Gaussian wave(Shah, 2018). An example theory for Gabor filter is shown as below;



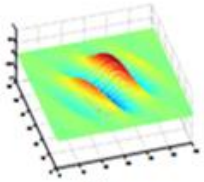
(a)

A Sinusoid oriented 30^0 with X-axis



(b)

A 2-D Gaussian

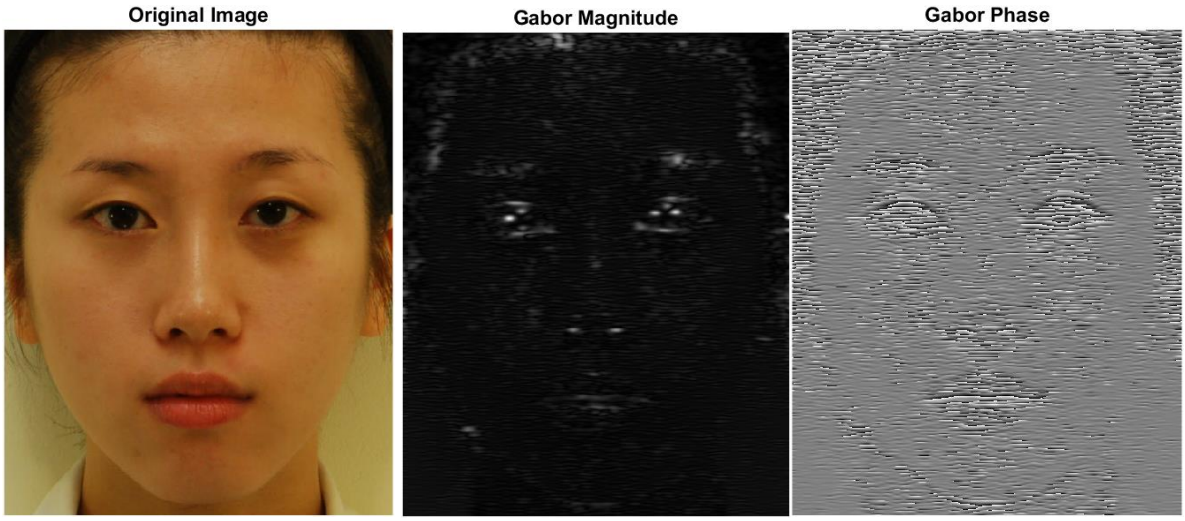


(c)

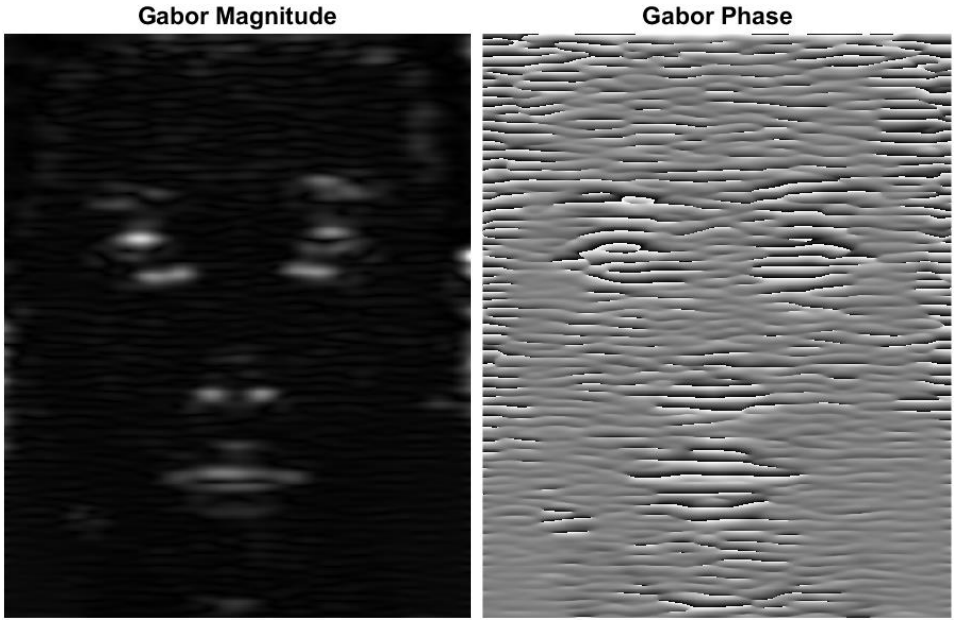
The corresponding 2-D Gabor filter

By adjusting wave parameters such as lamda, theta, gamma and sigma, different texture analysis can be done for a given image. For example, lamda which means the wavelength determines the width of the strips for the Gabor filter. Increasing the wavelength produces thicker strips and decreasing it produces thinner stripes. Theta controls the orientation of the Gabor filter.

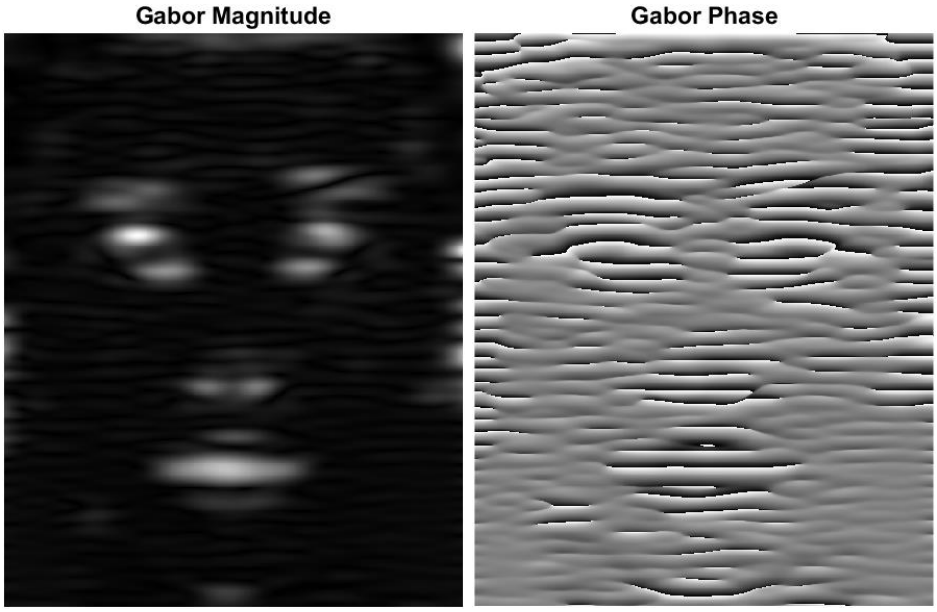
When this algorithm is applied to the neutral face image sample by setting wavelength as 4 and orientation as 90, the results for magnitude and phase for the image are as below;



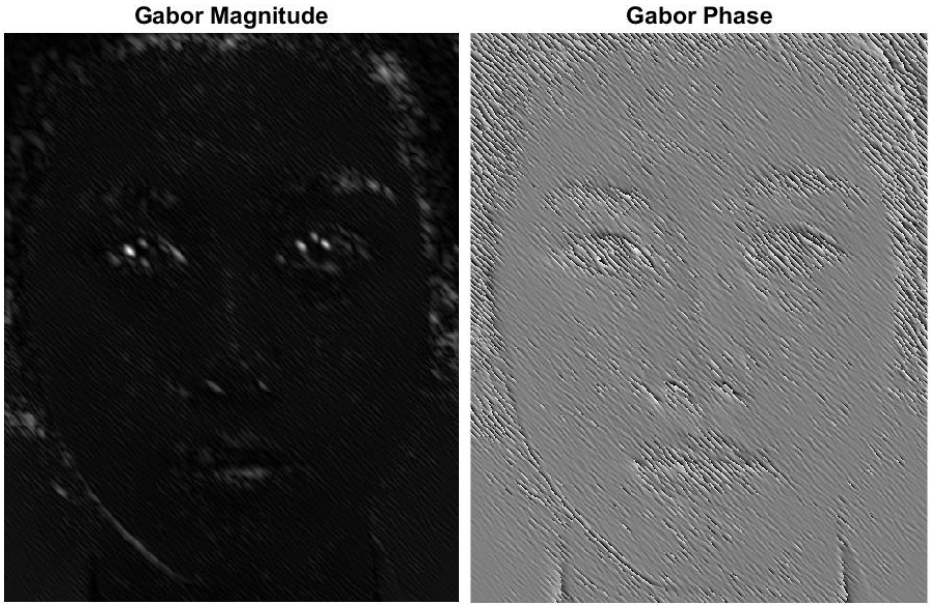
As the wavelength is increased to 10 without any change in the orientation, eyes, nose and mouth of the face became enlarged but the features of the image have lost robustness and became blurry in the Gabor magnitude analysis. Additionally, stripes became more horizontal in the Gabor Phase. The results are as below;



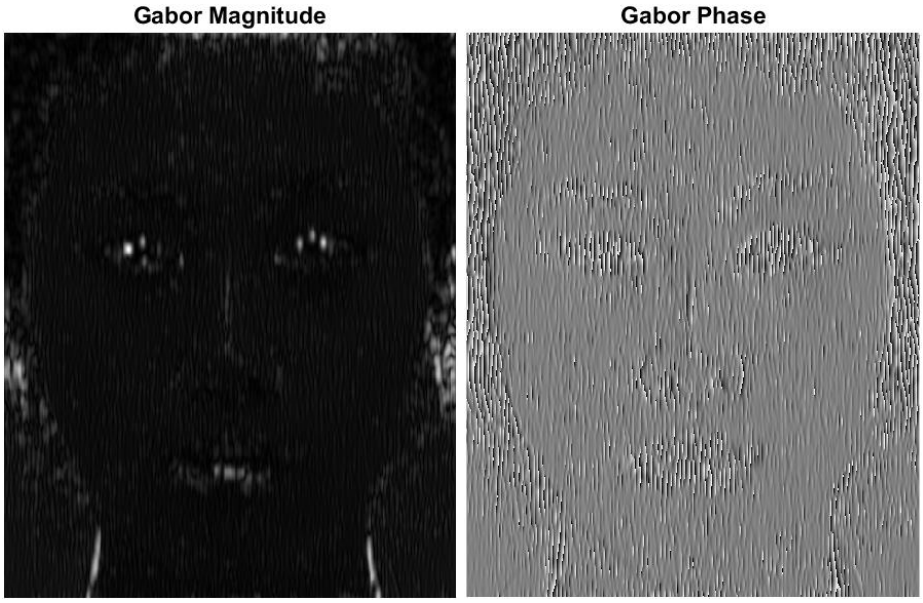
As the wavelength is increased even further up to 15, again with no change in orientation, it became harder to analyse the features of the face in a robust fashion. The result is as below;



When the orientation is decreased to 45 with wavelength is set to 4, the stripes in the Gabor Phase information became more vertical and the features of the face are clearer and the face is easier to be detected. Also, there is almost no change in the Gabor magnitude information. The results are as below;



When the orientation is decreased to zero, the stripes in phase information completely became vertical as shown below;



Therefore, by adjusting these parameters, the most optimum parameter values for face detection and its regarding feature extraction can be determined. In this case, it is 4 or 5 for wavelength and 0 for orientation.

The code for this algorithm is as below;

```
Im = imread('neutral.jpg');

ImG = rgb2gray(Im);

ImGauss = imgaussfilt(ImG);
```

```
wavelength = 4;

orientation = 90;

[mag,phase] = imgaborfilt(ImGauss,wavelength,orientation);

tiledlayout(1,3);

nexttile;

imshow(Im);

title('Original Image');

nexttile;

imshow(mag,[]);

title('Gabor Magnitude');

nexttile;

imshow(phase,[]);
```

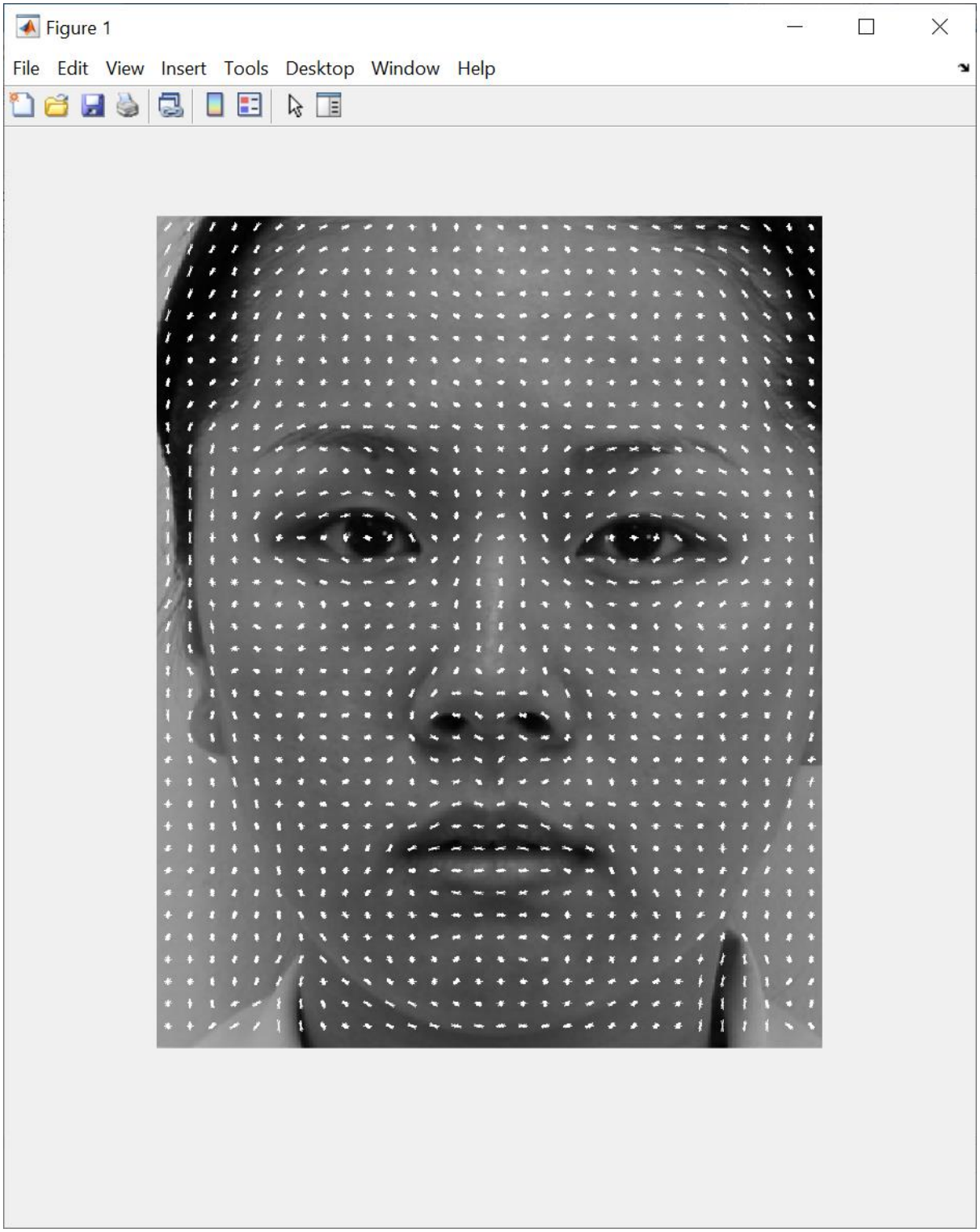
Histogram of Oriented Graph (HOG):

Histogram of oriented gradients (HOG) is a shape descriptor that counts occurrences of gradient orientations in localized portions of an image, such as in 8x8 or 16x16 cell sizes, and concatenates local orientations into a single histogram. HOG is mostly utilized for object detection but also is found to be useful to model the shape of the facial muscles by means of an edge analysis. Since facial expressions result from facial muscle movements, HOG features extracted from the mouth, brows and other region of interests can be used to form a feature vector. HOG descriptor is successfully experimented for facial expression recognition with use of SVMs by Pierluigi Carcagnì at National Research Council of Italy (Carcagnì et al.). Junkai Chen from The Hong Kong Polytechnic University (Chen et al.) also used HOG features to detect facial expressions (Chen et al.).

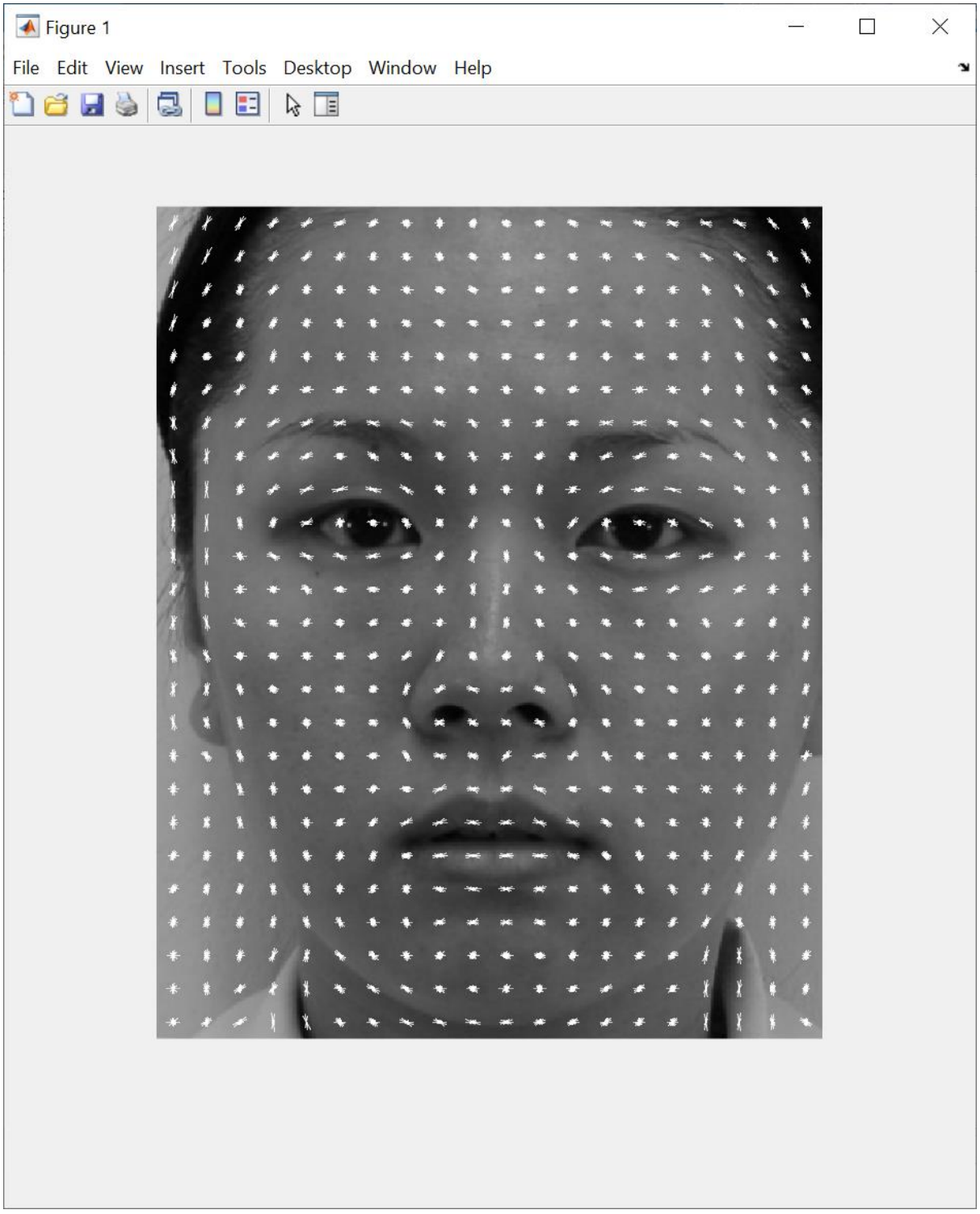
Junkai Chen, Zenghai Chen, ZHeru Chi, and Hong Fu from The Hong Kong Polytechnic University, proposed a system based on component detection and HOG feature extraction to be in conjunction with SVMs (Chen et al.). First Step after face detection and eye detection was to segment face into components, and later extracting Hog features. Then the extracted features from each component are put into a larger feature vector, which is then fed into SVMs. Success rate of this HOG feature based facial expression detection system suppressed that of other methods such as local binary patterns and Gabor pattern-based feature extraction methods, with 94.3% classification rate. (Chen et al.)

Below are our experimentations with MATLAB's extractHOGFeatures() function:

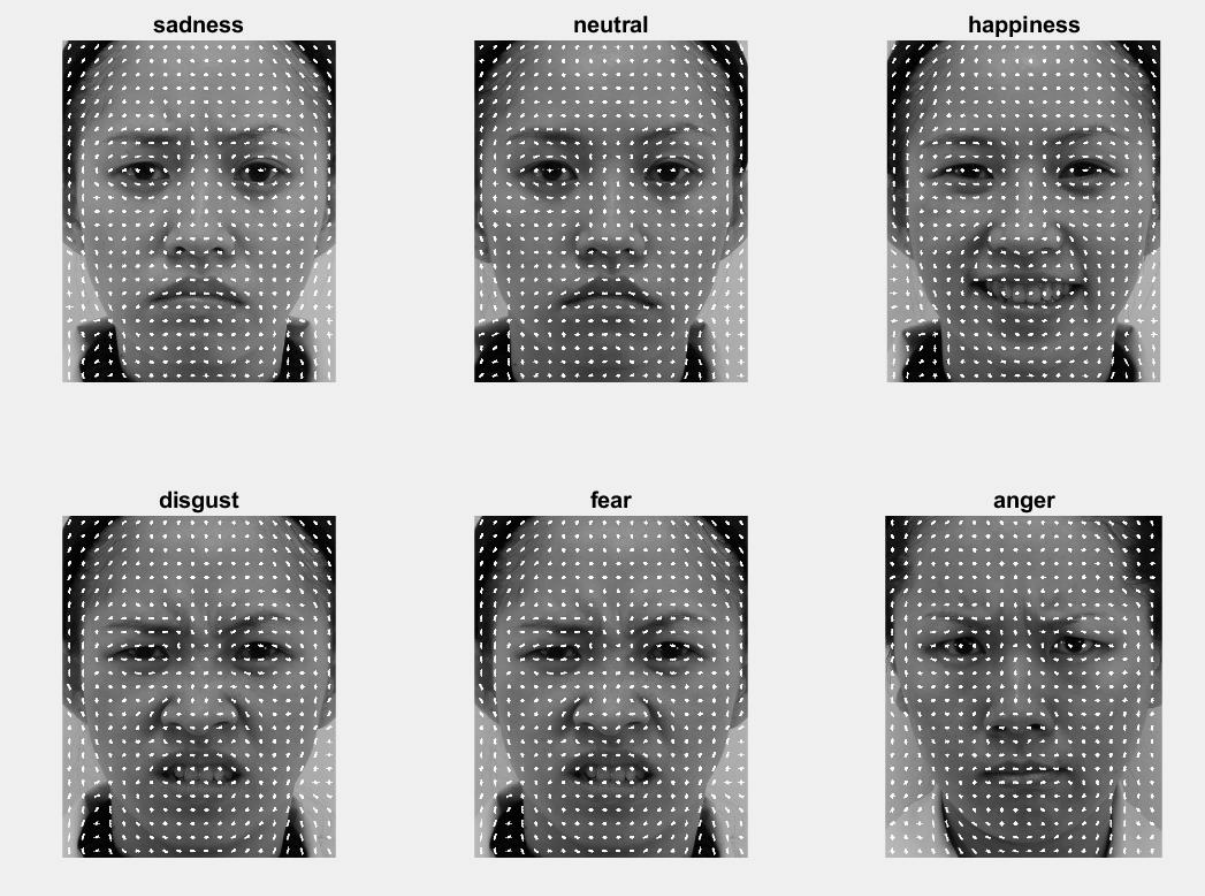
HOG Features Visualized with 16x16 cell size:



HOG Features Visualized with 24x24 Cell Size:



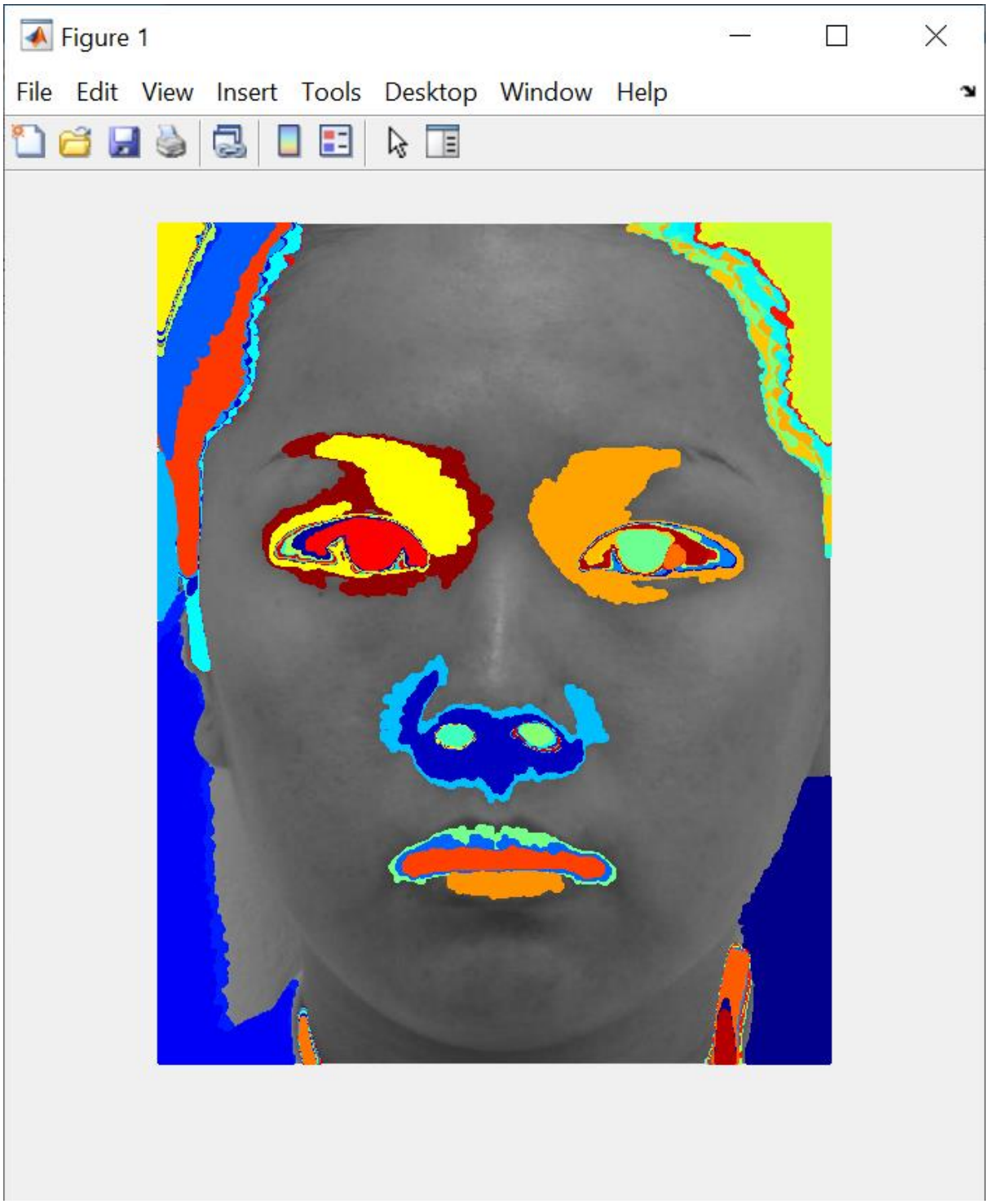
The extracted feature vector had dimensions 1x16416, which is a significant reduction of 600x480x3 image. These numbers can be further reduced by a synergetic implementation of MSER algorithm for region detection, or with any other method for component detection, and later extracting HOG features from those areas. This speed-up can increase robustness and decrease the computational cost of the operation, which is important in real-time applications.




```
clear all; close all; clc;

Im = imread("f04_dfh_nx.jpg");
I = rgb2gray(Im);
%I = histeq(I);


[featureVector,hogVisualization] = extractHOGFeatures(I,"CellSize",[24 24],
UseSignedOrientation=false);


figure;
imshow(I);
hold on;
plot(hogVisualization);
```

MSER Features:

Using MATLAB's detectMSERFeatures() function, we visualized the regions on top of the face. The regions-of-interest are the MSERs extracted. These regions of interest can be used to detect components for building feature vectors, and later other feature detection algorithms such as HOG feature extraction can be called to fill the vector components, and then fed on SVMs to be classified. MSERs can also be directly used with respect to blob sizes and fed on SVMs.



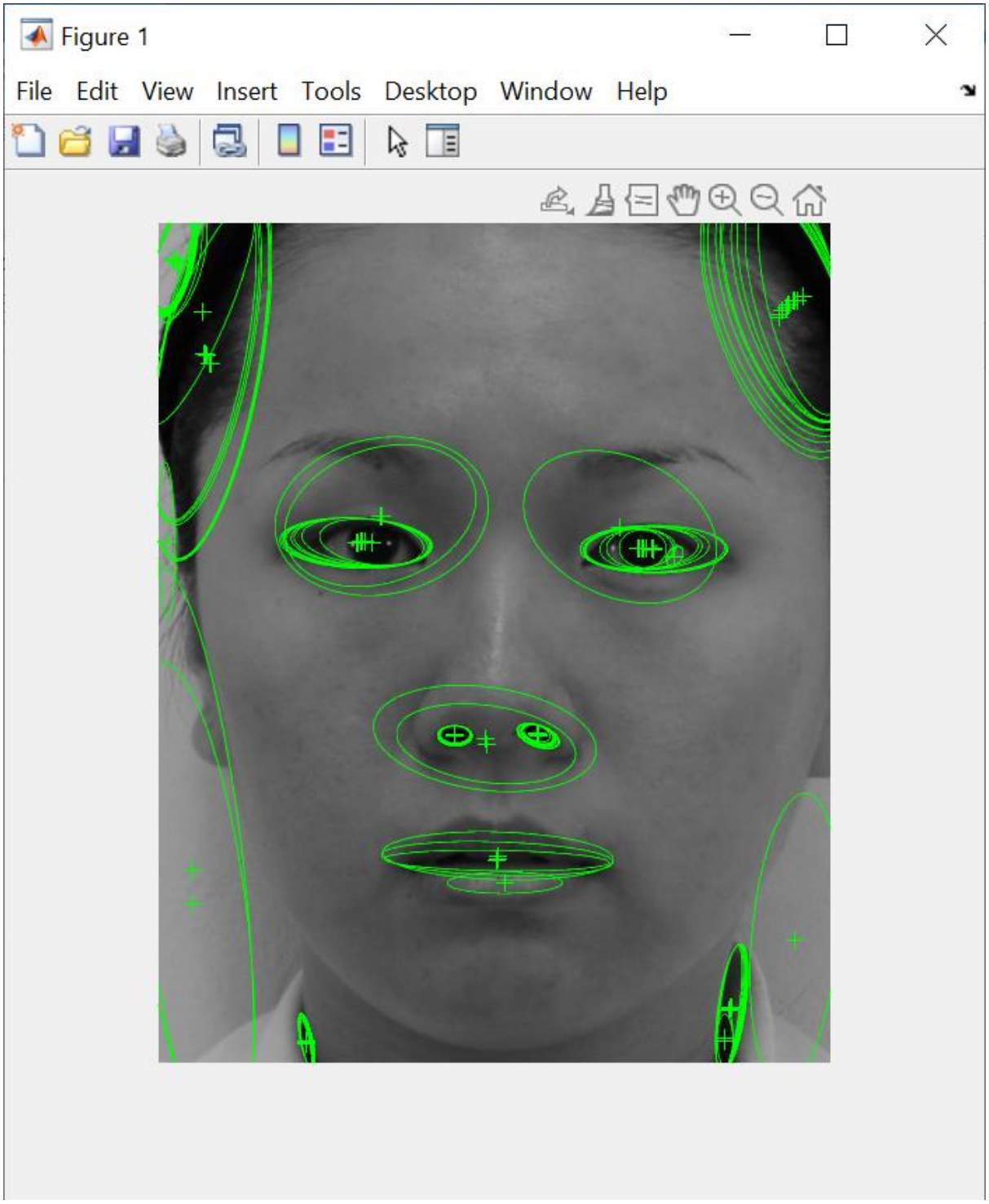
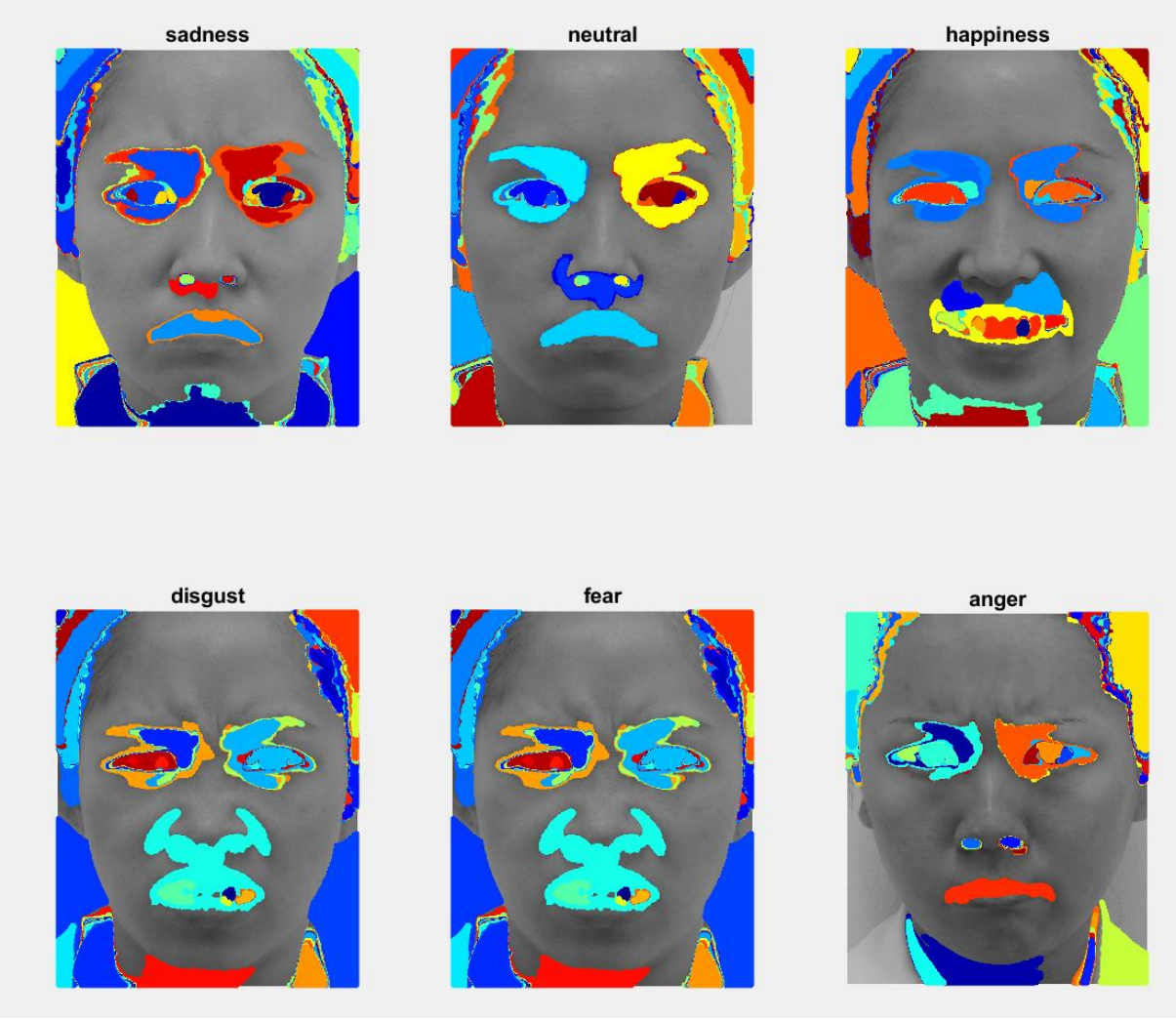


Figure 1
File Edit View Insert Tools Desktop Window Help




```
clear all; close all; clc;
Im = imread("f04_dfh_nx.jpg");
I = rgb2gray(Im);

%I = histeq(I);

regions = detectMSERFeatures(I, "MaxAreaVariation", 0.5);


figure;
imshow(I);
hold on;
plot(regions,'showPixelList',true,'showEllipses',false);
```

Features from accelerated segment test (FAST):

FAST is a corner detection algorithm and FAST corner detection algorithm is commonly used for extracting feature points in facial expression recognition problem. FAST algorithm is a local operator where a central pixel *p* is identified as an interest point or not. Only pixels that are some predetermined distance away are tested to decide whether the mid-point pixel p is a corner (feature point) or not. A threshold input for the required number of contiguous pixels that have either all their intensities higher than *Ip+t*, or all have intensities lower than *Ip-t. t* is another input parameter (Tyagi, 2019).

Below are our experimentations with MATLAB's detectFASTFeatures() function:

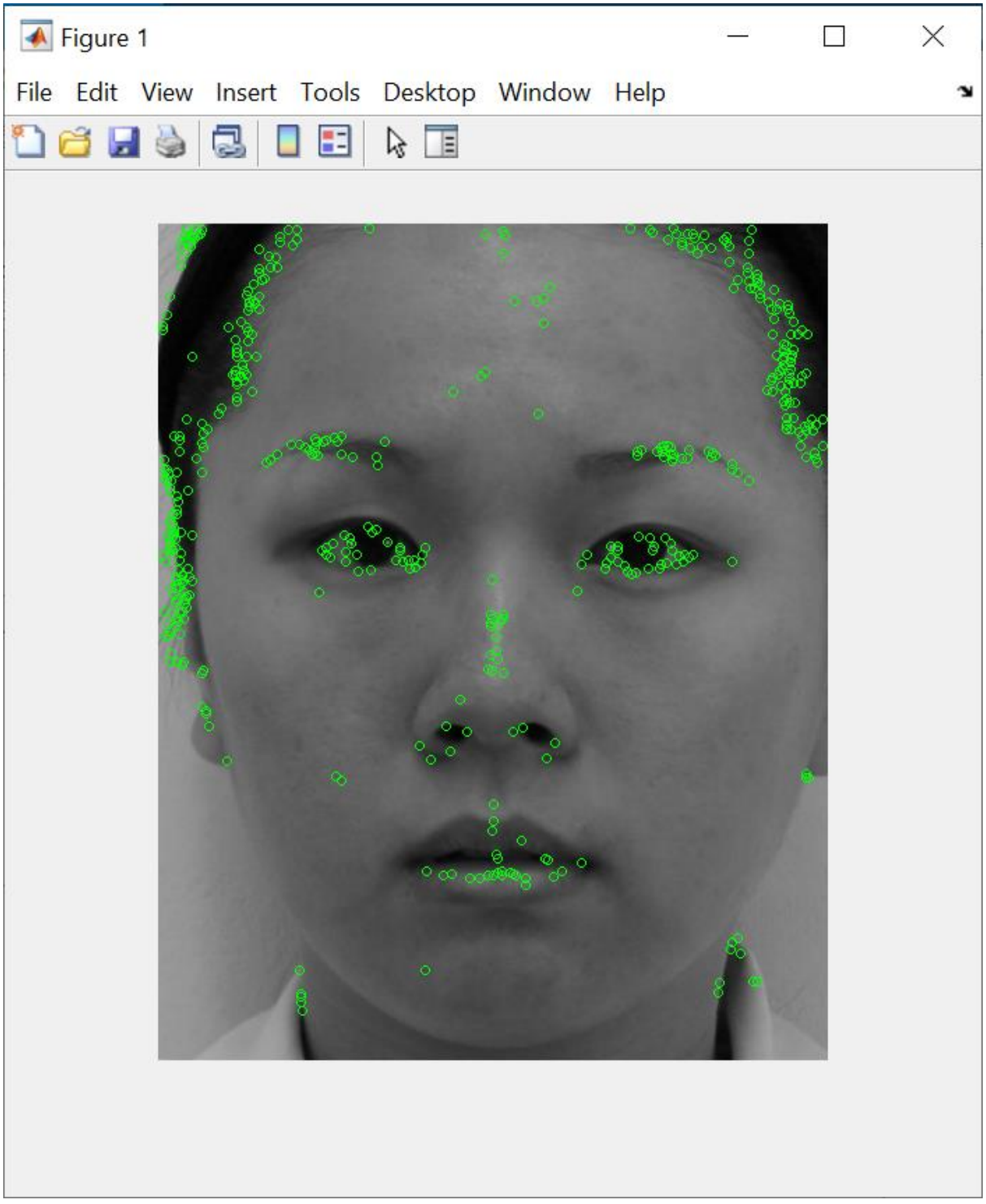
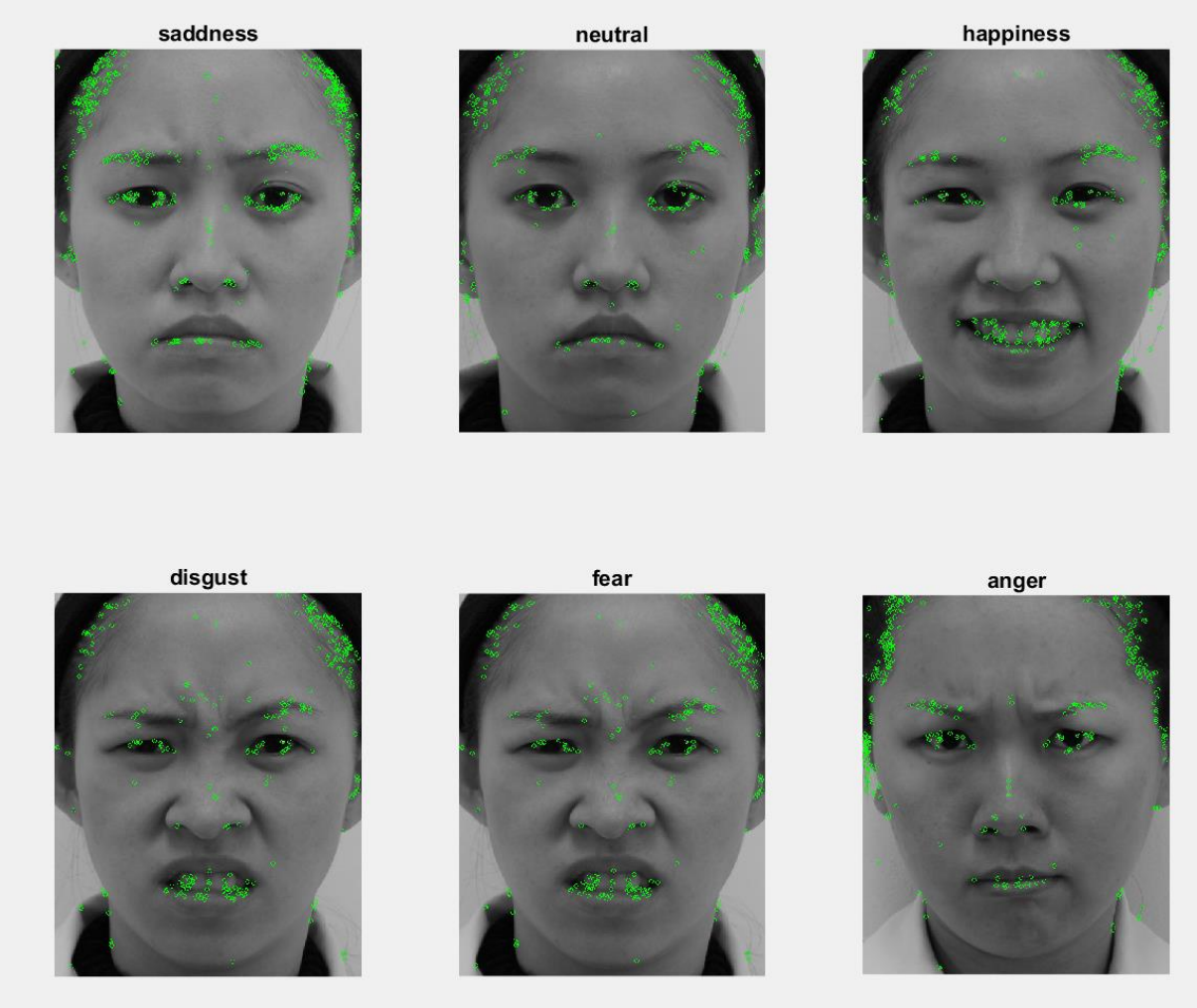


Figure 1
File Edit View Insert Tools Desktop Window Help




```
clear all; close all; clc;
Im = imread("f04_dfh_nx.jpg");
ImGray = rgb2gray(Im);

%ImGray = histeq(ImGray);

corners = detectFASTFeatures(ImGray, 'MinContrast',0.0001,'MinQuality',0.05);
J = insertMarker(ImGray,corners,'circle');

imshow(J)
```

Oriented FAST and rotated BRIEF (ORB) Feature Detector:

ORB is a recently developed technique from OpenCV labs in 2011 and is FAST based competitor to SIFT and is commonly used to detect facial features and is useful in facial expression recognition. Biggest advantage of ORB to SIFT is it being two orders faster than SIFT on the task of feature detection while being as performant as SIFT algorithm (Tyagi, 2019).

ORB algorithm uses a multiscale image pyramid to overcome the lack of orientation information from FAST features. The image pyramid consists of images that are versions of the original image at different resolutions. After creating the image pyramid, ORB uses FAST to detect keypoints in the image. After locating keypoints, ORB assigns an orientation keypoints depending on how the intensity change (Tyagi, 2019). Then, each keypoints with their orientations are taken by BRIEF and assigned to binary feature vectors (Tyagi, 2019).

One algorithm proposed by Ben Hernández-Pérez is the use of ORB features in fusion to extracted Local binary patterns with SVMs. (Hernández-Pérez., 2021) His approach to ORB feature detection was different in that he divided face into uniform regions based on size and randomly assigned number of features to be detected and if the number of feature points happened to exceed the limit number, non-maxima suppression was applied to choose the strongest features. (Hernández-Pérez., 2021) This way, he managed to circumvent the inconvenience of extracting many meaningless features. (Hernández-Pérez., 2021)

Below are our experimentations with MATLAB's extractORBFeatures() function:


Figure 1
File Edit View Insert Tools Desktop Window Help

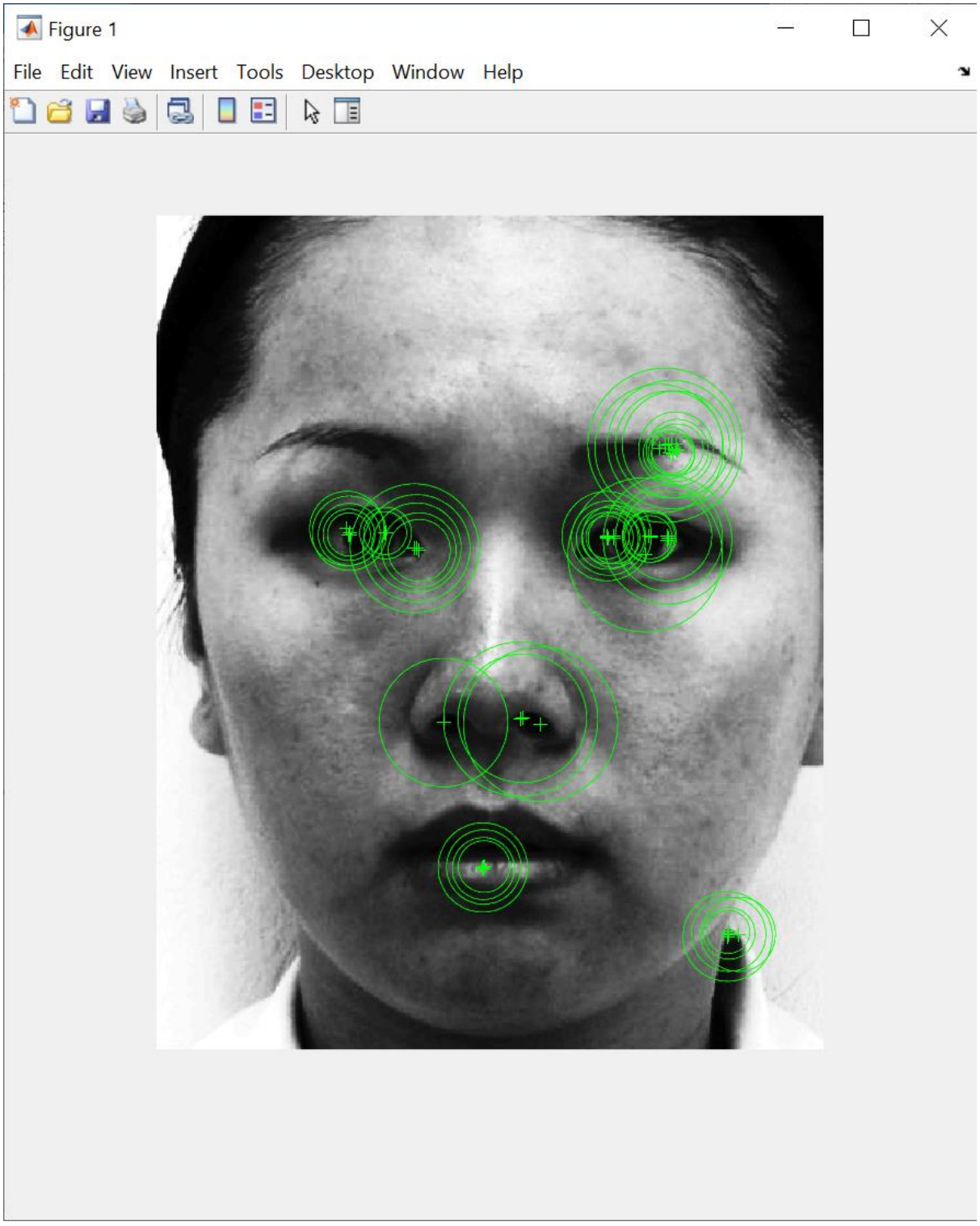


Figure 1
File Edit View Insert Tools Desktop Window Help


```
clear all; close all; clc;

Im = imread("f04_dfh_nx.jpg");
I = rgb2gray(Im);
points = detectORBFeatures(I);
I = histeq(I);
points2 = detectORBFeatures(I);



figure
imshow(I)
hold on
plot(points.selectStrongest(30), ShowScale=true)
hold on
plot(points2.selectStrongest(30), ShowScale=true)
```

Optical Flow:

Optical flow is commonly used technique to tracking of facial features. Optical flow can be efficiently used in conjunction with SIFT features. Optical flow can be used in conjunction with detected FAST features for tracking local feature points. This way movements in eyebrow, mouth or eyes can give valuable information to be fed into a Convolution Neural Network. One way to use optical flow for facial expression recognition is by utilizing sparse correspondence optical flow algorithms such as Cohn-Kanade optical flow algorithm.

When optical flow algorithm applied to emotion transition video frames from CK+ database, the result was as above. As the emotion starts from neutral and goes to peak happy, the velocity vectors track that motion. In areas such as forehead and hair where motion is almost nonexistent, the vectors can barely be seen in comparison to the ones where motion occurs such as areas next to lips and eyes as well as some parts of cheek and chin. The code that is used for this algorithm is as below;

```
faceDetector = vision.CascadeObjectDetector()
```

```
vidReader = VideoReader('neutral.mp4');

h = figure;

movegui(h);

hViewPanel = uipanel(h,'Position',[0 0 1 1],'Title','Plot of Optical Flow
Vectors');

hPlot = axes(hViewPanel);

while hasFrame(vidReader)

    frameRGB = readFrame(vidReader);

    bbox = step(faceDetector, frameRGB);

    frameGray = im2gray(frameRGB);

    newFrame = imcrop(frameGray,bbox);

    opticFlow = opticalFlowHS

    flow = estimateFlow(opticFlow,newFrame);

    imshow(newFrame)

    hold on

    plot(flow,'DecimationFactor',[5 5],'ScaleFactor',60,'Parent',hPlot);

    hold off

    pause(10^-3)

end
```

References

Carcagnì,Pierluigi, Coco,Marco Del, Leo,Marco & Distante,Cosimo "Facial expression recognition and histograms of oriented gradients: a comprehensive study."10.1186/s40064-015-1427-3

Chen, Junkai, Chen, Zenghai, Chi, Zheru & Fu, Hong, “Facial Expression Recognition Based on Facial Components Detection and HOG Features, Scientific Cooperations International Workshops on Electrical and Computer Engineering Subfields 22-23 August 2014, Koc University, ISTANBUL/TURKEY

Hernández-Pérez,Ben (2021), “Facial Expression Recognition with LBP and ORB Features”,hindawi.com, https://www.hindawi.com/journals/cin/2021/8828245/.

Kyrkou, C. (2018, May 24). *Object Detection Using Local Binary Patterns - Christos Kyrkou*. Medium.

https://ckyrkou.medium.com/object-detection-using-local-binary-patterns-50b165658368

Tyagi, Deepanshu, “Introduction to ORB (Oriented FAST and Rotated BRIEF)” ,medium.com, 1 January 2019, sourced from: https://medium.com/data-breach/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf

Tyagi, Deepanshu, “Introduction to FAST (Features from Accelerated Segment Test)”, medium.com, 2 January 2019, sourced from: https://medium.com/data-breach/introduction-to-fast-features-from-accelerated-segment-test-4ed33dde6d65

Shah, A. (2018, July 5). *Through The Eyes of Gabor Filter - Anuj shah (Exploring Neurons)*. Medium.

https://medium.com/@anuj_shah/through-the-eyes-of-gabor-filter-17d1fdb3ac97