Classification of Maize Kernels

using multispectral image analysis

by Morten Arngren

on July 28, 2008

1 Introduction

In this report a method for classification of grain is presented and analysed. It is based on multispectral image data and is used to form a probabilistic framework for classifying a small number of maize grain.

The quality of grain can be defined by many different parameters. One of the most distinct degradations is the group of fungal attacks, where small areas of the grain are infected with different types of fungus (black point, pink stain etc.). These types of degradations are very distinct in NIR range and has thus received some attention [**?**].

This analysis is however based on a simpler dataset with no fungal infections on the maize grain samples. The low quality distinction is included as miscoloured maize as shown below in figure 1.1 and also further described in section 3.

Figure 1.1: Pseudo RGB illustration of a healthy grain (*left*) and a miscoloured grain (*right*).

2 Classifying grain

In this analysis the objective is to identify potential low quality grain as outliers in a two-level classification structure. This is achieved by classifying the grain constituents in a first level followed by a simple analysis on the whole grain for outlier detection as depicted in figure 2.1. In this section the overall approach of the classification is described, further details will be described in subsequent sections.

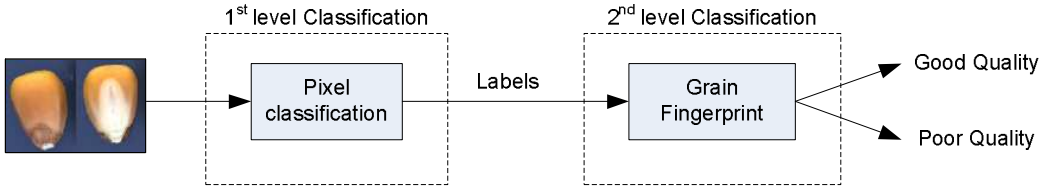

Figure 2.1: Simple overview of the 2 stage approach of classification.

For the initial level of classification we consider N multispectral pixels $\mathbf{S} = \{\mathbf{s}_n\}_{n=1}^{N}$ and want to classify each pixel $\mathbf{s}_n$ into K different classes of constituents $\mathcal{C}$, i.e. model the probability $p(\mathcal{C}|\mathbf{S}, \Theta)$, where Θ denote the model parameters. This posterior probability can also be expressed by *Bayes'* as

$$p(\mathcal{C}|\mathbf{S}, \Theta) = \frac{p(\mathbf{S}|\mathcal{C}, \Theta)p(\mathcal{C})}{\sum_k p(\mathbf{S}|\mathcal{C}, \Theta)p(\mathcal{C})} = \frac{p(\mathbf{S}|\mathcal{C}, \Theta)p(\mathcal{C})}{p(\mathbf{S}|\Theta)} \tag{2.1}$$

It is further assumed that the class prior $p(\mathcal{C})$ is uniform with equal probability where no prior information exists, i.e. $p(\mathcal{C}) = \frac{1}{K}$. This means the conditional probability $p(\mathcal{C}|\mathbf{S}, \Theta)$ is directly proportional to the likelihood $p(\mathbf{S}|\mathcal{C}, \Theta)$.

In this study we model the conditional likelihood $p(\mathbf{S}|\mathcal{C}, \Theta)$ by 2 similar discriminant models

- *Canonical Discriminant Modeling / Analysis (CDA)*
  CDA is a linear discriminant model which maximises the class separation based on between- and within class covariance and is described in section 5.3.
- *Quadratic Discriminant Modeling / Analysis (QDA)*
  QDA is a discriminant model with an individual covariance matrix for each class leading to a quadratic discriminant and is described in section 5.3.

As both CDA and QDA are supervised models the labels for each training sample $\mathcal{C}_n$ must be provided. These labels are inferred from the training set by the unsupervised *K-Means* algorithm prior to the training of the CDA or QDA model. This is discussed in detail in section 5.1.

Multispectral data are typically high dimensional and by classifying into K classes, where $K < \dim(\mathbf{s})$ it can sometimes be convenient to compress the data in order to save computational power. Under the assumption of isotropic noise, where the noise levels at each wavelength are assumed equal, the data $\mathbf{S}$ can be reduced to $K - 1$ dimensions. The compression is discussed further in section 5.2.

In the second level of classification the objective is to classify entire grain kernels based on the previous individual pixel classifications. This is achieved by forming a fingerprint for each maize grain and conducting simple classification using euclidean distance. This is discussed further in section 6.

Prior to any classification the kernels are extracted from the background by simple segmentation based on principal components. Instead of the raw spectra the 2nd order derivatives are extracted as features in order to remove brightness and illumination variations. The segmentation and feature extraction are both described in sections 3.1 and 4 respectively.

This entire structure of the proposed method is illustrated in figure 2.2.

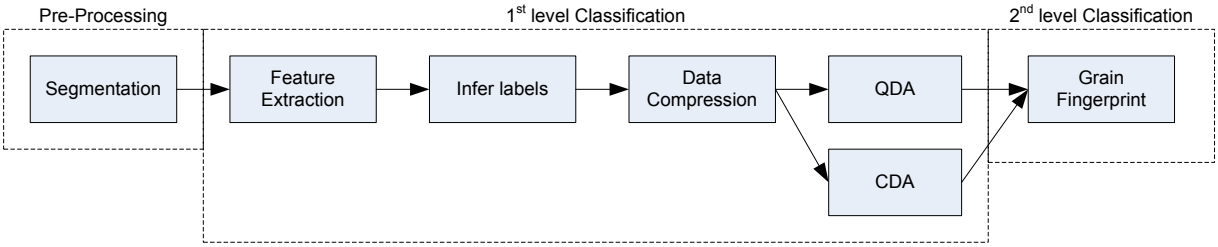

Figure 2.2: Structure overview of the proposed approach showing the processing pipeline.

Some of the steps include supervised modeling and thus requires the dataset to be split up in training- and testset. During inferation of the model parameters the training- and test are subjected to the relevant processing steps. This is described in details in the following sections.

3 Multispectral Image Data Set

The dataset is formed by 8 grain samples in this case maize kernels. A multispectral camera (Videometer system) with $Z = 17$ bands were used to acquire two images, one of each side of the grain samples. These 2 images were each cropped to 600×700 to encapsulate only the maize and afterwards appended to form one single 1200×700 image shown left in figure 3.1.

The dataset is split up in a training set consisting of the top 6 grain samples (top row) and a test set comprising of the front and backside of the lowest 10 samples in pairs (bottom 2 rows), i.e. total of 5 testsamples. A testsample is depicted right in figure 3.1 showing all 17 bands and a Pseudo-RGB image of all 5 test samples are shown in figure 3.2.

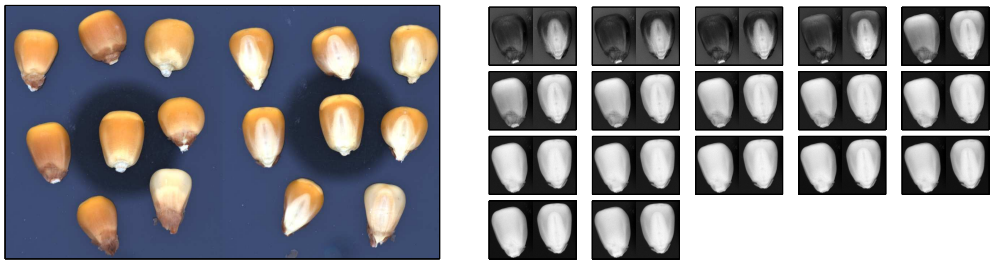

Figure 3.1: Left: The entire dataset in Pseudo-RGB colors. *Right*: All 17 bands for a single test sample.

Each pixel in the image represents a spectrum of 17 bands ranging from 430-970nm. leading to a vectorspace representation of all the pixels in a 17×840000 matrix $\mathbf{S}$ including the background. A typical spectra for 10 random pixels is shown right in figure 3.3.

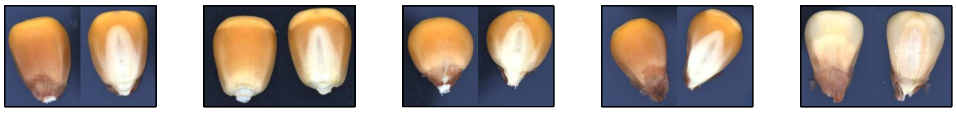

Figure 3.2: Pseudo-RGB image of all test grain samples, front and backside.

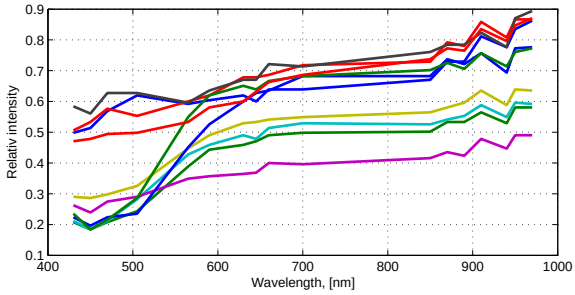

Figure 3.3: Raw spectra from 10 random pixels.

In the left image in figure 3.1 it is clear to identify a grain sample which is more white than the rest. Unfortunately the dataset does not include any fungal infections and for this purpose the white maize sample is hence considered an outlier in the analysis.

3.1 Background Segmentation

Prior to any feature extraction of the grain samples the background is removed by segmentation. Initially this could be achieved by selecting a specific wavelength image and applying a simple threshold to discriminate the background. However finding such discriminative wavelength may be dependent on the dataset and can hardly be automated.

A more robust approach is to infer an image which maximises the difference between the foreground and background. This corresponds to finding the maximum variance of the datapixels assuming an appropriate background has been used and apply a simple threshold. This can be formulated as the eigenvalue decomposition of the covariance matrix of the datapixels Σ expressed as

$$\Sigma = \mathbf{U}\Lambda\mathbf{U}^T \qquad \Leftrightarrow \qquad \Sigma\mathbf{u}_z = \lambda_z\mathbf{u}_z \quad \text{for } z = 1, \ldots, Z \tag{3.1}$$

where $\mathbf{U} = \{\mathbf{u}_z\}_{z=1}^{Z}$ are the orthonormal eigenvectors pointing in the direction of max. variance with $\|\mathbf{u}_z\| = 1$ and $\Lambda = \text{diag}\{\lambda_z\}_{z=1}^{Z}$ are the corresponding eigenvalues. Refer to [1] for a more detailed description of eigenvalue decomposition and PCA.

By projecting all datapixels onto these eigenvectors $\mathbf{U}$ the principal components (PC) of the data can be found. The first PC corresponds to the largest eigenvalue λ_1 and represents the max. variance in the dataset in this case the foreground vs. background. This leads to an efficient segmentation by applying a simple threshold to the 1st PC image.

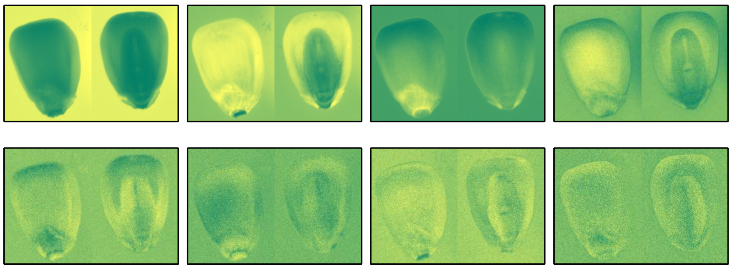

Figure 3.4: Principal component images for the 1st test grain sample in ascending order.

Figure 3.4 illustrates the 8 first principal components for a test grain sample showing how the 1st principal component varies between the foreground and background - suitable for discrimination.

This segmentation approach leads to a training set denoted as the $Z \times N_{train}$ matrix $\mathbf{S}_{train}$ with $N_{train} = 91997$ samples and a test set $\mathbf{S}_{test}$ consisting of the amount of samples as listed in table 3.1.

Sample	1	2	3	4	5
No. of samples	31193	33595	27460	26307	29954

Table 3.1: Number of active pixels in each of the test set samples.

4 Feature Extraction

In this analysis the extracted features must have the property of being being discriminative wrt. the different constituents of the grain. Initially the raw and unprocessed spectra $\mathbf{s}_n$ can be used, but since they also hold brightness information we risk classifying light from dark areas as the periphery of the grain samples.

In order to exclude the brightness information the 2nd order derivative of the spectra $\mathbf{S}$ can be used as this excludes the mean of the spectra and is hence appropriate for classification. This means the 2nd order features $\mathbf{x}_n$ can be expressed as

$$\mathbf{x}_n = \frac{\partial^2 \mathbf{s}_n}{\partial \lambda^2} \qquad \forall\, n \in \{1, \ldots, N\} \tag{4.1}$$

where λ denotes the wavelength of the spectra. This means all the features are now denoted $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^{N}$. To illustrate figure 4.1 shows 10 random raw spectra and the corresponding 2nd order derivatives.

The 2nd order derivates from the figure further indicate that the midrange from ∼600nm to ∼850nm might not hold discriminative information and can possibly be omitted. At this stage in the analysis however no compression of the feature space is performed and the size of the training set matrix and test set matrix thus remain the same.

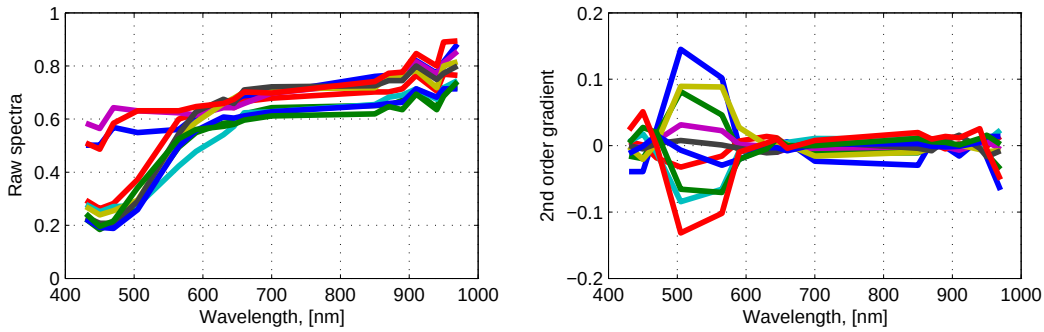

Figure 4.1: Raw spectra from 10 random pixels and the corresponding 2nd order derivatives.

5 Classifying Grain Constituents, 1st level

In the 1st level of classification the different constituents of the grain in the trainingset are classified to model the conditional likelihood $p(\mathbf{X}|\mathcal{C}, \Theta)$, refer to (5.2).

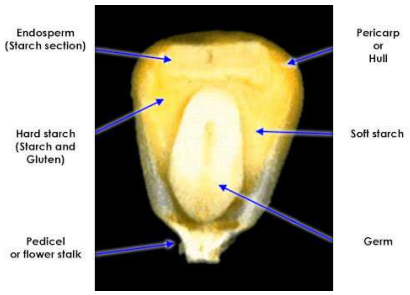

Figure 5.1: Maize kernel constituents.

A maize kernel can be divided into many different compounds on the macroscopic level as illustrated in figure 5.1 [1]. As the number of multispectral bands Z only allows a certain resolution we are limited in detecting similar constituents (e.g. soft and hard starch). We therefore limit our analysis to classifying into $K = 3$ classes: *Germ*, *Starch* and *Endosperm*.

5.1 Unsupervised Modeling, K-Means

In order to employ the supervised models CDA and QDA the trainingset must be provided with a set of labels from each datasample $\mathbf{x}_n$. These labels $l_n \in \{1, \ldots, K\}$ are inferred from the training data by clustering the different constituents using the K-Means algorithm.

The K-Means algorithm is one of the most simple clustering methods and is an exclusive classification, where each sample $\mathbf{x}_n$ is associated to only one cluster by the label l_n based on euclidean distance in feature space. Refer to [1] for a detailed description of the K-Means algorithm.

The performance of the K-Means algorithm is quite sensitive to the amount of clusters K given as a parameter. If K is too low we might obtain an *underfit* and suffer from a bias failing to adapt to the underlying structure of the dataset $\mathbf{X}$. If K is too high we risk achieving an *overfit*, where each cluster has only one association in the extreme case. Figure 5.2 gives an illustration of the two cases.

[1] Image taken from Food and Agriculture Organization of the United Nations (FAO), http://www.fao.org/inpho/content/compend/text/ch23_01.htm.

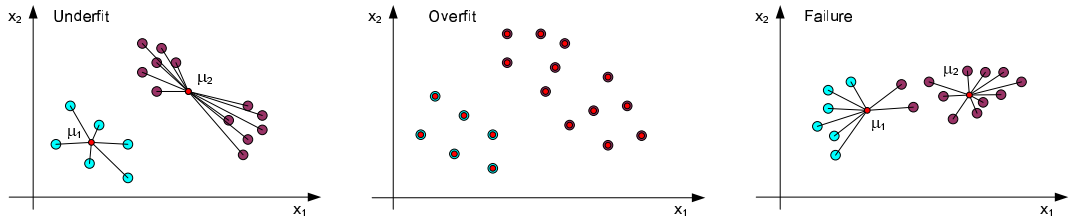

Figure 5.2: K-Means under- and overfit.

In our case with multispectral image data a minority of the pixels may not belong to only one class. Such pixels are particularly evident at transitions between materials in the scene and in systems with low spatial resolution. This means the exclusive clustering using the K-Means is not optimal and may result in suboptimal representation of the inferred model parameters.

As argued earlier the grain constituents can be divided into 3 distinct classes, i.e. $K = 3$. Using the trainingset the K-means clustering is applied with 3 classes and the resulting pixel classifications are depicted in figure 5.3.

Figure 5.3: The clustered constituents of the training maize with $K = 3$ classes.

The illustration shows how the different constituents in the training grain has been identified with reasonable accuracy and how the different regions are similar across the kernels due to the 2nd order derivative features $\mathbf{x}$.

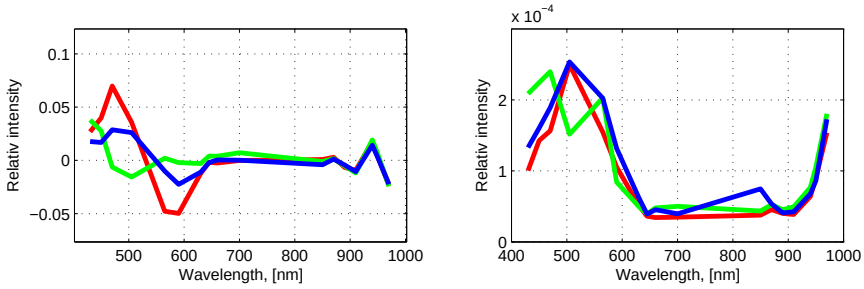

Figure 5.4: The mean spectra and corresponding variances for the 3 classes.

The class means μ_k shown in figure 5.4 reveals how the classes differ in the lower wavelength region from 400nm to 650nm. This corresponds to the red/yellow area in the visible range, as expected.

The K-means algorithm is also sensitive to the initialisation of the cluster means μ_k. In these simulations each cluster center were initialised to any random datasample in the dataset. From several independent simulations this random initialisation showed little influence as we achieved same final cluster centers in every run.

5.2 Compression / Dimension Reduction

The classification of the individual pixels in the training image leads to K cluster centers μ_k. These class means μ_k now span a $Z' = K - 1$ dimensional subspace, i.e. in our case of $K = 3$ classes they form a 2D

plane in 17 dim. space.

Under the assumption of isotropic noise, where all wavelengths have equal noise variance[2] the data can be projected onto this plane while ensuring maximum separation of the different class means μ_k in the smallest subspace possible. This corresponds to projecting the data onto the informative direction based on the label information.

The hyperplane is identified by the Z' eigenvectors estimated from the PCA applied on the K 17 dim. cluster means μ_k. This means from the eigenvalue decomposition given in (3.1) we can express the projection of the datasamples $\mathbf{X}$ onto the 2D subspace as $\mathbf{X}'$ given by

$$\mathbf{X}' = \mathbf{U}_{2D}^T \cdot \mathbf{X} \tag{5.1}$$

where $\mathbf{U}_{2D}$ denote the 2 eigenvectors $\{\mathbf{u}_1, \mathbf{u}_2\}$ with the largest associated eigenvalues $\{\lambda_1, \lambda_2\}$. The projected training and testdata onto the 2 dimension are shown in figure 5.5.

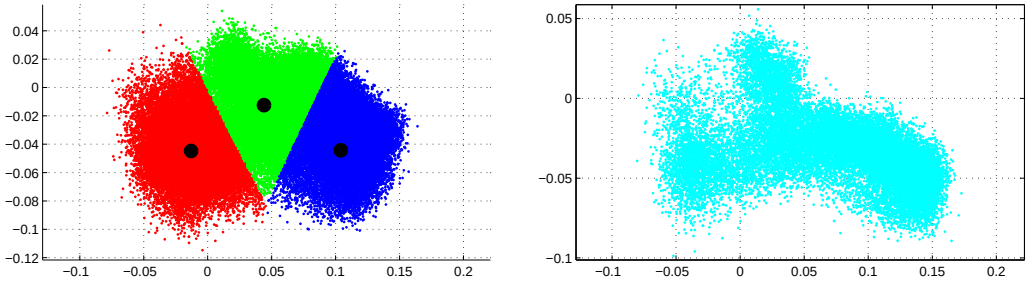

Figure 5.5: The scatterplot of the trainingdata (*left*) and testdata (*right*) of the 3rd grain sample after dimension reduction (only the first 20000 samples from each class are shown).

The new compressed feature set $\mathbf{X}'$ and the corresponding labels l_n achieved by the unsupervised clustering can now be appended and used to infer the model parameters for the two supervised models CDA and QDA.

5.3 Supervised Models

For the classification of the maize pixels the objective is to model the class posterior $p(\mathcal{C}|\mathbf{X}', \Theta)$. Assuming a uniform prior distribution of the classes $\mathcal{C}$, as argued earlier, the posterior becomes proportional to the likelihood, i.e. $p(\mathcal{C}|\mathbf{X}', \Theta) \propto p(\mathbf{X}'|\mathcal{C}, \Theta)$, see section 2.

The parameters of the model Θ needs to be inferred and this is achieved by maximizing the parameter posterior $p(\Theta|\mathbf{X}', \mathcal{C})$ expressed by Bayes as

$$p(\Theta|\mathbf{X}', \mathcal{C}) = \frac{p(\mathbf{X}'|\mathcal{C}, \Theta)p(\Theta)}{p(\mathbf{X}'|\Theta)} \tag{5.2}$$

If we again assume a uniform distribution over the parameters $p(\Theta)$ the posterior becomes proportional to the likelihood as before, i.e. $p(\Theta|\mathbf{X}', \mathcal{C}) \propto p(\mathbf{X}'|\mathcal{C}, \Theta)$. This means we can estimate the parameters Θ via max. likelihood and model the classification using the same distribution $p(\mathbf{X}'|\mathcal{C}, \Theta)$. In addition we are actually employing a generative model to conduct the classification as we model the data $\mathbf{X}'$ via the likelihood $p(\mathbf{X}'|\mathcal{C}, \Theta)$.

The likelihood of the model parameters $p(\mathbf{X}'|\mathcal{C}, \Theta)$ can further be expanded as

[2] Isotropic noise most likely not the case, but are assumed due to simplicity.

$$p(\mathbf{X}'|\mathcal{C},\Theta) = p(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N|\mathcal{C},\Theta) \tag{5.3}$$

As many distributions include exponential terms it is often more convenient to use the logarithm of the likelihood. Assuming i.i.d. (independent and identically distributed) data the likelihood factorise into $p(\mathbf{X}') = \prod_n p(\mathbf{x}'_n)$. The log-likelihood of the model parameters can hence be expressed as

$$\mathcal{L}(\Theta) = \ln p(\mathbf{X}'|\mathcal{C},\Theta) = \sum_{n=1}^{N} \ln p(\mathbf{x}'_n|\mathcal{C},\Theta) \tag{5.4}$$

In this analysis the likelihood function $p(\mathbf{X}'|\mathcal{C}_k,\Theta)$ is modeled by two different models, CDA and QDA. These models are described in the following sections.

5.3.1 Quadratic Discriminant Model / Analysis

The most general of the two is the QDA model where the likelihood $p(\mathbf{x}'_n|\mathcal{C},\Theta_k)$ for each class is represented by a Gaussian distribution with a full covariance matrix Σ_k expressed as

$$p(\mathbf{x}'_n|\mathcal{C},\Theta_k) = \frac{1}{\sqrt{(2\pi)^Z|\Sigma_k|}} \exp\Big(-\tfrac{1}{2}(\mathbf{x}'_n-\mu_k)\Sigma_k^{-1}(\mathbf{x}'_n-\mu_k)^T\Big) \tag{5.5}$$

where $\Theta = \{\mu_k,\Sigma_k\}_{k=1}^K$ for the QDA model. This means the log-likelihood $\mathcal{L}$ can be simplified to

$$\mathcal{L}(\Theta_k) = \sum_{n\in\mathbf{C}_k} -\tfrac{1}{2}(\mathbf{x}'_n-\mu_k)\Sigma_k^{-1}(\mathbf{x}'_n-\mu_k)^T + \mathbf{c}_k \tag{5.6}$$

where $\mathbf{c}_k$ is the normalisation constant and N_k denote the number of samples in class k. To estimate the class means μ_k the log-likelihood is maximised by setting the derivate wrt. μ_k to zero leading to

$$\mu_k = \frac{1}{N_k}\sum_{n\in k}\mathbf{x}'_n \tag{5.7}$$

By a similar approach the covariance Σ_k can be estimated to

$$\Sigma_k = \frac{1}{N_k}\sum_{n\in k}(\mathbf{x}'_n-\mu_k)(\mathbf{x}'_n-\mu_k)^T \tag{5.8}$$

In the QDA model the full covariance matrix Σ_k is estimated, i.e. no constraints. In case of relatively few high dimensional samples the estimation of Σ_k can potentially lead to singularities due to poorly estimated variances in certain directions. This is the curse of dimensionality. Inverting such a singular covariance matrix Σ_k in (5.5) can further lead to numerical instabilities.

In our case however the datasamples are already compressed to 2 dimension in the pre-processing step and hence we do not suffer from the curse of dimensionality. due to the large amount of samples.

In cases where such compression are not feasible other methods can be used to avoid ill-conditioned covariance matrices. A common approach is to enforce a constraint assuming isotropic covariance, where $\Sigma_k = \sigma^2\mathcal{I}$, described in the following section.

5.3.2 Canonical Discriminant Model / Analysis

The CDA model is a variant of the QDA model described above as it also models the likelihood $p(\mathbf{x}'_n|\mathcal{C}, \Theta_k)$, while inducing isotropical covariance for each class k in a sub-dimensional space through linear projection of the data $\mathbf{x}'$ into $\mathbf{x}'' = \mathbf{A}\mathbf{x}'$, where $\mathbf{A}$ is the transformation matrix.

This linear transformation is determined by the informative canonical projections, which maximises the separation between the different classes k, while constraining isotropical covariance for each class, i.e. $\Sigma_k = \mathcal{I}$. This is achieved by maximising the cost function J expressed as the *generalised Rayleigh quotient* or *Fisher's ratio* [4] by

$$J(\mathbf{a}) = \frac{\mathbf{a}^T\mathbf{B}\mathbf{a}}{\mathbf{a}^T\mathbf{W}\mathbf{a}} = \frac{\mathbf{a}^T(\mathbf{W}^{-\frac{1}{2}}\mathbf{B}\mathbf{W}^{-\frac{1}{2}})\mathbf{a}}{\mathbf{a}^T\mathbf{a}} \tag{5.9}$$

where $\mathbf{A} = \{\mathbf{a}_{z'}\}_{z'=1}^{Z'}$ is the set of linear transformation vectors and Z' is dimensionality of the compressed data as defined above. Further $\mathbf{W}$ is the *Within-Class* covariance expressing the covariance of each class separately and $\mathbf{B}$ is the *Between-Class* covariance denoting the variance between the classes. Both matrices are symmetric, where $\mathbf{W}$ is positive definite and $\mathbf{B}$ is positive semidefinite and are given by

$$\mathbf{W} = \frac{1}{K(N_k - 1)} \sum_k \sum_{n \in k} (\mathbf{x}'_n - \mu_k)(\mathbf{x}'_n - \mu_k)^T \qquad \wedge \qquad \mathbf{B} = \frac{1}{N} \sum_k N_k (\mu_k - \mu_{\mathcal{C}})(\mu_k - \mu_{\mathcal{C}})^T \tag{5.10}$$

where $\mu_{\mathcal{C}}$ denote the common mean of all the classes given by

$$\mu_{\mathcal{C}} = \frac{1}{N} \sum_n \mathbf{x}'_n = \frac{1}{N} \sum_k N_k \mu_k \tag{5.11}$$

The cost function in (5.9) induces the isotropical covariance constraint in the transformed space by including the within-class covariance $\mathbf{W}$ term in the denominator. The corresponding transformation vectors in $\mathbf{A}$ point in the canonical directions (hence the name) and are not necessarily orthonormal and thus includes both a scaling and a non-orthogonal transformation. In the transformed space however the orthogonality is still preserved as $\mathbf{A}^T\mathbf{W}\mathbf{A} = \mathcal{I}$, i.e. isotropical covariance. The principle can be illustrated in figure 5.6.

Figure 5.6: Original data (*left*) and the transformed data with isotropical covariance (*right*).

For finding the optimal $\mathbf{A}$ the cost J in (5.9) has the property of being invariant to scalings of $\mathbf{W}$ and we can thus optimise J while constraining the denominator to 1, i.e. $\mathbf{a}^T\mathbf{W}\mathbf{a} = 1$. This means we can formulate the following optimisation problem

$$\min_{\mathbf{a}} -\tfrac{1}{2}\mathbf{a}^T\mathbf{B}\mathbf{a} \qquad \text{s.t.} \qquad \mathbf{a}^T\mathbf{W}\mathbf{a} = 1 \tag{5.12}$$

This leads to the constrained optimisation using Lagrange multipliers

$$\mathcal{L} = -\tfrac{1}{2}\mathbf{a}^T\mathbf{B}\mathbf{a} + \tfrac{1}{2}\lambda(\mathbf{a}^T\mathbf{W}\mathbf{a} - 1) \tag{5.13}$$

Applying the *Karush-Kuhn-Tucker* first order necessary conditions [2], the optimisation can be expressed as

$$\mathbf{B}\mathbf{a} = \lambda\mathbf{W}\mathbf{a} \qquad \Rightarrow \qquad \mathbf{W}^{-1}\mathbf{B}\mathbf{a} = \lambda\mathbf{a} \tag{5.14}$$

This formulation is known as the *Generalized Eigenvalue Decomposition* and can be solved easy. An alternative approach to find the informative projection directions $\mathbf{a}_{z'}$ can also be expressed via Fisher's linear discriminant, see [2] [3] for details.

Finally the data samples can be linearly transformed by projecting them onto the space spanned by $\mathbf{a}_{z'}$, i.e.

$$\mathbf{X}'' = \mathbf{A}^T \cdot \mathbf{X}' \tag{5.15}$$

The resulting covariance of $\mathbf{X}''$ is now $\Sigma_{\mathbf{X}''} = \mathbf{A}^T\mathbf{W}\mathbf{A} = \mathcal{I}$. The constraint of the transformed within-class covariance set to unity, i.e. $\mathbf{a}^T\mathbf{W}\mathbf{a} = 1$ corresponds to $\Sigma_k = \mathcal{I}$ for each class k, as the linear transformation enforces isotropical covariance. This means each class can be represented by a Gaussian distribution with unity noise covariance matrix expressed as

$$p(\mathbf{x}'_n|\mathcal{C}, \Theta_k) = \frac{1}{(2\pi)^{Z/2}} \exp\Big(-\tfrac{1}{2}(\mathbf{x}'_n - \mu_k)^2 \Big) \tag{5.16}$$

where $\Theta = \{\mu_k\}_{k=1}^K$ for the CDA model. The mean μ_k is estimated similarly as before in (5.7).

It should be noted that since the informative projections has already been found in the compression step, the transformation of data is only susceptible to rotation and scaling in the CDA model. The canonical directions found in $\mathbf{A}$ can however be used for further compression of the data to dimension K', by applying the K' first directions sorted by the associated eigenvalues. The parameters for the QDA model $\Theta = \{\mu_k, \Sigma_k\}_{k=1}^K$ and the CDA model $\Theta = \{\mu_k\}_{k=1}^K$ can now be inferred from the trainingset depicted in figure 5.5.

Having the models defined with inferred parameters new pixels can be classified by estimating the appropriate likelihood using (5.5) or (5.16). As we assume uniform priors these likelihood terms are proportional to the posterior probability, as already argued. Applying the QDA and CDA model on the testset we achieve the scatterplot illustrated in figure 5.7.

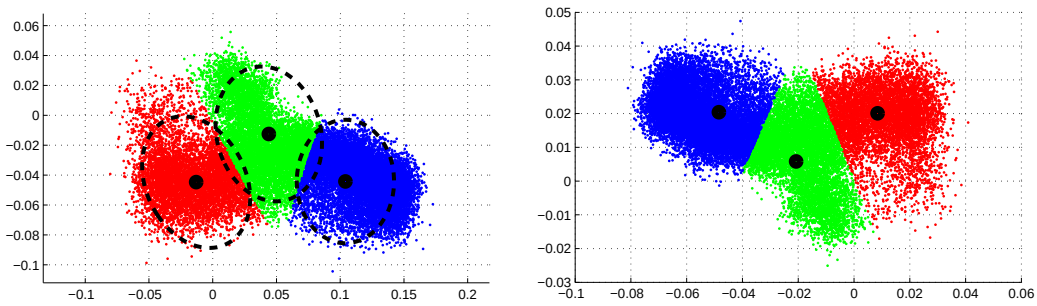

Figure 5.7: The scatterplot of the testdata of the 3rd grain sample after pixel classification using the QDA model (*left*) and the CDA model (*right*).

The left scatterplot using the QDA model reveals how the Gaussian distributions for each class has almost the same covariance shape. This means the quadratic discriminant is almost linear which is also evident

from the plot. The right figure illustrates the scatterplot using the CDA model and shows how the linear transformation has resulted in a $180°$ rotation and a scaling of the datapoints of approx. $\frac{1}{2}$, i.e. in our case the estimated canonical vectors both have the length of appr. $||\mathbf{a}|| \approx 0.5$.

The corresponding pixel classifications for the both model are shown in figure 5.8.

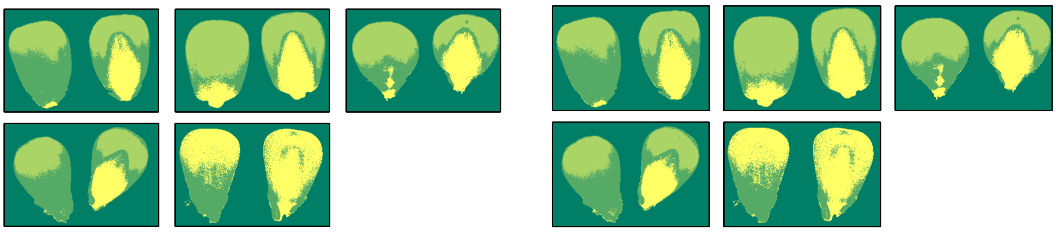

Figure 5.8: The individual test pixels classified using the QDA model (*left*) and the CDA model (*right*).

All the test grain images of both models indicate a successful classification of the pixels, where the overall structure has been captured. By comparison of the models there is no clear difference between the two. A few pixels can however be identified as different, but is without significance. In the further analysis the CDA model is thus used.

6 Classifying Grain Samples, 2nd level

In the 2nd level of classification the objective is to identify potential outlier grain kernels based on a fingerprint for each grain kernel denoted $\mathbf{Y} = \{\mathbf{y}_m\}_{m=1}^{M}$, where M is the amount of grain kernels. The distance between the individual fingerprints can then evaluated to find any outliers.

The fingerprint $\mathbf{Y}$ can be formed in many ways depending on the detection objective (e.g. coloring, fungal infections etc.). In our case there are no fungal infections in the dataset and hence we focus on detecting color differences. Such a fingerprint can be formed for each kernel as the distribution of the pixel class members, i.e. in this case a K dimensional vector $\mathbf{y}_m$ estimated as the normalised histogram of the different labels l_k, for $k = 1, \ldots, K$. For the grain dataset the fingerprint are shown in figure 6.1

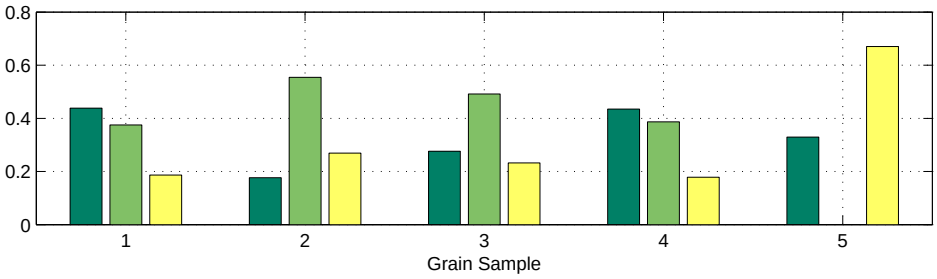

Figure 6.1: The fingerprints of the grain kernels defined as the normalised distribution of the pixel labels l.

The figure clearly reveals how the fingerprints for the first 4 grain kernels are fairly similar in shape, whereas the 5th kernel is completely missing the middle label, as expected.

For outlier detection a mean fingerprint μ_Y can for instance be estimated supervised based on inlier examples and any new fingerprint can then be classified by simple euclidean distance to this mean μ_Y.

7 Conclusion

In this analysis a two stage structure was proposed for classification of maize grain kernels based on multisprectral images analysis.

The initial level classified individual pixels based on a probabilistic framework comparing the two supervised models, QDA and CDA. By a simple clustering of pixels using K-Means and subsequent PCA we have shown how multispectral data can be compressed without loss of classification performance. In addition the 2nd order derivative of the spectra proved successful as extracted features.

In the study two similar models CDA and QDA were described and compared in terms of performance on pixel classification. The result revealed similar properties of the 2 models.

In a second stage a grain fingerprint was formed based on simple histogram distributions to further classify whole maize grain kernels. Due to the limitations of the dataset only a simple fingerprint was generated based on simple histogram statistics.

The two stage classification structure proposed proved successful in classifying maize kernels based on multispectral visible color information.

7.1 Limitations & Further Research

First of all the entire analysis is only limited to the best capabilities of the dataset. Our case with only 8 grain kernels is not the most representative dataset for all the types, as there are many varieties of maize and in particular grains with any fungal infections are not represented.

In case of fungal attacks in the dataset the unsupervised classification on the trainingset using K-Means would then be expanded to include these extra classes, i.e. $K = 4$ or 5.

The fingerprints $\mathbf{Y}$ presented include no geometrical information about the grain kernels and is hence susceptible to deform variations of the kernel shape. In addition one of the disadvantages of using the distribution of the labels l as a fingerprint is this method will discard any abnormalities in small areas.

References

[1] Morten Arngren. Modelling cognitive representations. Master's thesis, DTU Informatics, Intelligent Signal Processing, April 2007.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Verlag, 2006.

[3] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer Verlag, 2003.

[4] Max Welling. Fisher linear discriminant analysis. Technical report, Department of Computer Science, University of Toronto.

Appendix A: MATLAB code.

MATLAB code - Classification of Grain.

```
%% Classification of Maize kernels from Multispectral Image Analysis

clear;
path(path, 'Toolbox/MSItools');

% Model Parameters
K          = 3;             % Amount of clusters
M          = 5;             % Amount of test grains
segm_thrs  = 0/255;         % Threshold for background segmentation.
color      = ['rgbcmyk'];   % Colorcodes for plotting different classes.

%% Load data and initialise

% Load image and reduce image dimensions, X & Y
load 'Hyperspectral/Majs_3.mat'; X=[301,900]; Y=[161,860];
HIM = im(Y(1):Y(2),X(1):X(2),:) / 255;
load 'Hyperspectral/Majs_4.mat'; X=[301,900]; Y=[201,900];
HIM = [HIM im(Y(1):Y(2),X(1):X(2),:) / 255];

% Swap wavelengths
wv2 = [ wv(1:4); wv(14); wv(5:12); wv(15:17); wv(13) ];
wv = wv2; clear wv2;
im             = HIM(:,:,1:4);
im(:,:,5)      = HIM(:,:,14);
im(:,:,6:13)   = HIM(:,:,5:12);
im(:,:,14:16)  = HIM(:,:,15:17);
im(:,:,17)     = HIM(:,:,13);
HIM = im; clear im;

[Y X Z] = size(HIM);

%%% Plot
figure(1), plotRGB(HIM, wv);
figure(2), ShowAllBands(HIM, wv,5);

%% Segment
```

```
% Extract Principal Components, PCA
S_    = reshape(HIM, [X*Y Z])';
S_    = S_ - mean(S_,2)*ones(1,size(S_,2));
S_S_  = (S_*S_') / size(S_,2);
[E D] = svd(S_S_);
SPca  = E'*S_;
clear S_S_;

% Reformat to image
for z = 1:Z
  SPcaIm(:,:,z) = reshape( E(:,z)'*S_ , [Y X] );
end

%%% Segment into train and test set
XYTest = [ [ 51 200; 301 500;  641  790; 271 470]'; ...
           [241 390; 271 470;  851 1000; 241 440]'; ...
           [411 560; 231 430; 1021 1170; 271 470]'; ...
           [181 330; 501 700;  751  900; 471 670]'; ...
           [391 540; 441 640;  971 1120; 481 680]'; ];

for i = 1:M
  imTest{i} = [ HIM(XYTest(2*i-1,2):XYTest(2*i,2),XYTest(2*i-1,1):XYTest(2*i,1),:) ...
                HIM(XYTest(2*i-1,4):XYTest(2*i,4),XYTest(2*i-1,3):XYTest(2*i,3),:) ];
  tmp       = [ SPcaIm(XYTest(2*i-1,2):XYTest(2*i,2),XYTest(2*i-1,1):XYTest(2*i,1),:) ...
                SPcaIm(XYTest(2*i-1,4):XYTest(2*i,4),XYTest(2*i-1,3):XYTest(2*i,3),:) ];
  idx2DTest{i} = segmHIM(tmp, 1, segm_thrs, 1, 0);
end

idx2DTrain  = segmHIM(SPcaIm(1:240,:,:), 1, segm_thrs, 1, 0);
nTrainTotal = size(idx2DTrain,1);

% Extract spectras from active pixels only
for z = 1:size(HIM,3)
  tmp         = HIM(1:240,:,z);
  STrain(z,:) = tmp(idx2DTrain);
  for i = 1:M
    tmp           = imTest{i}(:,:,z);
    STest{i}(z,:) = tmp(idx2DTest{i});
  end
end
clear S_ tmp;

[YTrain XTrain] = size(SPcaIm(1:240,:,1));
[YTest  XTest ] = size(imTest{1}(:,:,1));

%%% Plot
figure(20), ShowAllBands(HIM(), wv, 5); colormap summer;
figure(21), ShowAllBands(SPcaIm(:,:,1:8), wv(1:8), 4); colormap summer;
figure(22), plotScatter(SPca(:,ceil(size(SPca,2)*rand(10000,1))), 6);

figure(25), for m = 1:M
              subplot(150+m), plotRGB(imTest{m}, wv);
            end

%% UNSUPERVISED LEARNING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Extract features
```

```
% Mean, Var., Kurtosis, Derivatives, Fractiles ...

% 2nd order derivative, numerical.
SFeatTrain = gradient(gradient(STrain'))';

for i = 1:M
  SFeatTest{i} = gradient(gradient(STest{i}'))';
end
Z = size(SFeatTrain,1);

%%% Segment training data using K-Means
[idxKM CC] = kmeans(SFeatTrain', K, 'MaxIter', 200);

% Extract ID's and samples from each class k
for k = 1:K
  idxSCTrain{k} = find(idxKM==k);        % Index for idx
  setTrain{k}   = SFeatTrain(:,idxSCTrain{k});
  nTrain(k)     = length(setTrain{k});
end
for i = 1:M
  setTest{i} = SFeatTest{i};
  nTest{i}   = length(setTest{i});
end
clear SFeatTest;

%%%%% Plot
  % Plot random spectra
  tmp = ceil(90000*rand(1,10));
  figure(40), subplot(121), plot(wv, STrain(:,tmp), 'LineWidth', 3);
              xlabel('Wavelength,␣[nm]'); ylabel('Raw␣spectra'); grid;
              subplot(122), plot(wv, gradient( gradient(STrain(:,tmp)' )'), ...
                  'LineWidth', 3),
              xlabel('Wavelength,␣[nm]'); ylabel('2nd␣order␣gradient'); grid;

  figure(41), plot(wv,var(STrain'), 'r', 'LineWidth', 2);
              xlabel('Wavelength,␣[nm]'); ylabel('Variance'); grid;

  % Plot, spectra
  figure(42), clf, hold on;
              for k = 1:K
                subplot(121), plot(wv, mean(setTrain{k},2), ...
                              ['-' num2str(color(k))], 'LineWidth', 3); hold on;
              end
              grid;
              xlabel('Wavelength,␣[nm]'); ylabel('Relativ␣intensity');
              for k = 1:K
                subplot(122), plot(wv, var(setTrain{k}'), ...
                              ['-' num2str(color(k))], 'LineWidth', 3), hold on;
              end
              grid;
              xlabel('Wavelength,␣[nm]'); ylabel('Relativ␣intensity');

%% SUPERVISED LEARNING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Reduce dim. of data to K-1 dim.
% Means of classes in Z dim.
for k = 1:K
```

```
  muTrain(:,k) = mean(setTrain{k},2);
end

% Between-Class covariance in full dim.
S_      = muTrain - sum( (diag(nTrain./nTrainTotal)*muTrain')' ,2) * ones(1,K);
Sigma2B = S_*S_' / size(S_,2);

% Project setTrain onto subspace based on PCA of means
[E_ D] = svd(Sigma2B);
clear S_ S_S_;
for k = 1:K
  setTrain_{k} = E_(:,1:K-1)' * setTrain{k};
end
muTrain_ = E_(:,1:K-1)' * muTrain;
for i = 1:M
  setTest_{i} = E_(:,1:K-1)' * setTest{i};
end

% Plot
figure(50), clf; hold on;
             for k = 1:K
               plot(setTrain_{k}(1,1:1:20000), setTrain_{k}(2,1:1:20000), ...
                    ['.' num2str(color(k))]);
               plot(muTrain_(1,k), muTrain_(2,k), 'ok', 'LineWidth', 5);
             end
             hold off; grid; title('Projected␣K-1␣traindata␣space');
figure(51), clf;
            plot(setTest_{k}(1,1:1:20000), setTest_{k}(2,1:1:20000),'.c');
            grid; title('Projected␣K-1␣testdata␣space');


%%% CANONICAL DISCRIMININANT ANALYSIS
% Within-Class covariance, average of all K classes in K-1 dim.
Sigma2W = zeros(K-1,K-1);
for k = 1:K
  S_      = setTrain_{k} - muTrain_(:,k)*ones(1,nTrain(k));
  Sigma2W = Sigma2W + S_*S_' / K * (nTrain(k)/nTrainTotal);
end
% Between-Class covariance in K-1 dim.
S_      = muTrain_ - sum( (diag(nTrain./nTrainTotal)*muTrain_')' ,2) * ones(1,K);
Sigma2B = S_*S_' / size(S_,2);

[E D] = eigs(Sigma2B, Sigma2W);

% Project testset onto subspace
muTrain__ = E' *  muTrain_;
for i = 1:M
  setTest__{i} = E' *  setTest_{i};
end

% Euclidian distance to centers
for i = 1:M
  for k = 1:K
    dist_(k,:) = sum( (setTest__{i} - muTrain__(:,k)*ones(1,nTest{i})).^2 );
  end
  p_{i} = dist_ ./ (ones(K,1)*sum(dist_));        % Distributions
  clear dist_;
end
```

```
for i = 1:M
  [a C_CDA{i}] = min(p_{i});
  for k = 1:K
    setTestC_CDA{k} = setTest__{i}(:,find(C_CDA{i}==k)); % FIX FOR all grain
  end
end

%%% PLOT, CDA
figure(55), clf; hold on;
             for k = 1:K
               plot(setTestC_CDA{k}(1,:), setTestC_CDA{k}(2,:), ...
                   ['.' num2str(color(k))]);
             end
             for k = 1:K
               plot(muTrain__(1,k), muTrain__(2,k), 'ok', 'LineWidth', 5);
             end
             hold off; grid; title('CDA, Classified testdata');


%%% QUADRATIC DISCRIMINANT ANALYSIS
% Train, estimate mean & full covariance in N-1 dim.
for k = 1:K
  S_             = setTrain_{k} - muTrain_(:,k)*ones(1,nTrain(k));
  Sigma2Full{k} = S_*S_'/ (size(S_,2)-1);
end

%%% Classification, QDA
for i = 1:M
  for k = 1:K
    % FULL, logp
    i_sigma2          = inv( Sigma2Full{k} );
    Gauss_norm        = - 0.5*( (K-1)*log(2*pi) + logdet(Sigma2Full{k}) );
    dist              = setTest_{i} - muTrain_(:,k)*ones(1,nTest{i});
    logpFull{i}(k,:) = Gauss_norm - 0.5 * (sum( dist.*(i_sigma2*dist), 1 ));
  end
end
clear dist;

% Find max. likelihood
for i = 1:M
  [a C_QDA{i}] = max(logpFull{i});
  for k = 1:K
    setTestC_QDA{k} = setTest_{i}(:,find(C_QDA{i}==k));
  end
end


% Plot training set
figure(60), clf; hold on;
            for k = 1:K
              plot(setTrain_{k}(1,1:10000),setTrain_{k}(2,1:10000), ...
                  ['.' num2str(color(k))]);
              plot(muTrain_(1,k), muTrain_(2,k), 'ok', 'LineWidth', 5);
            end
            for k = 1:K
              circle = 2*[cos(2*pi*(1:101)/100); sin(2*pi*(1:101)/100)];
              [E, D] = eig(Sigma2Full{k});
```

```
              x      = E(:,1:2)*diag(sqrt(diag(D(1:2,1:2)))) * circle;
              plot(x(1,:)+muTrain_(1,k), x(2,:)+muTrain_(2,k), ...
                  '--k', 'LineWidth', 2);
            end
            hold off, grid;
            title('Training data, scatterplot');


figure(61), clf; hold on;
            for k = 1:K
              plot(setTestC_QDA{k}(1, 1:end ), setTestC_QDA{k}(2, 1:end ), ...
                   ['.' num2str(color(k))]);
              plot(muTrain_(1,k),muTrain_(2,k), 'ok', 'LineWidth', 5)
            end
            for k = 1:K
              circle = 2*[cos(2*pi*(1:101)/100); sin(2*pi*(1:101)/100)];
              [E, D] = eig(Sigma2Full{k});
              x      = E(:,1:2)*diag(sqrt(diag(D(1:2,1:2)))) * circle;
              plot(x(1,:)+muTrain_(1,k), x(2,:)+muTrain_(2,k),  ...
                  '--k', 'LineWidth', 3);
            end
            hold off, grid; % axis equal;
            title('Test data, scatterplot');


%% GRAIN CLASSIFICATION

%%% Grain fingerprint
% Trainset
FPTrain = nTrain;

% Testset
for i = 1:M
  FPTest_CDA(:,i) = hist(C_CDA{i}, K) / nTest{i};
  FPTest_QDA(:,i) = hist(C_QDA{i}, K) / nTest{i};
end

%%% Dist to mean
for i = 1:M
  FP_Dist_CDA(i,:) = sqrt( sum( (FPTest_CDA(:,i) - FPTrain').^2 ) );
  FP_Dist_QDA(i,:) = sqrt( sum( (FPTest_QDA(:,i) - FPTrain').^2 ) );
end

% Plot histograms
figure(70), bar(FPTest_CDA'); colormap copper; xlabel('Grain Sample'); grid;



%% PLOT

IM = ones([YTrain XTrain]);
for k = 1:K
  IM( idx2DTrain(find(idxKM==k)) ) = k+1;        % K-Means
end
figure(200), imagesc(IM); colormap summer; title('K-Means, Train'); axis off;
clear IM;
```

```
% Plot images
for i = 1:M
  tmp_CDA = ones([YTest XTest]); tmp_QDA = ones([YTest XTest]);
  for k = 1:K
    tmp_CDA( idx2DTest{i}(find(C_CDA{i}==k)) ) = k+1;
    tmp_QDA( idx2DTest{i}(find(C_QDA{i}==k)) ) = k+1;
  end
  IM_CDA(:,:,i) = tmp_CDA;
  IM_QDA(:,:,i) = tmp_QDA;
end
figure(203), ShowAllBands(IM_CDA, [1:M], 3); colormap summer;
figure(204), ShowAllBands(IM_QDA, [1:M], 3); colormap summer;
```