CHAPTER 6

FEATURE EXTRACTION AND DEFECT DETECTION

Defect identification and classification from fruit images is becoming increasingly significant in a variety of applications since quality control plays a prominent role in the market. All the selected classifiers use a two-step process during classification, as listed below.

(i) Feature Extraction

(ii) Defect Classification

Feature extraction is a process, which transforms the large input data into a form that reduces redundancy and that allows comparisons between mango fruit images by extracting unique properties. This reduced representation is termed as “feature vector”. If the features extracted are carefully chosen, only relevant information from the input data is extracted. This reduced representation of data can be then used to perform the desired task instead of the large sized input data

6.1. FEATURE EXTRACTION

The first step during defect detection is feature extraction (Phase III). Feature extraction has long been an important topic in image processing and pattern recognition domain and has been studied by many authors. Feature extraction can be viewed as finding a set of vectors which effectively represent the information content of a mango fruit while reducing the dimensionality. It is desirable to extract features that have the ability to discriminate between predefined categories that the classifier groups data into. Although numerous feature extraction algorithms have been proposed and successfully applied, it is application dependent and the analysis of which features works best for a particular application is still a challenging question.

In Phase III of the research work, two important features which improve the performance of AESDDSM are selected. Two types of feature vectors were generated

(Table 6.1). These two features were used because of the advantages that can be obtained during defect detection and classification.

Recently, the usage of multiple features for classification has gained wide attention (Ha *et al.*, 2008, Vadivel *et al.*, 2009). According to Deselaers *et al.* (2008), the classification performance can be enhanced by employing multiple features, as each feature extracted characterizes certain aspect of image content and using more than one feature can provide a more adequate description of the image. This research work, in continuation with the interest on using multiple features, also considers merging or fusing the two feature vectors constructed, to further improve the feature set created and at the same also improves the overall process of defect detection and classification.

TABLE 6.1

DETAILS OF FEATURES EXTRACTED

Color Feature Vector
• Color histogram
Texture Feature Vector

• **GLCM-based Features**
  • First Order Features (4)
  ○ Mean
  ○ Standard Deviation
  ○ Energy
  ○ Entropy
  • Second Order Features (4)
  ○ Homogeneity

- Correlation
- Contrast
- Coarseness

6.1.1. Color Feature Extraction

Color is a popular choice of feature as it is readily available with no extra processing. In this research work, a color histogram is constructed and used as a feature vector for detecting and classifying defects in mango fruit. Image histogram, with its wide application in image analysis, is a graphical representation showing a visual impression of the distribution of image data. It plots the number of pixels for each tonal value. A color histogram is a representation of the distribution of colors in an image. For digital images, a color histogram represents the number of pixels that have colors in each of a fixed list of color ranges that span the image's color space, the set of all possible colors.

The color histogram can be built for any kind of color space, although the term is more often used for three-dimensional spaces like RGB or HSV. In this research work, the RGB color space model is used. This color space specifies colors via three numbers, each representing the three color components, red, green and blue. An RGB (Red, Green, Blue) color space is any additive color space based on the RGB color model. In the RGB color model, red, green, and blue light are added together in various ways to reproduce a broad array of colors. RGB is a convenient color model for computer graphics because the human visual system works in such a way that is similar to an RGB color space. The color histogram using RGB color space model is constructed using the procedure given in Figure 6.1.

6.1.2. Texture Feature Extraction

Texture feature is a set of metrics calculated to provide information about the spatial arrangement of color or intensities in the selected region of the mango fruit

(Shapiro and Stockman, 2001). Texture features can be extracted using either statistical approaches or transformation approaches.

Haralick *et al.* (1973) presented one of the most widely used statistical approaches to texture analysis, namely, the spatial gray level dependence matrix (SGLDM) or the gray level co-occurrence matrix (GLCM) approach. They listed 14 texture features that utilize the spatial relationship amongst gray level values of pixels with in a region. A comprehensive review on statistical algorithms is provided by Haralick (1979) and Gool *et al.* (1985). The GLCM is a tabulation of how often different combinations of pixel brightness values (grey levels) occur in the mango fruit region.

Statistical methods analyze the spatial distribution of gray values, by computing local features at each point in the image, and deriving a set of statistics from the distributions of the local features. Depending on the number of pixels defining the local feature, the statistical methods can be classified into first-order (one pixel), second-order (two pixels) and higher-order (three or more pixels) statistics.

Step 1 : Measure of pixels : An image I is a set of pixels and at each pixel, measure some m-dimensional property. For example, each pixel of an RGB image is a 3-dimensional vector, which can be formally written as

$$f_I : R \subset \Re^2 \rightarrow M \subset \Re^m \tag{6.1}$$

Step 2 : Create finite partition of M : The finite partition of M can be created using Equation (6.2)

$$M = \bigcup_{k=1}^{K} B_k \tag{6.2}$$

where B_k are subsets of M and are called bins and k is the label of the bin. The number bins used during experimentation was nine.

Step 3 : Construct histogram from bins: The histogram is then be constructed from the bins using the indicator function, $b_k(x)$, (Equation 6.3).

$$b_k(x) = \begin{cases} 1 & \text{if } f_I(x) \in B_k \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

where x is the element of the image. The indicator function converts the image into a histogram vector,

$$\vec{H} = (h_1,...,h_k) \text{ with } h_k = \frac{\int_{x \in R} b_k(f_I(x))dx}{\int_{x \in R} 1dx} \tag{6.4}$$

Figure 6.1 : Procedure to Construct Color Histogram

The basic difference is that first-order statistics estimate properties (e.g. average and variance) of individual pixel values, ignoring the spatial interaction between image pixels, whereas second- and higher-order statistics estimate properties of two or more pixel values occurring at specific locations relative to each other.

The four first order statistical features, namely, mean, standard deviation, energy and entropy, are defined in Equations (6.5) to (6.8).

$$\text{Mean}(\mu) = \sum_{i=0}^{G-1} ip(i) \tag{6.5}$$

$$\text{Standard Deviation}(\sigma) = \sqrt{\sum_{i=0}^{G-1} (i-\mu)^2 p(i)} \tag{6.6}$$

$$\text{Energy(E)} = \sum_{i=0}^{G-1} [p(i)^2] \tag{6.7}$$

$$\text{Entropy(H)} = \sum_{i=0}^{G-1} p(i) \log_2 [p(i)] \tag{6.8}$$

The GLCM uses co-occurrence matrix to extract texture features using statistical equations. A co-occurrence matrix is a matrix or distribution that is defined over an image to be the distribution of co-occurring values at a given offset. GLCM features are extracted for 'n×n' primitive template matrix in the directions 0°, 45°, 90°, and 135°, and then averaging is done to make them direction invariant. In the present application, GLCM features have been calculated at distances 1 and 2 respectively. The selected second order GLCM features, namely, homogeneity and correlation are estimated using Equations (6.9) and (6.10).

$$\text{Homogeneity (HO)} = \sum_{i,j=0}^{N-1} \frac{P_{i,j}}{1+(i-j)^2} \tag{6.9}$$

$$\text{Correlation (CO)} = \frac{\sum_i \sum_j (i,j)P_{ij} - \mu^2}{\sigma^2} \qquad (6.10)$$

Coarseness has a direct relationship to scale and repetition rates and an image will contain textures at several scales. Coarseness aims to identify the largest size at which a texture exists, even where a smaller micro texture exists. Computationally, the averages at every point over neighbourhoods the linear size of which are powers of two are taken first. The average over the neighbourhood of size 2^k x 2^k at the point (x, y) is given using Equation (6.11).

$$A_k(x,y) = \sum_{i=x-2^{k-1}}^{x+2^{k-1}-1} \sum_{j=y-2^{k-1}}^{y+2^{k-1}-1} \frac{f(i,j)}{2^{2k}} \qquad (6.11)$$

Then at each point the difference between pairs of averages corresponding to non-overlapping neighbourhoods on opposite sides of the point in both horizontal and vertical orientations is estimated. In the horizontal case this is estimated using Equation (6.12).

$$E_{k,h}(x, y) = |A_k(x+2^{k-1}, y) - A_k(x-2^{k-1}, y)| \qquad (6.12)$$

At each point, the best size which gives the highest output value, where k maximizes E in either direction is selected. The coarseness measure (CM) is then the average of $S_{best}(x, y) = 2^k_{best}$ over the picture.

Contrast aims to capture the dynamic range of grey levels in an image, together with the polarization of the distribution of black and white. The first is measured using the standard deviation of grey levels and the second kurtosis α_4. The contrast measure is therefore defined as in Equation (6.13).

$$F_{con} = \sigma^2 / (\alpha_4)^n \qquad \text{where } \alpha_4 = \mu_4/\sigma^4 \qquad (6.13)$$

Here, α_4 is the fourth moment about the mean and σ^2 is the variance. Experimentally, it was found that n = ¼ (0.25) gives the closest agreement to human measurements and provides maximum discrimination between textures. Hence this value is used during performance evaluation.

6.2. DEFECT DETECTION

As mentioned previously, in Phase IV an enhanced RVM classifier is proposed and is analyzed with SVM

6.2.1. Support Vector Machine (SVM)

Support Vector Machines (SVMs) were introduced by Vapnik (1995) (Basu *et al.*, 2002) and have proved to be fast effective classifiers and works effectively with high dimensional datasets also (Eirinaki, 2009). A Support Vector Machine (SVM) is a concept in computer science for a set of related supervised learning methods that analyze data and recognize patterns that are mainly used for classification and regression analysis. The standard SVM takes a set of input data and predicts, for each given input, which of two possible classes the input is a member of, which makes the SVM a non-probabilistic binary linear classifier.

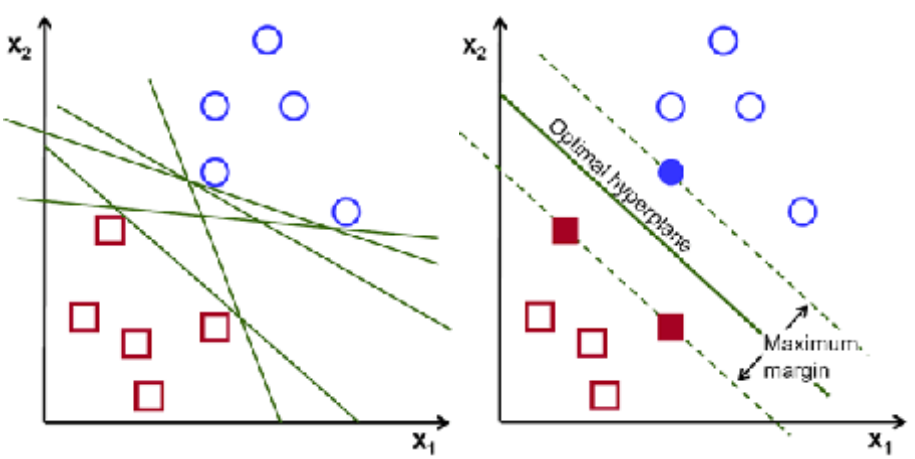
Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

Although SVMs were originally designed as binary classifiers, approaches that address a multi-class problem as a single “all-together” optimization problem exist (Weston and Watkins, 1999). A multi-class classification task usually involves separating data into training and testing sets. Each instance in the training set contains one ‘target

value" (i.e. class labels) and several “attributes" (i.e. features). The goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data given only the test data attributes. Mathematically SOM can be described as follows.

Considering the binary classification case, let $((x_1, y_1) \ldots (x_n, y_n))$ be the training dataset where xi are the feature vectors that represent the observations and $y_i \in (-1, +1)$ be the two labels that each observation can be assigned to. From these observations, SVM builds an optimum hyperplane (a linear discriminant in the kernel transformed higher dimensional feature space) that maximally separates the two classes by the widest margin by minimizing the objective function. For a linearly separable set of 2D-points which belong to one of two classes, find a separating straight line is shown in 6.2a. In this example, there exist multiple straight lines that separate the data points into two groups. Deciding the optimal divider is an intuitive criterion.

In general, a line is considered bad if it passes too close to the points because it will be noise sensitive and it will not generalize correctly. Therefore, the goal here is to find the line passing as far as possible from all points. Thus, the operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance to the training examples. Twice, this distance receives the important name of **margin** within SVM’s theory. Therefore, the optimal separating hyperplane *maximizes* the margin of the training data. Example of an optimal hyperplane is shown in Figure 6.2b.



(a) **(b)**

Figure 6.2 : Support Vector Machine Hyperplane

- **Hyperplane Computation**

Let the hyperplane be defined as

$$f(x) = \beta_0 + \beta^T x \tag{6.18}$$

where β is known as the weight factor and β_0 is the bias. The optimal hyperplane is represented in an infinite number of different ways by scaling of β and β_0.

As a matter of convention, among all the possible representations of the hyperplane, the one chosen is

$$| \beta_0 + \beta^T x| = 1 \tag{6.19}$$

where x symbolizes the training examples closest to the hyperplane. In general, the training examples that are closest to the hyperplane are called **support vectors** and this representation is known as the **canonical hyperplane**. Now, the result of geometry that gives the distance between a point x and a hyperplane $\{\beta, \beta_0\}$ is estimated using Equation (6.20).

$$\text{distance} = \frac{|\beta_0 + \beta^T x|}{||\beta||} \tag{6.20}$$

In particular, for the canonical hyperplane, the numerator is equal to one and the distance to the support vectors is Equation (6.21).

$$\text{distance}_{\text{support\_vectors}} = \frac{|\beta_0 + \beta^T x|}{||\beta||} = \frac{1}{||\beta||} \tag{6.21}$$

Let M denote the margin which is twice the distance to the closest examples (Equation 6.22).

$$M = \frac{2}{||\beta||} \tag{6.22}$$

Now, the problem of maximizing M is equivalent to the problem of minimizing a function L(β) subject to some constraints. The constraints model the requirement for the hyperplane to classify correctly all the training examples x_i Formally, it can be defined as Equation (6.23).

$$\min_{\beta,\beta_0} L(\beta) = \frac{1}{2}||\beta||^2 \qquad \text{subject to } y_i(\beta^T x_i + \beta_0) \geq 1 \ \forall i \tag{6.23}$$

where y_i represents each of the labels of the training examples.