IN5520/IN9520 Mandatory term project-2 2018

Feature Evaluation and Classification

Hao, Zhao

Department of Geosciences,

University of Oslo

Problem Description

In this project of feature evaluation and classification, we work on the image classification problem by using the selected image features and multivariate Gaussian classifier. The following core elements of theory are used in the feature evaluation and classification.

Gray Level Co-occurrence Matrix

The image features are derived from the Gray Level Co-occurrence Matrix (GLCM), which is a statistical tool to measure the texture of an image by counting the co-occurring gray level at a given pixel offset(dx,dy). The co-occurrence matrix is defined as equation (1)

$$C(i,j) = \sum_{x=1}^{n}\sum_{y=1}^{m}\begin{cases}1, & if(I(x,y) = i \; and \; I(x+dx, y+dy) = j \\ 0, & otherwise\end{cases} \quad (1)$$

Where index I and j represent the pixel values, the x, y are the spatial coordinates within the image, and dx, dy are the displacement of analyzing point pairs which are moving in the defined direction and offset. By select the specific direction and offset, the GLCM matrix can be used to classify the different image textures.

Bayes Rule

The Gaussian type of classification is based on the Bayesian decision theory. The rule of Bayes theory, defined as equation (2), is to derive the posterior probability $P(\omega_s, x)$ based on the prior probability $P(\omega_s)$, and the likely hood $P(x, \omega_s)$. For the classification problem, p(x,ω_s) is the probability density function that models the likelihood for observing gray level x if the pixel belongs to class ω_s, p(ω_s,x) represent the posterior probability that the pixel belongs to class ω_s, and the P(x) is a scaling factor to ensure that the summed probabilities equal to 1.

$$P(\omega_s, x) = \frac{P(x,\omega_s).P(\omega_s)}{P(x)} \quad (2)$$

Multivariate gaussian classification

The multivariate gaussian classification algorithm is a Bayesian rule-based approach used to handle the multivariate classification problem. The likelihood $P(x, \omega_s)$ is modeled by the probability density function (3). This pdf is parameterized by the mean vector μ and the covariance matrix Σs. By combing the prior probability $P(\omega_s)$ and scaling factor $P(x)$, we can derive the posterior probability $P(\omega_s, x)$ for classification.

$$P(x, \omega_s) = \frac{1}{\frac{l}{(2\pi)^{\frac{1}{2}}|\sum s|^{\frac{1}{2}}}} e^{\left[-\frac{1}{2}(x-u_s)' \sum_s -1 \, (x-u_s)\right]} \quad (3)$$

Numerical Result

1. Choosing GLCM images to work with

In this feature evaluation and classification project, we are going to use one mosaic picture, which contains 4 different textures images, and it's corresponding manually GLCM features to train the multivariate Gaussian classifier and apply the classification to itself and two other testing images. The training used figure and class label display at Fig. 1, and the two figures used for the classification test show in Fig.2.

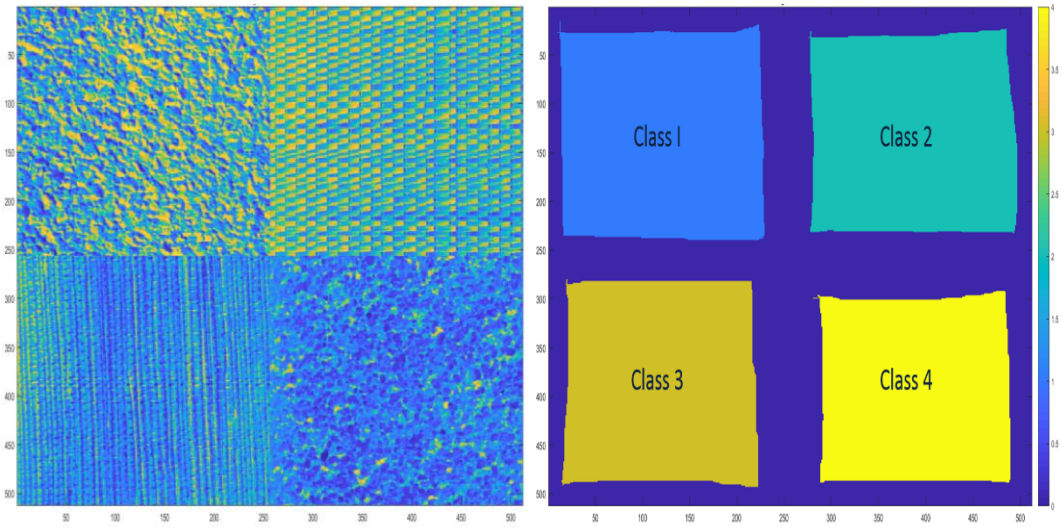

Figure 1. Mosaic picture used for Training(left) and given Class labels(right)

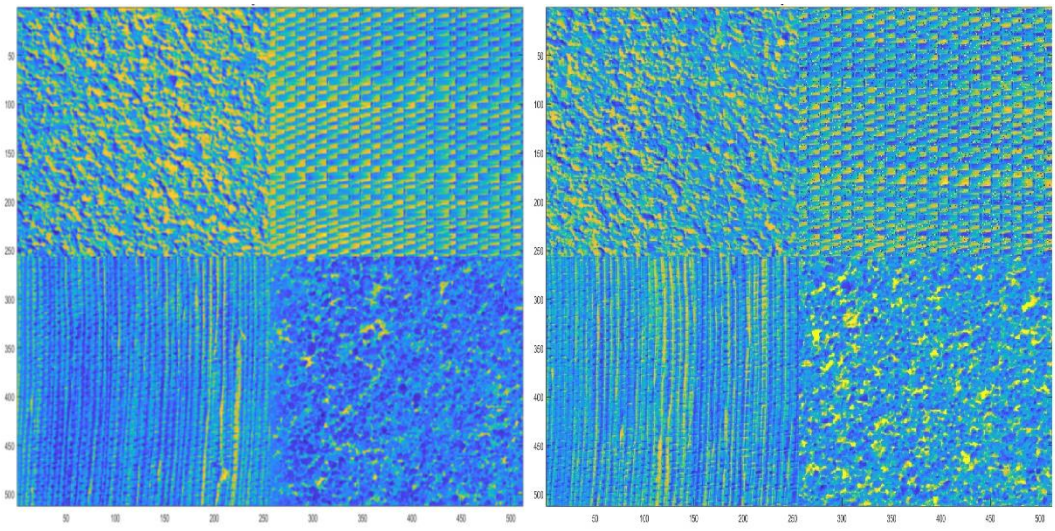

Figure 2. Mosaic pictures 1 (left) and 2 (right) used for Testing

Prior to derive the features for the training use of Gaussian multivariate classifier, we use the Gray Level Co-occurrence Matrix (GLCM) to analysis the texture of each sub-images. All the textures have derived the GLCM matrices with the analysis angle {0,45,90 and 135 degree}. In order to find the best GLCM matrices which be used to characterize the features of the 4 textures we analyze and choose the GLCM as following:

With respect to the texture-1 (cf. Fig. 3), this texture is composed of strong gray level variation but without dominant pattern directions. The corresponding GLCM matrices shows similar trend in most of the analysis directions. Similar to texture-1, the texture-4 (cf. Fig. 6), does have any particular direction but has less gray level variation compared to texture-1 which could also be observed in its GLCM matrices. The texure-2 (cf. Fig. 4), and texture-3 (cf. Fig. 5), have the pattern with apparent direction along 0 and 90 degree respectively, thus the corresponding GLCM with these directions could be selected to the GLCM feature calculation. Thus, in this project, we think the combination of GLCM derived in directions 0 and 90 degrees are the good basis for the GLCM features calculation and classification. In the implementation, we derived the 0+90 GLCM matrices for all textures and included in figures3 to figures 6.

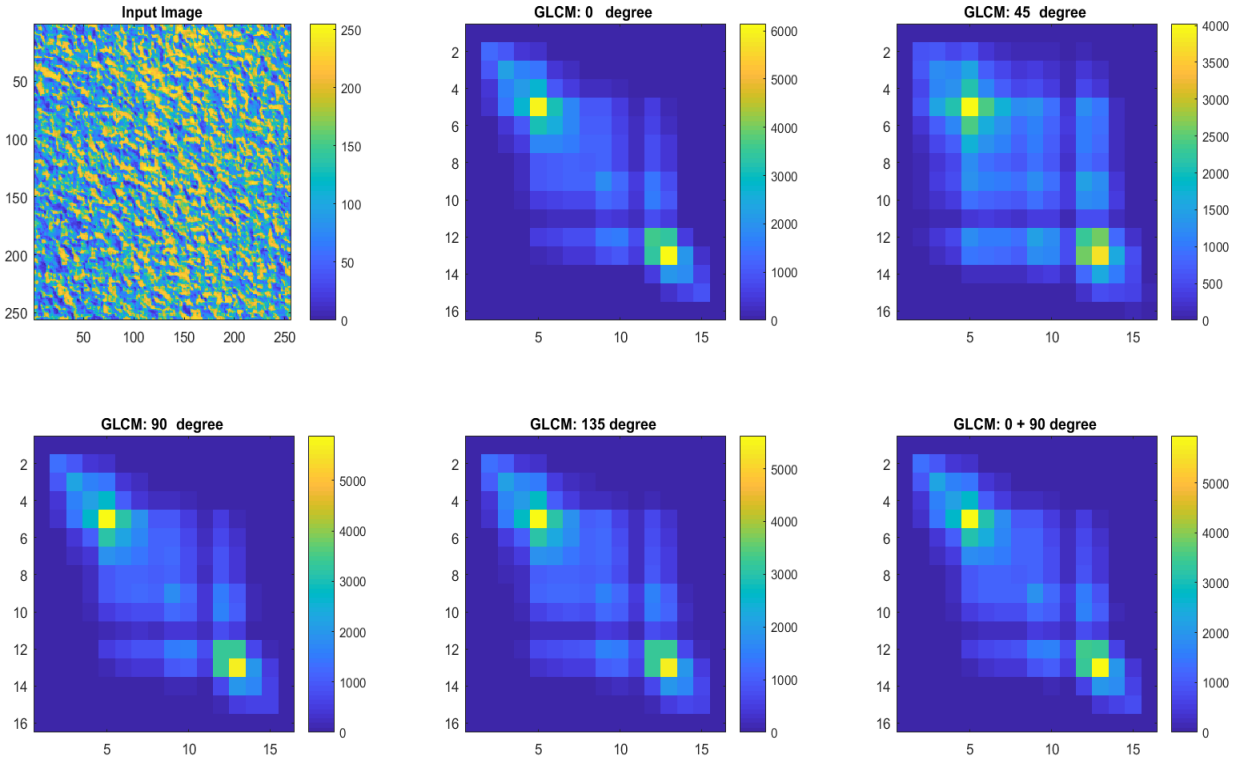

Figure 3. Texure-1(top left) and corresponding GLCM matrices (0,45,90,135 and 0+90 degree)

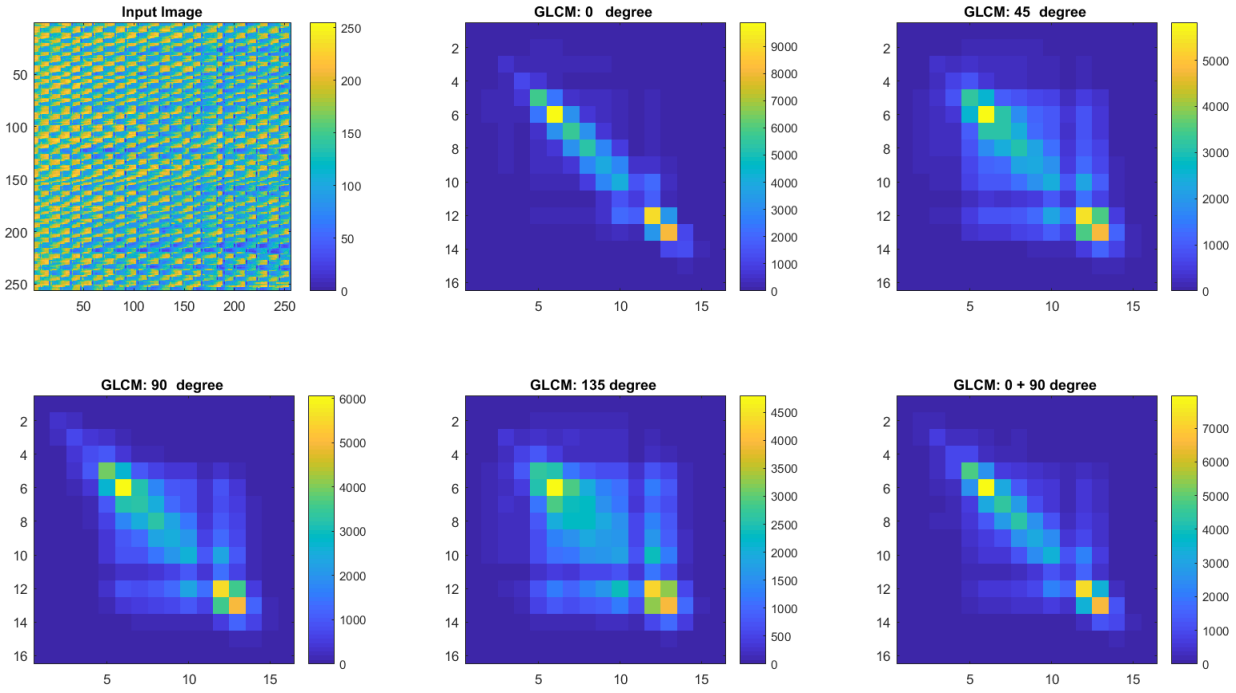

Figure 4. Texure-2(top left) and corresponding GLCM matrices (0,45,90,135 and 0+90 degree)

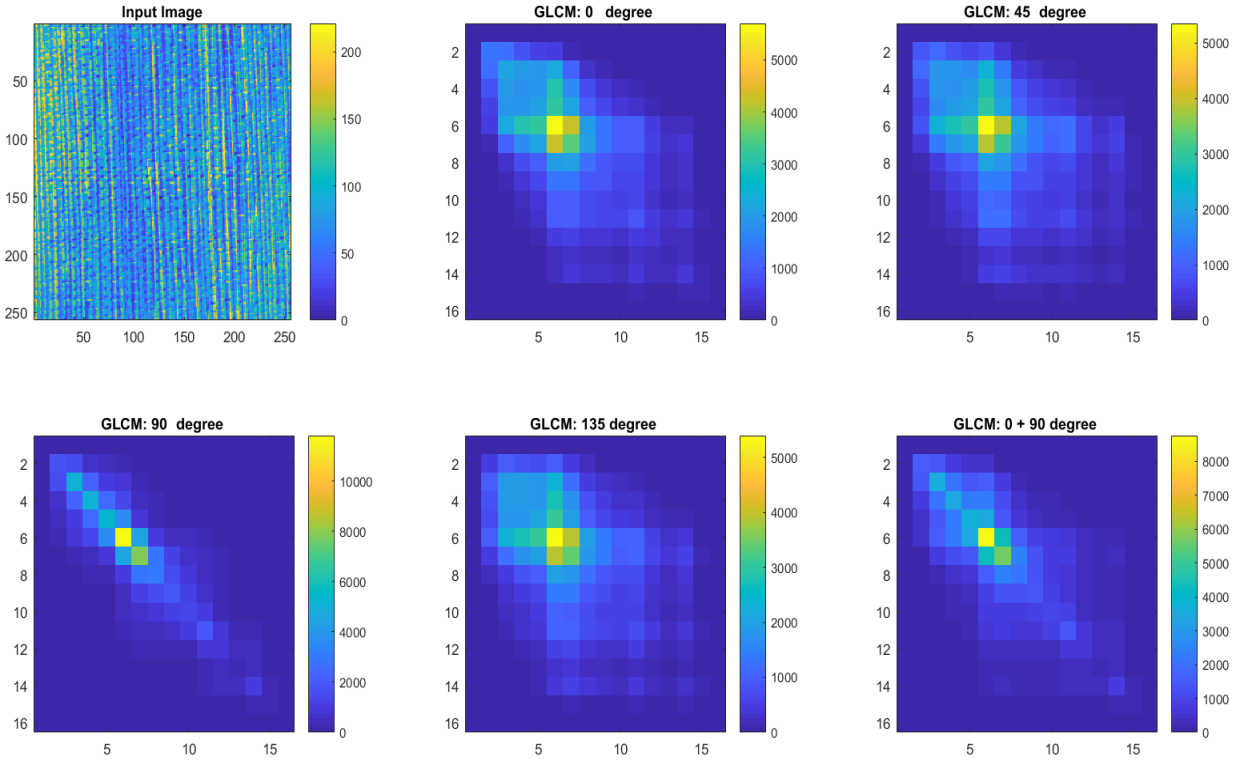

Figure 5. Texure-3(top left) and corresponding GLCM matrices (0,45,90,135 and 0+90 degree)

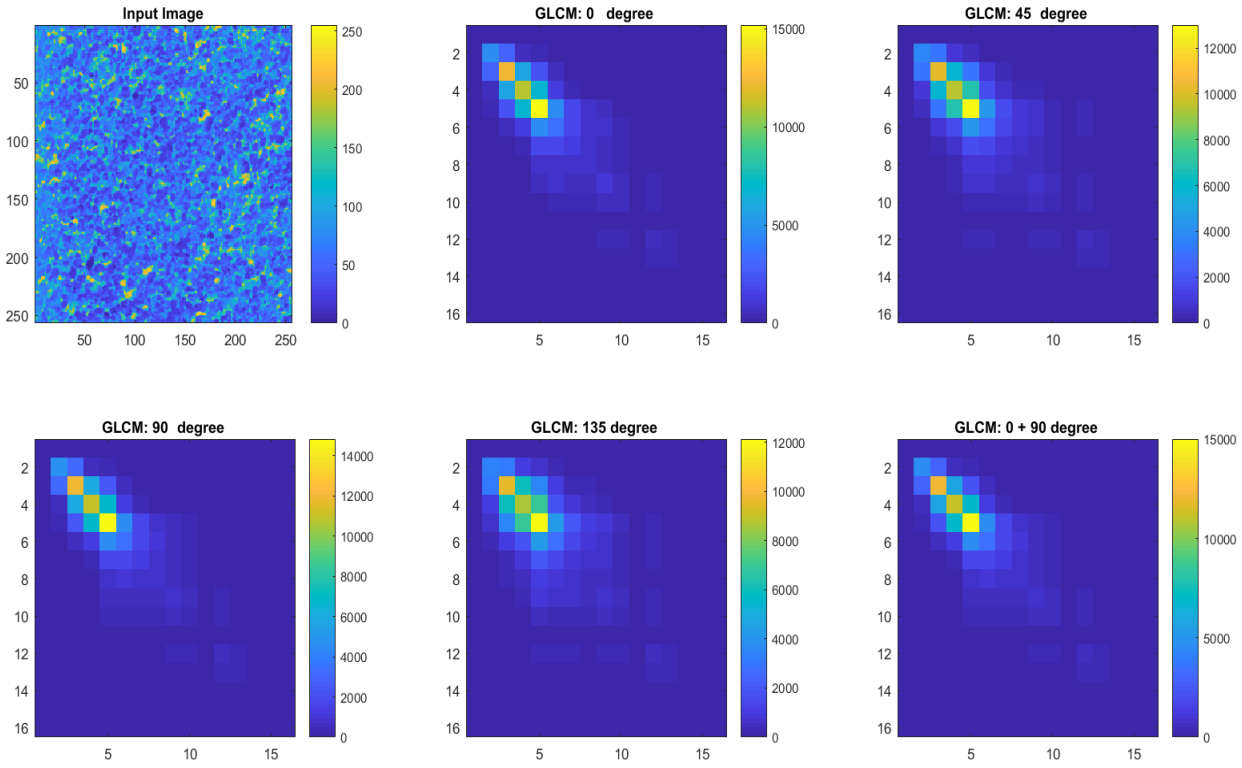

Figure 6. Texure-4(top left) and corresponding GLCM matrices (0,45,90,135 and 0+90 degree)

2. Discussing new features by subdividing the GLCM matrices

Based on the selected GLCM matrices generated in the direction of 0 and 90 degrees, we generated the GLCM quadrant-based features. These features can be defined as following: we divide the 16x16 GLCM matrix into four quadrants, and then calculate the ratio of local summed probability to the total probability. We firstly derived the 4 features as below:

$$FeatureQ1\text{: } Q1 = \frac{\sum_{i=1}^{8}\sum_{J=1}^{8} \mathrm{P(i,j)}}{\sum_{i=1}^{16}\sum_{J=1}^{16} \mathrm{P(i,j)}}$$

$$FeatureQ2\text{: } Q2 = \frac{\sum_{i=1}^{8}\sum_{J=9}^{16} \mathrm{P(i,j)}}{\sum_{i=1}^{16}\sum_{J=1}^{16} \mathrm{P(i,j)}}$$

$$FeatureQ3\text{: } Q3 = \frac{\sum_{i=9}^{16}\sum_{J=1}^{8} \mathrm{P(i,j)}}{\sum_{i=1}^{16}\sum_{J=1}^{16} \mathrm{P(i,j)}}$$

$$FeatureQ4\text{: } Q4 = \frac{\sum_{i=9}^{16}\sum_{J=9}^{16} \mathrm{P(i,j)}}{\sum_{i=1}^{16}\sum_{J=1}^{16} \mathrm{P(i,j)}}$$

Apart from the above-mentioned quadrant-based features Q1 to Q4. We also noticed that the first quadrant of GLCM matrix shows sufficient features which are likely to be able to use in the differentiation of the 4 different features. Thus, we also include 4 additional features by sub-divide the first quadrant and derive the features with below formulation.

$$Feature Q5: Q5 = \frac{\sum_{i=1}^{4}\sum_{J=1}^{4} \mathrm{P(i,j)}}{\sum_{i=1}^{16}\sum_{J=1}^{16} \mathrm{P(i,j)}}$$

$$Feature Q6: Q6 = \frac{\sum_{i=1}^{4}\sum_{J=5}^{8} \mathrm{P(i,j)}}{\sum_{i=1}^{16}\sum_{J=1}^{16} \mathrm{P(i,j)}}$$

$$Feature Q7: Q7 = \frac{\sum_{i=5}^{8}\sum_{J=1}^{4} \mathrm{P(i,j)}}{\sum_{i=1}^{16}\sum_{J=1}^{16} \mathrm{P(i,j)}}$$

$$Feature Q8: Q8 = \frac{\sum_{i=5}^{8}\sum_{J=5}^{8} \mathrm{P(i,j)}}{\sum_{i=1}^{16}\sum_{J=1}^{16} \mathrm{P(i,j)}}$$

Based on the above formulation, we derived the quadrant-based feature values Q1-Q8 for the 4 textures. Figure 7 show the comparison of feature values for the different textures.

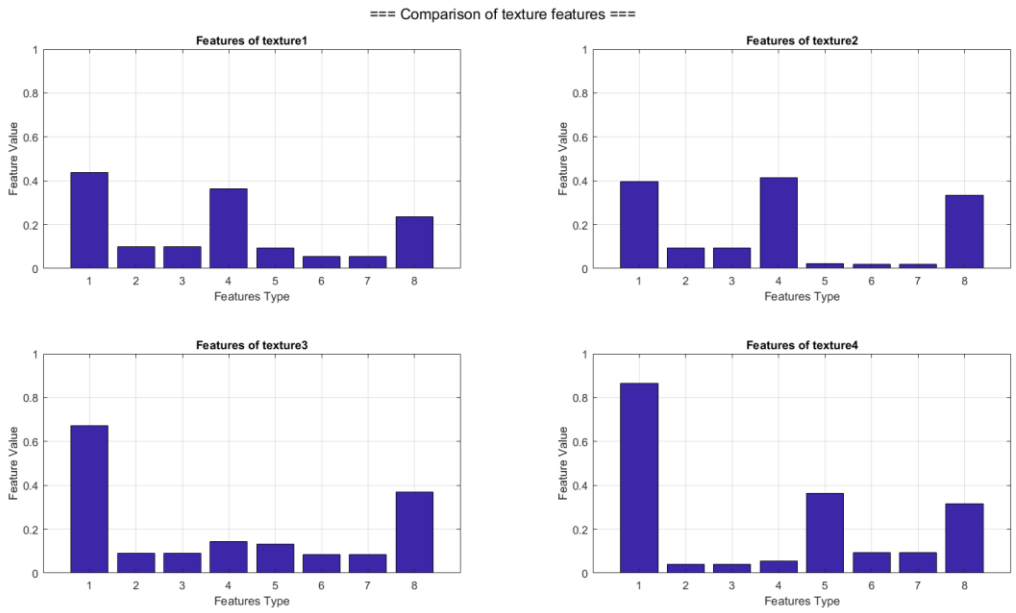

Figure 7. Feature values comparison for texture-1(top left), texure-2(top right), texture-3(bottom left) and texture-4(bottom right)

From the statistic of the quadrant-based feature (cf. Fig. 7), we see that the possibility of using different feature combination to differentiate the texture. In this project, we tried different combination, and propose to use the features 5, 6 and 8 for the following multivariate gaussian classification.

3. Selecting and implementing a subset of these features.

Before we start the classification, we derived the 8 quadrant-based features by using the sliding windows (size 31x31) with gray level 16 and direction 0+90 degrees on the training mosaic image. The figures 8,9 and10 present the derived quadrant-based features for training image and two testing images.

As mentioned, in the classification stage we tried different feature combinations, and choose the features 5, 6 and 8 from the training and testing image for the later multivariate gaussian classification.

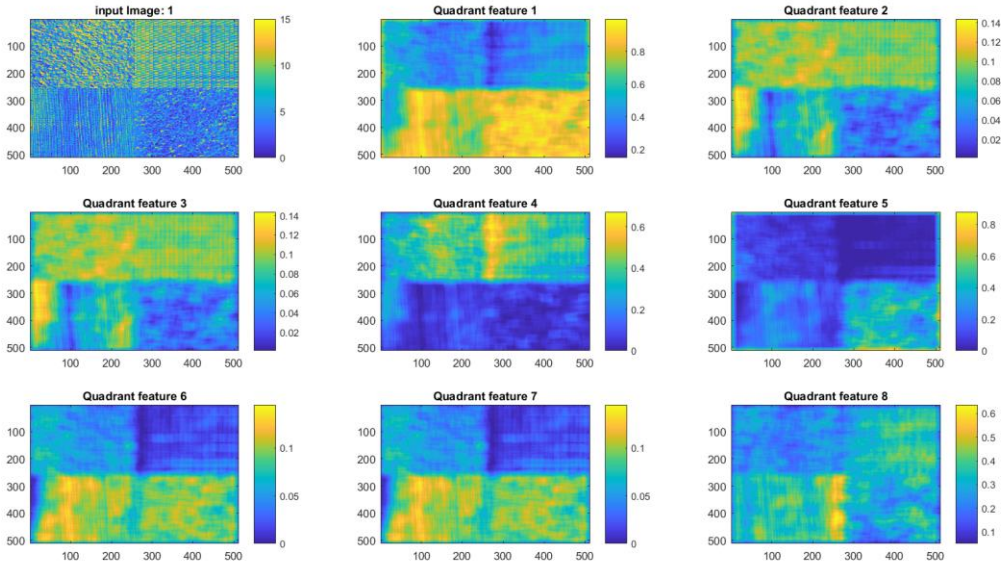

Figure 8. Training mosaic image (top left) and derived 8 quadrant-based features.

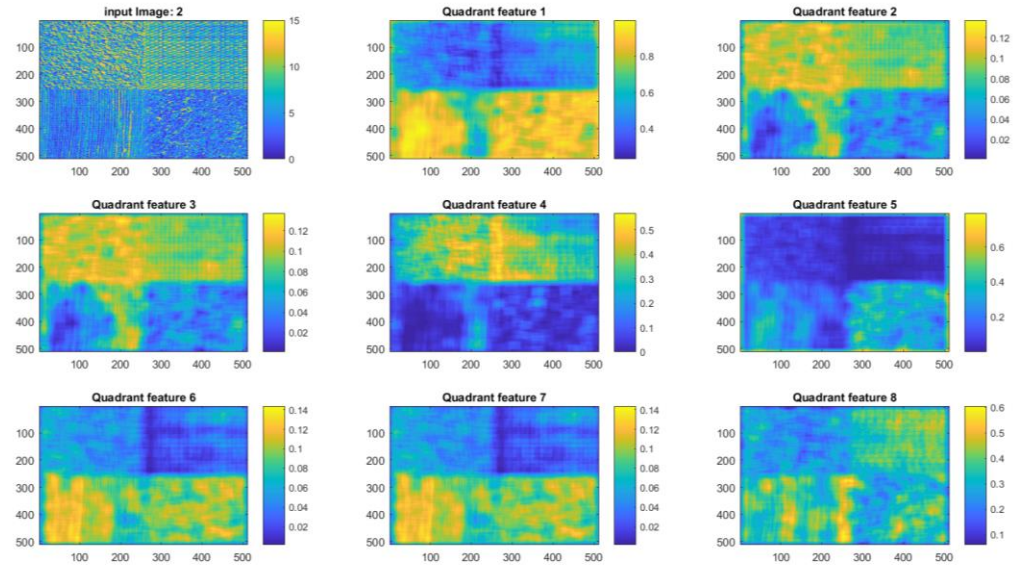

Figure 9. Testing mosaic image-1 (top left) and derived 8 quadrant-based features.

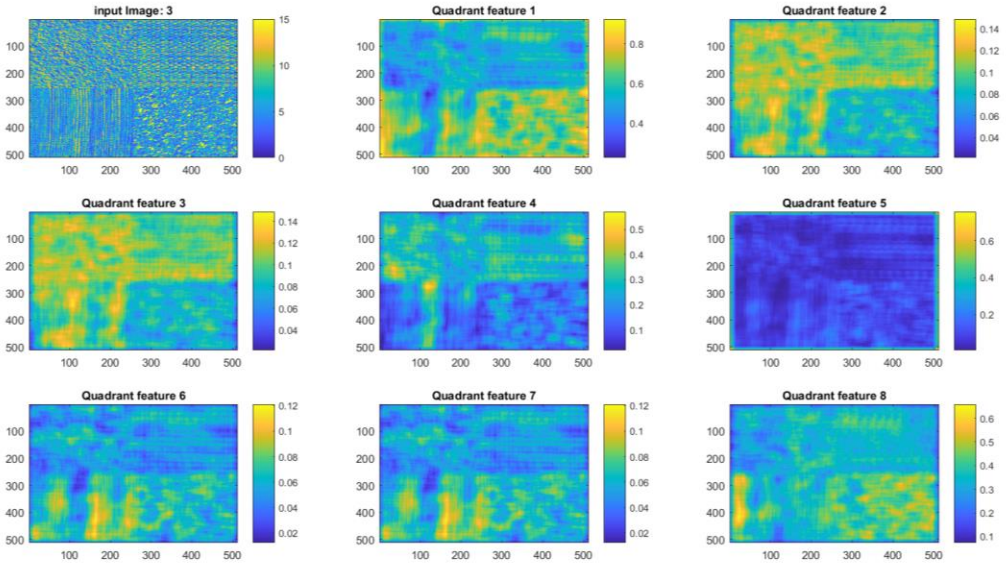

Figure 10. Testing mosaic image-2 (top left) and derived 8 quadrant-based features.

4. Implement a multivariate Gaussian classifier.

In this section, we implemented the multivariate gaussian classifier, the code is attached in the appendix section, and enclosed in the hand-in project code zip file.

5. Training the classifier based on the feature subset from point 3.

Based on the selected features (quadrant feature5, 6 and 8), we train the multivariate Gaussian classifier with the provided known class labels by analyze the mean value and covariance matrix for each class. The training dataset is also used for the verification for training accuracy. The following figure-11 shows the classification result.

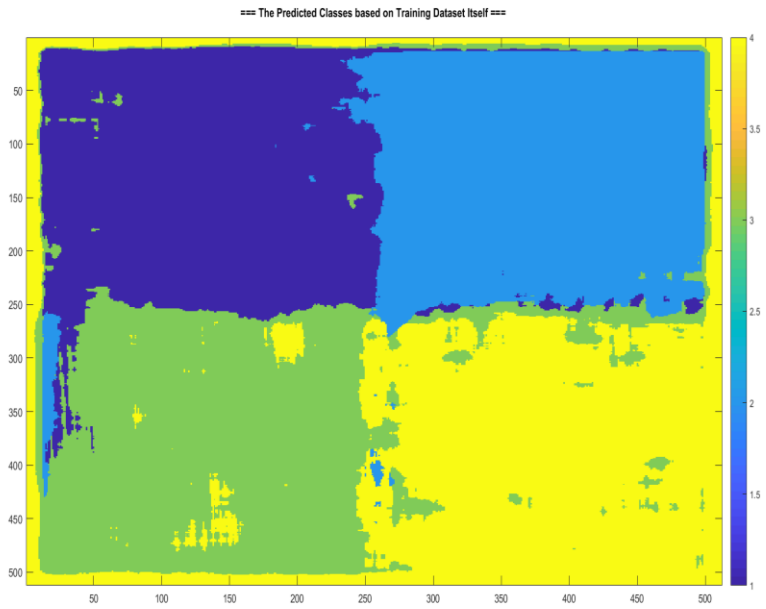

Figure 11. classification result based on training data

The confusion matrix, which is a table used to describe the performance of classification model, is based on the comparison the count of predicted class label to the true know class label. Based on the confusion matrix derived from the selected features of training image (cf. Fig. 12), we can see that the training has gain comparative high accuracy. The average overall accuracy is 95% : and the accuracy for individual classes (1-4) are 99.0%, 99.7%, 93.6% and 97.2%

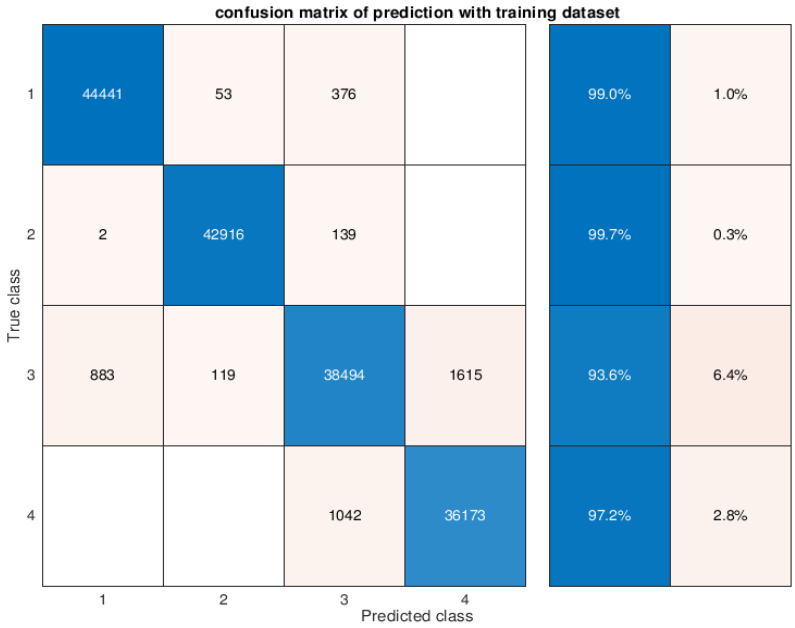

Figure 12. Confusion matrix of multivariate Gaussian classification by using training data

6. Evaluation of classification performance on the test dataset using the set of features selected in point 3.

In this section, we used the selected features from two mosaic picture for the classification testing. The training is based on the features from training mosaic image. The following figure13 and 14 are the classification result based on the testing dataset. The figure15 and 16 show the confusion matrices. We can see that the average overall accuracy for test image-1 is :88.9 %, and the accuracy for individual classes (1-4) are 97.3%, 93.8%, 85.1% and 77.4%. And for the testing image-2, we had comparatively low prediction accuracy, the average overall accuracy is 43%, and the prediction accuracy for individual classes are 59.8%, 25%, 73.8% and 7.2%.

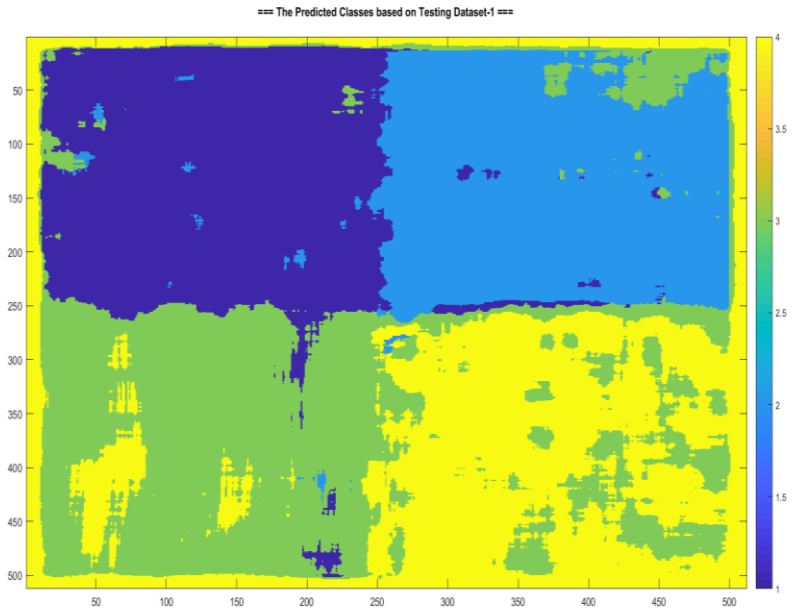

Figure 13. classification result based on testing data set-1

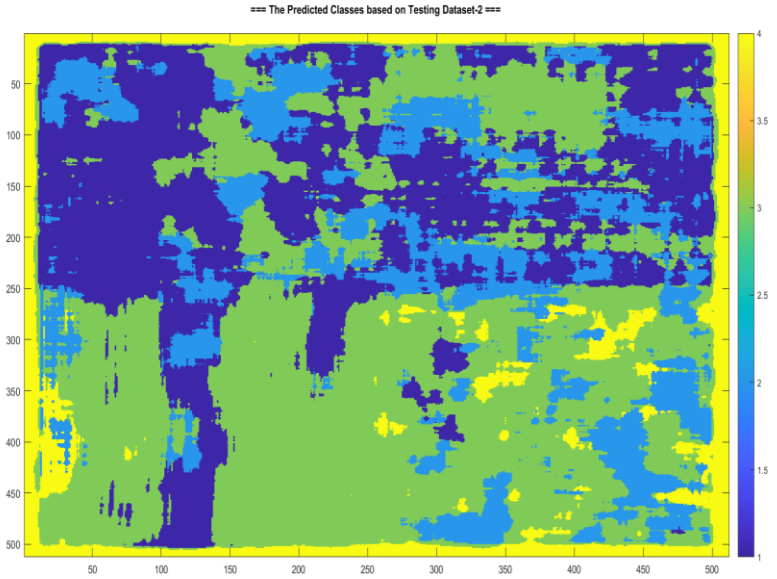

Figure 14. classification result based on testing data set-2

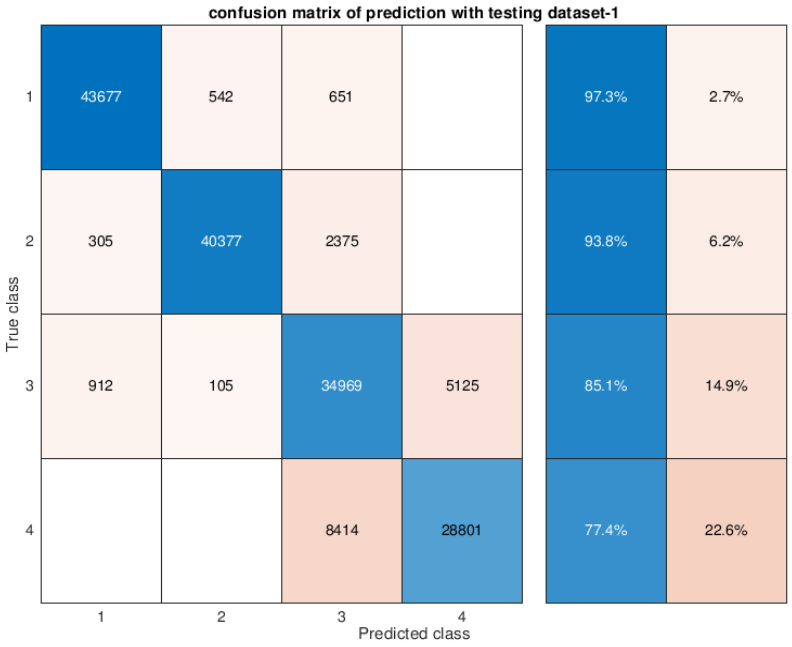

Figure 15. Confusion matrix of multivariate Gaussian classification by using testing data1

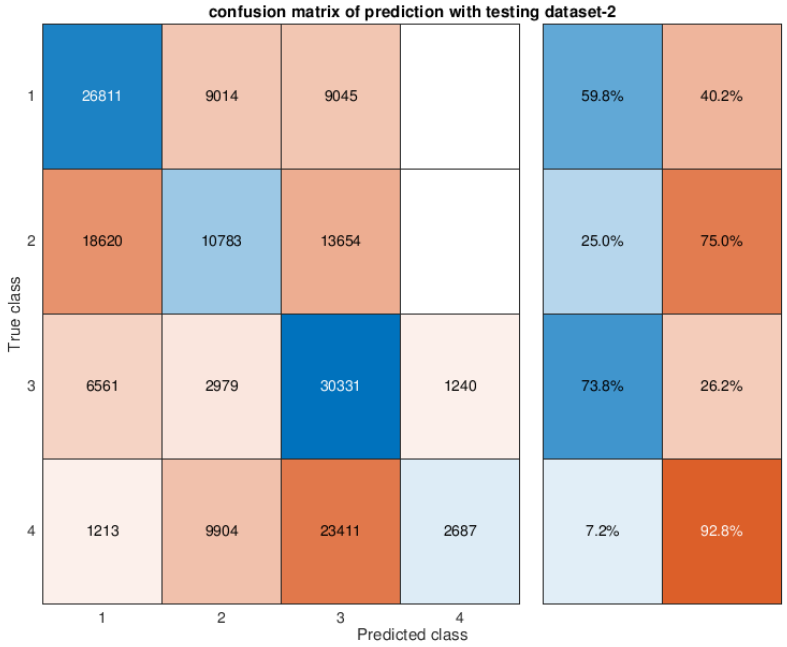

Figure 16. Confusion matrix of multivariate Gaussian classification by using testing data2

Appendix.

MatLab source code for multivariate Gaussian classifier.

```
function [ class_predict ] =
Multivariate_Gaussian_Classifier(TainingFeaturesArray,ClassLabels,TestFeatureArray)

%Function to apply multivariate Gaussian classication
% Code by Hao, Nov 06,2018
%---- input ----
%TainingFeaturesArray   : A 3D matrix (m x n x L) includes L features with feature size: m x n
%ClassLabels            : The predefined class labels
%TestFeatureArray:      : The test feature array (3D matrix with same dimension as
TainingFeaturesArray)
%---- output ----
%class_predict: The predicted class labels.

%% datasets for the code test only
if(nargin==0)

    test_input    = load('output_project2.mat');
    GLCM_features = test_input.GLCM_features;

    %sort the selected features into 3d matrix
    TainingFeaturesArray        = zeros(512,512,3);
    TainingFeaturesArray(:,:,1) = GLCM_features{1,5};
    TainingFeaturesArray(:,:,2) = GLCM_features{1,6};
    TainingFeaturesArray(:,:,3) = GLCM_features{1,8};

    %load the predefined mask label
    load('training_mask.mat');
    ClassLabels = training_mask;

end

%% Basic parameters setting

%derive the samples of features(m*n) and feature numbers
[m,n,L] = size(TainingFeaturesArray);

% derive the number of class (exclude 0)
C = length(unique(ClassLabels))-1;

%% sort the features from 3d format to 2d format
feature_mat_training = zeros(L,m*n);
feature_mat_test     = zeros(L,m*n);

for iL = 1:L
    tmp1   = TainingFeaturesArray(:,:,iL);
    tmp2   = TestFeatureArray(:,:,iL);
    feature_mat_training(iL,:) = tmp1(:);
    feature_mat_test(iL,:)     = tmp2(:);
end

%% derive the mean of the input feature array for each classes
Mu_s          = zeros(L,C);
ClassLabels   = ClassLabels(:)';

for iC = 1:C
    %get the features matrix belongs to current class
    tmp_feature   = feature_mat_training(:,ClassLabels==iC);

    %derive the mean of features
```

```
    Mu_s(:,iC) = mean(tmp_feature,2);
end

%% derive the covariance matrix and determinat of feature matrix
cov_mat = cell(1,C);
det_mat = cell(1,C);

for iC = 1:C
    %get the features matrix belongs to current class
    tmp_feature   = feature_mat_training(:,ClassLabels==iC);

    %get the covariance matrix based on features
    cov_mat{iC} = cov(tmp_feature');

    %get the determinant of covariance matrix
    det_mat{iC} = det(cov_mat{iC});
end


%% Apply the classification by multivariate gaussian classifier

%set the output class
class_predict = zeros(m,n);

%apply the classification
%--In this project, we directly use likelyhood to derive the class, since
%the input class has the identical prior probability)--

for im = 1:m
    for in = 1:n

        %get the feature vector
        X    = feature_mat_test(:,(in-1)*m+im);

        %calculate the probability
        Prob = zeros(1,C);
        for iC = 1:C

            if(abs(det_mat{iC}) <= 1e-20) %condition: if the matrix is singular, the determinant
is 0
                %calculate the posterior probability (use pseudo inversion in case the covariance
matrix is singular)
                Prob(iC) = 1/((2*pi)^(L/2)*sqrt(abs(det_mat{iC}))) * exp(-0.5*(X-Mu_s(:,iC)).' *
pinv(cov_mat{iC}) * (X-Mu_s(:,iC)));

            else
                %calculate the posterior probability
                Prob(iC) = 1/((2*pi)^(L/2)*sqrt(abs(det_mat{iC}))) * exp(-0.5*(X-Mu_s(:,iC)).' *
inv(cov_mat{iC}) * (X-Mu_s(:,iC)));
            end

        end

        %derive the class based on maximal probalibity
        class_predict(im,in) = find(Prob==max(Prob));

    end
end

end
```