A PROGRAMMING TOOL FOR ENHANCING THE TEACHING OF IMAGE PROCESSING

Alejandro Simon and Malek Adjouadi
Center for Advanced Technology and Education
Department of Electrical and Computer Engineering
Florida International University
10555 W. Flagler Street
Miami, FL 33174
Alejandro.Simon@4XLab.NET Adjouadi@fiu.edu

Abstract

A desktop computational platform that facilitates the teaching of digital image processing remains a needed tool for both engineers and computer scientists, especially when real-world applications are abound in pattern recognition, computer vision, geographic information systems, and biomedical imaging, to name a few. In line with the educational reform suggested by the National Science Education Standards [1], this type of teaching platform allows for the bridging of research to teaching through practical implementation of imaging algorithms [2]. The teaching through the proposed desktop computational platform has proven effective both in increasing the quality of project implementation and in research throughput of students in an academic setting. Different tools exist which currently allow students to manipulate images, but a fine balance must still exist between ease of developing image processing algorithms and the portability of the resulting code in user friendly computer platforms to allow for broader access. The teaching tool is as proposed which allows students to be exposed to developing and writing the code to their own algorithms through an effective graphical user interface. The program is expandable, and new algorithms can be added with ease. Once compiled, the program can be demonstrated as a stand alone tool, with no additional software being required. This software has been implemented successfully in a digital image processing course at Florida International University. The speed of completion of the assignments has increased, as well as the satisfaction of the students with their code. The training of the students using this GUI interface required one lecture session of two hours through an interactive presentation.

Index Terms – C++, digital image processing, education, image processing software.

Introduction

This study presents an open source image processing software which can be used as a teaching tool in an undergraduate or graduate level image processing course. The motivation to write this software platform was due to the scarcity of applications which could be used in a classroom setting that support easy coding of various image processing algorithms.

The designed software helps students perform their homework or laboratory projects in an image processing course by giving them the tools they need to quickly develop image processing routines. This is done by providing students with a framework in which images are represented as data structures which are simple to modify. This software is being used successfully in a graduate level image processing class.

With the aim to extend the use of the proposed software platform, a thorough description is provided of the program structure to include both the methods and techniques inherent to this structure and the means to integrate within it new software developments that will dynamically enhance the applications capability of this platform, stressing the nature of its open source characteristic.

Broadening the Scope of Image Applications

The scope of image processing programs can be viewed in a continuum of programming skills. One end of the spectrum represents those programs that require minimal or no programming skills, and are often used as procedures or subroutines which do not necessitate any additional manipulations or enhancements. The other end of the spectrum represents programs which require sharp skills where the user may be called to program each step of the image transformation process. Currently, several image processing programs exist, each filling a space in this continuum.

Several image processing tools that currently exist demonstrate different application principles of digital image processing [3],[4], and [5]. Such tools allow students to see interactive demos of chosen transforms or applications. The importance of a small portable solution to teaching image processing has been shown by [6]. Similar approaches to the solution proposed in this study have been developed by [7],[8] where a collection of basic transforms as well as a GUI were integrated. Some tools have demonstrated the utility of providing a solution in which the GUI and the image processing algorithms are equally important, such as [9]. Other tools choose to tackle the problem of offering real-time image processing; such an approach is described by[10].

Furthermore, several image processing software platforms have been developed with an educational focus, using Matlab, essentially. An illustrative example of such tools is presented by [11]. Such an environment provides facilities for rapid prototyping and development, but the resulting code depends for execution on the Matlab environment being installed. Similar digital image processing tools exist for Mathematica [12].

This particular program as proposed and designed advances the state of the art by implementing a stand-alone image processing platform that can be demonstrated with a default set of functionality, or expanded with additional new image applications. The program processes the most popular formats of bitmap files and can be expanded by the user to support all image formats. The image transformation code is easy to write due to a framework of functions created to handle the conversion between image file and two-dimensional arrays.

Program Description

The developed software platform has several outstanding features which facilitate its use. It presents an easy to use Graphical User Interface (GUI), as well as offers a practical framework for image processing. The overall result is a program that allows students a complete understanding of the different steps necessary for developing an image processing algorithm. Students can concentrate on developing the image processing programs without dealing with the complexities of the file format of the image processed. The program handles the various aspects of file loading, saving, and displaying. This is implemented as a Multiple Document Interface (MDI) application, allowing for several files to be opened simultaneously and different image processing algorithms applied to them. An illustration of the different runtime image processing tools displayed in windows platform is as shown in Figure 1. All actions of the program can be scripted in batch files and executed from within the program. Specially named batch files can be executed upon program startup or opening a new image processing window. Each image processing window includes an undo/redo feature which is only limited by the amount of free space on disk. Multiple instances of the program can be run simultaneously. In case the image transformation code developed by the student executes an illegal operation and crashes the program, the next time the program is executed, the temporary files left on disk are

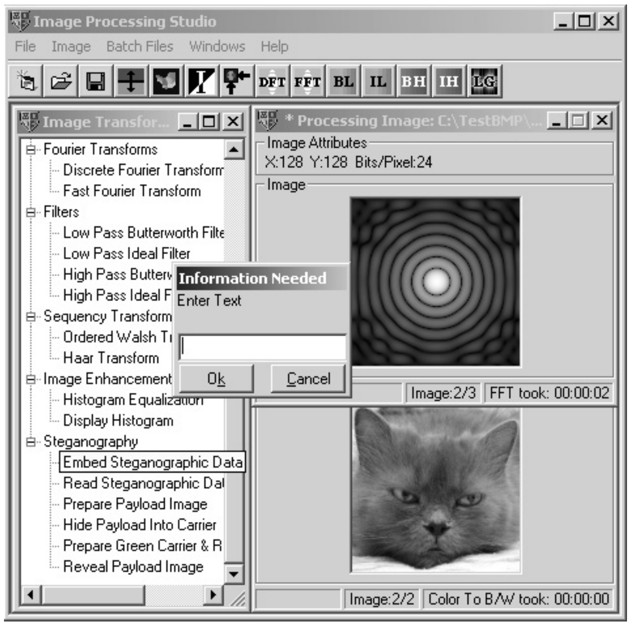
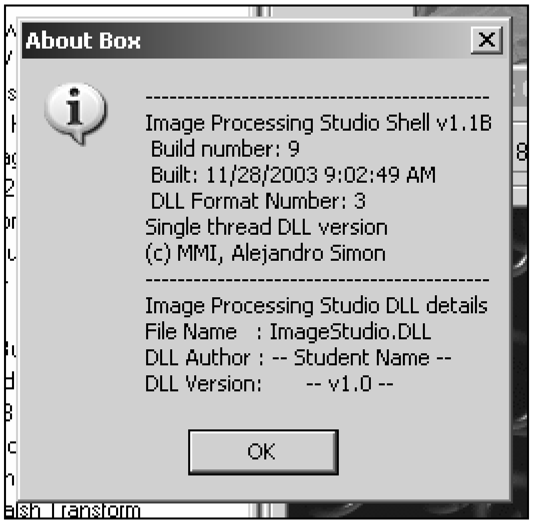


Figure 1: Program snapshot showing multiple windows opened. An easy to use interface provides access to the transforms. The Image Transform List window contains the list of transforms supported by the DLL module. The power of the Fourier spectrum of a circle is shown in the topmost window. The active window is collecting the parameters needed for a transform.

detected and the user is prompted to delete them.

The program is controlled by a familiar interface. This interface includes a menu system, an icon bar for quick access to common functions, as well as a toolbar. The toolbar lists all available image algorithms which have been programmed by the student. The author of the image processing algorithms is displayed in the credits for the program; this feature is useful in live demos in establishing authorship and the time of program creation, in order to monitor continued enhancements by the same or different author. An example of this feature is shown in Figure 2.

The image processing shell supports various threading models for the student developed image algorithms. Students can use the single threading model during the first assignments, free to move at any time to a multi threaded model, which supports simultaneous execution of several algorithms. Single threaded student code executes slightly faster resulting in an unresponsive program while the algorithm executes. Multithreaded code executes slightly slower, however it results in a program which is able to demonstrate several algorithms at once. The image processing shell adapts automatically to the threading model employed in the image processing code.



```
----------------------------------------
Image Processing Studio Shell v1.1B
 Build number: 9
 Built: 11/28/2003 9:02:49 AM
 DLL Format Number: 3
Single thread DLL version
(c) MMI, Alejandro Simon
----------------------------------------
Image Processing Studio DLL details
File Name   : ImageStudio.DLL
DLL Author : -- Student Name --
DLL Version:      -- v1.0 --
```

Figure 2: Upon program startup, the DLL is queried for the author name, which is displayed in the About Box. This mechanism ensures a compiled DLL is personalized with the author information.

Methods and Techniques Used

Background

Images are composed of a collection of pixels, which can be represented as a combination of the three primary colors: red, green and blue. Each primary color is allocated 8 bits to represent all the color graduations between black and its maximum intensity. Thus, one pixel stored in RGB format with 8 bits per color will take 24 bits, or 3 bytes to represent.

Each uncompressed image storage file format takes a different approach to storing the pixels in the image. One approach is to store the RGB value for each pixel consecutively, with some filler bytes to pad the length of each row to a multiple of 4 bytes. Bitmap files employ the padding to a multiple of 4 bytes, which may allow for faster access to pixels in 32bit memory architectures. The RGB approach results in a rather large file where there are no limitations to the number of different colors to display. Another approach is to create a table, or palette of all the colors used. Instead of storing the RGB value of each pixel, the table code for the matching RGB value is stored. The palette approach gives smaller files, however only a select number of colors can be present in the image.

Bitmap files support different image storage formats. The 24 bits per pixel format does not necessitate a palette. Other formats such as 256 colors or 16 colors require a palette. This proposed software platform supports reading and writing to any bitmap file. The format of the file read can change during a transformation. This is to enable the student to develop conversion function between formats such as 256 colors and 24 bit color.

Bitmap files are composed of 4 sections, as illustrated in Figure 3. Some sections are optional, depending on the file format. The palette is not present in 24 bit images, while 256 color images use it.

The first section is the header, which contains information about the file size and the start of the image data. This section is used to determine if the file is valid via the signature check. The presence of a palette is also specified in this section.

The information header section contains details about the image. This section is used to determine the dimensions of the image, as well as the image type: 16 colors, 256 colors, or 24 bit color.

The palette section is only present in non 24-bit images. This section is composed of several RGB color entries, each 4 bytes long. The first byte is reserved, and should be 0. There is no entry for the color code in this table. The position of the entry determines the code for this color. The next three bytes contain the RGB triplet, stored as Blue, Green, and Red.

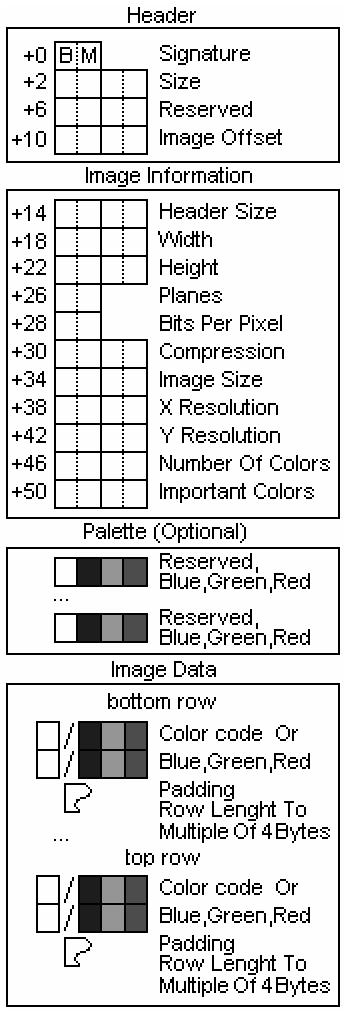


Figure 3: Details of Bitmap Image file format. The header and the image information sections are always present. They indicate the presence of an optional palette. The image data section stores 3 bits per pixel if no palette is present. Otherwise, the color codes are stored. If less than 8 bit per pixel are used, several pixels are packed in the same byte.

The image data section contains the data for the pixels for both the palette modes and the 24 bit per pixel mode. The bottom row is stored first, in a left to right order. After the row is complete, padding bytes set at 0 will be added to bring the row length to a multiple of 4 bytes. In the palette mode, the image data will contain 1-byte entries to represent the pixels in each row. In the 24 bit per pixel mode, each pixel will be stored as three bytes for the Blue, Green, and Red components.

Program Structure

The designed program is structured as two distinct modules with specific functions. The GUI module is responsible for file handling, image visualization, executing batch files, and user interaction. The image transformation module is where the image processing algorithms are executed. This module is responsible for receiving the data representing the different sections of an image, transforming them, and returning them to the GUI module for display.

The GUI module has been developed in Visual Basic. This language was chosen due to the ease of development. The GUI is compiled into an executable (.EXE) file. The image transformation module has been developed using the C language. The execution speed of this module was an important factor in deciding the language to use. This module, which is compiled into a dynamic linkage library (.DLL) file, exposes a collection of functions to the GUI. These functions are used to determine the transforms available, and request execution of a particular transform.

Students need access to the project files to build the image processing DLL module. This module communicates with the GUI to make all its functionality available to the user. It is not necessary to modify the GUI code unless the student wishes to add new functionality to it. Therefore, students can access the project files that build the GUI module, but it is not necessary for completing any part of the course work. This makes previous knowledge of C and programming techniques, the prerequisites needed for developing transforms.

Algorithmic Flow of Processing Steps

Loading and transforming an image utilizes all modules of this program. All the steps can be performed interactively by selecting options in the program, or can be scripted in the batch files supported by the program.

The images are loaded and displayed in the GUI module. When a transformation is requested, the GUI queries the DLL for the size of the returned image. This mechanism was implemented to support transforms that return larger images as well as image format conversion functions.

After the GUI or image processing shell obtains the size of the data returned, sufficient memory is allocated. Afterwards, the DLL transform entry point function is called with a numerical code representing the transform requested. The control remains inside the DLL until the transformation is finished and the image is returned to the GUI module. The entry point function sets some variables and executes the appropriate transform based on the numerical code received from the GUI. The transform converts the image into easy to manipulate arrays, modifies the content in the arrays, and converts them back to the image format. When the transform finishes execution, the image header is updated if there were changes to the variables stored in it. Afterwards, control returns to the GUI.

The GUI receives the modified image data and uses it to create a new image file on disk. This file is used to provide the undo/redo functionality. This complete process of image transformation is illustrated in Figure 4.

Inclusion of New Transform Code

To add new image transformations, students merely need access to the projects for the DLL. The GUI source code requires no modification when inserting new transform code.

Figure 4: Illustrates the programming tools necessary to compile the projects and how they interact to transform an image file. Both batch file and interactive processing are shown.

In this process, the following steps must be followed:

- The GetDLLInfo function returns the name of the author for display in the GUI. The DLL version number is returned as well. This function must be modified with the actual name of the author of the functions. This is only done once.

- The Interrogate function must be modified to add the transform to the Image Transform List window in the GUI. The transform will be added to the list with a user defined transform code. The list of parameters the function needs is specified as well. A category can also be created to group similar transforms together.

- If the transform returns more data, an entry needs to be created in the MemoryRequired function. This step is optional and only needed if the transform returns an image which is larger than the original. The worst case memory requirements are specified here. This ensures that the memory assigned to the image will be large enough to accommodate the added data. If the returned image has a bigger palette than the original image, this step needs to be performed as well.

- The DoTransform function is the entry point where the image, as well as the numerical code representing the transform is received. An entry needs to be added to call the function that will perform the image transformation.

- The last step is to declare the function which will perform the image transformation.

This series of steps will modify the DLL code to add a new image transform. These steps will include all the necessary declarations to make the image transform visible to the shell and available for calling.

The Image Transformation Framework

The DoTransform DLL function receives from the graphic processing shell the BMP headers, the palette if it exists, and the image data. All modifications to the BMP data must be performed in the DLL as the imager shell will not modify them. This scheme allows for maximum flexibility of the DLL.

The image processing DLL contains several functions composing a rapid image transformation framework. The functions are already developed and functional, but they can be modified by the user of the DLL if the need arises, enhancing the usability of this program as a platform for image processing. These functions transform image data into easy to manipulate arrays. The functions work in pairs, one converts the image to an array, while another converts the array data back into an image.

For 24 bit per pixel images, a pair of functions called ImageToArrays and ArraysToImage provides the conversion of the RGB image data into three arrays: the Red, Green and Blue planes. The ImageToArrays function correctly extracts the pixel data for all rows while skipping the padding bytes added. The ArraysToImage function reverses the process by combining the three plane arrays into a properly formed collection of pixel data. The padding bytes are also considered here.

For images containing a palette, ImageToArray and ArrayToImage functions are provided. They handle the conversion from the palette format to a bi-dimensional array containing the RGB combinations for all palette entries.

The DoTransform DLL function obtains the width, height, and bit per pixel properties from the header, and makes it available to the transforms.

The parameters which have been collected from the user are also made accessible to the transforms. The transforms are allowed to return the title of the action they performed on the image, as well as a status string, which will be displayed to the user in a message box.

The functions ExtractString and ExtractValue are used to easily extract a single parameter from the string in which the Shell encodes the parameters collected from the user.

All these functions compose the framework that allows students to productively code new transforms. This framework simplifies the writing of transform code by providing functions to access the image data.

Nevertheless, the learning experience is not hindered since all the code needed to extract the pixel data from an image is available in the DLL project, and is accessible to students at all times. The image processing shell involvement from a transformation point of view is just for opening, saving, and displaying the files, with no processing required.

Developing New Image Transforms

With the image processing framework in place, developing a transformation is now as simple as modifying an array. The typical steps in any transform are as follows:

- The transform code must declare several variables. Since the images are bi-dimensional, two variables must be declared for the loops which will iterate through all image pixels.

- To store the image data, a provided bi-dimensional array data type must be used. The quantity of arrays to declare changes if the transform expects a palette image or a 24 bit per pixel image. For a palette image, one array is sufficient, while three arrays are needed for a 24 bit per pixel image.

- The function then checks if the image type is palette or 24 bit per pixel. This step can be used to execute a format specific algorithm or to return with an error message.

- Once this check passes, the appropriate function to split the image data into the arrays is called. If the image contains a palette, it too is converted into an array.

- Now, the data arrays are processed, and the image transformation is applied to them.

- After the transformation is finished, the image data is reassembled from the arrays using the inverse function, and the transform returns.

This sequence of events is typical for all transforms; complex transforms simply expand the image transformation step over a series of functions.

Setup of the developing Environment.

The developing environment for the image processing shell is Microsoft Visual Basic 6.0, which comes as part of Microsoft Visual Studio 6.0. The developing environment for the image processing DLL is Microsoft Visual C++ 6.0, which also comes in the Visual Studio package.

The image processing shell needs the Visual Basic runtime files to execute. They are automatically installed as part of Visual Basic, or they can be installed separately.

A full installation of the Visual Studio package will satisfy all the requirements for compiling both the Shell and the DLL projects, as well as executing the Shell. This is a typical setup found in a computer lab; therefore students should have no shortage of development computers.

It is worth stating that students need not modify the image processing shell in order to develop new image transformations. For this reason, access to Visual Basic, or the image processing shell project files is not mandatory.

The minimum requirements for developing image transforms will entail that Microsoft Visual C++ 6.0, as well as the project files for the image processing DLL must be installed. This will allow compilation and debugging of the DLL file. Furthermore, to execute the image processing shell, the Visual Basic 6.0 runtime files must be installed.

For demonstrating the transforms, the requirements are even lower, with only three files needed. Only the image processing shell executable file and the image processing transform DLL file are actually needed if the computer has already installed the Visual Basic runtime files. These two files are under 300kb in size. Otherwise, the runtime installer should be accessible. The three files fit on a 1.44MB floppy disk, which makes this project transportable.

Discussion and Results

Implementation of Program

This program has been used in two semesters in a graduate level image processing course. This solution replaced previously existing code which limited students in relation to the implementation platform, proprietary architecture, and debugging capabilities.

By switching to this solution, students were allowed access to a broader selection of computers in which to develop and debug their code. They are also using tools which are more familiar, accessible, and user-friendly. As a result, student productivity has increased, just as their theoretical know how is put to test in this more practical computer platform.

Note that some of the figures presented in the text have been captured directly from the screen, and saved as bitmap files. The presented program has been used to convert them to black and white format to comply with publication requirements.

Conclusion

A fine balance is reached by a program designed with the intent to improve the research and educational qualities in the delivery and in project implementation of an image processing class. The designed program offers enough built in functionality to allow students to be productive, while the computer platform is more accommodating. The motivation remains that classroom experience for students should integrate an environment that supports hands-on experience, as has been shown from results of this study and related work such as in [13]

This program has thus been successfully used to support teaching of an image processing class. Features have been added as a result of student feedback and it has grown into a mature solution. Students have been quite satisfied with the quality of their finished work. They appreciate the flexibility offered them to develop code locally in the laboratory, or at home. The portability of the compiled solution with all the required software modules fitting on a commonly available 1.44MB floppy disk is valued as well.

In its present form, the developed program handles only bitmap files. Foreseeable modifications to the shell could be made to allow for any file to be loaded and passed as is to the image processing DLL for modification. Optional modifications have been also made to allow transforms to execute in parallel by using multi-threading technology. All these modifications can be implemented at the expense of the relative simplicity of the solution in its present form.

The source code and compiled demos for the software described in this paper can be downloaded from:
http://www.cate.fiu.edu/software/imagestudio

Acknowledgments

The authors gratefully acknowledge the support from the National Science Foundation Grants EIA-9906600 and HRD-0317692, and the Office of Naval Research Grant N00014-99-1-0952.

References

1. *National educational technology standards (NETS) and performance indicators for teachers* [Web Page]. International Society for Technology in Education 2000. [Online] Available: http://cnets.iste.org/teachstand.html.

2. *How people learn: Bridging research and practice*. Edited by S. Donovan, J. Bransford and J. Pellegrino, *Committee of Learning Research and Educational Practice, Commission on Behavioral and Social Sciences and Education,* National Research Council. Washington DC: National Academy Press.

3. U. Rajashekar, G.C Panayi, F.P. Baumgartner and A.C. Bovik, “The SIVA Demonstration Gallery for signal, image, and video processing education”, *Education, IEEE Transactions* on, vol. 45, no. 5, pp. 323-335, 2002.

4. G.W. Donohoe and P.F. Valdez, “Teaching digital image processing with Khoros”, *Education, IEEE Transactions* on, vol. 39, no. 2, pp. 137-142, 1996.

5. J. Campbell, F. Murtagh and M. Kokuer, “DataLab-J: a signal and image processing laboratory for teaching and research”, *Education, IEEE Transactions on*, vol. 44, no. 4, pp. 329-335, 2001.

6. R.H. Bamberger, “Portable tools for image processing instruction”, *Image Processing, 1994. Proceedings.* ICIP-94., IEEE *International Conference*, vol. 1, no. 1, pp. 525-529, 1994.

7. D. Roman, M. Fisher and J. Cubillo, “Digital image processing-an object-oriented approach”, *Education, IEEE Transactions on*, vol. 41, no. 4, pp. 331-333, 1998.

8. K.R. Burger, “Teaching two-dimensional array concepts in Java with image processing examples”, *Technical Symposium on Computer Science Education archive Proceedings of the 34th SIGCSE technical symposium on Computer science education*, vol. 35, no. 1, pp. 205-209, 2003.

9. T. Parveen, “PphotoSuite: a windows based digital image processing program”, *The Journal of Computing in Small Colleges archive*, vol. 19, no. 3, pp. 147-156, 2004.

10. J.A. Robinson, “A software system for laboratory experiments in image processing”, *Education, IEEE Transactions on*, vol. 43, no. 4, pp. 455-459, 2000.

11. S.L. Eddins and M.T. Orchard, “Using MATLAB and C in an image processing lab course”, *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, vol. 1, no. 1, pp. 515-519, 1994.

12. Digital Image Processing Package for Mathematica. Wolfram Research. [Online]. Available: http://www.wolfram.com/products/ applications/digitalimage/

13. K. Bowyer, G. Stockman and L. Stark, “Themes for improved teaching of image computation”, *Education, IEEE Transactions on*, vol. 43, no. 2, pp. 221-223, 1996.

Biographical Information

Alejandro Simon received his Bachelor and M.S. degree in Computer Engineering from Florida International University. He has been a research associate in the Center for Advanced Technology and Education and has contributed in the areas of Digital Logic, Distributed Computing, and Image Processing. He has developed an educational interface for the development of image processing algorithms and has helped in the setting of a distributed cluster that led to research work in .NET Web Services.
Email address: Alejandro.Simon@4XLab.NET

Malek Adjouadi is a joint faculty member with the department of Electrical and Computer Engineering and Biomedical Engineering at Florida International University. He is also the founding and current director of the Center for Advanced Technology and Education (NSF-CATE) funded by the National Science Foundation since 1993. He received his BS degree in EE from Oklahoma State University and his MS and Ph.D. degrees also in EE from the University of Florida. Malek's earlier work on computer vision to help blind persons led to his testimony to the US Senate Committee of Veterans Affairs on the subject of technology to help disabled persons. Malek is also co-leading with Dr. Prasanna Jayakar the joint neuro-engineering program between FIU and Miami Children's Hospital in researching methods for understanding key brain dysfunctions such as epilepsy. His research interests are in vision-based guidance systems, machine vision applications, biomedical imaging and diagnostics, and man-machine interfaces to help people with disabilities (visual and motor).
Phone: (305) 348 – 3019, fax: (305) 348 – 3707
Email address: adjouadi@fiu.edu