Real Time Face Detection System

Nafi Rushdi Nafi Hamdan
Master in Computer Science
Advisor: Dr. Jeffrey Duffany
Electrical & Computer Engineering and Computer Science Department
Polytechnic University of Puerto Rico

Abstract *— This paper presents details of a prototype for research project on face detection systems. The prototype uses Emgu CV cross platform .Net wrapper with the Intel OpenCV image processing library and C# .Net. The library allows capture and processing of image from a camera in real time. Principal Component Analysis (PCA) with Eigen face is used on the technology. Details of the project such as code samples, architecture diagram, system design ER diagrams, use cases, activity diagrams, data flow diagrams, class diagram, and sequence diagram are presented. Details of the testing, and feasibility are also given.*

Key Terms *— Emgu CV, Face Detection, Intel OpenCV, Principal Component Analysis.*

INTRODUCTION

Face recognition is a computer based system to recognize faces by scanning the face in real-time using a camera, and matching the image to stored images from the database [1]. Data validation, matching facial attributes is the key to this technology, and the objective is to find the identity of the person, unknown or unknown, under the scanner [2]. The technology of face recognition forms an important part of bio-metric face recognition systems, used in offices, sensitive installations, to counter and detect terror activities, in crime fighting, and in many other security systems [3].

The system proposed in this paper basic Dotnet API‘s to interact and acquire the output of a local camera used for scanning such as a webcam or other attached cameras. The output from the camera is used to recognize faces in real time. The proposed system offers many advantages over existing systems [4]. Some of the existing systems such as human guards, security, locks and keys, finger print scanning, surveillance cameras, magnetic swipe cards, etc., are not very reliable. They can be hacked or misused and allow identity theft. This paper, presents findings from a research project on face recognition system [2].

LITERATURE REVIEW

Initial attempts at creating facial recognition systems were by Woody Bldesoe and others and these attempts were very crude since computer science and applications were very rudimentary. Problems such as variability of head rotation, facial expression lighting intensity, aging, and other factors, limited the scope of these efforts. Later, experiments were conducted by the University of Bochum, University of Southern California, Massachusetts Institute of Technology, and others were more promising. They were used by some banks for identification of customers. In the next development phases, high-resolution face images were developed along with 3D and iris scans. These applications were much better than humans in recognizing faces and they were more powerful [5].

Over the decades, the rate of error in detecting face has reduced. Systems are now available were face recognition algorithms are used to extract features of the face, such as shape of nose, relative position and shape of eyes, mouth, lips, skin and texture recognition, thermal cameras, and other techniques. In general, face recognition algorithms are of two types, geometric, and photometric. Geometric approach looks at landmarks of the face, while statistical converts the face into values that are matched with templates to remove variations. The latest Apple X phone uses advanced face recognition system. The main problem of these systems is the speed of cross-referencing and matching thousands of records in the database, with the least number of errors [6].

Several airports in Australia, Ottawa, Panama use an airport-wide surveillance system with face recognition systems, designed to spot and identify potential terrorists and other unwanted elements. These systems are still subject to problems of lighting, expression, pose, noise, additional features such as bear and glasses. Some concerns and criticism are raised about the effectiveness of these systems. It appears the systems installed in London have not recognized even one offender. However, crime rate is down by 34%, and the reason is that the fear of advanced face recognition cameras in public places, acts as a major deterrent. There are some fears of a total surveillance society, where privacy of people is under threat [7].

PROPOSED SYSTEM

The project uses Emgu CV cross platform .Net wrapper with the Intel OpenCV image processing library and C# .Net. The library allows capture and processing of image from a camera in real time. Principal Component Analysis (PCA) with Eigen face is used on the technology. The system is made of two phases, enrolment phase and the authentication / verification phase. Enrolment phase facilitates capture and processing of user biometric data for use by system in subsequent authentication operations. During this phase, users are enrolled, features and characteristics of the face are extracted with a process called feature extraction, and a model is created in a template [8].

The modeling process is important and the subjects face is mapped to construct the face image. After the model is created and the signals are process, a quality check is run on the templates extracted from the face. If the required amount of features is extracted, then the template is stored in the facial database. Facial recognition is done with complex algorithms, and with mathematical and metrical techniques. These are used to develop a raster model, a digital format of the face. When a person's face is to be matched, then the image is processed and compared at a pixel level for faster and accurate results. Running these comparisons and algorithms requires machines with large computational power, since algorithms, functions, and routines are to be run on a large number of templates [9]. Figure 1 illustrates the system design.

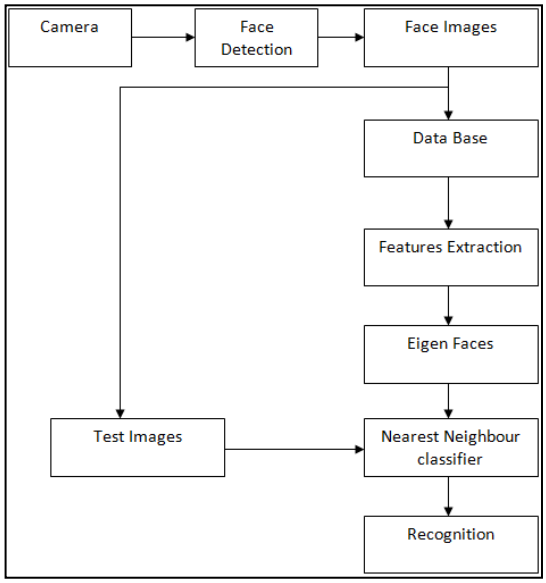


Figure 1
System Design

As seen in Figure 1, the system consists of different components and processes handlers. Camera, either a web cam or any other camera is used to capture images of the subjects face. This is used for face detection, and the face images are stored in the database.

The structured database allows features extraction with Eigen face values, which are extracted from the images. A test can be run to find the effectiveness of the system by running tests with the nearest neighbor classifier. When the test is run, then face recognition is carried out. Even if a match is not found, an alert is sent to the system. There can be instances of false-positives and false-negatives, and these are indicated as alerts to the users. Depending on the circumstances, corrective action can be taken [10].

Advantages of the Proposed System

The following advantages of this project are evident. Extraction of similar facial areas is carried out, and identification and authentication is based on individual facial features. Easy adaptation with

existing IT systems with flexible integration for many types of video monitoring systems. A 1: n matching is provided and the system supports diverse graphic and video formats as well as live cameras. The system uses an open CV based multiple-matching face detection and combination of eye-zone extraction and facial recognition. It uses recognition based on neural network technology, with short processing time, high recognition rate, and recognition regardless of vantage point and facial changes. The system is accurate with fast identification, provides high usability and security, and has a user friendly design.

Data Flow Diagram

Data Flow Diagram (DFD) is used to analyze and describe data movement through a diagram. This is an important tool used to develop other components. Logical data flow diagram is used to represent data from input to output in a logical and independent manner from physical components of the system. A physical data flow diagram presents the movements of data between departments and workstations. Each component is given a descriptive name. The process is later given a number, used for identification. DFD is carried out at different levels. Each process in the lower level is broken down into a detailed DFD at the next level. The upper level is the content diagram and it has a single process bit that is useful in understanding the system. The process in the context level diagram is later expanded into other processes at the first DFD level [2].

The reason for exploding the process into sub process is that the levels of different details can be understood in greater detail. This explosion is continued until the required amount of detail is obtained. DFD is also known as a bubble chart and it helps to understand the system requirements and in identifying important transformations that lead to a program development of the system design. It thus becomes the initial point for design at the lowest level. A series of bubbles are joined by the data flows in the system [4]. The DFD for the face recognition system is given in Figure 2.

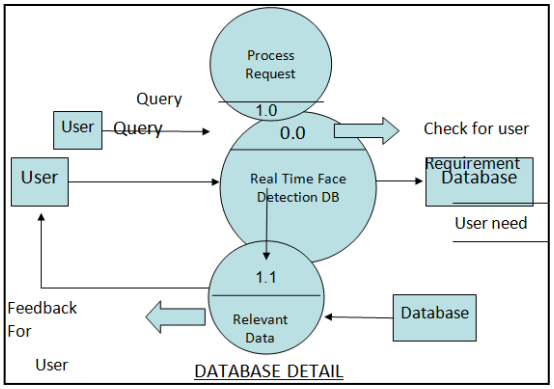


Figure 2
DFD for Face Recognition System

From Figure 2, it is seen that the user sends the query and this is taken as a process request, which is passed to the real time face detection database. The request is sent to the requirement database to check for the user. If the user records exist or if the scan is matched, then the request is processes, a match found and feedback given to the user, with the relevant data. Figure 3 gives the Level 1 illustration of the DFD.

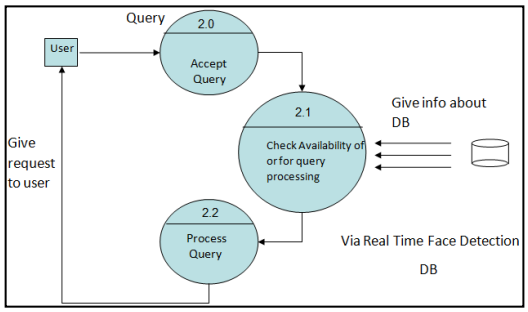


Figure 3
Level 1 DFD for Face Recognition System

As seen in Figure 3, the exploded view of the DFD. User sends a query to the system and it accepts the request. System checks availability of data for query processing and sends the request to the database. Database verifies the records, matches the request with data and send to the system, where the query is processes. This is then sent to the user as a request.

ER Diagram

The entity relation diagram, maps the relations between different entities of the system. Figure 4 presents the ER diagram of the system.

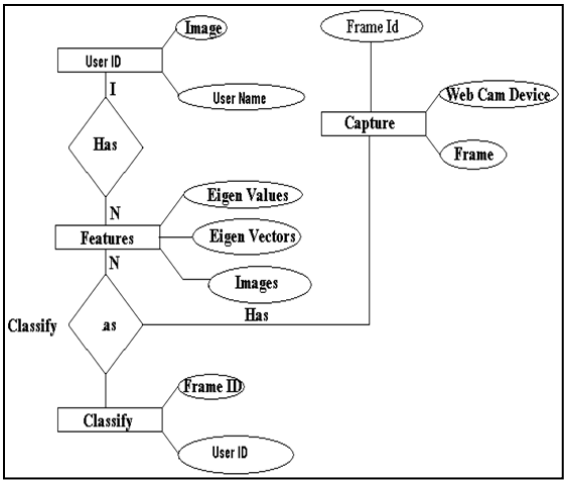


Figure 4
ER Diagram

As seen in Figure 4, the entity User Id has attributes like Image and User Name. The entity User Id is connected to another Entity Features. Features have attributes like Eigen Values, Eigen Vectors and Images. Thus, a User Id can have 1 or many features like Eigen Values, Eigen Vectors and Images.

The features of User Id are connected to another entity Classify which has entities like Frame Id and User Id. Thus, features are classified into Frame Id and User Id. The entity Features is connected to another entity Capture which has attributes Frame Id, Webcam Device and Frame. Thus, Features of the User Id captured has Frame Id, Webcam Device and Frame associated to it.

Use Cases

Two use case diagrams are given in the test case. Figure 5 presents the first use case diagram. The use case is to click a photo using webcam; user has to perform certain activities. User has to initially keep his face in front of the Webcam. The webcam will use face recognition technology to detect the face in front of it and when the face is recognized it will process and save the image in the system. In Use Case 1, User has used a webcam and saved his face Image successfully in the system.

The System is the User in this Case. It manages all security related issues. It recognizes the face of the User who had in the Use Case 1 used a webcam to detect and save his face image. After face matching it will check whether a new face is to be registered or the previous face with existing information needs to be modified. The Image is ultimately saved to the system. Thus, the insertion or modification i.e. updating of an image and its corresponding information in the database is done accordingly.

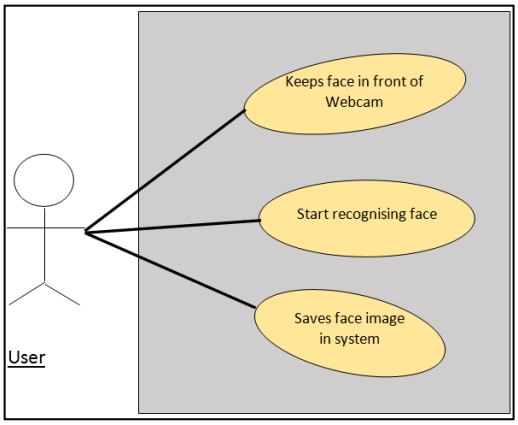


Figure 5
Use Case Diagram 1

Figure 6 illustrates the second use case diagram.

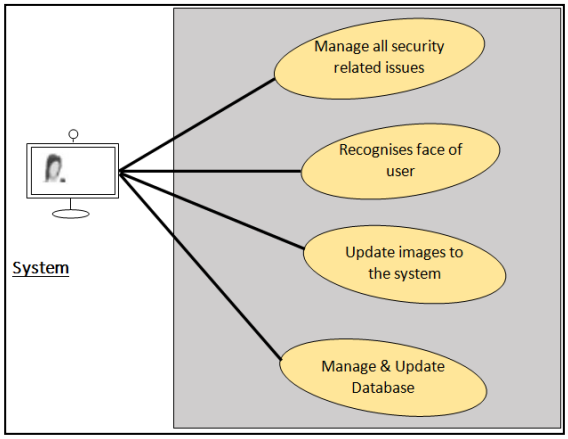


Figure 6
Use Case Diagram 2

There are three actors involved in this Activity Diagram namely - User, Webcam and Features Extract. User uses a webcam to capture Image i.e. his face. The face captured by the webcam is sent to extract certain notable features of the face that can help recognize it. These features are then sent back to the user. The user will then classify the features and send it back to the Features extract through the

webcam. The Features Extract will verify and send the expression of the user which it has recognized back to the user.

Class Diagram

Figure 7 illustrates the class diagram of the system. There are two classes present here namely, Capture and Classification. Class Capture has attributes like Frame No and Bitmaps. It means every Image that is captured is classified by Frame No and their Bitmaps. Class Capture has operations Capture and Save defined. The Image captured thus can be captured again if not matching or it can be saved if it matches. Class Classification has attributes Images and Features. Thus it means every Image that has been captured can be classified as Image and Features of the Image. The operation for Class Classification is Classify.

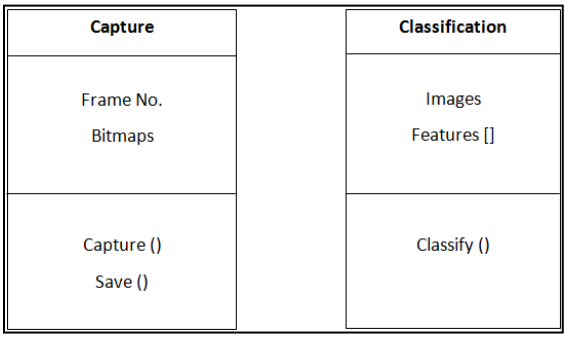


Figure 7
Class Diagram

Sequence Diagram

The sequence diagram illustrates the sequence of operations. Figure 8 illustrates the sequence diagram.

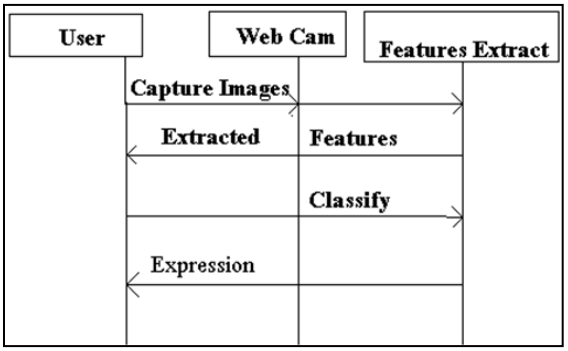


Figure 8
Sequence Diagram

User saves his face image using the webcam. If the face is not recognized and found to be invalid, the user is exited from the system. The process of recognizing the image again is started. If the Image is found to be valid, then the particular menu is displayed for the user. He can choose the options he wants from the menu. The face image which has been matched is then saved to the system and the database is updated with the latest information of the user.

Activity Diagram

Figure 9 shows the activity diagram. The activity begins the user who receives a request with an invalid face. The request is processed through face recognition. If the face is invalid, the user exits the system. If the face is valid, then a menu is displayed and the image is saved.

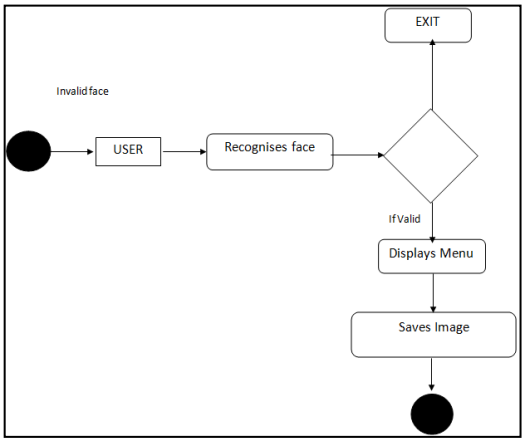


Figure 9
Activity Diagram

FACE RECOGNITION APP

The Face Recognition Prototype Software App is based on the Emgu CV cross platform. Code for the app was obtained from open source repository. Emgu CV has two layers, layer 1 or the basic layer and layer 2 or the second layer. Layer 1 carries the structure, enumeration,, functions mappings that show the ones on OpenCV. Layer 2 has the classes that use features of .Net. Image is specified by the generic parameters such as color and depth, and an 8bit unsigned Grayscale image is done in Emgu CV by calling a function. Image pixels are stored in a 3D

array. When a new image is stored, an identification number, and the pixel values are stored as numbers. When a face is scanned, the image is converted into pixel values and matched with the stored records.

Figure 9
Welcome Screen

If there is a match, then the person is accepted, if there is a mismatch with the stored values, then the request is denied [11].

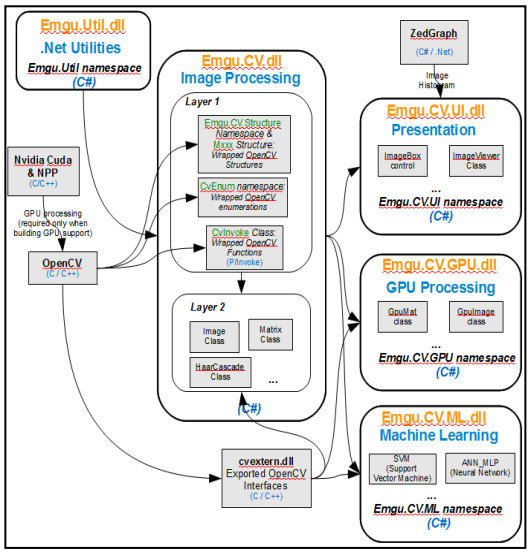


Figure 10
System Architecture [11]

Sample codes, obtained from Emgu CV are given in Figure 11. The code is compiled and run as a sub-routines [11].

On the Project Executable folder called “Faces” was created and images were stored as BMP formats. Names of the images were face1.bmp, face2.bmp, etc. and the names were stored in a file “TrainedLabels.txt”, in the same folder. It is a flat file with CSV structure and data can be obtained from the folder “bin/x86/Debug/Faces/”. A Euclidean algorithm was used to compare the images and to select the image that are match the face on the camera [11]. Next images show more sample codes.

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("FaceRecognitionSecuritySystem")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("FaceRecognitionSecuritySystem")]
[assembly: AssemblyCopyright("Copyright ©  2018")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("56a8a119-710f-4df8-bd83-6223b41f5e3a")]
```

Figure 11
Sample Code 1

```
using System;

using System.Diagnostics;
using Emgu.CV.Structure;
using Emgu.CV;

namespace FaceRecognitionSecuritySystem
{
    /// <summary>
    /// An object recognizer using PCA (Principle Components Analysis)
    /// </summary>
    [Serializable]
    public class EigenObjectRecognizer
    {
        private Image<Gray, Single>[] _eigenImages;
        private Image<Gray, Single> _avgImage;
        private Matrix<float>[] _eigenValues;
        private string[] _labels;
        private double _eigenDistanceThreshold;

        /// <summary>
        /// Get the eigen vectors that form the eigen space
        /// </summary>
```

Figure 12
Sample Code 2

```
using System;

using System.Windows.Forms;

namespace FaceRecognitionSecuritySystem
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}
```

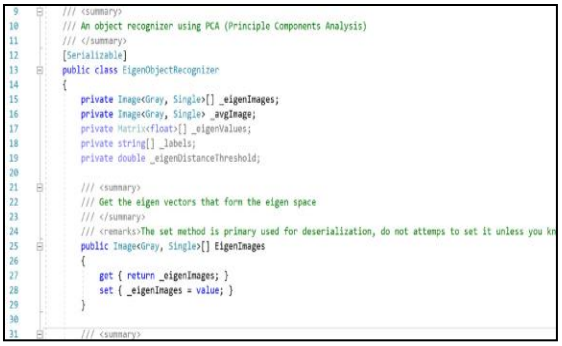
Figure 13
Sample Code 3

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <startup>

    <supportedRuntime version="v2.0.50727"/></startup>
</configuration>
```

Figure 14
Sample Code 4

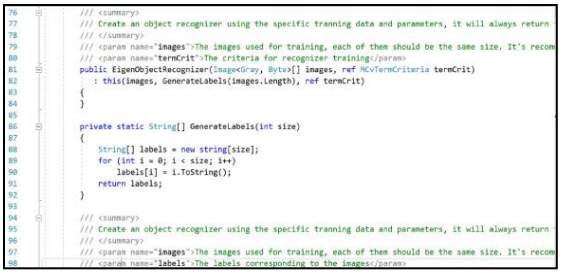
```
using System;


/// <summary>
///   A strongly-typed resource class, for looking up localized strings, etc.
/// </summary>
// This class was auto-generated by the StronglyTypedResourceBuilder
// class via a tool like ResGen or Visual Studio.
// To add or remove a member, edit your .ResX file then rerun ResGen
// with the /str option, or rebuild your VS project.
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilde
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
internal class Resources {

    private static global::System.Resources.ResourceManager resourceMan;

    private static global::System.Globalization.CultureInfo resourceCulture;

    [global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance", "CA1811:Avoi
    internal Resources() {
    }
```

Figure 15
Sample Code 5



```
/// <summary>
/// An object recognizer using PCA (Principle Components Analysis)
/// </summary>
[Serializable]
public class EigenObjectRecognizer
{
    private Image<Gray, Single>[] _eigenImages;
    private Image<Gray, Single> _avgImage;
    private Matrix<float>[] _eigenValues;
    private string[] _labels;
    private double _eigenDistanceThreshold;

    /// <summary>
    /// Get the eigen vectors that form the eigen space
    /// </summary>
    /// <remarks>The set method is primary used for deserialization, do not attemps to set it unless you kn
    public Image<Gray, Single>[] EigenImages
    {
        get { return _eigenImages; }
        set { _eigenImages = value; }
    }

    /// <summary>
```

Figure 16
Sample Code 6



```
/// <summary>
/// Create an object recognizer using the specific tranning data and parameters, it will always return
/// </summary>
/// <param name="images">The images used for training, each of them should be the same size. It's recom
/// <param name="termCrit">The criteria for recognizer training</param>
public EigenObjectRecognizer(Image<Gray, Byte>[] images, ref MCvTermCriteria termCrit)
   : this(images, GenerateLabels(images.Length), ref termCrit)
{
}

private static String[] GenerateLabels(int size)
{
    String[] labels = new string[size];
    for (int i = 0; i < size; i++)
        labels[i] = i.ToString();
    return labels;
}

/// <summary>
/// Create an object recognizer using the specific tranning data and parameters, it will always return
/// </summary>
/// <param name="images">The images used for training, each of them should be the same size. It's recom
/// <param name="labels">The labels corresponding to the images</param>
```

Figure 17
Sample Code 7

PROJECT IMPLEMENTATION

The Project is loaded in Visual Studio 2008. Visual Studio for Design and coding of project was done with this system. All databases were created and stored in SQL Server 2005. Tables were created and query run to store data or record the project. The software and hardware used in the research are detailed as follows:

Hardware Requirement: i3 Processor Based Computer, 1GB-Ram, 5 GB Hard Disk.

Software Requirement: Windows XP, Windows 7 (ultimate & enterprise), Visual studio 2008, SQL Server 2005.

Operating System: Windows XP, 7(ultimate & enterprise).

Languages: Asp.Net with C# (.Net 2008)

Database System: MS-SQL Server 2005

Documentation Tool: MS - Word 2010

These are the basic hardware and software requirements and they can be enhanced if required.

PROJECT TESTING

Since the project would be implemented on a large scale, testing was done to trace bugs and defects. Tests were run to identity the desired output for all types of inputs. This approach is essential for a successful project. System testing was done to find if user requirements were met. The code for the new system was written with ASP.NET and with C#, which was also used as the interface for front-end designing. The system was tested with users and all applications were tested for different use cases. Some defects and bugs were found in some of the components, and these were corrected before implementation. Form and data flow was as per the requirements.

The testing done here was System Testing checking whether the user requirements were satisfied. The code for the new system has been written completely using ASP .NET with C# as the coding language, C# as the interface for front-end designing. The new system has been tested well with the help of the users and all the applications have been verified from every nook and corner of the user. Figure 18 illustrates the different levels of testing used.

As seen from Figure 18, client's needs were met by acceptance testing, while requirements were met by system testing. Design specifications were met by integration testing, and coding specifications were met by unit testing.

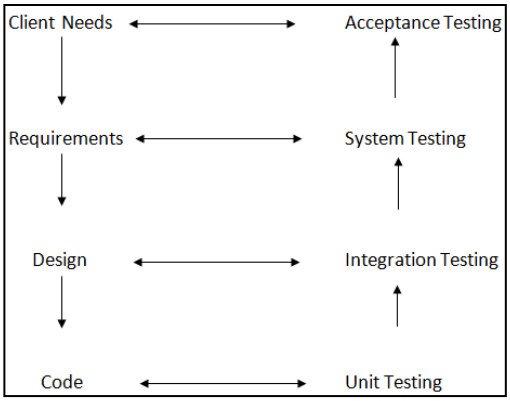


Figure 18
Levels of Testing

FEASIBILITY

The project was evaluated for Technical Feasibility, Economic Feasibility, and Operational Feasibility. For technical feasibility, the system was verified to see if the components and systems were feasible to develop and implement. The project is technically feasible because, all the technology needed for our project is readily available. For economic feasibility, the project is feasible since all the hardware, cameras, and software are available and not expensive to procure. Cost of development is less when compared to financial benefits of the application. For operational feasibility, different operational factors such as manpower, time, and equipment requirements were examined. The project is operationally feasible as the time requirements and personnel requirements are met. The project was completed by for team members in three months.

CONCLUSIONS

The paper detailed the research on face recognition system and developed a prototype. The prototype was run and used for sample face recognition. Important aspects of the project such as the details of the proposed system, various assets such as system design, ER diagrams, use case diagrams, activity diagrams, etc. were presented. Methods of testing and feasibility were also presented. The conclusion is that the project is feasible, useful, and it can be implemented.

Sequence Diagram

The future work needs to include a larger database and image sample, where tests are run to identify unknown faces from public databases.

REFERENCES

[1] Y. P. Chen, C. H. Liu, K. Y. Chou, and S. Y. Wang, "Real-time and low-memory multi-face detection system design based on naive Bayes classifier using FPGA," in *Automatic Control Conference (CACS), 2016 International*, IEEE, 2016, pp. 7-12.

[2] C. N. Kumar Ravi, & A. Bindu, "An efficient skin illumination compensation model for efficient face detection," in *IEEE Industrial Electronics, IECON 2006-32nd Annual Conference*, IEEE, Paris, France, 2006, pp. 3444-3449.

[3] M. Bappaditya, "Face recognition: Perspectives from the real world," in *Control, Automation, Robotics and Vision (ICARCV)*, 2016 14th International Conference on, IEEE, 2016, pp. 1-5.

[4] C. O. Manlises, J. M. Martinez, J. L. Belenzo, C. K. Perez, & M. Kristine, T. A. Postrero., "Real-time integrated CCTV using face and pedestrian detection image processing algorithm for automatic traffic light transition," in *Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, 2015 International Conference on, IEEE, 2015, pp. 1-4.

[5] S. Yang, P. Luo, C. Loy, & X. Tang, "From facial parts responses to face detection: A deep learning approach," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3676-3684.

[6] S. Yang, P. Luo, C. Loy, & X. Tang, "Wider face: A face detection benchmark," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5525-5533.

[7] S. Yang, P. Luo, C. Loy, & X. Tang,"Faceness-Net: Face detection through deep facial part responses," in *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[8] Z. Stefanos, C. Zhang, & Z. Zhang, "A survey on face detection in the wild: past, present and future," in *Computer Vision and Image Understanding*, 138, 2015, pp. 1-24.

[9] Z. Kaipeng, Z. Zhang, Z. Li, & Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," in *IEEE Signal Processing Letters*, 23, no. 10, 2016, pp. 1499-1503.

[10] Z. Chenchen, Y. Zheng, K. Luu, & M. Savvides, "CMS-RCNN: contextual multi-scale region-based CNN for

unconstrained face detection," in *Deep Learning for Biometrics*, Springer, Cham, 2017, pp. 57-79.

[11] Emgu CV. (2018, April 9). *Emgu CV Open CV 3x* [Online]. Available: http://www.emgu.com/wiki/index.php/Main_Page.