CHAPTER 1

INTRODUCTION AND MOTIVATION

Today, the world of the smart devices such as the personalized digital interactive TV is a fast developing area with well established and proven research concepts .The future interactive TV should be designed to learn for it-self and to prioritize the available programs for the final viewer according to the accepted knowledge. The introduction of the interactive TV brings new challenges as well as new opportunities to the existing world of TV services. The interactive TV applications enable the consumer to actively give feedback on the TV program. This feedback can be collected by the application and fed-back to the content provider. The information on the viewer preferences or reactions could be used for the planning of the TV program. This would require that the user feedback is collected in real-time and that the whole content delivery process is adapted to effectively use this new information. The personalized interactive TV services such as user specific program guide require consumer authentication and consumer input, which is traditionally done by logging in to the set-top box through the remote control, which is not a very user-friendly procedure. To win wide consumer acceptance, the future interactive TV should be friendly and personalized, which implies that next-generation interfaces will be aware of the people in their immediate environment and know who they are and even what they like or dislike. The investigation of simple and intuitive human-machine interfaces is therefore important for its widespread proliferation. The personalized future interactive TV will communicate with us more like humans. A key component of that interaction will be its abilities to continually detect and recognize our faces, and even our expressions, and therefore understand our internal emotional states. Considering the above requirements, in this paper, we introduce a novel structure of the future interactive TV and investigate a robust real-time face analysis system based on multi-class pattern recognition to enable consumer authentication and even his or her preference feedback for the personalized interactive TV services.

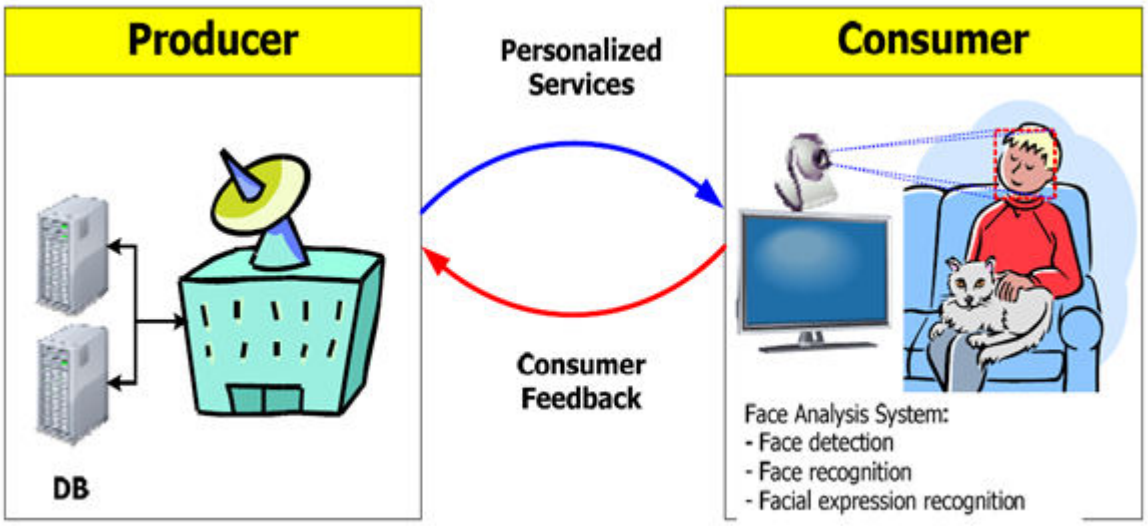


Fig.1.1 The overall structure of the future interactive TV

Fig. 1.1 illustrates the proposed architecture of the future interactive TV. Face analysis system on the set-top box connected to the video camera continually detects and tracks the TV viewers. Simultaneously recognizing their identities and facial expressions, it can give feedback on the information about the emotions or preferences of the viewers to the service producer in real-time. Then, the service producer can adaptively provide personalized user interfaces and contents for each viewer. To design the cognitive real-time face analysis system, in this paper, we present a novel and universal algorithm for multi-class pattern classification: face detection, face recognition, and facial expression recognition, which uses a strong classifier based on the simple and discriminative multi scale and multi-position Local Binary Pattern (Msp LBP) features and neural network.

CHAPTER 2

PROBLEM STATEMENT

Cognitive Face Analysis for future interactive T.V. is a real-time, user response based feedback system. The proposed system must be able to state the mood of the user watching the television program. For this, the system must be able to capture the user face images and generate corresponding histograms. The expressions of the user must be evaluated using concepts like Msp-LBP operator. The input image must be matched with closest matching pattern, thus finding the ideal expression match. To implement this, three important steps must be used – face detection, face recognition and facial expression evaluation.

Face detection is a fundamental task as a preprocessing step for many applications such as face tracking and recognition, and facial expression recognition. Face recognition, as one of the primary biometric technologies, has several advantages over other biometric technologies: It is natural, nonintrusive, and easy to use. It can also be used as the initial step toward interpreting human actions and intensions. Facial expressions are facial changes in response to a person's internal emotional state, intention, or social communication. Facial expression recognition is therefore important for automatic feedback of information about the viewer's reactions, intentions, and preferences in the future interactive TV. Face detection, face recognition, and facial expression recognition are the typical problems of multi-class pattern classification. To deal with these kinds of multi-class pattern classification problems, we propose a novel and universal boosting technique called Msp LBP. LBP is defined as a gray-scale invariant texture measure in a local neighborhood.

The final output must give the expression of the user after watching the program after evaluating the exact expressions of the user while watching the program.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 Feasibility Study

The feasibility study plays a major role in the analysis of the system. The very decision of the system analyst, to design a particular system depends whether the system is feasible or not. Hence, the feasibility study forms the very basic of the system. The feasibility study can be categorized into:

3.1.1 Technical Feasibility

It has been determined that the technology support needed for the proposed system is available and that this technology can use the resultant input image patterns. Technical evaluation has also evaluated the existing system to find that it can be upgraded in keeping with the system needs.

3.1.2 Operational Feasibility

There are two aspects of operational feasibility for the system, technical performance and acceptance. The system can provide correct data without compromising on the quality of image. Also it has been determined that the system will be accept images and generate histograms of the input images.

3.1.3 Economic Feasibility

The economic feasibility of the system is mainly concerned with its financial aspects. It determines whether the project is economically feasible. As the hardware and software are already available easily in the market, no further investment is to be made in that direction. It was decided that the project was technically feasible because of the following:

Necessary technology exists to do what is suggested.

The system would be expandable if so decided.

3.2 Software Development Model Used

Several models exist to streamline the development process. Each one has its pros and cons, and it is up to the development team to adopt the most appropriate one for the project. We should choose the software development model according to the complexity and nature of the project development.

We have used the **Waterfall Model** for our project. The Waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. The progress is seen as flowing steadily downwards through the phases of analysis, design, coding, testing and maintenance.

Requirement Analysis- All possible requirements of the system to be developed are captured in this phase. Requirements are set of functionalities and constraints that the end-user expects from the system. In the project, we had to collect the images in this phase.

Design- Before a starting for actual coding, it is highly important to understand what we are going to create and what it should look like. The requirement specifications from first phase are studied in this phase and system design is prepared. We select the suitable algorithms and finalize the flow and working of the project.

Implementation- On receiving system design documents, the work is divided in modules/units and actual coding is started. The system is first developed in small programs called units, which are integrated in the next phase. We have used Matlab for coding.

Testing and Maintenance- Generally, problems with the system developed (which are not found during the development life cycle) come up after its practical use starts, so the issues related to the system are solved after deployment of the system. Here, we check for the accuracy of the facial expression matching.

3.3 Hardware and Software Requirement

3.3.1 Hardware Requirement

Processor: Pentium /core 2 duo

Hdd: 5 GB or more

Monitor: svga color

Modem: 56 kbps

Fdd: 1.44 Mb

Camera: Above 2 Mp

3.3.2 Software Requirement

Operating system: windows XP/windows 7

MATLAB version 7 or above

CHAPTER 4

PROJECT DESIGN

4.1 Architecture

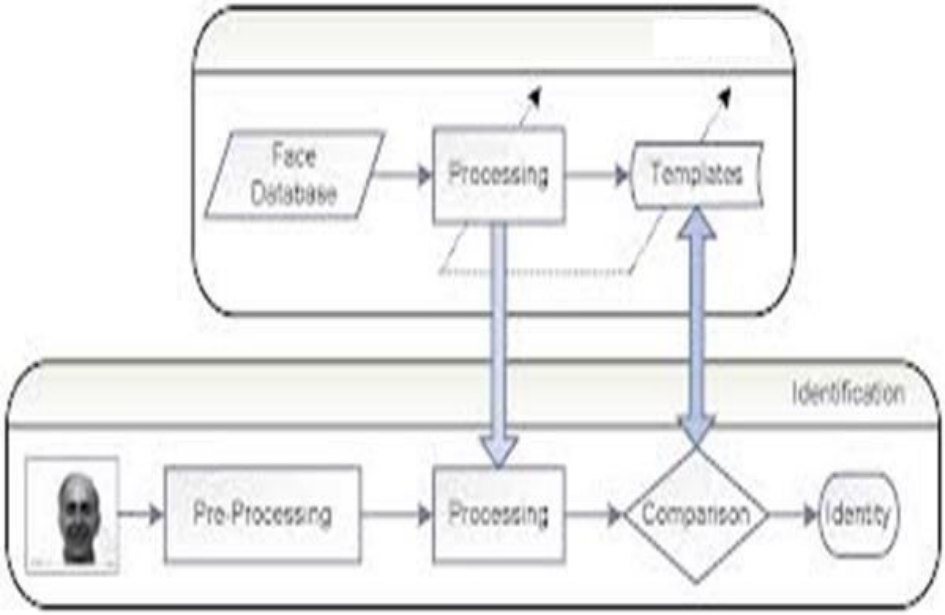


Fig. 4.1.1 Block diagram of the proposed system.

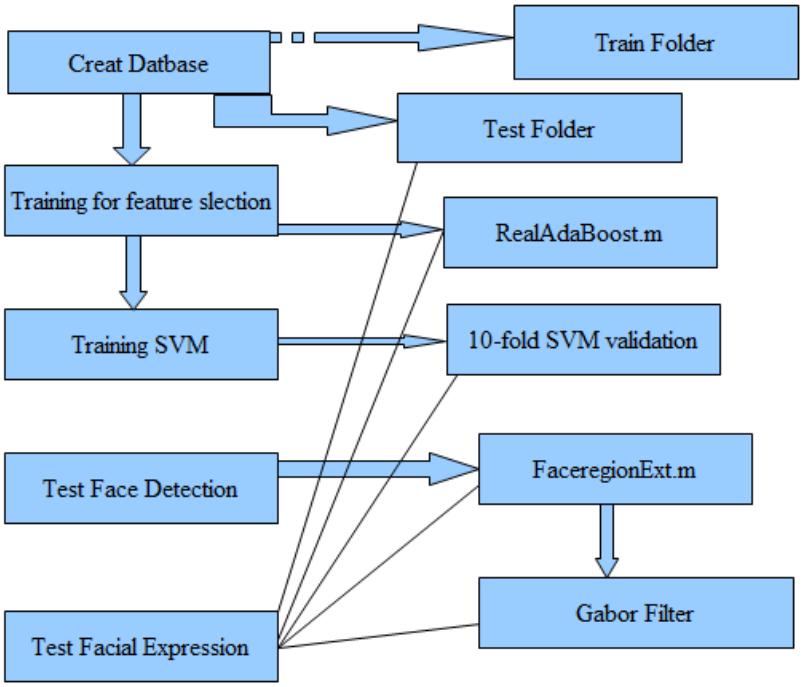


Fig. 4.1.2 Flow diagram of the proposed system.

4.1.1 Gabor Features

The Gabor wavelet representation of images allows description of spatial frequency structure in the image while preserving information about spatial relations. The Gabor filter provides an excellent basis for object recognition and face recognition. In this report, we use 2-D Gabor filters for the purpose of extracting facial features.

$$G_j(\vec{x}) = \frac{k_j^2}{\delta^2}\exp(-\frac{k_j^2 x^2}{2\delta^2})[\exp(i\vec{k}_j\vec{x}) - \exp(-\frac{\delta^2}{2})]$$

$$\vec{k}_j = \begin{pmatrix} \omega_{jx} \\ \omega_{jy} \end{pmatrix} = \begin{pmatrix} \omega_v \cos\phi_u \\ \omega_v \sin\phi_u \end{pmatrix},$$

$$\varphi u = u \cdot \pi 8$$

Where delta is the scale factor of the Gaussian envelope, which decides the scale and bandwidth of the bandwidth. The Wjx and Wjy are separate frequencies in direction x and y. The 2-d Gabor filter is a family of narrowband band-pass filters. It has good resolution both in spatial field and frequency field. It also has obvious speciality of orientation selection and frequency selection. In this study, 5 different spatial frequencies were used with wave numbers wv = (2pi, 4pi, 8pi, 16pi, 32pi) and 12 orientations from were used.

$$S_\phi(J, J') = \frac{\sum_j a_j a'_j \cos(\phi_j - \phi'_j - \vec{d}\vec{k}_j)}{\sqrt{\sum_j a_j^2}\sqrt{\sum_j a_j'^2}}$$

Gabor-face vectors constitute 64x64x5x8 dimension features (for 40 filters). However the dimension of Gabor-face vector is very high. Here the efficient features are extracted in Gabor-face vector using Ada Boost with lesser gabor features.

4.1.2 Support Vector Machines

“Support Vector Machines are a maximal margin hyper plane classification method that relies on results from statistical learning theory to guarantee high generalization performance.” In our system the classification of expression is done using Support vector machine. They are sets of unique methods mainly used for the purpose of regression and classification. JAFFE expression database was used significantly to evaluate expression recognition in frontal still images. Performance evaluates using 10-fold cross validation strategy.

4.2 Pipe and Filter Architecture Style

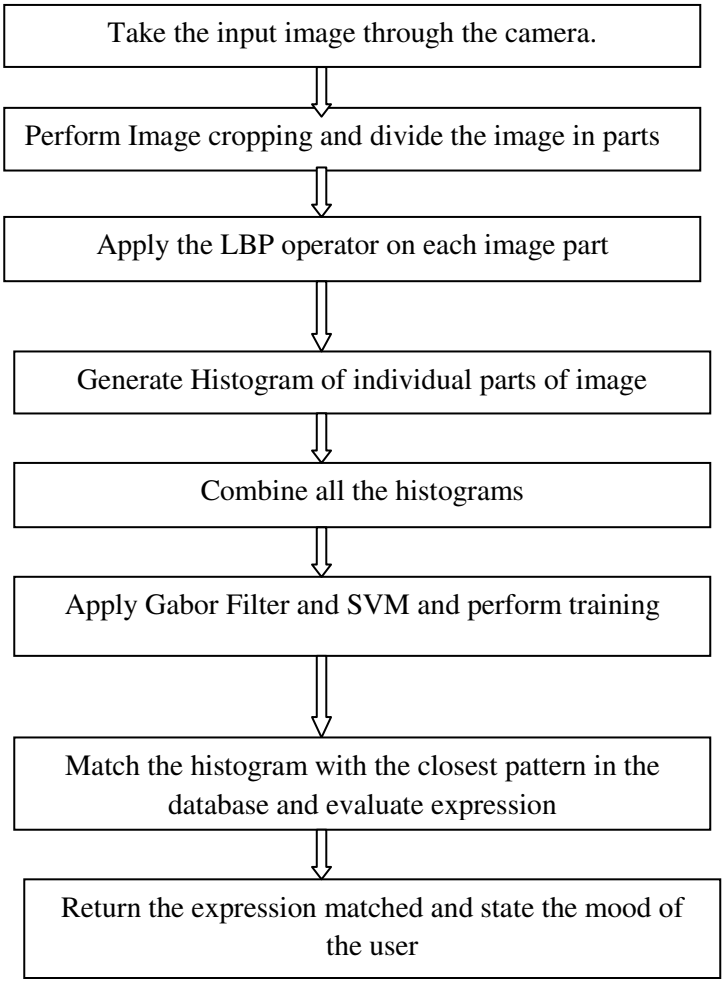


Fig 4.2.1 Pipe and Filter Architectural Style

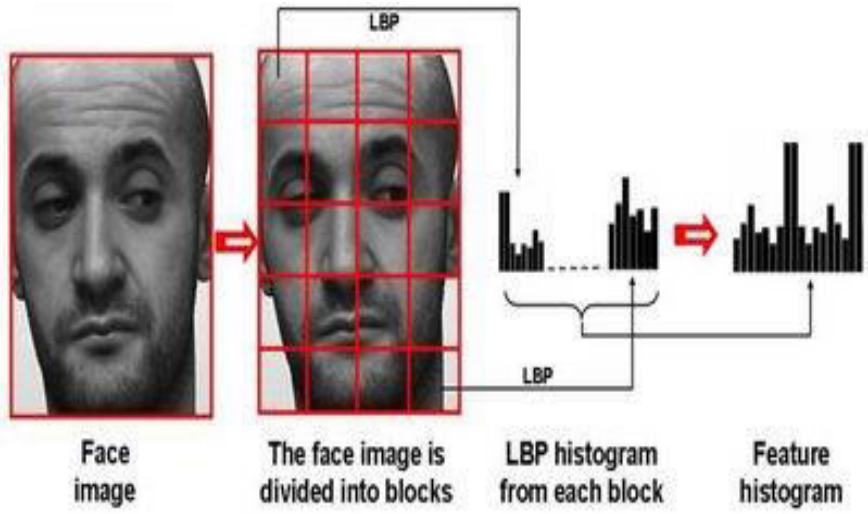


Fig 4.2.2 Msp-LBP feature vector generation when using the LBP Features

Step 1: Take a visual image (RGB image) as input.

Step 2: Resize the image.

Step 3: Crop the image.

Step 4: The face image is divided into various blocks.

Step 5: Now perform the application of LBP operator. For this, consider the center pixel. Get the values of all its neighboring pixels and convert it into binary values and get the final decimal value.

Step 6: Repeat step 5 for the parts of the image.

Step 7: Now compare the value of each pixel to the center pixel. Based on its value, differentiate the pixel and find pattern.

Step 8: After applying LBP, create the histogram of individual image parts.

Step 9: Fuse all the individual histograms into one final histogram of the entire face.

Step 10: Extract the facial features using Gabor features and SVM concept.

Step 11: A Close match is found based on histogram matching and training using SVM.

Step 12: Return the expression this facial image as the output.

Step 14: End

4.3 Data Flow Diagram

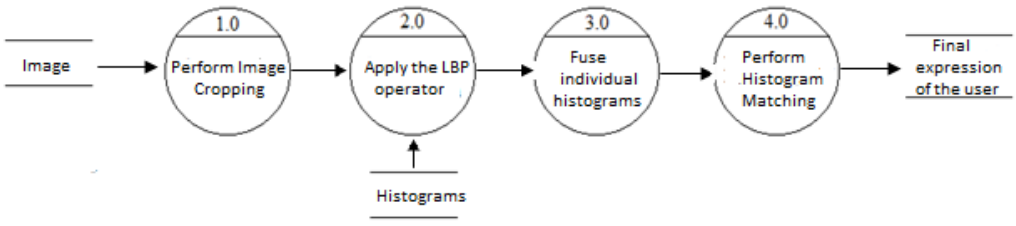


Fig. 4.3.1 Dataflow Diagram for overall system

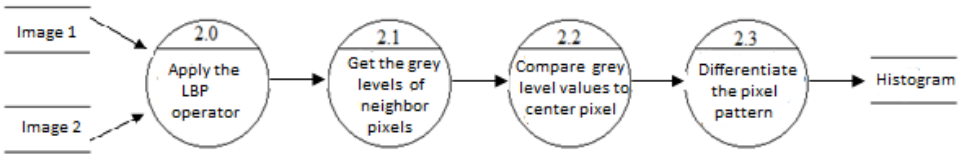


Fig. 4.3.2 Dataflow Diagram for Apply LBP operator

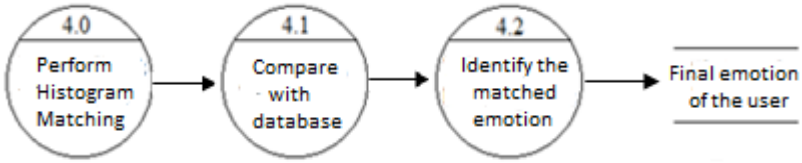


Fig. 4.3.3 Dataflow Diagram for Performing Histogram Matching

4.4 UML Diagrams

4.4.1 Use-case diagram

Use-case diagram is specifying the different users & their functionalities. The use-case diagram has number of use-case which describes different functions that can be performed by the actors and actors. The admin can be involved in various scenarios like login, insert, update etc. The Viewer can be involved in various scenarios like adding & updating content in databases.

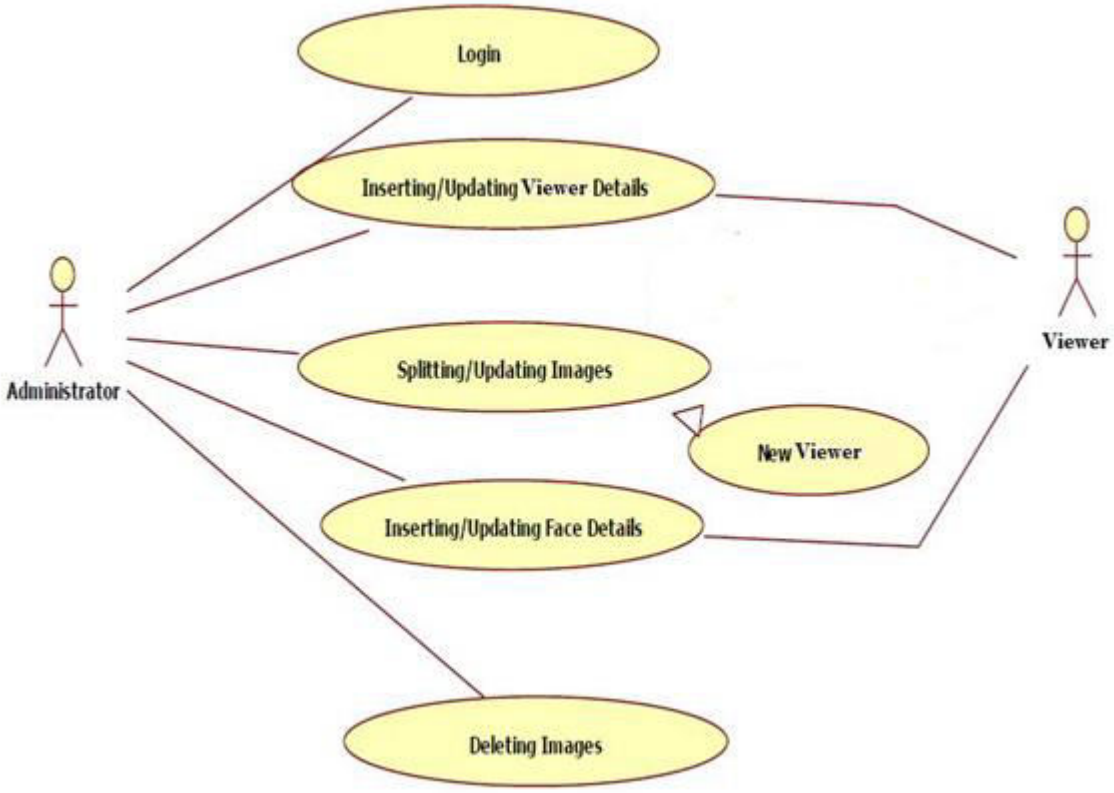


Fig.4.4.1.1 Use-case diagram

4.4.2 Sequence diagram

In the Sequence diagrams, the admin uploads & updates contents in System which is the database consisting of the images of all the viewers in the system . Similarly the admin users can ask details, view as well as split images in the database.

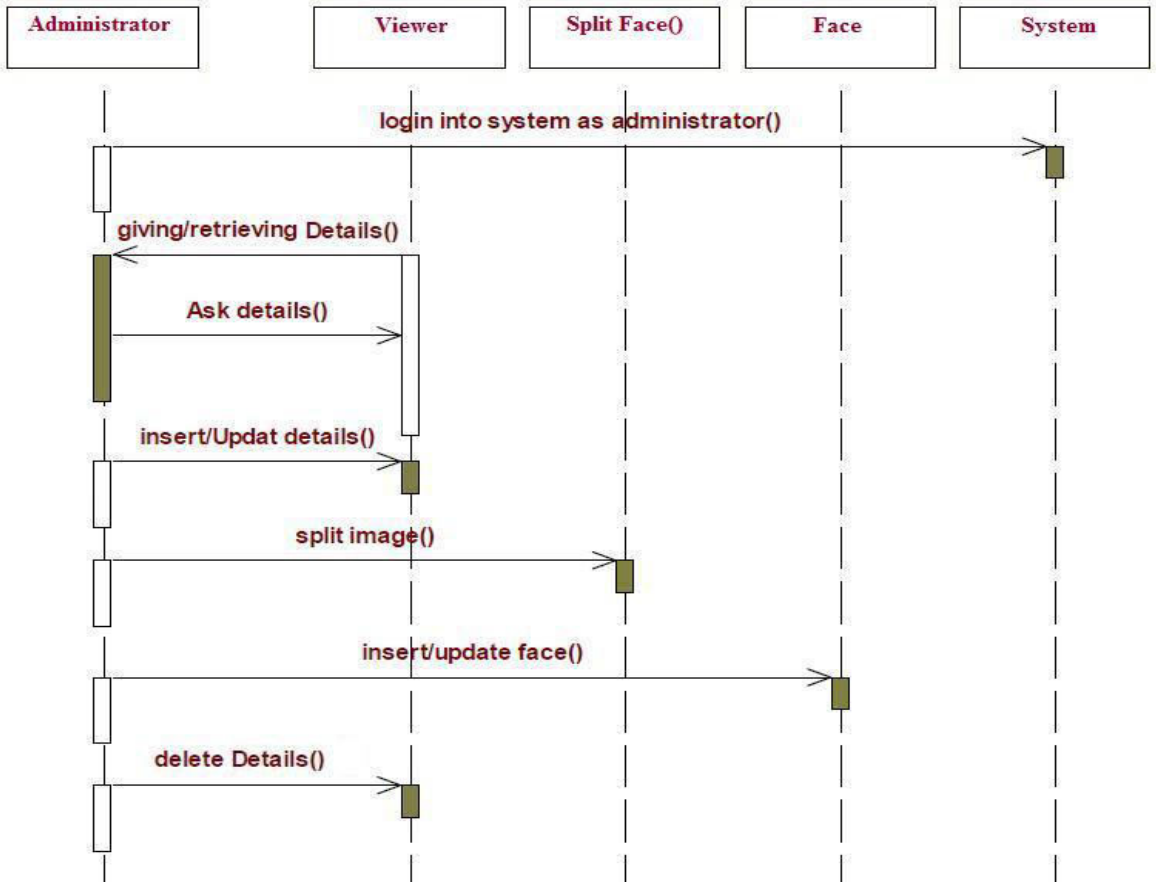
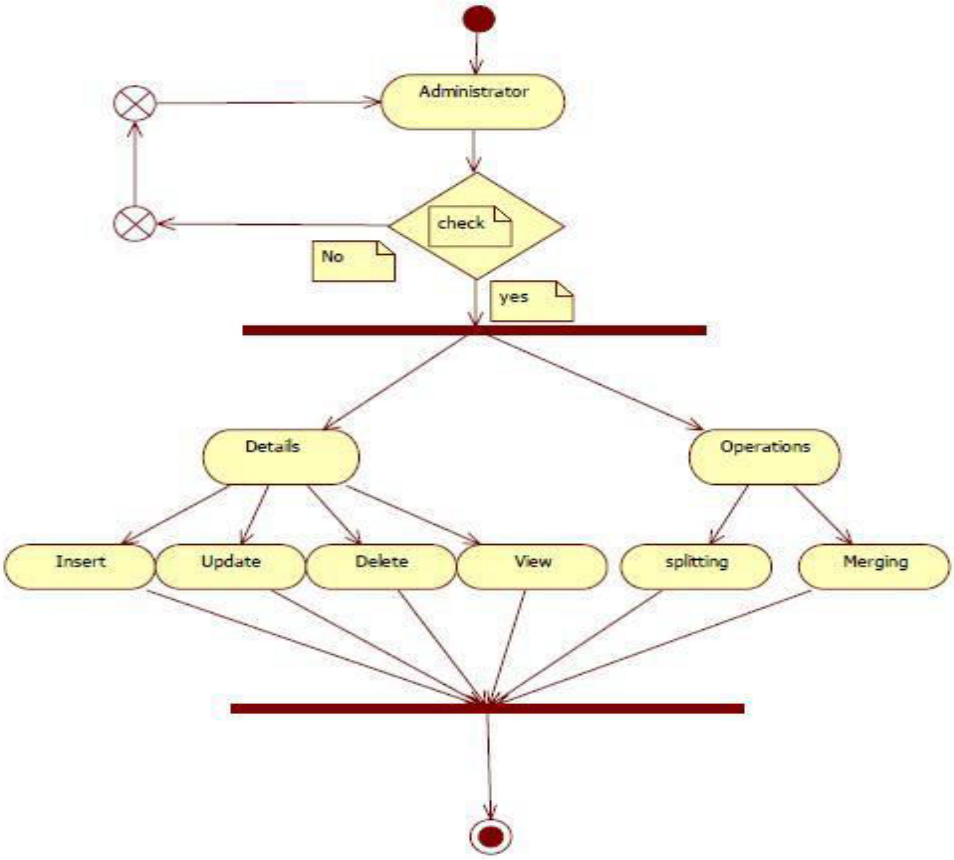


Fig. 4.4.2.1 Sequence Diagram

.

4.4.3 Activity diagram

Activity Diagram is specifying the flow or the sequence of events. In the Activity diagram, the user first needs to register and then login, then only he/she can proceed further & access the system. After successfully logged in the user can search for programs in the database. After searching the registered user, the admin can confirm the identity.



4.4.3.1 Activity Diagram

CHAPTER 5

IMPLEMENTATION DETAIL

5.1 Excerpts From Code

5.1.1 Input from Camera

```
clc ;                                                    % clearing the command window
n = input(' Enter the number of photos to be taken: ');
intervel = input(' Enter the time(seconds) gap between succeessive photos: ');
photosave = input(' Do you want to save the files(y/n): ','s');
disp('Please wait...');
outputFolder = fullfile(cd, 'frames');
if ~exist(outputFolder, 'dir')
mkdir(outputFolder);
end
obj = videoinput('winvideo',1);
preview(obj);
disp('Press Enter to start after webcam initialization.');   % webcam starts automatically
pause;
disp('First shot will taken after 1 second');
pause(1);
for i=1:n
img=getsnapshot(obj);
image(img);
if(photosave == 'y')
outputBaseFileName = sprintf('fr%d.png',i);
outputFullFileName = fullfile(outputFolder, outputBaseFileName);
imwrite(img,outputFullFileName,'jpg');
end
pause(intervel);
end
closepreview;
disp('The program successfully taken the photos');
disp('Done.');
```

5.1.2 User Interface

```
clear all;
clc;
close all;
while (1==1)
   choice=menu('Facial Expression Recognition',...
```

```
            'Create database',...
            'Training for feature selection',...
            'Training SVM',...
            'Test Face Detection',...
            'Test Facial Expression',...
            'Exit');
    if (choice ==1)
                                    % create database
        CreateDatabase;
    elseif (choice ==2)
                                    % feature selection sby AdaBoost
        TrainFeaSelectECOC;
    elseif (choice == 3)
                                     % training by SVM
        TrainSVM1toallECOC;
    elseif (choice == 4)
                                     % testing face detection
        TestFaceDetect;
    elseif (choice == 5)
                                     % test facial expression recognition on photos
        TestPhoto1toall;
    elseif (choice == 6)
                                     % exit program
        clear all;
        clc;
        close all;
        return;
    end
```

5.1.3 Creating database

```
clear;
clc;
close all;

%*****************************************************
% select training folder

[file_name file_path] = uigetfile ('*.*','Select for training');
if file_path == 0
   return;
end
szTrainPath = file_path;
szLableFilename = strcat(szTrainPath,'list');

fid=fopen(szLableFilename);
imageLabel=textscan(fid,'%s %s %s %s %s');
fclose(fid);
```

```
mask = [];

numImage = length(imageLabel{1});          % Total Observations: Number of Images in training set
TrainImages='';
for i = 1:numImage
        TrainImages{i,1} = strcat(szTrainPath, imageLabel{5}(i));
end

traindb = [];
for i = 1:numImage
   s=imageLabel{5}(i);
   tt=cell2mat(s);
   ss = tt(4:5);
   yapp(i) = ClassNo(ss);
end
for i = 1:numImage
   t=TrainImages{i,1};
   s= cell2mat(t);
   im1=double(imread(s));
   im2 = FaceRegionExt(im1);
   if (length(im2) < 2)
      kkk=0;
   end
   aa=FaceExpFea (im2);
   traindb(:,i) = aa(:);
   disp(sprintf('Loading Train Image # %d',i));
end
for i = 1:numImage
   s=imageLabel{5}(i);
   tt=cell2mat(s);
   ss = tt(4:5);
   traindblabel(i) = ClassNo(ss);
end

save traindb traindb;
save traindblabel traindblabel;
%##########################################################################
% select testing folder

[file_name file_path] = uigetfile ('*.*','Select for testing');
if file_path == 0
   return;
end
szTestPath = file_path;
szLableFilename = strcat(szTestPath,'list');

fid=fopen(szLableFilename);
imageLabel=textscan(fid,'%s %s %s %s %s');
fclose(fid);
```

```
mask = [];

numImage = length(imageLabel{1});               % Total Observations: Number of Images in training set
TestImages='';
for i = 1:numImage
        TestImages{i,1} = strcat(szTestPath, imageLabel{5}(i));
end

testdb=[];
for i = 1:numImage
   s=imageLabel{5}(i);
   tt=cell2mat(s);
   ss = tt(4:5);
   yapp(i) = ClassNo(ss);
end
for i = 1:numImage
   t=TestImages{i,1};
   s= cell2mat(t);
   im1=double(imread(s));
   im2 = FaceRegionExt(im1);
   if (length(im2) < 2)
      kkk=0;
   end
   aa=FaceExpFea (im2);
   testdb(:,i) = aa(:);
   disp(sprintf('Loading Test Image # %d',i));
end
for i = 1:numImage
   s=imageLabel{5}(i);
   tt=cell2mat(s);
   ss = tt(4:5);
   testdblabel(i) = ClassNo(ss);
end
save testdb testdb;
save testdblabel testdblabel;
```

5.1.4 Training for features

```
% loading train data and test data

load traindb;
load traindblabel;
load testdb;
load testdblabel;
classnum=6;

MaxIter = 5;                                % boosting iterations
```

```
traindblabel1=traindblabel;
testdblabel1=testdblabel;

mECOC=[
   1 0 0;                                   %Angry
   0 1 0;                                   %Disgust
   0 1 1;                                   %Fear
   0 0 1;                                   %Happy
   1 1 0;                                   %Sad
   1 0 1                                    %Surprise
   ];
err=zeros(classnum,2);
traindb=[traindb testdb];
for i=1:size(mECOC,2)
   traindblabel1=-ones(size(traindblabel,1),size(traindblabel,2));
   testdblabel1=-ones(size(testdblabel,1),size(testdblabel,2));
   for iclass=1:classnum
      if mECOC(iclass,i)==1
         for isam=1:length(traindblabel)
            if traindblabel(isam) == iclass
               traindblabel1(isam)=1;
            end
         end
         for isam=1:length(testdblabel)
            if testdblabel(isam) == iclass
               testdblabel1(isam)=1;
            end
         end
      end
   end

   traindblabel1=[traindblabel1 testdblabel1];
   weak_learner = tree_node_w(1);                % pass the number of tree splits to the constructor
   if i==1
      [RLearners1 RWeights1] = RealAdaBoost(weak_learner, traindb, traindblabel1, MaxIter);
      ResultTrainR1 = sign(Classify(RLearners1, RWeights1, traindb));
      err(i,1)  = sum(traindblabel1 ~= ResultTrainR1) / length(traindblabel1);
   elseif i==2
      [RLearners2 RWeights2] = RealAdaBoost(weak_learner, traindb, traindblabel1, MaxIter);
      ResultTrainR2 = sign(Classify(RLearners2, RWeights2, traindb));
      err(i,1)  = sum(traindblabel1 ~= ResultTrainR2) / length(traindblabel1);
   elseif i==3
      [RLearners3 RWeights3] = RealAdaBoost(weak_learner, traindb, traindblabel1, MaxIter);
      ResultTrainR3 = sign(Classify(RLearners3, RWeights3, traindb));
      err(i,1)  = sum(traindblabel1 ~= ResultTrainR3) / length(traindblabel1);
   end

   if i==1
      disp('  TrainErr');
```

```
   end

end
disp(err);
save RLearners1 RLearners1;
save RLearners2 RLearners2;
save RLearners3 RLearners3;
```

5.1.5 SVM

```
clear all;
close all;
load traindb;
load traindblabel;
load testdb;
load testdblabel;

load RLearners1;
load RLearners2;
load RLearners3;

classnum=6;
traindb=(traindb)/256;
testdb=(testdb)/256;

c = 1000;
lambda = 1e-7;
kerneloption= 2;
kernel='gaussian';
verbose = 1;

traindb=[traindb testdb];
traindblabel=[traindblabel testdblabel];

traindb_svmlabel=traindblabel';
testdb_svmlabel=testdblabel';

a=[];
for i=1:3
   if (i==1)
      for k = 1 : length(RLearners1)
        out=get_dim(RLearners1{k});
         a=[a out];
      end
   elseif i==2
      for k = 1 : length(RLearners2)
        out=get_dim(RLearners2{k});
         a=[a out];
```

```
        end
    elseif i==3
        for k = 1 : length(RLearners3)
          out=get_dim(RLearners3{k});
           a=[a out];
        end
    end
end

idxarray=zeros(1,size(traindb,1));

traindb_svm = [];
testdb_svm = [];
for k = 1 : length(a)
   if (idxarray(a(k))==1)
       continue;
   end
  traindb_svm=[traindb_svm traindb(a(k),:)'];
  testdb_svm=[testdb_svm testdb(a(k),:)'];
  idxarray(a(k))=1;
end


dbidx = zeros(10, size(traindb_svm,1));
for i=1:size(traindb_svm,1)
    ifold=round(rand(1)*10 + 0.5);
    dbidx(ifold,i)=1;
end
totaldb=traindb_svm;
totaldb_label=traindb_svmlabel;

trainrecograte=[];
testrecograte=[];
for iFold=1:10
    traindb_svm=[], traindb_svmlabel=[], testdb_svm=[], testdb_svmlabel=[];
    for isam=1:size(totaldb,1)
        if (dbidx(iFold,isam) == 1)
            testdb_svm=[testdb_svm' totaldb(isam,:)'];
            testdb_svmlabel=[testdb_svmlabel' totaldb_label(isam,:)'];
        else
            traindb_svm=[traindb_svm' totaldb(isam,:)'];
            traindb_svmlabel=[traindb_svmlabel' totaldb_label(isam,:)'];
        end
    end
    clear xsup, clear w, clear b, clear nbsv;

    nbclass=6;
```

```
[xsup,w,b,nbsv]=svmmulticlassoneagainstall(traindb_svm,traindb_svmlabel,nbclass,c,lambda,kernel,kernelo
ption,verbose);
   [ytrain,maxi] = svmmultival(traindb_svm,xsup,w,b,nbsv,kernel,kerneloption);
   [ytest,maxi] = svmmultival(testdb_svm,xsup,w,b,nbsv,kernel,kerneloption);

   filename=sprintf('recograteSVM1toall_fold_%d.txt',iFold);
   fid = fopen(filename, 'w');
   recognum=0;
   numImage=size(traindb_svm,1);
   for k=1:numImage
      if ytrain(k) == traindb_svmlabel(k)
         recognum = recognum + 1;
      end
   end
   trainrecograte(iFold) = recognum / numImage*100;
   fprintf (fid, 'train performace  %.2f\n',trainrecograte(iFold));

   recognum=0;
   numImage=size(testdb_svm,1);
   for k=1:numImage
      if ytest(k) == testdb_svmlabel(k)
         recognum = recognum + 1;
         fprintf (fid, '%d  %d : true\n', testdb_svmlabel(k), ytest(k));
      else
         fprintf (fid, '%d  %d : false\n', testdb_svmlabel(k), ytest(k));
      end
   end
   testrecograte(iFold) = recognum / numImage*100;
   fprintf (fid, 'test performace  %.2f',testrecograte(iFold));

   fclose(fid);

   save xsup xsup, save w w, save b b, save nbsv nbsv;

end
meantrainrecog = sum(trainrecograte)/10;
meantestrecog = sum(testrecograte)/10;
filename=sprintf('recograteSVM1toall_fold.txt');
fid = fopen(filename, 'w');
fprintf (fid, 'train    test\n');
for i=1: 10
   fprintf (fid, '%.2f   %.2f\n',trainrecograte(i), testrecograte(i));
end
fprintf (fid, '%.2f   %.2f\n',meantrainrecog, meantestrecog);
fclose(fid);
```

5.1.6 Checking facial Expression

```
[file_name file_path] = uigetfile ('*.*');
if file_path == 0
   return;
end
im1=double(imread([file_path,file_name]));

% face detection

im2 = FaceRegionExt(im1);
aa=FaceExpFea(im2);
img = aa(:);

% loading adaboost machine

load RLearners1;
load RLearners2;
load RLearners3;

% SVM
load xsup, load w, load b, load nbsv;

a=[];
for i=1:3
   if (i==1)
      for k = 1 : length(RLearners1)
        out=get_dim(RLearners1{k});
         a=[a out];
      end
   elseif i==2
      for k = 1 : length(RLearners2)
        out=get_dim(RLearners2{k});
         a=[a out];
      end
   elseif i==3
      for k = 1 : length(RLearners3)
        out=get_dim(RLearners3{k});
         a=[a out];
      end
   end
end
idxarray=zeros(1,size(img,1));

fea = [];
for k = 1 : length(a)
  if (idxarray(a(k))==1)
     continue;
  end
```

```
  fea=[fea img(a(k),:)'];
  idxarray(a(k))=1;
end

fea=double(fea);
fea=(fea)/256;
[ypred,maxi] = svmmultival(fea,xsup,w,b,nbsv,kernel,kerneloption);

facialexp = ClassName(ypred);

subplot(1,2,1); imshow(im1,[]);   title(file_name);
subplot(1,2,2); imshow(im2,[]); title(facialexp);
```

5.2 Output

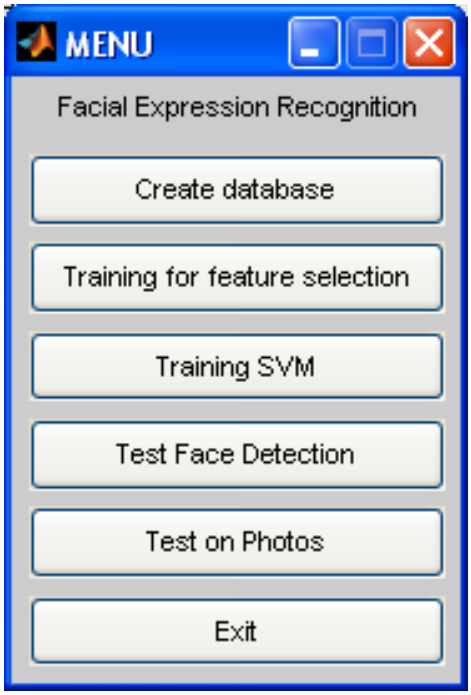


Fig. 5.2.1 Main Menu

Fig. 5.2.2 Detection of Facial Expression

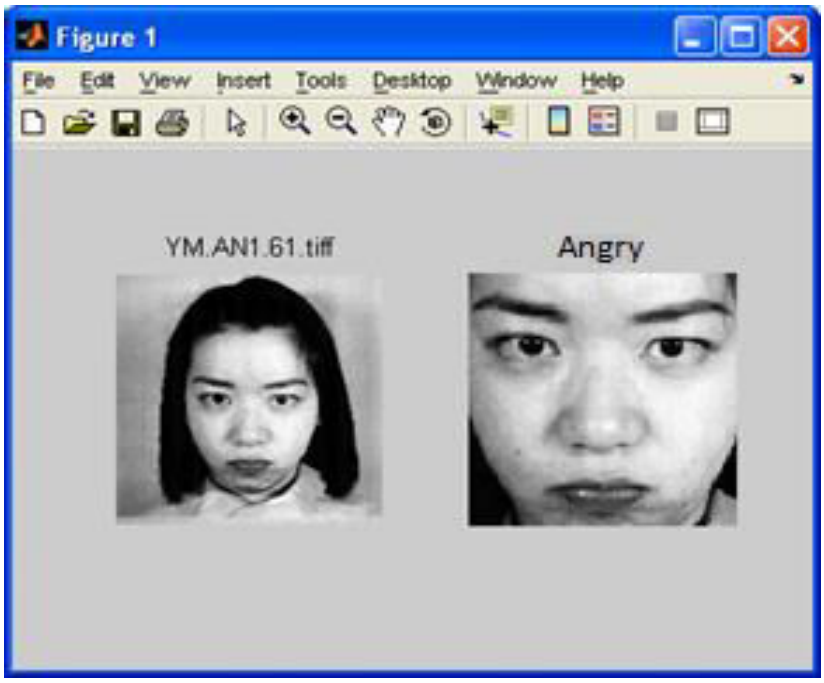


Fig. 5.2.3 Expression Detected

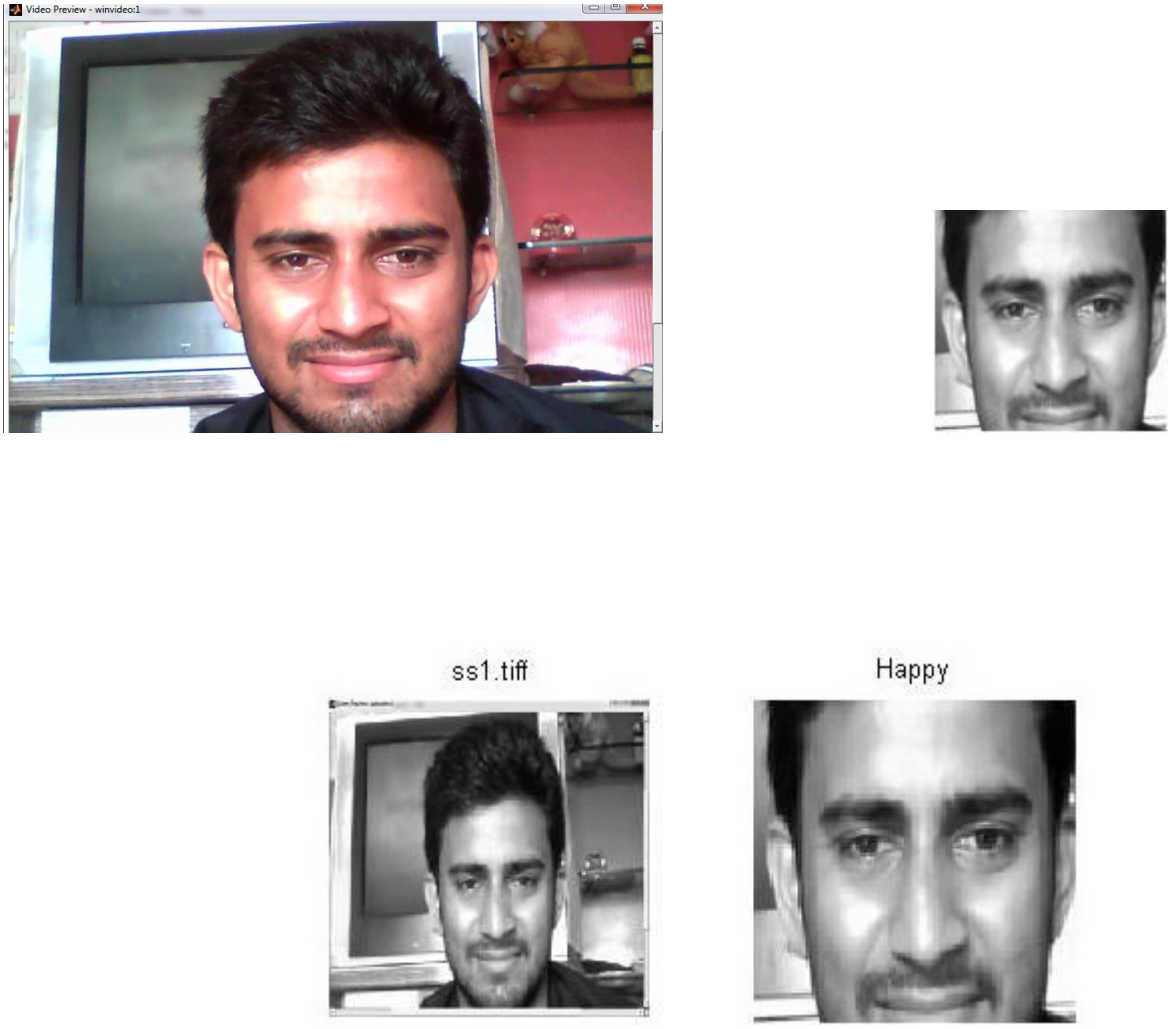


Fig. 5.2.4 Random Face detected and final output

CHAPTER 6

TECHNOLOGY USED

6.1 MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

-Math and computation

-Algorithm development

-Modeling, simulation, and prototyping

-Data analysis, exploration, and visualization

-Scientific and engineering graphics

-Application development, including Graphical User Interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non interactive language such as C or FORTRAN.

The MATLAB system consists of five main parts:

6.1.1 The MATLAB language.

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

6.1.2 The MATLAB working environment.

This is the set of tools and facilities that you work with as the MATLAB user or programmer. It includes facilities for managing the variables in your workspace and importing and exporting data. It also includes tools for developing, managing, debugging, and profiling M-files, MATLAB's applications.

6.1.3 Handle Graphics.

This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete Graphical User Interfaces on your MATLAB applications.

6.1.4 The MATLAB mathematical function library.

This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigen values, Bessel functions, and fast Fourier transforms.

6.1.5 The MATLAB Application Program Interface (API).

This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

CHAPTER 7

TEST CASES

7.1 Testing

7.1.1 White Box Testing

White-box testing is a method of testing software that tests internal structures or workings of an application. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs.

7.1.2 Black Box Testing

Black-box testing is a method of software testing that tests the functionality of an application. The tester is only aware of what the software is supposed to do, but not how i.e. when he enters a certain input, he gets a certain output; without being aware of how the output was produced in the first place.

7.2 Test Cases

7.2.1 Test Case I: Size of image

While designing the test cases for the application, the main point was the size of image. The image captured through camera must be of size 256x256 and hence it is necessary to capture the image to a more specific extent. Facial specifications need to be distinct and accurate.

7.2.2 Test Case II: Accuracy

It provides better results at the cost of complexity. More the no. of block the image is divided into, more the number of histograms generated and more accurate the results. Also, more space and time required.

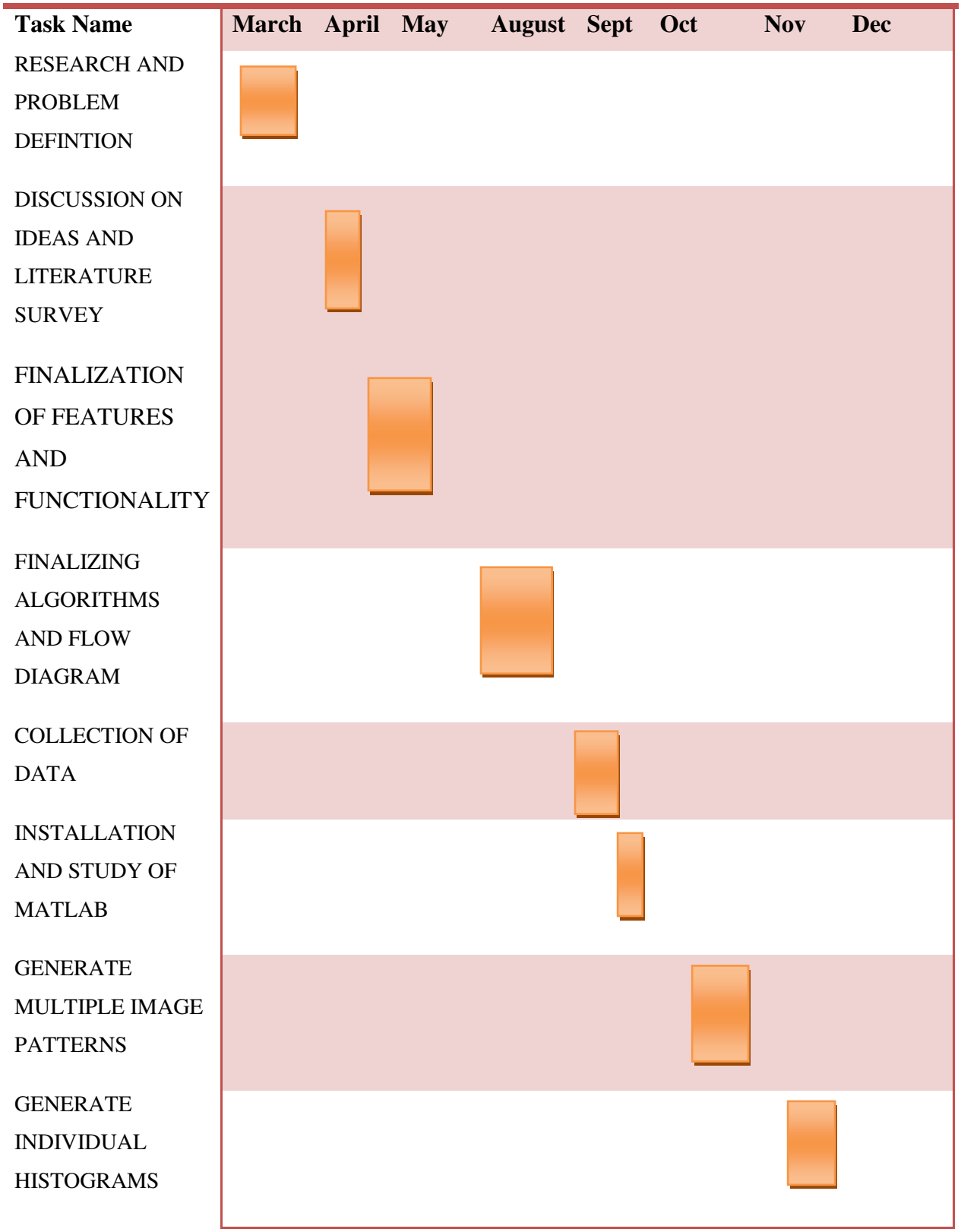
CHAPTER 8

PROJECT TIMELINE

		Task Name	Duartion	Start	Finish
1	✓	RESEARCH AND PROBLEM DEFINTION	25 DAYS	Mon 04-03-2013	Fri 29-03-2013
2	✓	DISCUSSION ON IDEAS AND LITERATURE SURVEY	12 DAYS	Fri 29-03-2013	Wed 10-04-2013
3	✓	FINALIZATION OF FEATURES AND FUNCTIONALITY	20 DAYS	Tue 23-04-2013	Sun 12-05-2013
4	✓	FINALIZING ALGORITHMS AND FLOW DIAGRAM	25 DAYS	Fri 26-07-2013	Tue 20-08-2013
5	✓	COLLECTION OF DATA	15 DAYS	Sun 25-08-2013	Sun 08-09-2013
6	✓	INSTALLATION AND STUDY OF MATLAB	7 DAYS	Wed 11-09-2013	Wed 18-09-2013
7	✓	GENERATE MULTIPLE FACE PATTERNS	20 DAYS	Fri 20-09-2013	Fri 10-10-2013
8	✓	GENERATE INDIVIDUAL HISTOGRAMS	20 DAYS	Tue 14-01-2014	Mon 03-02-2014
9	✓	APPLY MSP-LBP OPERATOR AND HISTOGRAM MATCHING	30 DAYS	Wed 05-02-2014	Fri 07-03-2014
10	✓	COMBINE RESULTS	7 DAYS	Tue 11-03-2014	Tue 18-03-2014
11	✓	EVALUATE EXPRESSION MATCHING	8 DAYS	Wed 19-03-2014	Wed 26-03-2014
12	✓	TESTING	7 DAYS	Thu 27-03-2014	Thu 03-04-2014
13	✓	DOCUMENTATION	5 DAYS	Fri 04-04-2014	Wed 09-04-2014
14	✓	PRESENTATION	5 DAYS	Thu 10-04-2014	Tue 15-04-2014

Table 8.1 Timeline Information

2013


Task Name
March
April
May
August
Sept
Oct
Nov
Dec
RESEARCH AND PROBLEM DEFINTION
DISCUSSION ON IDEAS AND LITERATURE SURVEY
FINALIZATION OF FEATURES AND FUNCTIONALITY
FINALIZING ALGORITHMS AND FLOW DIAGRAM
COLLECTION OF DATA
INSTALLATION AND STUDY OF MATLAB
GENERATE MULTIPLE IMAGE PATTERNS
GENERATE INDIVIDUAL HISTOGRAMS

2014

Task Name	Jan	Feb	March	April
APPLY MSP-LBP OPERATOR AND PERFORM HISTOGRAM MATCHING				
COMBINE RESULTS AND EVALUATE EXPRESSIONS				
TESTING				
DOCUMENTATION				
PRESENTATION				

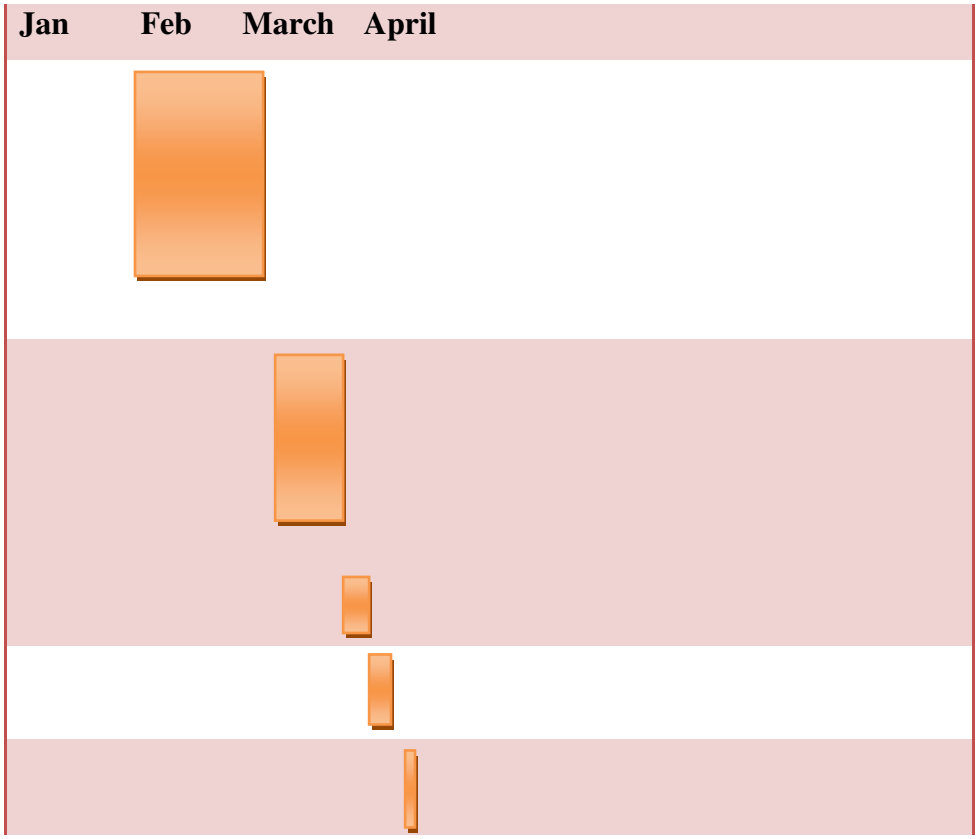


Table 8.2 Timeline Chart

CHAPTER 9

TASK DISTRIBUTION

Sr. no	Completion (%)	Task Name	Duration	Start	Finish	Team Members	Predecessor
1	100	RESEARCH AND PROBLEM DEFINTION	25 DAYS	04/03/13	29/03/13	Sarwanlal Devra Saurabh Bora Karan Chauhan	
2	100	DISCUSSION ON IDEAS AND LITERATURE SURVEY	12 DAYS	29/03/13	10/04/13	Sarwanlal Devra Saurabh Bora Karan Chauhan	1
3	100	FINALIZATION OF FEATURES AND FUNCTIONALITY	20 DAYS	23/04/13	12/05/13	Sarwanlal Devra Saurabh Bora Karan Chauhan	2
4	100	FINALIZING ALGORITHMS AND FLOW DIAGRAM	25 DAYS	26/07/13	20/08/13	Sarwanlal Devra Saurabh Bora Karan Chauhan	3
5	100	COLLECTION OF DATA	15 DAYS	25/08/13	8/09/13	Sarwanlal Devra Saurabh Bora Karan Chauhan	4

6	100	INSTALLATION AND STUDY OF MATLAB	7 DAYS	11/09/13	18/09/13	Sarwanlal Devra Saurabh Bora Karan Chauhan	5
7	100	GENERATE MULTIPLE IMAGE PATTERNS	20 DAYS	20/09/13	10/10/13	Sarwanlal Devra Saurabh Bora Karan Chauhan	6
8	100	GENERATE INDIVIDUAL HISTOGRAMS	20 DAYS	14/01/14	03/02/14	Sarwanlal Devra Saurabh Bora Karan Chauhan	7
9	100	APPLY MSP-LBP OPERATOR TO INDIVIDUAL HISTOGRAMS AND PERFORM MATCHING	30 DAYS	05/02/14	07/03/14	Sarwanlal Devra Saurabh Bora Karan Chauhan	8
10	100	COMBINE RESULTS	7 DAYS	11/03/14	18/03/14	Sarwanlal Devra Saurabh Bora Karan Chauhan	9
11	100	EVALUATE EXPRESSION MATCHING	8 DAYS	19/03/14	26/03/14	Sarwanlal Devra Saurabh Bora Karan Chauhan	10
12	100	TESTING	7 DAYS	27/03/14	03/04/14	Sarwanlal Devra Saurabh Bora	11

						Karan Chauhan	
13	100	DOCUMENTATION	5 DAYS	04/04/14	09/04/14	Sarwanlal Devra Saurabh Bora Karan Chauhan	12
14	100	PRESENTATION	5 DAY	10/04/14	15/04/14	Sarwanlal Devra Saurabh Bora Karan Chauhan	12

Table 9.1 Task Distribution

CHAPTER 10

CONCLUSION AND FUTURE WORK

In this paper, we introduced a novel architecture of the future interactive TV and proposed a real-time face analysis system for providing automatic viewer-personalized services. To enable delivery of personalized services such as a personalized electronic program guide more user-friendly, the future personalized TV should interact with us more like humans. A key component of that interaction will be its abilities to detect, track, and recognize our faces and even understand our internal emotional states that are expressed by facial expressions. To achieve this goal, in this paper, we presented a novel and universal algorithm for multi-class pattern classification: face detection, face recognition, and facial expression recognition, which uses a strong classifier based on the simple and discriminative Msp LBP features that are trained by the proposed neural learning algorithm. To verify the feasibility of the proposed approach, three challenging experiments, in terms of face detection, face recognition, and facial expression recognition, were carried out on the widely used face databases.

Face detection on the MIT-CBCL, Yale and MIT+CMU test sets, face recognition on our laboratory face database and the Yale Face Database B, and facial expression recognition on the JAFFE face database have shown that the Msp LBP features can capture more information about image texture and structure than both the traditional Haar-like features and raw image pixel intensities, and so are more distinctive, and classifier with only a small number of discriminative Msp LBP features yields more discrimination power and better classification performance.

Extensive experimental results have shown that the proposed face analysis system including face detection, face recognition, and facial expression recognition provides outstanding performance with high recognition rates and fast processing speed of over 15 frames per second. It is efficient enough to be applied to the personalized real-time interactive TV.

In future, complex Msp-LBP operations and their combinations can be explored to improve robustness of proposed future interactive T.V. approach to get near to 100% results.

CHAPTER 11

REFERENCES

1. A. Pentland and T. Choudhury, "Face recognition for smart environments," IEEE Computer, vol. 33, no. 2, pp. 50-55, Feb. 2000. T. Isobe, M. Fujiwara.

2. H. Kaneta, "Development and features of a TV navigation system," IEEE Trans. Consumer Electronics, vol. 50, no. 1, pp. 393-399, Nov. 2003.

3. F. Zuo and P. H. N. de with, "Real-time embedded face recognition for smart home," IEEE Trans. Consumer Electronics, vol. 51, no. 1, pp. 183-190, Feb. 2005.

4. His – Chieh Lee, Chia –Ying Wu and Tzu-Miao Lin "Facial Expression Recognition Using Image Processing Techniques and Neural Networks"