CHAPTER 8
SOFTWARE IMPLEMENTATION

8.1 INTRODUCTION

This chapter discusses the implementation details of the echocardiographic image analysis in terms of class diagrams for various modules of the general block diagram shown in Chapter 1. A comprehensive customized framework has been developed using Microsoft Visual Studio C#.NET 2005/2010 on Windows platform. Several additional tools have been integrated to make the implementation easy and to reduce the development time. The design is based on the object oriented approach, making the implementation more readable, easily understandable, scalable, modifiable, etc.

In general the implementation is based on the top down approach and consists of four steps: segmentation, boundary extraction, CBIR, and classification. The class diagram will show the details of implementation of these steps and relevant methods that accomplishes the intended task. Some of the supporting tools are open source products making the implementation convenient. Since database and data mining are important components of the research, DBMS - Oracle 10g - is used. All queries are written using SQL *PLUS in a standard database environment.

8.1.1 SOFTWARE TOOLS USED FOR THE DESIGN

A number of software tools, computer languages, middleware, databases, etc, have been used throughout the implementation.

- AForge .NET Image processing toolbox
- Oracle 10g DBMS
- Oracle Data Access Components (ODAC) for Windows [Oracle10g, 2010]
- MATLAB with .NET compiler builder for Windows
Out of these AForge.NET is an open source tool helpful for the current research domains. AForge.NET is a C# framework [AForge, 2010] designed for developers and researchers in the fields of Computer Vision and Artificial Intelligence - image processing, neural networks, genetic algorithms, machine learning, robotics, etc.

The framework comprises of a set of libraries and sample applications, which demonstrate their features:

- AForge.Imaging - library with image processing routines and filters;
- AForge.Vision - computer vision library;
- AForge.Video - set of libraries for video processing;
- AForge.Neuro - neural networks computation library;
- AForge.Genetic - evolution programming library;
- AForge.Fuzzy - fuzzy computations library;
- AForge.Robotics - library providing support of some robotics kits;
- AForge.MachineLearning - machine learning library, etc.

The work on the framework’s improvement is in constant progress. The advantage is that the classes provided by AForge.NET framework can directly be used as an API in C# programs, thus making the development faster and easier.

### 8.1.2 THE MAIN MODULE

Since the user must get an interactive usage of the echo analysis software, a professional UI look and feel type of application has been designed on Windows platform. To develop applications with advanced features, C#.NET supports form design which is used in the current implementation of the software [Andrew Trolsen, 2007] [John Sharp, 2002] [Karli, 2006].

The main module comprises of one major class called CBIR that includes several member data and methods. Users can open a query echo image (i.e. input image) and apply various analysis methodologies and the output will be displayed in the form of images and/or numeric values. For CBIR the image database path can be specified which contains appropriate echo images. Similarly, users can segment the input image, trace the boundaries of LV and other cardiac chambers, find the quantitative parameters, analyze the color Doppler images by extracting the color segment, etc. Figure 8.1 depicts the main form that contains all these features in terms of menu items.
Fig. 8.1 Main Menu showing major image analysis components

The main menu consists of the following items: Clustering, 2D Echo Quantification, Color Doppler, Classification, and Preprocessing.

In Clustering, K-Means algorithm options for 2D echo and color Doppler images are made available. Using 2D Echo Quantification, one can trace the boundary of heart cavities either manually or automatically.

Fig. 8.2 Submenu items of Clustering

Since the analysis varies with respect to 2D echo, all relevant methods are given here. The naïve-Bayesian classification of echo images can be performed either using conventional SQL or OLAP based queries. The submenu of Clustering is as shown in Figure 8.2. The menu also consists of Convention K-Means to segment the 2D echo image using SQL method for $k=3$ and $k=5$, and the K-Means for color Doppler images.

Fig. 8.3 Submenu items of 2D Echo Quantification
The submenu of 2D Echo Quantification is as shown in Figure 8.3. The user can trace the boundary of any cavity by specifying the initial contour interactively by clicking New Image option. But this option first segment the image then gets the object using active contour algorithm. Whereas the next option Presegmented Image does not segment the image but traces the cavity directly.
In the third option the user need not specify even the initial contour, meaning it is fully automatic tracing. The last option can be clicked if the user wishes to avoid penetration of the contour through gaps. The submenus of *color Doppler* and *Classification* are as shown in Figure 8.4 and Figure 8.5 respectively.

Fig. 8.7 Class Diagram of the Main Form

Finally, for the CBIR work, the image database path and the query images are to be specified and various features would be computed and stored in the feature database. When the
user clicks the Search button top \( k \) most relevant/nearest images will be retrieved and displayed in the thumbnail view. Figure 8.6 illustrates the working of CBIR module for color Doppler images.

### 8.1.3 Class Diagram

An important data structure for this work is the feature vector structure. This structure consists of data members for color Doppler image features such as histogram, contrast, edge frequency, texture features such as entropy, energy, etc., and 2D echo image cardiac chamber dimensions such as diameter, height, area, etc. Figure 8.7 shows the important methods under this class.

The first and the last boxes in Figure 8.7 show the member data and methods of the main class [John Sharp, 2002] [Karli, 2006] [CSharpStation, 2010]. These methods create objects of other classes, or call static methods present in other classes to accomplish the requested task. For example, to segment a 2D echo image the following code snippet is required:

```csharp
QuickKMeansClass fkm = new QuickKMeansClass();
fkm.DeleteTables();
Bitmap segImage = fkm.kMeans(queryImage, iter);
```

The `fkm` object is of type `QuickKMeansClass` and `kMeans` method is called to segment the input image `queryImage` with number of iterations `iter` send as a parameter.

#### Member Data

The structure definition called `fvt` is the main feature vector useful for CBIR and other operations. This contains the feature data of all the features that will be extracted from the 2D and color echo images through different methods. The partial list is as shown below:

```csharp
public struct fvt
{
    // Color Histogram
    public string imgid;
    public double median_red;
    public double mean_red;
    public double sd_red;
    public double median_green;
    public double mean_green;
}
```
public double sd_green;
public double median_blue;
public double mean_blue;
public double sd_blue;

// CDF Features
public double cdf_mean_red;
public double cdf_mean_green;
public double cdf_mean_blue;
public double cdf_sd_red;
public double cdf_sd_green;
public double cdf_sd_blue;
public double cdf_contrast_red;
public double cdf_contrast_green;
public double cdf_contrast_blue;
public double cdf_EG_red;
public double cdf_EG_green;
public double cdf_EG_blue;
public double cdf_kurtosis;
public double cdf_skewness;

// Color Doppler Edge Density
public double cdf_ed_mean;

// 2D Echo
public double tde_LVA;
public double tde_LVV;
public double tde_EF;
public double tde_LAA;
public double tde_LAV;

// Texture
public double energy;
public double entropy;
public double contrast;
public double homo;
public double edefrequency;

// Edge Direction Histogram
public double[] edh;
public int[] rhist;
public int[] ghist;
public int[] bhist;

..........................
}
Another structure that is used for ranking the retrieved images during CBIR task is as shown here:

```csharp
public struct cbirImgStruct
{
    public double distance;
    public string imgFile;
}
```

Apart from the `fvt` structure which is a consolidated feature vector used for Euclidean distance computation, individual structures are maintained for each feature. In the proposed CBIR model the user can select/deselect from a list of features. The feature vector(s) corresponding to the selected feature(s) only need to be picked from the feature database and the rest will be zero. This will not affect the distance calculation. This selection can be modified at run time giving the user flexibility and provide a feedback mechanism in a novel way. Few examples are shown here.

```csharp
// Color Histogram structure
public struct ColorHistStruct
{
    // color histogram
    public string imgid;
    public double median_red;
    public double mean_red;
    public double sd_red;

    public double median_green;
    public double mean_green;
    public double sd_green;

    public double median_blue;
    public double mean_blue;
    public double sd_blue;
}
```

```csharp
// 2D Echo image structure
public struct TwoDEchoStruct
{
    // color Doppler Features
    public string imgid;
```

203
public double LVArea;
public double LVVolume;
public double EF;

public double LAArea;
public double LAVolume;
..........................
}

Public/Private Methods

Some of the important methods that are responsible to carry out data mining and CBIR operations are discussed here. The Form Load entry point does not call any major method, but initializes some of the member data and disable some of the menu/submenu, or buttons in the form. Most of the buttons/menu items are grayed out until the user selects the query image. Once the image is successfully loaded, then one of the many options can be selected and appropriate method(s) is invoked as given below:

(1) K-Means Algorithm (Conventional Approach)
   // Static method - no object creation required
   Bitmap segImage = kMeans.kMeansSegmentation(queryImage, k);

(2) Fast K-Means SQL (k = 3)
   FastKMeansClass fkm = new FastKMeansClass();
   Bitmap segImage = fkm.kMeans(queryImage, iter);

(3) Quick K-Means SQL (k = 3)
   QuickKMeansColorClass kmc = new QuickKMeansColorClass();
   Bitmap segImage = kmc.kMeansColor(queryImage, iter);

(4) CBIR Module
   EuclideanDistance(threshold);

This method takes care of computing the distance from the query image feature vector data to all the images stored in the feature database and ranks them. If certain features are not applicable or user has not selected, then such features would automatically contain zero. The parameter threshold would retrieve only those many top most similar images. To build the query image feature vector, the following code is used:
QueryImagefvt(queryColHist, queryCDF, queryED, queryTDE, queryTex, queryShapeData, queryIntHaar, ...........);

And the distance can be calculated as,

$$d = \text{Eucldist}(\text{TempQueryFeatureVector}, \text{TempFeatureVector}[\text{index}]);$$
8.1.4 **CLASS DIAGRAM OF K-MEANS CLUSTERING METHOD (CONVENTIONAL)**

To compare the practical execution time of the proposed SQL based K-Means with the conventional algorithm, a set of classes have been developed and is shown in Figure 8.8. Here, JCA is the main class in which `startAnalysis()` methods contains the basic nested iterations for number of data points, number of clusters, and number of iterations. Other methods are just the supporting ones to get the clustered data points.

8.1.5 **CLASS DIAGRAM OF ECHO IMAGE SEGMENTATION**

This class diagram includes the segmentation process through Fast SQL K-Means clustering algorithm and boundary tracing with the help of active contour. In addition to this, variations of the SQL based design are also given. For example, QuickK-Means, K-Means for color Doppler images, etc. do appear in the same diagram and is shown in Figure 8.9.

The three SQL based classes namely `FastKMeansClass`, `QuickKMeans`, and `KMeansColorClass` follow almost similar kind of design. The only difference is the number of tables and methodology used in writing the SQL queries.
Fig. 8.9 Class Diagram of Segmentation and boundary tracing
The method $kMeans()$ first initializes, assuming that the table data is already deleted) all the tables using a call $Initialize(pixels)$ where pixels is the array that contains the image pixel data. Another crucial method is $kMeans\_Update()$ which updates all the tables for pres-specified number of iterations. The following connection string is used that communicates to Oracle 10g tables:

```csharp
string oradb = "Data Source=orcl;User Id=scott;Password=tiger;";
```

And all SQL commands are executed using the following code snippet:

```csharp
OracleCommand cmd = new OracleCommand(sql, conn);
cmd.ExecuteNonQuery();
```
where, sql is the SQL query string which is ultimately executed by the Oracle SQL engine [Oracle-parallel, 2010]. SNGFilter() as given in the class diagram removes the unwanted black spots in the segmented image as part of the post processing.

8.1.6 CLASS DIAGRAM FOR CBIR

This diagram is split and shown as three portions due to the shortage of space. The method Compute2DEchoFeatures() preprocesses the image, segments and then extract the two dimensional echo image features and returns it in the form of a structure by using active contour approach.

```java
ActiveContour ac = new ActiveContour();
CavityDimension Lventricle = ac.GetContourFeatures();
```

It can either trace the internal boundary or external boundary depending upon the value sent via the parameter: ‘type’. The class diagram for the next part of this module is as shown in Figure 8.11.

Several classes such as ColorHistogram, EdgeDirectionHistogram, HaarWavelets, etc., help to obtain features from the 2D echo and color Doppler images and for each of this a structure is used to store the same. From these data structures it is easy to export to the image feature database. The class diagram for the final part of this module is as shown in Figure 8.12.
CHAPTER 8 - SOFTWARE IMPLEMENTATION

Fig. 8.11 Class Diagram of CBIR (Features) – contd.
This part of the diagram is for retrieving the texture and statistical feature extraction classes. As explained earlier supporting structures are used to store these features for later processing. An additional class, *ImageThumbnail* class is used to display the retrieved image thumbnail view and it is developed using the *imageView* and *listview* controls of C#.NET.

### 8.2 SUMMARY

A comprehensive customized framework has been developed using Microsoft Visual Studio C#.NET 2005/2010 on Windows platform. The implementation is based on the top down approach and consists of four steps: segmentation, boundary extraction, CBIR, and classification. Several image processing tools, data mining tools, and open source software components are used in the implementation.