DSP Project Abstracts
ELC 152
October 2009

Project Code:

ELC152-oct2009-01

Title:

Image Resizer Using Bicubic Interpolation

Proponents:

Claudelle S. Abellanosa
Beatriz Elena L. Ocampo
Andeonette A. Villanueva

Abstract:

Bicubic Interpolation, one interpolation method out of many, is more advanced than other some other interpolation methods, such as the Nearest-Neighbor Interpolation and Bilinear Interpolation. Implementation of this Image Resizer is done with the use of wxDevC++. A graphical user interface is created, which can open, resize, and save images in RGBtext, BMP, JPG, and PNG formats. This project is a modification of the Bilinear Interpolation Image Resizing project done by a previous group.

Notes:

Project Code:

ELC152-oct2009-02

Title:

Ball Recognition

Proponents:

Steve Ryan C. Andalis
Michael L. Ruiz
Miguel Antonino L. Varela

Abstract:

This paper discusses the methodology into properly recognizing and counting circular objects. This was done by passing an image through a series of digital imaging techniques properly sequenced to achieve the desired output. The image signal passes through the color filter first to isolate the relevant pixels. The relevant pixels or those that fall within the user pre-defined color will be masked as having a value 1 and 0 for the background. The image counting algorithm will then be implemented. The algorithm detects groups of contiguous pixels and counts them. The final filter will then recognize the shape of the contiguous pixels and count the circles in them.

Notes:

Project Code:

ELC152-oct2009-03

Title:

Harris Points Identification

Proponents:

Oliver Ryan S. Ang
Gazelo D. Arcilla
Genesis P. Mendoza

Abstract:

Harris Points Identification is a C++ program developed using wxDevC++ and the wxWidgets library which aims to detecting the corners in a given input image. The program uses the Harris corner detection in order to get the interest points and corners of the images.

The algorithm outputs four versions of the input image. First one is the input image, second is a monochrome version of the image, third is the corner response version of the image and last is the output image with the corner points. The program can load several image file types including .rgbtext, .jpg, .bmp, .png., .gif and .tiff. But the program can only save into .jpg, .bmp and .png file types.

The application was done using wxform of wxWidgets in order to create an easy-to-use graphical user interface (GUI).

Notes:

Project Code:

ELC152-oct2009-04

Title:

XMCS Player with Equalizer

Proponents:

Jose Fernando V. Angeles
Adrienne Mae G. Riña
Albert C. San Juan

Abstract:

This project is an improvement of the console–based XMCS Player/Recorder. Aside from creating a graphical user interface for the XMCS Player, a real–time audio equalizer is also implemented using ten different band–pass filters having center frequencies of 31.5 Hz, 63 Hz, 125 Hz, 250 Hz, 500 Hz, 1000 Hz, 2000 Hz, 4000 Hz, 8000 Hz, and 16000 Hz. These center frequencies correspond to the standard for graphic equalizer center frequencies (ISO R 266–1997 or ANSI S1.6–1984).

The project implements Infinite Impulse Response filters for the band–pass filters used in the equalizer. The playback process conforms to the standard Windows API for multimedia playback. The program was constructed using C++ and wxWidgets library for the graphical user interface.

Finite Impulse Response filters were also considered. However, these required a large number of filter coefficients, and much processing power. Designing FIR filters on a warped frequency scale was then considered. The filters were optimized by combining FIR coefficients, for filters made with the same warping factor, and filter order. Optimization of filters needed the development of two console-based programs (parameter_generator.exe and optimization.exe). Despite these measures, much computing speed was still required.

Notes:

Project Code:

ELC152-oct2009-05

Title:

Audio Player and Effects

Proponents:

Mark Edison B. Angeles
Eloy Ceasar B. Bello IX
Mark Edward S. Manio

Abstract:

This project deals with the addition and introduction of effects to audio files and consequently playing these files in order to hear and see the difference between the original audio track and the track with effects. Given these tasks, the WAV file type was chosen mainly because of the high quality samples that it provides.

A program that will be able to detect, read and save the individual samples from a given audio file is needed. This program will allow the user to access the individual samples of the audio as well as other vital information such as the Chunk Size, Number of Channels and the Sampling Rate. The addition of effects to the audio is done by first converting the audio samples from the time domain to the frequency domain. Once the samples have been converted to the frequency domain, it would be easier to manipulate the said samples. The conversion will be done through Fast Fourier Transforms.

Once the effects have been implemented, a program that can play the two audio tracks will be implemented so as to see the implications of the effects to the audio tracks.

Notes:

Project Code:

ELC152-oct2009-06

Title:

Graphical Audio Waveform Display

Proponents:

John Kester C. Begonia
June Carlo D. Chico
Peter Angelo P. Santos

Abstract:

This project aims to develop a graphical audio waveform display for a simple .wav file. In order to achieve this, the project was divided into three main parts: the audio data samples extraction, the audio samples waveform graphing, and the development of a graphical user interface.

In the project, an extensive programming code is housed in which the data samples and several audio information is extracted once a .wav (WAVE) audio file and its corresponding directory is loaded and located by the user interface. After this, a visual representation of the .wav audio data samples will be developed through the graphing of the extracted points in a continuous time domain which cohere to how the overall audio file is interpreted based partly on the obtained audio format information. The plotting can be made possible through a developed graphical window which will serve as the graphing location for the points. The data points which are representations of the extracted data samples are interpreted as amplitudes and are scaled down to some factor in order to affix the graph more clearly in the fixed graphical window. Once the waveform has been produced, the user can view and analyze the graphical audio representation and some of the important digital audio data format information by using the simple functions embedded on the common computer external input interfaces such as the mouse buttons and the keyboard's arrow keys. To provide additional ease and automation, the loading button will also serve as a reloading interface where a new .wav file can be placed in the program to replace the waveform and the format information displays of the previously loaded audio file. The color of the waveform will also be available for change depending on the user's choice based on the default given set of waveform colors.

The project has been developed using techniques, functions and program libraries utilizing on the wxDev-C++ language development platform. Libraries which were based on the windows WIN32 API were the ones largely capitalized.

Notes:

Project Code:

ELC152-oct2009-07

Title:

Gaussian Noise Simulation

Proponents:

Jose Paolo B. Bernardo
Lawrence Lean R. Sze

Abstract:

This project aims to simulate Gaussian Noise through a user-provided WAV file. The main device of coding will be C++ under the Bloodshed DevC++ 4.9.9.2, which uses a GNU GCC compiler. The program is a console-based application which creates a copy of the WAV file, and creates another WAV file which is run through the Gaussian noise application. Both are named “Before Noise” and “After Noise” for convenience. A third file is also created. It is a text file named “data_bin” showing the frequency of use of each possible value of Gaussian Noise sample. The headers “WAVE” and “rifffile” provided manipulation on WAV files. The headers “process” and “random” handles the Gaussian noise creation and the Sample Counting.

Notes:

Project Code:

ELC152-oct2009-08

Title:

Pre-Image Stitching: Real-Time Error Metric Computation

Proponents:

Marnelli D. Canlas
Thea Mae A. Osorio
Grace U. Yap

Abstract:

The project is a C++ program implemented primarily using the wxDragImage and wxImage libraries, which aims to compute for the error metric of two overlapped images of any file type using the sum of squared of differences algorithm. This is of main use in carrying out image stitching processes as the location with the least pixel difference between two images is significantly correlated to the best location where stitching may be done. As the project title indicates, a real-time computation of the error metric is executed in response to mouse events.

The project includes a graphical user interface (GUI), wherein two images may be opened. The images to be opened are inputted by the user, both of which can be moved by the user along with the movement of the mouse. As the mouse together with the selected image is moved, the resultant error metric is shown on the status bar. The program involves the normalization of the RGB values of the two images and the detection of their overlapping areas, after which the computation for the error metric is performed.

Notes:

Project Code:

ELC152-oct2009-09

Title:

Image Filter Class

Proponents:

Addison S. Chan
Miguel Carlo S. Sobrio
James Edward A. Tan

Abstract:

Image Filter Class is a C++ program developed using wxDevC++ compiler and the wxWidgets library. The program is able to load an image of variable size, be able to apply image filters on the desired image, and output a new filtered image file.

The group created a graphical user interface using wxWidgets and incorporated preset filters such as blur, greyscale, monotone, resize, and hue available in the wxImage class. The user has the option to load one's own filter and apply it to a desired image. The filter kernel to be loaded is in .txt format which contains the dimensions of the filter as well as the coefficients that will be used for convolution. The RGB values are convolved to the kernel individually and the resulting output is a new filtered image that can be saved as BMP, JPEG, and PNG format.

Notes:

Project Code:
ELC152-oct2009-10

Title:
Ear Biometrics

Proponents:
Justin Dale Chan
Jeanine Ysabel Chua
Billy Joel Santos

Abstract:
A C++ program was developed using wxDev-C++, wxImage, and wxString to explore and extract the geometrical features that uniquely identify the ear. The program takes an input image and analyzes it using the methods and algorithms based on Michal Choras' work on ear biometrics.

The program has a user interface wherein a person can select the ear image to be analyzed. After getting the input image, the program first converts it to a grayscale image. An edge detection algorithm is done on the grayscale image to produce a binary image showing the contours of the ear. Next, the binary image is normalized by determining its centroid and setting it as the reference point for the following feature extraction process. The method of feature extraction includes counting the number of intersection points between the binary image and a set of circles, which is done through the application of Bresenham's circle algorithm, and calculating the sum of the distances of these points using sorted vectors and the Pythagorean Theorem.

The program creates a text file containing the file name, dimensions, and centroid of the image, along with the file names of the output binary image and the image with Bresenham's circle. Finally, the properties of the feature vectors are presented –the radii of the circles used, and their corresponding number of intersection points and sum of distances between those points.

Notes:

Project Code:

ELC152-oct2009-11

Title:

Interactive Image Stitcher via Cross-Correlation

Proponents:

Elvis T. Chua
Katrina Jean S. Elnar
Ma. Cristine M. Nepomunceno

Abstract:

CroCorIS v1.0 is an interactive image stitcher that functions by stitching two images containing overlapping regions or regions present in both images. The employed algorithm was normalized cross-correlation, which measures pixel similarity by calculating the normalized cross-correlation coefficient with values ranging from -1 to 1. A cross-correlation coefficient of 1 suggests perfect similarity. This application outputs the resulting stitched image as well as the cross-correlation image that serves as a plot of the obtained normalized cross-correlation coefficients.

To make the image stitcher application interactive and user-friendly, the software was programmed using wxDevC++ , which provides several tools and functions for creating and designing a graphical user interface (GUI). Hence users are allowed to load, stitch images and view output images by simply clicking on specific buttons. Moreover, wxDevC++ enables the application to handle images of different formats such as JPEG, BMP, PNG, GIF, and TIFF. Also, image stitching can be done against two images of different formats. For the application's full interactivity, detailed yet comprehensible help files and user-guide documents may be easily accessed by clicking on the instructions button.

Notes:

Project Code:

ELC152-oct2009-12

Title:

Audio Player with Playlist

Proponents:

Kristoffer Paolo C. Cruz
Michael Angelo S. Dantes
Ma. Kristle G. Dimarucut

Abstract:

The development and implementation of a simple audio player with a playlist function in C++ using the wxWidgets, vector and Audiere 1.9.4 libraries are aimed for this project. Through this simple audio player, the user is able to load any number of music files, arrange them according to how he chooses in the manner of a playlist, clear everything or delete something from this playlist, and finally, play the music files in the playlist order that he has defined. Several well-known music file types such as MP3, WAV and OGG are supported by the program. The user is also allowed to load not only from one folder but from any part of the computer file system. In order to make the program experience easy for the user, the program is also implemented in a GUI (Graphical User Interface) with proper audio player list boxes and control buttons.

In order to load a particular music file and play it, accessing the computer's sound card is most important. The sound card or audio card is a computer expansion card which controls and processes the audio signals that are inputted to and outputted from the computer. In this project, since the sound file comes from within the computer memory, this audio is simply loaded into a temporary buffer memory and then converted from digital audio samples to an analog signal to be outputted to the speakers. In order to access the computer's sound card, *Audiere 1.9.4*, a music and sound effects C++ library, is used. Then, to create an arbitrary amount of sound buffers, the use of the built-in Vector library is implemented by the program. Lastly, in order to create the overall GUI for the program, the *wxWidgets* libraries are used.

The program is therefore written such that it follows a general flow: 1) Browsing and Loading to Song List, 2) Loading to Playlist, and 3) Playing. The browse button is first clicked in order to browse for the desired music file. The chosen individual file is automatically loaded to a temporary buffer memory, and the complete path of the file is displayed on the upper list. After all the sound files are loaded, particular files are then selected and loaded in the desired order or deleted and cleared through the click of a button. The play button can be hit at any instance as long as the playlist is not empty. When the play button is hit, the files loaded in the playlist are played one after the other.

Notes:

Project Code:

ELC152-oct2009-13

Title:

Pitch Shifting in the Frequency Domain

Proponents:

Daphine Ferrer
Louel Lacasandile
Jarwin Tee

Abstract:

This project implements a pitch shifting algorithm of an audio signal without changing its time scale. The file name of the audio to be shifted, the pitch shift direction, and the shifting factor will be given as inputs by the user. A pitch-shifted output .wav audio file will then be generated by the program.

To implement pitch shifting, raw PCM data from the file will first be extracted by the program. After which, the data will be divided into chunks or frames. Each frame will then undergo a Fast Fourier Transform performed using the FFTW library. Arrays of real and imaginary values will then be yielded by the FFT. The respective frequencies of the real and imaginary values will be represented by the indices of these arrays. These indices will then be multiplied by a shifting factor in order to change the pitch of the signal. An additional interpolation is also implemented to accommodate non-integer shifting factors. An inverse FFT will then be performed, and the resulting values will be appended into the output .wav file after its header. The audio signal may be shifted by factors that are multiples of 0.1, ranging from -2 (octave lower) and 2 (an octave higher).

A GUI was also developed using wxDev to provide ease of use. The user has the option to play the altered audio file after it has been loaded and modified by the program using the Audiere library.

Notes:

Project Code:

ELC152-oct2009-14

Title:

Optical Character Recognition using Pixel Centroid Matching

Proponents:

Jonathan Gonzales
Carlo Jamandre

Abstract:

Optical Character Recognition using Pixel Centroid Matching is a program that extracts machine-editable text from image files. It uses a novel approach by using Pixel Centroids in identifying and matching Roman alphabetical letters both in uppercase and lowercase.

The application utilizes the Pixel Centroid of characters for identification. In the implementation, a set of 10 points is used by the program to compare characters, one accounts for the whole character image, and the other nine points are computed from the partitions of the image into nine sections in an 3x3 array manner.

The user selects a bitmap file to be processed via a file picker box in the user interface. Once the convert button is clicked, the original image is processed and separated into individual character images of bitmap type. Text files containing the locations of these characters and the calculated pixel centroids are outputted for the user's reference. The primary output is a text file which contains the characters which have been detected and matched to the characters contained in the database. The database used in the program is based from the font style Tahoma.

Although the results show that the database currently provided in the program and its usage restrict compatibility with other font styles, for font style Tahoma and Arial Optical Character Recognition through Pixel Centroid Matching is an effective approach with satisfactory results.

Notes:

Project Code:

ELC152-oct2009-15

Title:

Pitch Correction Software

Proponents:

Lenvic Elicer R. Lesigues
Marvin N. Muñoz
Elgin C. Peteza

Abstract:

This Pitch Correction Software is a console-based C++ program that processes audio files, specifically of the WAV file format. Using Fast Fourier Transform and the FFTW C library, the program samples a certain sound file and uses FFT to change its fundamental frequency, thereby changing its pitch.

Running the program would open a graphical user interface. The input file can be loaded into the program by choosing File > Open in the Menu Bar or by pressing Ctrl+O. The input file must be in the same directory of the program. Once a valid sound file is inputted, the user must choose the resulting pitch by choosing through the dropdown menu. Pressing the Change Pitch button would prompt the program to take samples and perform FFT, and shift the frequency by a factor *f*, thereby changing the pitch. A shift to the right would result in a higher pitch, and a shift to the left would result in a lower pitch.

Once the sound processing is done, a separate output file is written and saved in the same directory of the program and input file. The output file is named “product.wav” and would now reflect the change in pitch.

Notes:

Project Code:

ELC152-oct2009-16

Title:

Project *Colorization*: Automatic Colorization of Grayscale Images

Proponents:

Ana Catherina C. Ramirez
Johnoelle S. Santiago
Patrick Christian L. Uyengco

Abstract:

A program, called *Colorization*, was developed, using C++ programming language, to perform automatic color traits transfer via pixel matching. The *Colorization* program accepts two images: a colored source image and a grayscale target image, and converts both from RGB to LUV. The color transfer is done by scanning and matching each pixel in the target image to a suitable pixel from the source image. Color samples from the source image are selected by using random jittered sampling. The pixel's luminance value, given by the parameter L in the LUV color model, and the neighborhood pixel statistics, consisting of the standard deviation and mean of the luminance values of the surrounding 5x5 pixel neighborhood, are the criteria for selecting the best matching pixel from the source image. Upon finding the best matching pixel, its U and V parameters (chromaticity) are transferred to the target pixel while keeping its original luminance value. In order to display the image, a pixel's LUV values are transformed back to RGB color model.

Notes: