CSE185
Introduction to Computer Vision
Lab 06: Edge Detection and Face Recognition

Instructor: Prof. Ming-Hsuan Yang

TA: Tiantian Wang & Tsai-Shien Chen

Sobel Filtering

- Sobel filter computes the gradients of input image:

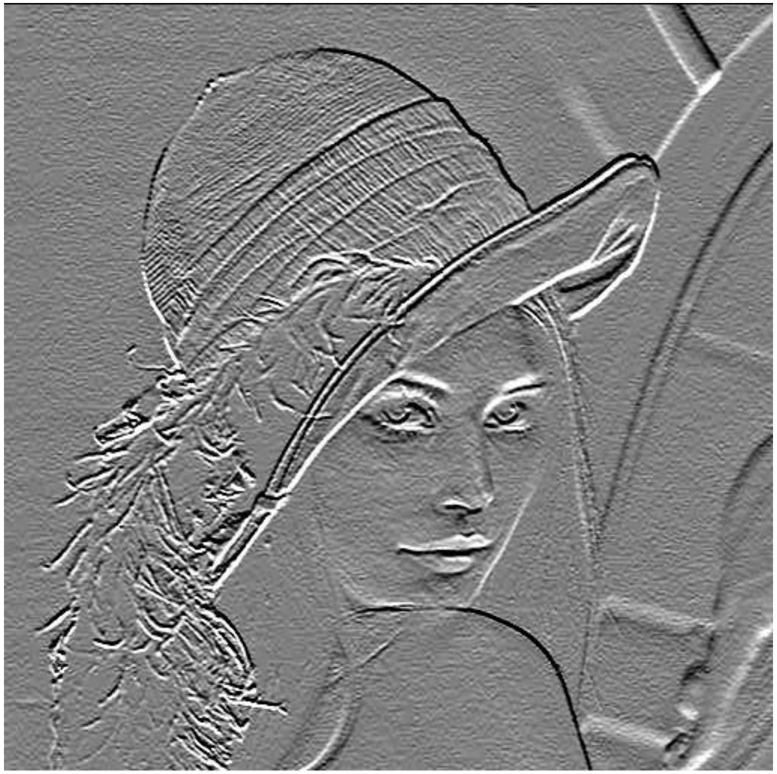
$$H_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \text{ or } H_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

Sobel Filtering

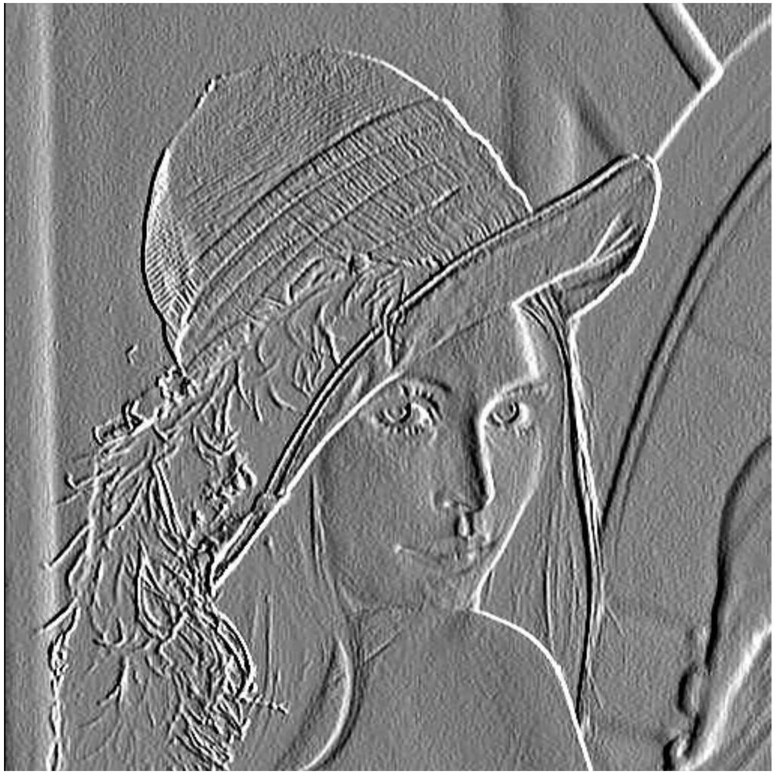
- Sobel filter computes the gradients of input image:

$$G_y = H_y \otimes I \;\text{ and }\; G_x = H_x \otimes I$$

convolution, or spatial filtering



Horizontal Edge Response



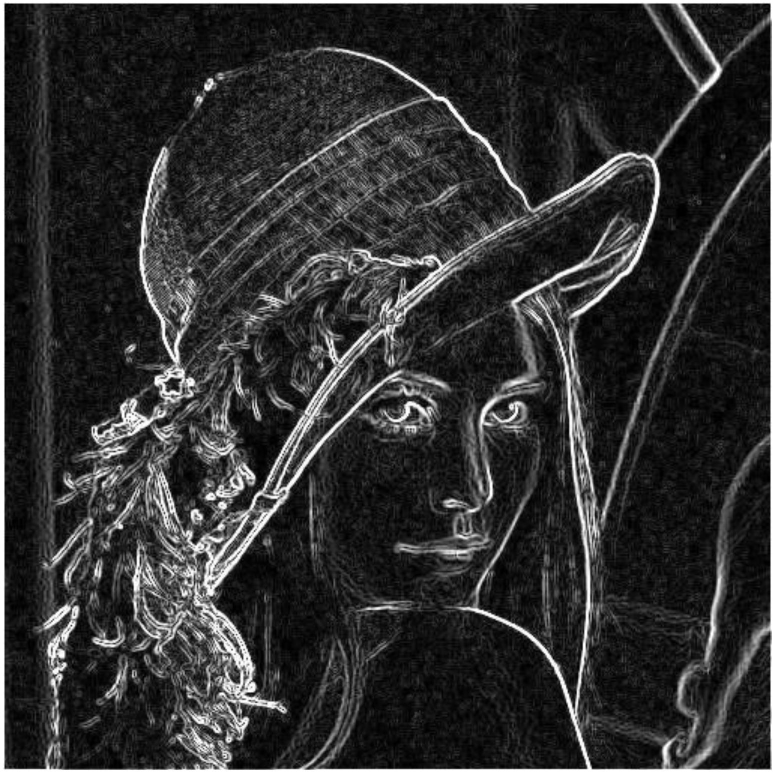
Vertical Edge Response

Sobel Filtering

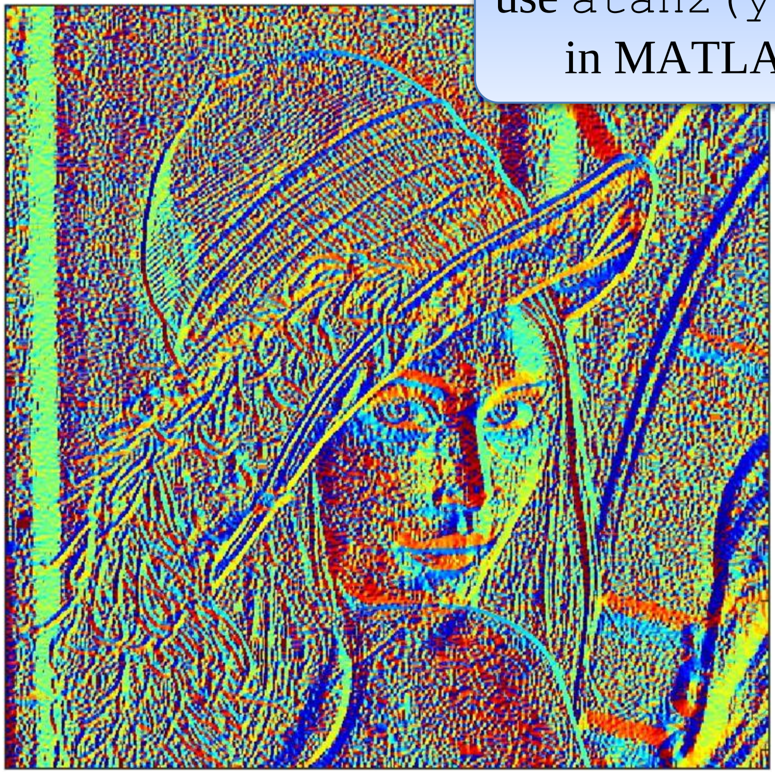
- Compute gradient magnitude and orientation:

$$M = \sqrt{G_y^2 + G_x^2} \text{ and } \theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

use `atan2(y, x)` in MATLAB



Magnitude

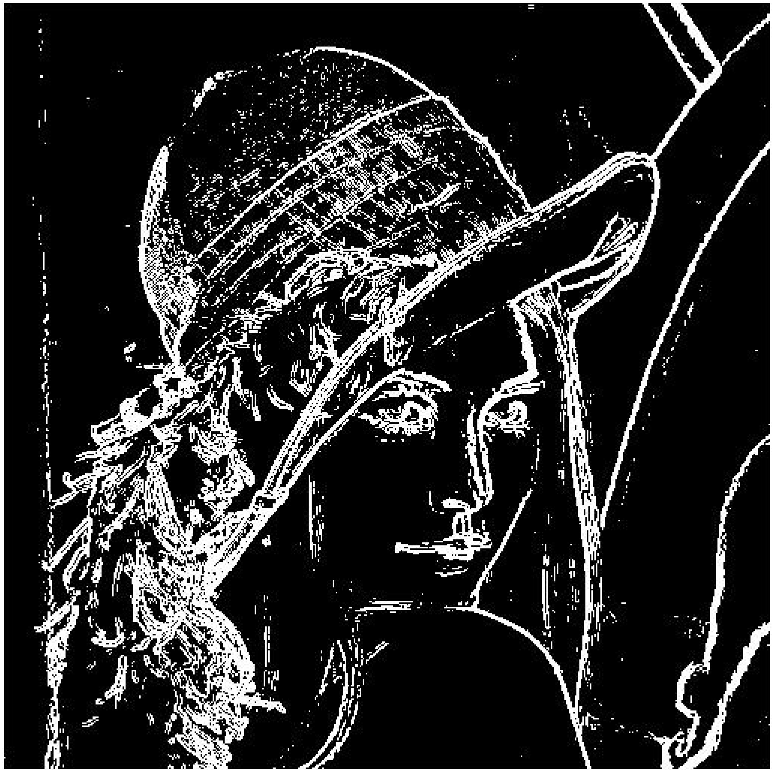


Orientation

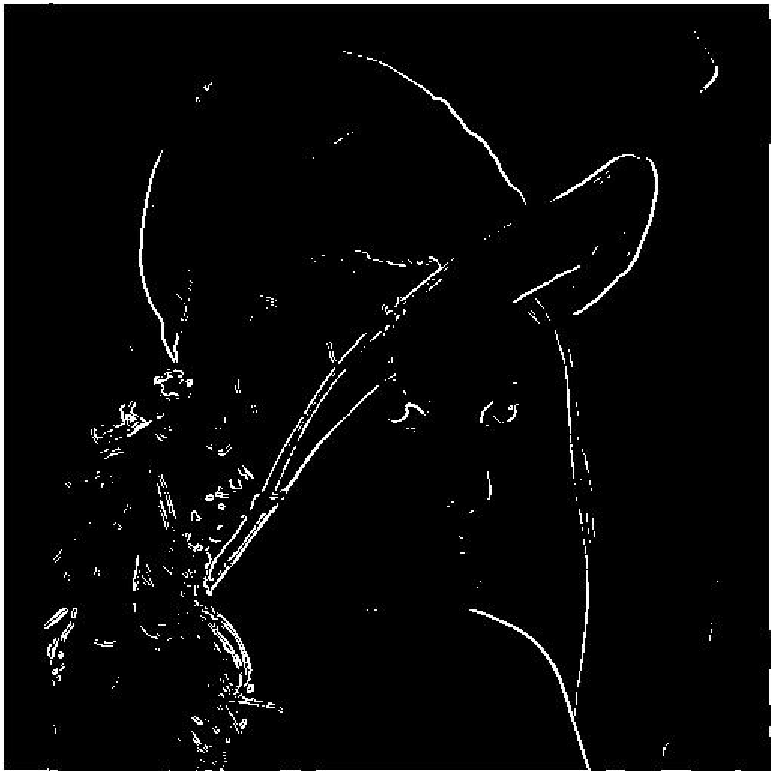
Edge Detection

- How to localize edge positions?
- Apply a threshold:

$$edge = (gradient\ magnitude) > threshold$$



threshold = 0.3



threshold = 1

sobel_feature.m

- In sobel_feature.m:

```
function [magnitude, orientation] = sobel_feature(img)

    % horizontal edge
    Hy = [1, 2, 1; 0, 0, 0; -1, -2, -1];
    % vertical edge
    Hx = [1, 0, -1; 2, 0, -2; 1, 0, -1];

    %% Sobel filtering


    %% compute gradient magnitude and orientation
    magnitude = img;
    orientation = img;

end
```

use `imfilter` or your sobel_filter.m from lab03

lab06_edge.m

- In lab06_edge.m:

```
img = im2double(imread('lena.jpg'));

%% compute gradient magnitude and orientation with
Sobel filter
[magnitude, orientation] = sobel_feature(img);

%% apply thresholding to detect edge
threshold = 0.3;
e = magnitude > threshold;
```

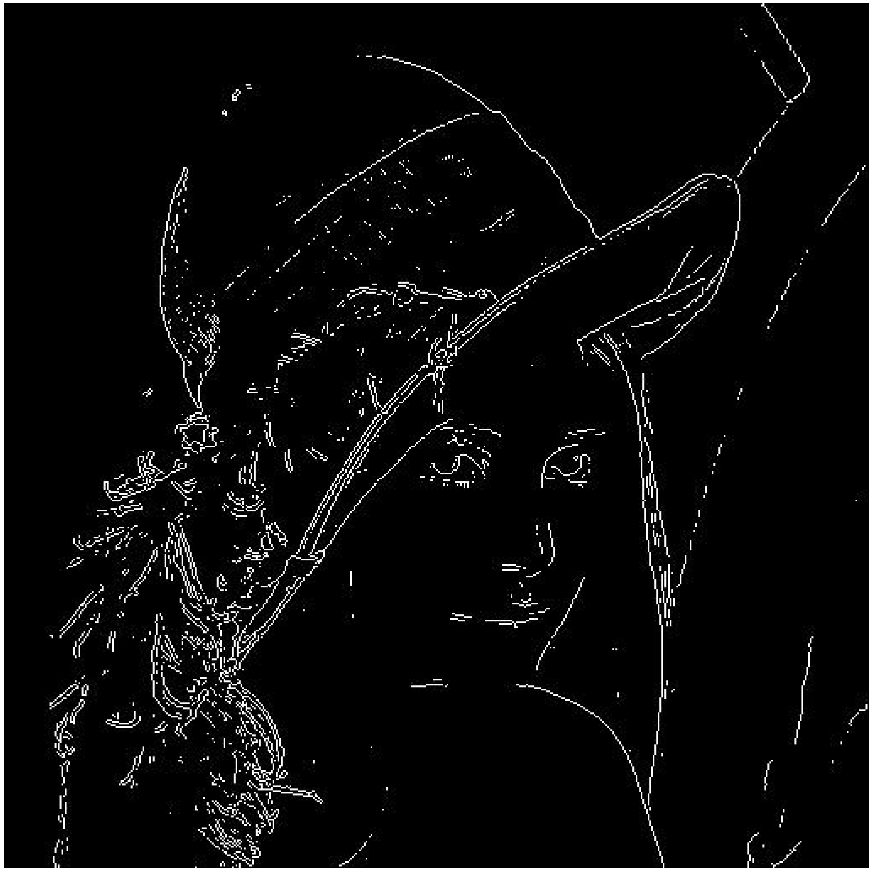
From Sobel Filter to Canny Edge Detection

- How to remove noise and detect accurate edges?
- Canny edge detection:
  - Apply Gaussian filter to input image in order to remove noise
  - Compute image gradient
  - Apply non-maximum suppression to detect local maxima
  - Apply double thresholding to detect weak and strong edges
  - Apply hysteresis thresholding to localize connected edges

Sobel Edge Detector

- In MATLAB, use `edge(img,  'Sobel')`

Input Image

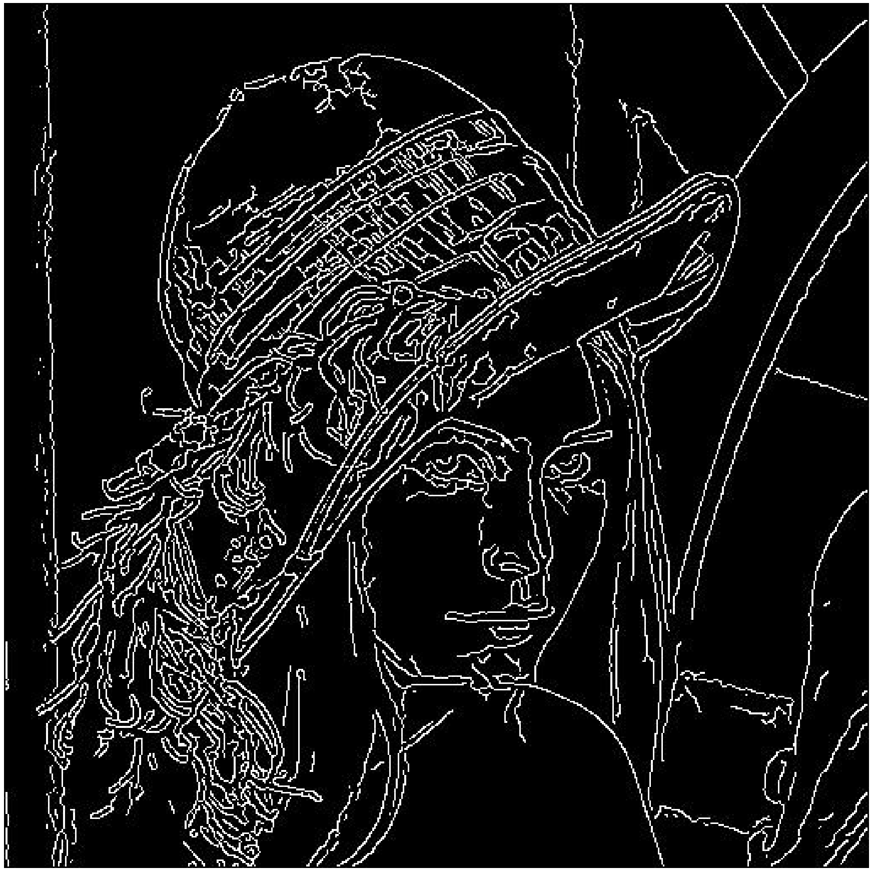


Sobel Edge Detection

Canny Edge Detector

- In MATLAB, use `edge(img, 'Canny')`

Input Image



Canny Edge Detection

lab06_edge.m

- In lab06_edge.m:

```
%% use built-in function to detect edge
e1 = img; % change img to sobel edge detection
e2 = img; % change img to canny edge detection

figure, imshow(img);
figure, imshow(e1); title('Sobel Edge');
figure, imshow(e2); title('Canny Edge');
```

AT&T Face Dataset

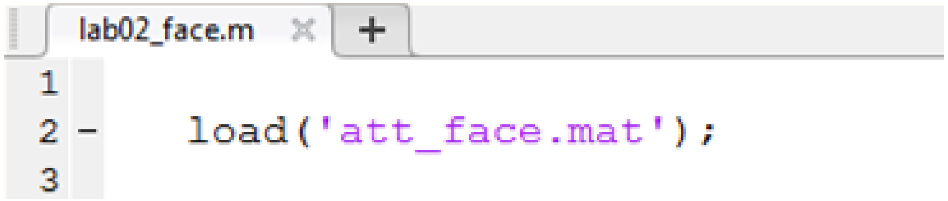
- There are 40 training images and 160 testing images from 40 different identities/people (labeled as 1 to 40).

AT&T Face dataset: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

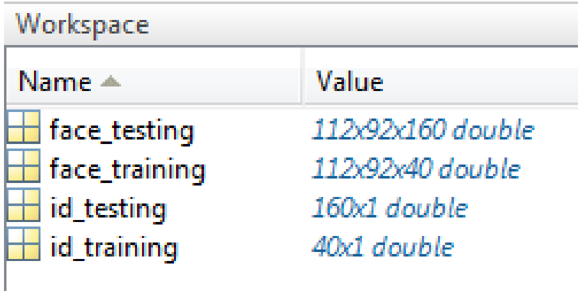
AT&T Face Dataset

- You don’t need to download the dataset. We have prepared a mat file `att_face.mat`, which contains 4 variables:
  1. `face_training` (112 × 92 × 40): training images
  2. `face_testing` (112 × 92 × 160): testing images
  3. `id_training` (40 × 1): the id/label of training images
  4. `id_testing` (160 × 1): the id/label of testing images

- Use `load(‘att_face.mat’)` to load the mat file to your workspace:



```
load('att_face.mat');
```



Name	Value
face_testing	112x92x160 double
face_training	112x92x40 double
id_testing	160x1 double
id_training	40x1 double

Face Recognition

- Our goal: predict the id/labels for 160 testing images

- Steps:
  1. For each testing image, compute the 40 square errors from the training images
  2. Find the index with the minimum square error
  3. Calculate the accuracy between your predict labels and the ground truth labels (`id_testing`)

lab06_face.m

- In lab06_face.m:

```
for i = 1:num_testing
    %% extract testing image
    img_test = face_testing(:, :, i);
    vec_test = img_test(:);

    error = zeros(num_training, 1);
    for j = 1:num_training

        %% extract training image
        img_train = face_training(:, :, j);
        vec_train = img_train(:);

        %% compute the square error between feature vectors
        diff = vec_train - vec_test;
        error(j) = sum( diff .^2 );
    end
    %% find the image id with minimal error
    [~, min_id] = min(error);
    id_predict(i) = min_id;

end
```

use image intensity as feature vector

use image intensity as feature vector

element wise operation

lab06_face.m

- Compute accuracy:

```
%% compute accuracy
accuracy = sum(id_testing == id_predict)/num_testing;
fprintf('Accuracy = %f\n', accuracy);
```

- Accuracy = 0.7375 if using intensity as feature vectors

- Your job: change feature vectors to multi-scale Sobel features (magnitude or orientation)

multiscale_sobel_feature.m

- In multiscale_sobel_feature.m:

```
function feature = multiscale_sobel_feature(img, scale)

    % initialize feature vector
    feature = [];

    for i = 1:scale

        % compute sobel feature
        f = ???;

        % concatenate feature vector
        feature = cat(1, feature, f(:));

        % down-sample image by 2

    end
end
```

use your sobel_feature.m

lab06_face.m

- In lab06_face.m:

```
for i = 1:num_testing
    %% extract testing image
    img_test = face_testing(:, :, i);
    vec_test = multiscale_sobel_feature(img_test, scale);

    error = zeros(num_training, 1);
    for j = 1:num_training

        %% extract training image
        img_train = face_training(:, :, j);
        vec_train = multiscale_sobel_feature(img_train, 
scale);
        %% compute the square error between feature vectors
        diff = vec_train - vec_test;
        error(j) = sum( diff .^2 );
    end
    %% find the image id with minimal error
    [~, min_id] = min(error);
    id_predict(i) = min_id;

end
```

lab06_face.m

- In lab06_face.m:

```
% Using gradient magnitude as features:
%---------------------------------------%
% Scale |  Accuracy
%---------------------------------------%
%   1   |   0.5313
%---------------------------------------%
%   2   |
%---------------------------------------%
%   3   |
%---------------------------------------%
%
% Using gradient orientation as features:
%---------------------------------------%
% Scale |  Accuracy
%---------------------------------------%
%   1   |   0.5563
%---------------------------------------%
%   2   |
%---------------------------------------%
%   3   |
%---------------------------------------%
```

Assignment

1. Implement sobel_feature.m to compute gradient magnitude and orientation from Sobel filtering (save the output images as lena_sobel_magnitude.jpg, lena_sobel_orientation.jpg and lena_edge_threshold_*i*.jpg for edge detection thresholding at *i*)
2. Use `edge()` to apply Sobel and Canny Edge detection in lab06_edge.m (save the output images as lena_sobel.jpg and lena_canny.jpg)
3. Run lab06_face.m and understand the code
4. Implement multiscale_sobel_feature.m, and replace image feature vectors
5. Fill in the form in the bottom of lab06_face.m
6. Upload all output images and your sobel_feature.m, multiscale_sobel_feature.m, lab06_edge.m, lab06_face.m separately.