ECO-COMPUTING USING FACIAL RECOGNITION

101204

Varun Agarwal

Dr Pradeep Kumar Gupta

May 2014

Submitted in partial fulfilment of the Degree of

Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

& INFORMATION TECHNOLOGY

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT

CERTIFICATE

This is to certify that the work titled **Eco Computing using Facial Recognition** submitted by **Varun Agarwal** in partial fulfilment for the award of degree of Bachelor of technology of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor:

Name of Supervisor : Dr. Pradeep Kumar Gupta

Designation : Assistant Professor (Senior Grade)

Date :

ACKNOWLEDGEMENT

I would like to express my gratitude to all those who gave us the possibility to complete this project. I want to thank the Department of CSE & IT in JUIT for giving us the permission to commence this project in the first instance, to do the necessary research work.

I am deeply indebted to my project guide Dr. Pradeep Kumar Gupta, whose help, stimulating suggestions and encouragement helped me in all the time of research on this project. I feel motivated and encouraged every time I get his encouragement. For his coherent guidance throughout the tenure of the project, I feel fortunate to be taught by him, who gave me his unwavering support.

We are also grateful to **Mr. Amit Singh** (**CSE Project lab**) for his practical help and guidance

Varun Agarwal

TABLE OF CONTENT

ABSTRACT

The aim of this project is to develop a framework which can help reduce power consumption by putting the system into suspended state when it is idle. The application is proposed to run as a background service which will get activated when the system reaches a low threshold of computer activity. Using facial detection and recognition techniques the presence of the user on the system is inferred and the system is instructed accordingly. Using the idea of Eigen faces facial recognition has been implemented .A service has been installed into the operating system for integration of our code and to pass instructions to the system during the various phases. This service is compatible in all the existing system with windows 7 and 8 operating systems.

PROBLEM STATEMENT

To develop a power saving application using the functionalities of facial recognition. The basic aim of the project is to develop an application which runs in the background and while saving power at the same time doesn't affect user experience. A windows service running in the background launches all the required files as well as gives the necessary instructions to the system for successful deployment of the application.

MOTIVATION

Throughout the 20th and 21st century the science and technology have developed rapidly and focus has shifted gradually from personal computers to laptops. There are hundreds and thousands of laptops in any corporate office working day and night with the majority of them just sitting idle generating a lot of heat and consuming the ever decreasing natural resources. The heat generated is not only a source of global warming but for the corporates too it is a very big issue as they need to set up air conditioning system which adds up to total expenses. Using the concept of facial recognition this application tries to develop a service which would help to reduce the no of idle system and hence help in whatever way possible to help conserve environment and save costs for an organisation.

CHAPTER 1: INTRODUCTION

It is normal human tendency to take a break every few hours and sometimes these coffee breaks which are normally for 5 10 minutes due to some reason get elongated to a big stretch sometimes spanning 1 to 2 hours but often this is unknown. And as a normal person we realize that maybe once or twice a user may put his computer to sleep but with such monotonous life employees tend to ignore such little facts which may not directly affect them but in the long run and which such huge amount of laptops being worked if even a fraction of power can be saved per company as a whole might create a huge difference.

1.1 Carbon Credit

The idea being put forward tries to solve this problem without affecting the user experience as the whole service will run in background without prompting for any input from the user. If used commercially this proposal can be used to generate carbon credits which can be used in a variety of ways .From exchanging these credits for money to using these as barter for national and international agreements Corporate Carbon [**www.corporatecarbon.com.au/**] are creating new ways to cover up losses through innovative carbon management. According to the Gold Standard **[carbontradexchange.com/partners/standards]**, developed by 50 non-government organisations, the Federal Government's Greenhouse Friendly program and the NSW Government's Greenhouse Gas Abatement Scheme a variety of standards have been developed to calculate credits and the best way for carbon offset is tree plantation. The end result of our implementation is supposed to conserve power and it is well known that power consumed is power conserved.

1.2 Applications of facial recognition

There are many areas of life where applications of facial recognition have helped as ranging from ATM's to even attendance in college classrooms. With so much research in place finding out a technique with right proportion of processing and success percentage that can benefit the purpose led to the use of PCA based Facial Recognition system

1.3 API Used

The implementation starts with the user's photos being clicked in the background using Microsoft's Directshow API **[http://msdn.microsoft.com/ens /library/windows/desktop/**

dd375454(v=vs.85). aspx] which would generate user's photograph and story in the repository for facial detection and further use. The Microsoft DirectShow application programming interface (API) is a media-streaming interface made and used specifically for Windows operating system of Microsoft. Using DirectShow, the applications can be intended to perform high-quality video and audio playback or capture.

Matlab's computer Vision System toolbox **[http://www.mathworks.in/products/computer-vision/]** is being used for facial detection and PCA analysis using Eigen vectors is used for facial recognition. All the applications being launched are being automated by windows service. The Matlab programs being executed will be converted to dotnet.dll , this will help to make a universal program for laptops without having to install an extra software which can be a burdensome issue for any organisation.

Many components affect the success of a facial recognition system. Most research attempts are PCA or LDA algorithms,.Recognition rates of various distance measures are compared. It is observed there is an inconsistency of performance for each distance measure across each algorithm and face database. This infers that being able to determine the best distance measure before running the recognition algorithm will make the recognition system more successful. Eigenfaces is the approach proposed by **Turk and Pentland** . In this approach, face images are projected into a lower dimensional space using Principal Component Analysis (PCA). Each image is represented by a vector of weights needed to reconstruct the image. Distance measures are used to compute the difference between two vectors. The CSU Face Identification Evaluation System includes many common distance measures that are used to compute the similarity between two images and these methods are used in the facial recognition.

1.4 MATLAB as a Deployment Tool

The MATLAB Compiler Runtime (MCR) is an independent set of shared libraries that enables the implementation of compiled MATLAB applications or mechanisms on systems that do not have installed version of latest matlab. When used together, MATLAB,MATLAB Compiler, and the MCR empower you to produce and distribute numerical applications or

software components quickly and securely.[**http://www.mathworks.in/products/compiler/mcr/]**

1.5 Expected Result

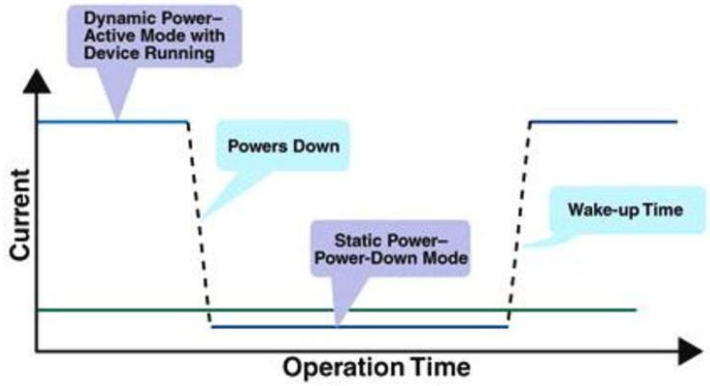


Figure 1.1

As seen in figure 1 it is seen that whenever CPU powers down a lot of power is saced and that is the major aim of our proposal. During the whole operation time the time for which the laptop is put to sleep it can be observed that power is conserved and however small amount it maybe it goes a long way to be a part of a large scale eco computing system.

FIGURE 1.2

FIGURE 1.3

These are the two set of faces that emerge when facial recognition is implemented on training data base where the second set of images are achieved after histogram equalization , normalisation and calculating the average face.

CHAPTER 2: CHALLENGES AND IMAGING CONCEPTS

This chapter focuses on the current confronts in the field of the recognition and some basic notions of image and video applications. These experiments and the imaging concepts are defined in detail in sub sections below respectively.

2.1 Challenges in facial recognition

Over the years, face recognition has added swift performance with the expansion of new approaches and techniques. Due to this growth, the rate of successful face recognition has been augmented to well above 90%. Despite of all this success, all face recognition techniques usually encounter common challenges of image visibility. These are lighting conditions variations, skin differences and face angle deviations. The challenges are explained in detailed manner below.

2.1.1 Difference in Lighting Conditions

The lighting circumstances in which the pictures are shot are not always alike because of the differences in time and place. This example of lighting differences could be because of the pictures taken in the internal room and the pictures taken in broad daylight. Due to these differences, a same person with alike facial expressions would seem differently in diverse pictures. As its outcome, if the individual has solo image store in the databank of face recognition, matching could be tough with the face detection under different lighting conditions.

2.1.2 Skin Colour Variations

Another trial for skin based face recognition scheme is the alteration in the skin colour due to the variance in the races of the individuals. Due to this difference, occasionally the true skin pixels have to be filtered out along with the noise persisting in an image. In count, the untrue skin background/non-background are not wholly filtered out during noise

filtering process. So, it is a rough task to choose the filter which would conceal the entire skin tones of different individuals and kick out false skin noise .

2.1.3 Variation in face angle or Orientation variation

The angle of the humanoid face from camera can be unlike in different situations. A frontal face uncovering algorithm can't work on non-frontal faces existing in the image because the geometry of facial features in frontal view is every time diverse than the geometry of facial features in non-frontal view. This is why orientation variation remains the difficult challenge in face detection system.

2.2 Important concepts of Imaging applications

2.2.1 Image Processing

Image processing is a technique of handling the image values, more accurately, the pixels in case of digital pictures. The purpose of image processing is to modify the input image such that the output image may change parametrically such as in colors and representation. Image processing is the basic part of the face recognition involving digital images. The processing can change the image representation from one color space to another color space. It can also assign different color values to targeted pixels for the purpose of keeping areas of interest in output image. Image processing is also used to increase or decrease image brightness, contrast and other morphological operations .

2.2.2 Color Space

Color space is the representation of image colors in two or more color components. Typical examples of color spaces are RGB, YCbCr and HSI color spaces. In each of these color spaces the color of a pixel at any point in an image is the combination of three color components. These color components vary from 0 to maximum value and this maximum

value depends on the bits per pixel. The different values in the range give different colors from black (0) to white (255) with 8 bits per pixel.

2.2.3 RGB colour Space

RGB color space is the combination of red, green and blue color components. For the 24 bits per pixel, the range of R, G and B varies from 0 to 255. If R, G and B are all 0 then the resulted color will be black. If R, G and B are all 255 then the output color will be white. The concept of the RGB color space is specified in figure 2.b.3. Here the x-axis represents blue color range, y-axis represents green color range and Z-axis represents red color range. As explained above, we can see that black color is represented at the origin and white color is represented at the other corner where red, green and blue are 255 each. Similarly we can have other color values at different corners of cube corresponding to different RGB values.

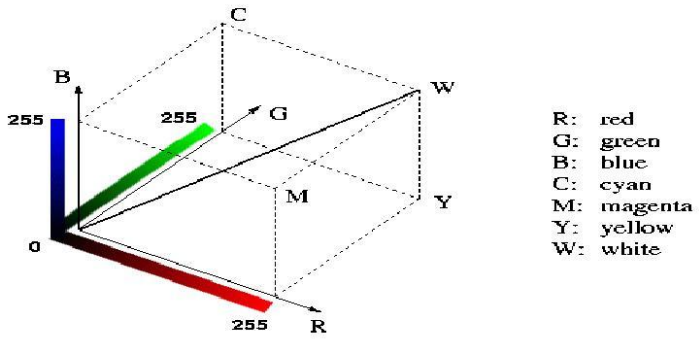


Figure 2.2.3: RGB color model [9]

CHAPTER 3: FRAMEWORK

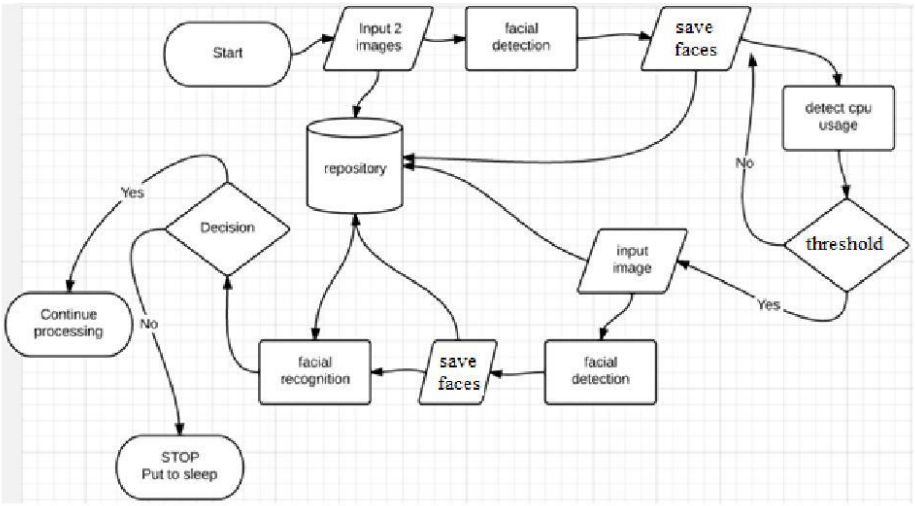


Figure 3.1

3.1 Framework Analysis

According to the figure 3.1

The phase 1 of the application begins with the execution of a batch file placed in a startup folder . The aim of the batch file is to execute an executable file based on Microsoft's Directshow API which provides function to click snapshots with multiple options which can

be customised specially for an user. The snapshots captured by the executable are saved in a directory which have to be used for further processing later.

It is then followed by Phase 2 where windows service launches a matlab application which uses Matlab's computer Vision System toolbox for facial detection .The faces detected are then stored into a directory which are used as a training database for comparison of photos at a later time

This is followed by the most essential part of the program which involves detection of cpu usage as per the threshold set by the user. This can be done in two ways , firstly by using command prompt to return the cpu storage in a file which can be retrieved later and other is to use powerbroadcast status of the windows service. It has various modes to read the current power status of a system , and using query suspend mode it is detected if the computer is ready or needs to be put to sleep

The next phase involves implementation of steps in phase 1 and 2 again as per the flowchart. The phase involves facial recognition amongst the two set of faced detected. This algorithm uses PCA analysis to perform this function. In the next phase instructions are given to the system for execution. These instructions are based on the result, if the facial recognition is successful the application returns to its second phase otherwise data is saved and computer is put to sleep

3.2 TEST BED

Microsoft Visual Studio	2012 Ultimate
Microsoft Windows 7	Home Basic Service Pack 1
Processor	Intel® Core™ i3 CPU M 380 @2.53 GHz
Matlab	2013b
API	Microsoft Directshow API
Matlab API	Computer Vision System Toolbox
System type	64-bit Operating system
Installed memory(RAM)	3.00 GB

CHAPTER 4: CODE ANALYSIS AND IMPLEMENTATION

4.1 Algorithm

STEP 1: Snapshot Capture

P_1 → Clicked picture 1

P_2 → Clicked picture 2

P_3 , P_4 →Clicked picture 3&4

T → time gap

Cd →Directory for training database

Cd1 → Directory for test database

STEP 2: Launch Windows Service Myservice

STEP 3: Execute batch file cam.exe [C++ program using DirectShow API]

P_1 ← Input 1 ┐
P_2 ← Input 2 ┘ Photos of user clicked using laptop camera

P_1 and P_2 saved in cd with a time gap t.

STEP 4: LAUNCH FACIAL DETECTION SYSTEM

Launch Matlab.exe

Use Computer Vision System Toolbox

Inputs:

Output:

Replace original pictures with detected facial images in the given directory (cd) in the repository

STEP 5: CHECK CPU USAGE

While(true)
Start while
Start cmd
Wmi cpu usage –c
If(cpu usage =0)
Start if
Repeat steps 3 & 4
Goto step 6
Else goto step 5
End if
End while

STEP 6: RUN RECOGNITION

Use windows service
Start Matlab.exe
Launch code[PCA Analysis]
Inputs →
If ((P_2= P_2 or P_2) (P_2 =P_1 or P_2))
Start if
Ignore (Continue processing)
Start else
Put computer to sleep
End else
End if

4.2 Code :

CODE

WINDOWS API: service1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Linq;
using System.ServiceProcess;
using System.Text;
using System.Threading.Tasks;

namespace WindowsService2
{
   public partial class Service1 : ServiceBase
   {
      public Service1()
      {
         InitializeComponent();
         if (!System.Diagnostics.EventLog.SourceExists("WindowsService2"))
```

```
        {
            System.Diagnostics.EventLog.CreateEventSource(
                "MySource", "MyNewLog");
        }
        eventLog1.Source = "MySource";
        eventLog1.Log = "MyNewLog";
    }

    protected override void OnStart(string[] args)
    {
        eventLog1.WriteEntry("In OnStart");
    }

    public bool CanHandlePowerEvent {
            get { return can_handle_power_event; }
            set {
                if (QuerySuspend)
{
                    ProcessStartInfo start = new ProcessStartInfo();

start.FileName = eco.exe;
```

```
// Do you want to show a console window?
start.WindowStyle = ProcessWindowStyle.Hidden;
start.CreateNoWindow = true;

// Run the external process & wait for it to finish
using (Process proc = Process.Start(start))
{
    proc.WaitForExit();
            // Retrieve the app's exit code
    exitCode = proc.ExitCode;
}} }

      //protected override void OnSessionChange(SessionChangeDescription
changeDescription)
      //{
      //    base.OnSessionChange(changeDescription);
      //}

      protected override void OnStop()
      {
         eventLog1.WriteEntry("stopped succesfully");
      }
```

```
        private void eventLog1_EntryWritten(object sender, EntryWrittenEventArgs e)
        {       }
    }
}
```

WINDOWS API:projectinstaller.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Configuration.Install;
using System.Linq;
using System.Threading.Tasks;

namespace WindowsService2
{
    [RunInstaller(true)]
    public partial class ProjectInstaller : System.Configuration.Install.Installer
    {
        public ProjectInstaller()
        {
            InitializeComponent();
```

```
    }

    private void serviceInstaller1_AfterInstall(object sender, InstallEventArgs e)
    {
    }  }}
```

Camera aaplication using directshow api

```
// DirectShow header file
#include <dshow.h>

// This is a workaround for the missing header
// file qedit.h which seems to be absent from the
// Windows SDK versions 7.0 and 7.1.
// To use the items defined in this dll, the
// DexterLib namespace must be specified.
// The items in question are:
//
//              DexterLib::_AMMediaType
//              DexterLib::ISampleGrabber
//              DexterLib::IID_ISampleGrabber
//
#import "qedit.dll" raw_interfaces_only named_guids
EXTERN_C const CLSID CLSID_NullRenderer;
```

```
EXTERN_C const CLSID CLSID_SampleGrabber;


// DirectShow objects

HRESULT hr;

ICreateDevEnum *pDevEnum = NULL;

IEnumMoniker *pEnum = NULL;

IMoniker *pMoniker = NULL;

IPropertyBag *pPropBag = NULL;

IGraphBuilder *pGraph = NULL;

ICaptureGraphBuilder2 *pBuilder = NULL;

IBaseFilter *pCap = NULL;

IBaseFilter *pSampleGrabberFilter = NULL;

DexterLib::ISampleGrabber *pSampleGrabber = NULL;

IBaseFilter *pNullRenderer = NULL;

IMediaControl *pMediaControl = NULL;

char *pBuffer = NULL;


void exit_message(const char* error_message, int error)

{

        // Print an error message

        fprintf(stderr, error_message);

        fprintf(stderr, "\n");
```

```
        // Clean up DirectShow / COM stuff
        if (pBuffer != NULL) delete[] pBuffer;
        if (pMediaControl != NULL) pMediaControl->Release();
        if (pNullRenderer != NULL) pNullRenderer->Release();
        if (pSampleGrabber != NULL) pSampleGrabber->Release();
a       if (pSampleGrabberFilter != NULL)
                        pSampleGrabberFilter->Release();
        if (pCap != NULL) pCap->Release();
        if (pBuilder != NULL) pBuilder->Release();
        if (pGraph != NULL) pGraph->Release();
        if (pPropBag != NULL) pPropBag->Release();
        if (pMoniker != NULL) pMoniker->Release();
        if (pEnum != NULL) pEnum->Release();
        if (pDevEnum != NULL) pDevEnum->Release();
        CoUninitialize();

        // Exit the program
        exit(error);
}

int main(int argc, char **argv)
```

```
{
        // Capture settings
        int snapshot_delay = 2000;
        int show_preview_window = 0;
        int list_devices = 0;
        int device_number = 1;
        char device_name[100];
        char filename[100];

        // Other variables
        char char_buffer[100];

        // Default device name and output filename
        strcpy(device_name, "");
        strcpy(filename, "image.bmp");

        // Information message
        fprintf(stderr, "\n");
        fprintf(stderr, "CommandCam  Copyright (C) 2012  Ted Burke\n");
    fprintf(stderr, "This program comes with ABSOLUTELY NO WARRANTY;\n");
    fprintf(stderr, "This is free software, and you are welcome to\n");
    fprintf(stderr, "redistribute it under certain conditions;\n");
```

```
fprintf(stderr, "See the GNU General Public License v3,\n");
fprintf(stderr, "<http://www.gnu.org/licenses/gpl.txt>\n");
fprintf(stderr, "\n");
fprintf(stderr, "http://batchloaf.wordpress.com\n");
fprintf(stderr, "This version 21-4-2012\n");
fprintf(stderr, "\n");

// Parse command line arguments. Available options:
//

{
	// Process next command line argument
	if (strcmp(argv[n], "/preview") == 0)
	{
		// Enable preview window
		show_preview_window = 1;
	}
	else if (strcmp(argv[n], "/devlist") == 0)
	{
		// Set flag to list devices rather than capture image
		list_devices = 1;
	}
```

```
        else if (strcmp(argv[n], "/filename") == 0)
        {
                // Set output filename to specified string
                if (++n < argc) //strcpy(filename, argv[n]);
                {
                        // Copy provided string into char buffer
                        strcpy(char_buffer, argv[n]);

                        // Trim inverted commas if present and copy
                        // provided string into filename char array
                        if (char_buffer[0] == '"')
                        {
                                strncat(filename, char_buffer, strlen(char_buffer)-2);
                        }
                        else
                        {
                                strcpy(filename, char_buffer);

}
        else if (strcmp(argv[n], "/delay") == 0)
        {
                // Set snapshot delay to specified value
```

```
            if (++n < argc) snapshot_delay = atoi(argv[n]);
            else exit_message("Error: invalid delay specified", 1);

            if (snapshot_delay <= 0)
                exit_message("Error: invalid delay specified", 1);
        }
        else if (strcmp(argv[n], "/devnum") == 0)
        {
            // Set device number to specified value
            if (++n < argc) device_number = atoi(argv[n]);
            else exit_message("Error: invalid device number", 1);

            if (device_number <= 0)
                exit_message("Error: invalid device number", 1);
        }
        else if (strcmp(argv[n], "/devname") == 0)
        {
            // Set device number to specified value
            if (++n < argc)
            {
                // Copy device name into char buffer
                strcpy(char_buffer, argv[n]);
```

```
                    // Trim inverted commas if present and copy
                    // provided string into device_name
                    if (char_buffer[0] == '"')
                    {
                        strncat(device_name, char_buffer, strlen(char_buffer)-
2);
                    }
                    else
                    {
                        strcpy(device_name, char_buffer);
                    }

                    // Remember to choose by name rather than number
                    device_number = 0;
                }
                else exit_message("Error: invalid device name", 1);
            }
            else
            {
                // Unknown command line argument
                fprintf(stderr, "Unrecognised option: %s\n", argv[n]);
```

```
                exit_message("", 1);
            }

            // Increment command line argument counter
            n++;
        }

        // Intialise COM
        hr = CoInitializeEx(NULL, COINIT_MULTITHREADED);
        if (hr != S_OK)
            exit_message("Could not initialise COM", 1);

        // Create filter graph
        hr = CoCreateInstance(CLSID_FilterGraph, NULL,

hr = CoCreateInstance(CLSID_CaptureGraphBuilder2, NULL,
                CLSCTX_INPROC_SERVER, IID_ICaptureGraphBuilder2,
                (void **)&pBuilder);
        if (hr != S_OK)
            exit_message("Could not create capture graph builder", 1);

        // Attach capture graph builder to graph
```

```
hr = pBuilder->SetFiltergraph(pGraph);
if (hr != S_OK)
        exit_message("Could not attach capture graph builder to graph", 1);


                        CLSID_VideoInputDeviceCategory, &pEnum, 0);
if (hr != S_OK)
        exit_message("No video devices found", 1);


// If the user has included the "/list" command line
// argument, just list available devices, then exit.
if (list_devices != 0)
{
        fprintf(stderr, "Available capture devices:\n");
        n = 0;
        while(1)
        {
                // Find next device
                hr = pEnum->Next(1, &pMoniker, NULL);
                if (hr == S_OK)
                {
                        // Increment device counter
                        n++;
```

```
                        // Get device name
                        hr = pMoniker->BindToStorage(0, 0,
IID_PPV_ARGS(&pPropBag));
                        VARIANT var;
                        VariantInit(&var);
                        hr = pPropBag->Read(L"FriendlyName", &var, 0);
                        fprintf(stderr, "  %d. %ls\n", n, var.bstrVal);
                        VariantClear(&var);
                    }
                    else
                    {
                        // Finished listing device, so exit program
                        if (n == 0) exit_message("No devices found", 0);
                        else exit_message("", 0);
                    }
                }
            }

            // Get moniker for specified video input device,
            // or for the first device if no device number
            // was specified.
```

```
VARIANT var;
n = 0;
while(1)
{
        // Access next device
        hr = pEnum->Next(1, &pMoniker, NULL);
        if (hr == S_OK)
        {
                n++; // increment device count
        }
        else
        {
                if (device_number == 0)
                {
                        fprintf(stderr,
                                "Video capture device %s not found\n",
                                device_name);
                }
                else
                {
                        fprintf(stderr,
                                "Video capture device %d not found\n",
```

```
                    device_number);
        }
        exit_message("", 1);
}
// If device was specified by name rather than number...
if (device_number == 0)
{
        // Get video input device name
        hr = pMoniker->BindToStorage(0, 0, IID_PPV_ARGS(&pPropBag));
        if (hr == S_OK)
        {
                // Get current device name
                VariantInit(&var);
                hr = pPropBag->Read(L"FriendlyName", &var, 0);

                // Convert to a normal C string, i.e. char*
                sprintf(char_buffer, "%ls", var.bstrVal);
                VariantClear(&var);
                pPropBag->Release();
                pPropBag = NULL;

                // Exit loop if current device name matched devname
```

```
                    if (strcmp(device_name, char_buffer) == 0) break;
                }
                else
                {
                    exit_message("Error getting device names", 1);
                }
            }
            else if (n >= device_number) break;
}

// Get video input device name
hr = pMoniker->BindToStorage(0, 0, IID_PPV_ARGS(&pPropBag));
VariantInit(&var);
hr = pPropBag->Read(L"FriendlyName", &var, 0);
fprintf(stderr, "Capture device: %ls\n", var.bstrVal);
VariantClear(&var);

// Create capture filter and add to graph
hr = pMoniker->BindToObject(0, 0,
                            IID_IBaseFilter, (void**)&pCap);
if (hr != S_OK) exit_message("Could not create capture filter", 1);
```

```
// Add capture filter to graph
hr = pGraph->AddFilter(pCap, L"Capture Filter");
if (hr != S_OK) exit_message("Could not add capture filter to graph", 1);

// Create sample grabber filter
hr = CoCreateInstance(CLSID_SampleGrabber, NULL,
        CLSCTX_INPROC_SERVER, IID_IBaseFilter,
        (void**)&pSampleGrabberFilter);
if (hr != S_OK)
        exit_message("Could not create Sample Grabber filter", 1);

// Query the ISampleGrabber interface of the sample grabber filter
hr = pSampleGrabberFilter->QueryInterface(
                DexterLib::IID_ISampleGrabber, (void**)&pSampleGrabber);
if (hr != S_OK)
exit_message("Could not get ISampleGrabber interface to sample grabber filter", 1);

// Enable sample buffering in the sample grabber filter
hr = pSampleGrabber->SetBufferSamples(TRUE);
if (hr != S_OK)
        exit_message("Could not enable sample buffering in the sample grabber", 1);
```

```
// Set media type in sample grabber filter
AM_MEDIA_TYPE mt;
ZeroMemory(&mt, sizeof(AM_MEDIA_TYPE));
mt.majortype = MEDIATYPE_Video;
mt.subtype = MEDIASUBTYPE_RGB24;
hr = pSampleGrabber->SetMediaType((DexterLib::_AMMediaType *)&mt);
if (hr != S_OK)
        exit_message("Could not set media type in sample grabber", 1);
// Add sample grabber filter to filter graph
hr = pGraph->AddFilter(pSampleGrabberFilter, L"SampleGrab");
if (hr != S_OK)
        exit_message("Could not add Sample Grabber to filter graph", 1);
hr = pGraph->AddFilter(pNullRenderer, L"NullRender");
if (hr != S_OK)
        exit_message("Could not add Null Renderer to filter graph", 1);

// Connect up the filter graph's capture stream
hr = pBuilder->RenderStream(
        &PIN_CATEGORY_CAPTURE, &MEDIATYPE_Video,
        pCap,  pSampleGrabberFilter, pNullRenderer);
if (hr != S_OK)
        exit_message("Could not render capture video stream", 1);
```

```
// Connect up the filter graph's preview stream
if (show_preview_window > 0)
{
        hr = pBuilder->RenderStream(
                        &PIN_CATEGORY_PREVIEW, &MEDIATYPE_Video,
                        pCap, NULL, NULL);
        if (hr != S_OK && hr != VFW_S_NOPREVIEWPIN)
                exit_message("Could not render preview video stream", 1);
}

// Get media control interfaces to graph builder object
hr = pGraph->QueryInterface(IID_IMediaControl,
                                (void**)&pMediaControl);
if (hr != S_OK) exit_message("Could not get media control interface", 1);

// Run graph
while(1)
{
        hr = pMediaControl->Run();
        // Hopefully, the return value was S_OK or S_FALSE
        if (hr == S_OK) break; // graph is now running
```

```
        if (hr == S_FALSE) continue; // graph still preparing to run
        // If the Run function returned something else,
        // there must be a problem
        fprintf(stderr, "Error: %u\n", hr);
        exit_message("Could not run filter graph", 1);
}

// Wait for specified time delay (if any)
Sleep(snapshot_delay);

// Grab a sample
// First, find the required buffer size
long buffer_size = 0;
while(1)
{
        // Passing in a NULL pointer signals that we're just checking
        // the required buffer size; not looking for actual data yet.
        hr = pSampleGrabber->GetCurrentBuffer(&buffer_size, NULL);
        // Keep trying until buffer_size is set to non-zero value.
        if (hr == S_OK && buffer_size != 0) break;
        // If the return value isn't S_OK or VFW_E_WRONG_STATE
        // then something has gone wrong. VFW_E_WRONG_STATE just
```

```
        // means that the filter graph is still starting up and
        // no data has arrived yet in the sample grabber filter.
        if (hr != S_OK && hr != VFW_E_WRONG_STATE)
                exit_message("Could not get buffer size", 1);
}
// Stop the graph
pMediaControl->Stop();
// Allocate buffer for image
pBuffer = new char[buffer_size];
if (!pBuffer)
        exit_message("Could not allocate data buffer for image", 1);
// Retrieve image data from sample grabber buffer
hr = pSampleGrabber->GetCurrentBuffer(
                &buffer_
if (hr != S_OK) exit_message("Could not get media type", 1);

// Retrieve format information
VIDEOINFOHEADER *pVih = NULL;
if ((mt.formattype == FORMAT_VideoInfo) &&
        (mt.cbFormat >= sizeof(VIDEOINFOHEADER)) &&
        (mt.pbFormat != NULL))
{
```

```
// Get video info header


// Create bitmap structure
long cbBitmapInfoSize = mt.cbFormat - SIZE_PREHEADER;
BITMAPFILEHEADER bfh;
ZeroMemory(&bfh, sizeof(bfh));
bfh.bfType = 'MB'; // Little-endian for "BM".
bfh.bfSize = sizeof(bfh) + buffer_size + cbBitmapInfoSize;
bfh.bfOffBits = sizeof(BITMAPFILEHEADER) + cbBitmapInfoSize;


// Open output file
HANDLE hf = CreateFile(filename, GENERIC_WRITE,
FILE_SHARE_WRITE, NULL, CREATE_ALWAYS, 0, NULL);
if (hf == INVALID_HANDLE_VALUE)
        exit_message("Error opening output file", 1);


// Write the file header.
DWORD dwWritten = 0;
WriteFile(hf, &bfh, sizeof(bfh), &dwWritten, NULL);
WriteFile(hf, HEADER(pVih),
                        cbBitmapInfoSize, &dwWritten, NULL);
// Write pixel data to file
```

```
            WriteFile(hf, pBuffer, buffer_size, &dwWritten, NULL);
            CloseHandle(hf);
        }
        else
        {
            exit_message("Wrong media type", 1);
        }
        // Free the format block
        if (mt.cbFormat != 0)
        {       CoTaskMemFree((PVOID)mt.pbFormat);
            mt.cbFormat = 0;
            mt.pbFormat = NULL;
        }
        if (mt.pUnk != NULL)
        {               // pUnk should not be used.
            mt.pUnk->Release();
            mt.pUnk = NULL;
        }
        // Clean up and exit
        fprintf(stderr, "Captured image to %s", filename);
        exit_message("", 0);
}
```

```
Clear all

 clc

 close all

TrainDatabasePath = 'C:\Users\VARUN\Desktop\TrainDatabase\';

TestDatabasePath = 'C:\Users\VARUN\Desktop\TestDatabase\';

TestImage = 'C:\Users\VARUN\Desktop\TestDatabase\1.jpg';

im = imread(TestImage);

T = CreateDatabase(TrainDatabasePath);

[m, A, Eigenfaces] = EigenfaceCore(T);

OutputName = Recognition(TestImage, m, A, Eigenfaces);

SelectedImage = strcat(TrainDatabasePath,'\',OutputName);

SelectedImage = imread(SelectedImage);

imshow(im)

title('Test Image');

figure,imshow(SelectedImage);

title('Equivalent Image');

str = strcat('Matched image is : ',OutputName);

disp(str)

function T = CreateDatabase(TrainDatabasePath)

TrainFiles = dir(TrainDatabasePath);

Train_Number = 0;
```

```
for i = 1:size(TrainFiles,1)
if
not(strcmp(TrainFiles(i).name,'.')|strcmp(TrainFiles(i).name,'..')|strcmp(TrainFiles(i).name,'Thu mbs.db'))
Train_Number = Train_Number + 1; % Number of all images in the training database
end
end
T = [];
for i = 1 : Train_Number
    str = int2str(i);
    str = strcat('\',str,'.jpg');
    str = strcat(TrainDatabasePath,str);
    img = imread(str);
    img = rgb2gray(img);
    [irow icol] = size(img);
    temp = reshape(img',irow*icol,1);   % Reshaping 2D images into 1D image vectors
    T = [T temp]; % 'T' grows after each turn
end
function [m, A, Eigenfaces] = EigenfaceCore(T)
m = mean(T,2); % Computing the average face image m = (1/P)*sum(Tj's) (j = 1 : P)
Train_Number = size(T,2);
A = [];
```

```
for i = 1 : Train_Number
   temp = double(T(:,i)) - m;
 Ai = Ti - m
   A = [A temp]; % Merging all centered images
end
L = A'*A; % L is the surrogate of covariance matrix C=A*A'.
[V D] = eig(L); % Diagonal elements of D are the eigenvalues for both L=A'*A and C=A*A'.
L_eig_vec = [];
for i = 1 : size(V,2)
   if( D(i,i)>1 )
      L_eig_vec = [L_eig_vec V(:,i)];
   end
end
Eigenfaces = A * L_eig_vec; % A: centered image vectors
function OutputName = Recognition(TestImage, m, A, Eigenfaces)
ProjectedImages = [];
Train_Number = size(Eigenfaces,2);
for i = 1 : Train_Number
   temp = Eigenfaces'*A(:,i); % Projection of centered images into facespace
   ProjectedImages = [ProjectedImages temp];
end
InputImage = imread(TestImage);
```

```
temp = InputImage(:,:,1);

[irow icol] = size(temp);

InImage = reshape(temp',irow*icol,1);

Difference = double(InImage)-m; % Centered test image

ProjectedTestImage = Eigenfaces'*Difference; % Test image feature vector

Euc_dist = [];

for i = 1 : Train_Number

   q = ProjectedImages(:,i);

   temp = ( norm( ProjectedTestImage - q ) )^2;

   Euc_dist = [Euc_dist temp];

end

[Euc_dist_min , Recognized_index] = min(Euc_dist);

OutputName = strcat(int2str(Recognized_index),'.jpg');
```

function [m, A, Eigenfaces] = EigenfaceCore(T)

```
 m = mean(T,2); % Computing the average face image m = (1/P)*sum(Tj's)    (j = 1 : P)

 Train_Number = size(T,2);

 A = [];

 for i = 1 : Train_Number

    temp = double(T(:,i)) - m; % Computing the difference image for each image in the
training set Ai = Ti - m

    A = [A temp]; % Merging all centered images

 end
```

```
L = A'*A; % L is the surrogate of covariance matrix C=A*A'.

[V D] = eig(L); % Diagonal elements of D are the eigenvalues for both L=A'*A and
C=A*A'.

L_eig_vec = [];

for i = 1 : size(V,2)

   if( D(i,i)>1 )

      L_eig_vec = [L_eig_vec V(:,i)];

   end

end

Eigenfaces = A * L_eig_vec; % A: centered image vectors
```

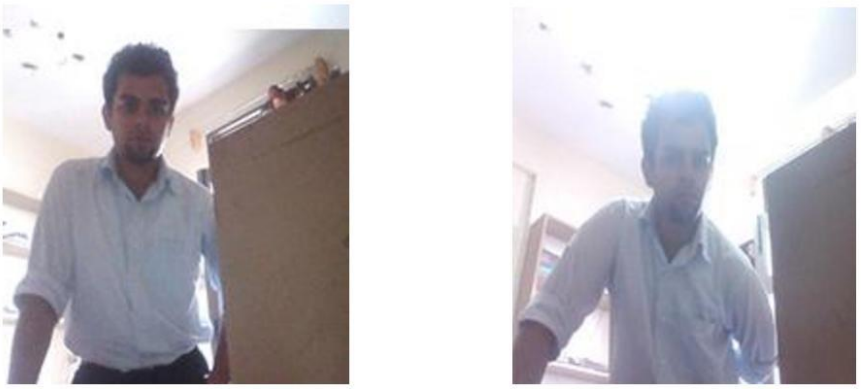
CHAPTER 5: RESULT

Eigenfaces

	1		3	4	5	6	7	8	9	10
1	0.8366	-1.9204	-1.3806	-0.6049	-8.1744	0.5829	7.3790	-19.6944	17.5970	-14.1171
2	0.1814	-0.4456	-1.3686	-1.5538	-9.1400	0.8316	7.2478	-19.2706	19.7610	-12.9015
3	0.2253	0.9877	-0.8068	-0.6536	-8.5714	0.8736	6.6152	-18.8522	22.0401	-11.2657
4	-0.4533	1.6604	0.5665	0.7268	-8.0785	0.0956	5.8733	-17.7039	21.9304	-10.4914
5	-1.8836	1.6317	2.0978	1.5698	-8.2427	-1.3863	5.8905	-16.4436	18.9177	-11.3564
6	-1.8964	0.2483	2.8630	1.7782	-8.4752	-1.3662	6.5589	-16.1463	16.7287	-11.4715
7	-2.5649	-1.7642	4.5074	-0.6908	-9.8816	-0.2759	7.2501	-17.6085	16.1274	-12.4744
8	-3.2527	-3.1485	4.5160	-1.6251	-10.6921	1.7734	8.8613	-18.7840	17.3071	-13.6928
9	-1.2097	-0.3227	-1.7141	-2.2007	-11.6316	-1.5381	9.5350	-17.1852	18.4946	-12.1530
10	-0.5212	-0.3875	-0.3787	-0.6170	-11.3470	-0.7851	7.8010	-16.9957	17.4437	-10.5814
11	-1.2390	-0.5330	-0.3332	1.1845	-11.1467	-0.5518	4.6596	-16.7215	16.8105	-10.7650
12	-0.5194	-0.5606	0.8343	1.7187	-10.4887	-0.7482	5.2436	-17.6798	17.6501	-12.8153
13	-0.4966	-0.5084	0.8902	1.6508	-11.2870	-1.6164	6.5324	-17.7343	19.4276	-14.2190
14	0.2250	-0.5102	0.9591	1.8413	-10.8658	-0.7490	5.8164	-17.4513	19.8137	-13.3244
15	-0.5163	-0.5538	0.9753	2.3028	-8.8650	3.2788	4.7310	-17.2463	19.6150	-9.9533
16	0.8652	-0.6310	1.0534	4.3868	-8.4669	7.2976	3.3356	-16.6586	18.5374	-6.1702
17	-2.6340	-1.1188	-2.9170	-0.9660	-3.2238	1.0563	3.1385	-15.7882	18.0010	-7.5801
18	-1.8829	-1.0855	-1.0402	-1.3431	-7.9925	2.7294	3.8921	-18.3259	17.7554	-10.1173
19	-1.8549	-1.1331	0.7621	-1.0965	-11.8539	4.4836	4.0932	-21.2445	17.2052	-13.9355
20	0.2685	-1.1625	2.3441	-0.3297	-12.6340	3.5212	2.1689	-21.2079	15.8872	-14.2579

Table 3: Table representing eigen values. Column 1 to 10 represent 10 Images from training database . Since in column 2 the eigen values are minimum for the majority of the cases it is seen that figure 5(a) is recognised with figure 5(b) which is the second image in the training database

MINIMUM EIGEN VALUES FOR RECOGNITION BETWEEN TWO IMAGES



(a) (b)

Figure 5: Test Image (a) and train database image (b) . Under different Lighting conditions and varying distance faicial recogniton positive using PCA analysis.

TEST IMAGE

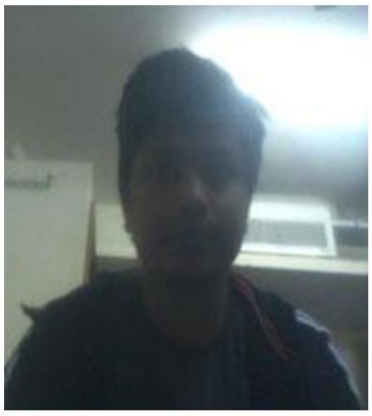


Figure 6 : Image to be used as sample training image for facial recognition of all test images.

SAMPLE DATABASE AND THEIR RECOGNITION

Figure 7: Test Image (a) and train database image (b) . Under different Lighting conditions and varying distance faicial recogniton positive using PCA analysis. If Minimum Euclid<0.8e008 and 0.35e008 facial recognition is positive as seen in table 4 where test image matches to training Image 2 as the minimum Eucledian distance is 0 in all the test cases

Euclideandistance <2x5 double>

	1	2	3	4	5
1	4.9683e+07	0	7.0153e+07	4.2601e+07	8.0790e+07
2	-2.0713e+07	0	-4.5767e+07	-1.4814e+08	3.4360e+06
3	-3.1828e+08	0	-1.7676e+07	-5.9383e+07	1.0934e+07

Table 4: Table representing Minimum Eucledian distance between images figure 7 (a) & (b)

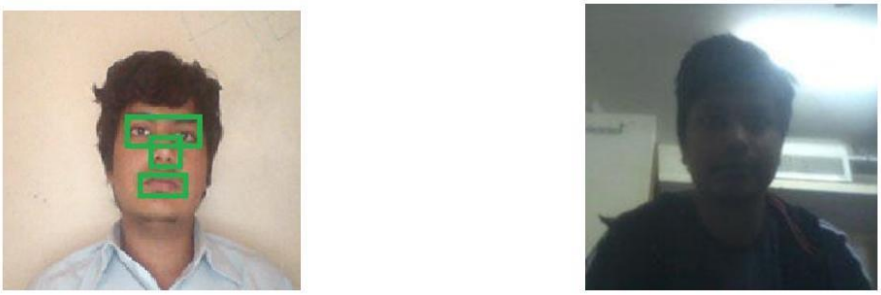


Figure 8: Test Image (a) and train database image (b) . Under different Lighting conditions and varying distance faicial recogniton positive using PCA analysis. If Minimum Euclid<0.8e008 and 0.35e008 facial recognition is positive as seen in table 5 where test image matches to training Image 2 as the minimum Eucledian distance is 0 in all the test cases

Euclideandistance <3x5 double>

	1	2	3	4	5
1	0	-4.9683e+07	2.0470e+07	-7.0817e+06	3.1107e+07
2	0	2.0713e+07	-2.5055e+07	-1.2743e+08	2.4149e+07
3	0	3.1828e+08	3.0060e+08	2.5889e+08	3.2921e+08

Table 5: Table representing Minimum Eucledian distance between images figure 8 (a) & (b)

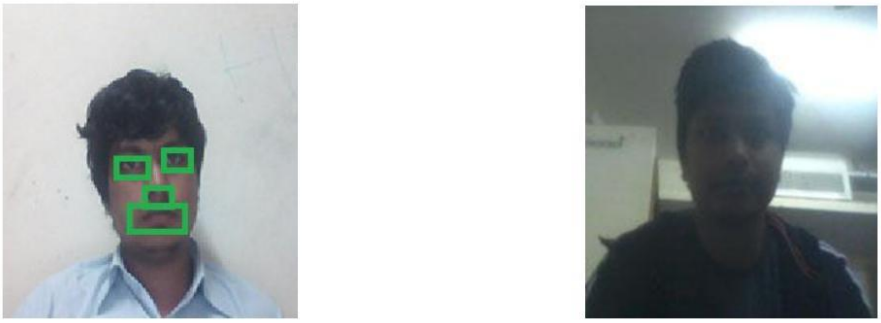


Figure 9: Test Image (a) and train database image (b) . Under different Lighting conditions and varying distance faicial recogniton positive using PCA analysis. If Minimum Euclid<0.8e008 and 0.35e008 facial recognition is positive as seen in table 6 where test image matches to training Image 2 as the minimum Eucledian distance is 0 in all the test cases

Euclideandistance <3x5 double>

	1	2	3	4	5
1	0	-5.3157e+07	1.6079e+07	-1.1808e+07	2.5682e+07
2	0	3.5711e+07	-1.3820e+07	-1.2072e+08	3.8047e+07
3	0	2.3053e+08	2.3701e+08	2.1745e+08	2.5829e+08

Table 6: Table representing Minimum Eucledian distance between images figure 9 (a) & (b)

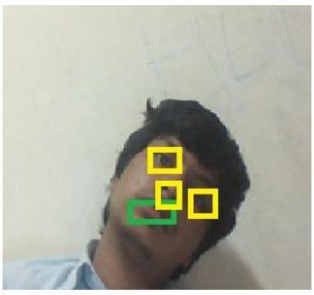
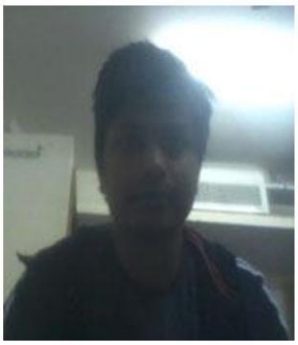


Figure 10: Test Image (a) and train database image (b) . Under different Lighting conditions and varying distance faicial recogniton positive using PCA analysis. If Minimum Euclid<0.8e008 and 0.35e008 facial recognition is positive as seen in table 7 where test image matches to training Image 2 as the minimum Eucledian distance is 0 in all the test cases

Euclideandistance <3x5 double>

	1	2	3	4	5
1	-2.0470e+07	-7.0153e+07	0	-2.7552e+07	1.0637e+07
2	2.5055e+07	4.5767e+07	0	-1.0237e+08	4.9203e+07
3	-3.0060e+08	1.7676e+07	0	-4.1707e+07	2.8610e+07

Table 7: Table representing Minimum Eucledian distance between images figure 10 (a) & (b)



Facial recognition result – positive

Figure 11: Test Image (a) and train database image (b) . Under different Lighting conditions and varying distance faicial recogniton positive using PCA analysis. If Minimum Euclid<0.8e008 and 0.35e008 facial recognition is positive as seen in table 8 where test image matches to training Image 2 as the minimum Eucledian distance is 0 in all the test cases

Euclideandistance <2x5 double>

	1	2	3	4	5
1	0	-3.8967e+07	3.0386e+07	1.5579e+06	4.1819e+07
2	0	1.9189e+07	-3.0280e+07	-1.3926e+08	2.4052e+07

Table 8: Table representing Minimum Eucledian distance between images figure 11 (a) & (b)

CONCLUSION

The face recognition and detection algorithms were thoroughly studied taking a number of test images and varying the conditions and variables. All the work mentioned above involved real time data. The PCA and MPCALDA success rates were given while for face detection, the success rate was different for different images depending on the external factors. The overall success rate was 95%.

It is seen that a 19V 4 cell Li Ion battery if uses our application works normally tends to last longer to 4 hours rather than the traditional 2 hour use.

APPENDIX A

SOFTWARE TOOLS USED

A.1. Microsoft visual C++

Microsoft Visual C++ (often abbreviated as MSVC or VC++) is a commercial integrated development environment (IDE) product from Microsoft for the C, C++, and C++/CLI programming languages. It features tools for developing and debugging C++ code, especially code written for the Microsoft Windows API, the DirectX API, and the Microsoft .NET Framework. It has used Microsoft's DirectShow API for developing source to click photos using webcam for our facial service.

A.1.1 Microsoft's DirectShow API

DirectShow (sometimes abbreviated as DS or DShow), codename Quartz, is a multimedia framework and API produced by Microsoft for software developers to perform various operations with media files or streams. It is the replacement for Microsoft's earlier Video for Windows technology. Based on the Microsoft Windows Component Object Model (COM) framework, DirectShow provides a common interface for media across various programming languages, and is an extensible, filter-based framework that can render or record media files on demand at the request of the user or developer. The DirectShow development tools and documentation were originally distributed as part of the DirectX SDK. Currently, they are distributed as part of the Windows SDK (formerly known as the Platform SDK).

A.1.2 MediaCapture API

The Media Capture API is a standard developed to allow web applications to access the media capture capabilities of a device This would allow a web application to include a function to record audio via the device's microphone, and take a picture or record a video with the device's camera.

A.2 MATLAB

MATLAB® is a high-level language and interactive environment for numerical computation, visualization, and programming. Using MATLAB, you can analyze data, develop algorithms, and create models and applications. The language, tools, and built-in math functions enable you to explore multiple approaches and reach a solution faster than with spreadsheets or traditional programming languages, such as C/C++ or Java™.

You can use MATLAB for a range of applications, including signal processing and communications, image and video processing, control systems, test and measurement, computational finance, and computational biology. More than a million engineers and scientists in industry and academia use MATLAB, the language of technical computing.

Image Processing Toolbox™ provides a comprehensive set of reference-standard algorithms, functions, and apps for image processing, analysis, visualization, and algorithm development. You can perform image enhancement, image deblurring, feature detection, noise reduction,image segmentation, geometric transformations, and image registration. Many toolbox functions are multithreaded to take advantage of multicore and multiprocessor computers.

Image Processing Toolbox supports a diverse set of image types, including high dynamic range, gigapixel resolution, embedded ICC profile, and tomographic. Visualization functions let you explore an image, examine a region of pixels, adjust the contrast, create contours or histograms, and manipulate regions of interest (ROIs). With toolbox algorithms you can restore degraded images, detect and measure features, analyze shapes and textures, and adjust color balance.

A.3.VISUAL STUDIO 2012 ULTIMATE

.NET Framework

Supported in: 4.5, 4, 3.5, 3.0, 2.0, 1.1, 1.0

.NET Framework Client Profile Supported in: 4, 3.5 SP1

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with Windows Forms or WPF applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight.

Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer,class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source-control systems (like Subversion and Visual SourceSafe) and adding new toolsets like editors and visual designers fordomain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer).

Visual Studio supports different programming languages by means of language services, which allow the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages includeC/C++ (via Visual C++), VB.NET (via Visual Basic .NET), C# (via Visual C#), and F# (as of Visual Studio 2010). Support for other languages such as M, Python, and Ruby among others is available via language services installed separately. It also supports XML/XSLT,HTML/XHTML, JavaScript and CSS. Individual language-specific versions of Visual Studio also exist which provide more limited language services to the user: Microsoft Visual Basic, Visual J#, Visual C#, and Visual C++.

A.4 Windows Service

In Windows NT operating systems, a Windows service is a computer program that operates in the background.[1] It is similar in concept to a Unix daemon.[1] A Windows service must conform to the interface rules and protocols of the Service Control Manager, the component responsible for managing Windows services.[2]

Windows services can be configured to start when the operating system is started and run in the background as long as Windows is running. Alternatively, they can be started manually or by an event. Windows NT operating systems include numerous services which run in context of three user accounts: System, Network Service and Local Service. These Windows components are often associated with Host Process for Windows Services. Because Windows services operate in the context of their own dedicated user accounts, they can operate when a user is not logged on.

Prior to Windows Vista, services installed as an "interactive service" could interact with Windows desktop and show a graphical user interface. In Windows Vista, however, interactive services are deprecated and may not operate properly, as a result of Windows Service hardening.[

Windows PowerShell can intrinsically manage Windows services via the following cmd lets:

- Get-Service
- New-Service
- Restart-Service
- Resume-Service
- Set-Service
- Start-Service
- Stop-Service
- Suspend-Service

powerStatus

Type: System.ServiceProcess.PowerBroadcastStatus

A PowerBroadcastStatus that indicates a notification from the system about its power status.

Return Value

Type: System.Boolean

When implemented in a derived class, the needs of your application determine what value to return. For example, if a QuerySuspend broadcast status is passed, you could cause your application to reject the query by returning false.

REFERENCES

- http://www.face-rec.org
- http://www.umiacs.umd.edu/~knkim/KG_VISA/PCA/FaceRecog_PCA_Kim.pdf
- IEEE Transactions on Image Processing
- Face Recognition Editors: Kresimir Delac and Mislav Grgic
- Face Processing: Advanced Modeling and Methods Editors: Wenyi Zhao and RamaChellappa
- www.microsoft.com
- www.c-sharpcorner.com/1/78/
- www.stackoverflow.com
- K. Jain (1989), 'Fundamentals of Digital Image Processing, Prentice-Hall
- S. Choi, C. Kim, C. Choi (2007), 'Shadow compensation in 2D images for face recognition,' Pattern Recognition,
- RULE-BASED FACE DETECTION IN FRONTAL VIEWS by Constantine Kotropoulos and Ioannis Pitas
- Face Recognition: A Convolutional Neural Network Approach by Steve Lawrence, C.Lee Giles, Ah Chung Tsoi and Andrew D. Back.