# IMAGE PROCESSING FOR EVERYONE

George C Panayi, Alan C Bovik and Umesh Rajashekar

Laboratory for Vision Systems,
Department of Electrical and Computer Engineering
The University of Texas at Austin, Austin, TX 78712-1084 USA
{panayi, bovik, umesh}@ece.utexas.edu

***ABSTRACT***

*The techniques of digital image processing have found a myriad of applications in diverse fields of scientific, commerical, and technical endeavor. Image processing education therefore needs to cater to a wide spectrum of people coming from different educational backgrounds. In this paper, we describe tools and techniques that facilitate a gentle introduction to image processing. We present novel LabVIEW-based image processing demonstrations that, when supplemented with web-based class lectures, illustrate the power and beauty of image processing algorithms.*

## 1 Introduction

Digital Image Processing (DIP) is a multidisciplinary science that borrows principles from diverse fields such as optics, surface physics, visual psychophysics, computer science and mathematics. The many applications of image processing include: astronomy, ultrasonic imaging, remote sensing, video communications and microscopy, among innumerable others.

In this paper, we discuss teaching visualization tools developed for the EE 371R - *Digital Image and Video Processing* course offered every Fall semester at the University of Texas at Austin. The aim of the course is to make DIP accessible to an audience of fairly mixed backgrounds, using numerous visual examples to supplement the theory with reasonable mathematical simplicity. Though designed for an undergraduate Electrical Engineering curriculum, EE 371R attracts many undergraduate and graduate students from various departments, such as geology, psychology, astronomy, computer science (to name a few) as well as local industry. Image acquisition, image processing theory, and practical applications are introduced from an operational perspective with some exposure to theory. To encourage a web-based educational system, all course material and visualization tools are made available to the students for downloading so that they can experiment with the tools at their leisure. MATLAB based assignments help reinforce the concepts. The *"Best Project Award"* for the best DIP class project (with a $100 first prize) motivates students to investigate and apply the concepts learned in the course to their respective fields. Beginning this Fall, students will also be provided camcorders and webcams to develop their projects. Introductory material covered in the course includes binary image processing, image analysis, and image enhancement, while the more advanced material covers Hough transforms, edge detection and video processing. Since visualization is invaluable to interpret the concepts, lecturing is accompanied with "in-class" LabVIEW demonstrations to illustrate the concepts being discussed.

Section 2 of this paper gives a brief overview of LabVIEW and its basic image processing functions, such as those embodied in the IMAQ Vision modules. In Section 3, we explain some of the visualization modules we have created and we conclude in Section 4.

## 2 LabVIEW

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a graphical programming language used as a powerful and flexible instrumentation and analysis software system in industry and academia. LabVIEW uses a graphical programming language - *G* to create programs called Virtual Instruments or VI (pronounced *vee-eye*) in a pictorial form called a block diagram, eliminating a lot of the syntactical details of other programming languages like C and MATLAB that use a text based programming approach. LabVIEW also includes many tools for data acquisition, analysis and display of results. The analysis library contains a multitude of functions in signal generation, signal processing, filtering and statistics. LabVIEW is available for all the major platforms and is easily portable across platforms. Each VI contains 3 parts:

1) The *front panel* contains the user interface like knobs, push buttons, graphs and many other controls (inputs) and indicators (outputs). Inputs can be fed using the mouse or the keyboard. *Fig 1a* shows a typical front panel.

2) The *block diagram* shown in *Fig 1b* is the VI's source code constructed in *G* and is the actual executable program. The block diagram has other lower-level VIs and built in functions. The blocks can be connected using wires to indicate the dataflow. Front panel objects have corresponding terminals on the block diagram to allow dataflow from the user to the program and back to the user.

3) Sub-VIs are analogous to subroutines in conventional programming languages.

[1] provides an introduction to LabVIEW. [2] provides more information on LabVIEW. In some applications such as image processing, execution speed is critical. LabVIEW is the only graphical programming system with a compiler that generates optimized code with execution speeds comparable to compiled C programs. Thus, LabVIEW has the ability to create stand-alone executable applications, which run at compiled execution speeds. Another advantage of LabVIEW is the fact that it includes built in applications, such as the IMAQ Vision for image processing. IMAQ Vision includes more than 400 imaging functions and interactive imaging windows and utilities for displaying and building imaging systems. IMAQ Vision gave us the opportunity to create examples for all the important functions in image processing, and use them for educational purposes. [3] provides information on IMAQ Vision.

## 3 Visualization modules

The front panel provides an excellent intuitive graphical user interface (GUI) to vary various parameters in the algorithm. These GUIs resemble the controls on many instruments and provide a user-friendly interface (hence the name Virtual Instrument).

We have developed a wide range of VI's that can be used in conjunction with class lectures. In this section, we describe few of the LabVIEW VIs that we developed. The reader can download other VI's from: http://pineapple.ece.utexas.edu/class/ee371r/Modules/demos.htm. [4] provides detailed information on how the VI s were created.

### *3.1 Analog to digital conversion*

Sampling and quantization help to transform the continuous domain image into a digital format. The effects of sampling and quantization can be visualized effectively with the following two VI's. This is a basic concept that must be understood by any practitioner in any field utilizing digital images.

#### 3.1.1 Quantization

The Quantization VI (front panel in *Fig 2a*) demonstrates the effects that different quantization levels have on images. This VI reads in an 8-bits/pixel image and creates an output image whose number of bit levels is

specified by the input parameter Number of bits. The user has the options to create images that have 1, 2, 4, or 8 bit levels. *Fig 2b* shows the effects of quantization.

3.1.2 Sampling

To demonstrate the effects that different sampling rates have on images, we created the Sampling VI whose front panel shown in *Fig 3a*. While there is a mathematical theory of sampling, the intuitive aspects of sampling can be understood by a diversity of users by visual observation of its effects. This VI reads in an image and sub-samples it to the user-specified size. The user has the option to sub-sample the image to the sizes of 256*256, 128*128, 64*64, and 32*32, using our VI. *Fig 3b* shows the sub-sampling of the input image to the specified size. The sub-sampled image is scaled to the size of the input image, by duplicating columns and rows of the sub-sampled image. In this image, we can see the effects that sampling has on images. The effects of sampling become more pronounced when the sampling rate is decreased. Aliasing effects can be demonstrated when severe undersampling occurs.

*3.2 Binary Image processing*

Binary images have only two possible "gray levels" and can be represented using only one bit per pixel. Besides developing VIs for thresholding grayscale images to binary, we also demonstrate the effects of binary morphology. Morphological operations are defined by moving a structuring element over the image to be modified, in such a way that it is centered over every image pixel at some point. When the structuring element is centered over a region of the image, a logical operation is performed on the pixels covered by the structuring element, yielding a binary output. We created the Morphology VI for demonstrating the effects of various morphological operations on binary images. In this VI, we implemented the seven following morphological operations: Median, Dilation, Erosion, Open, Close, Open-Clos, and Clos-Open. The user has the option to test all the above morphological operations on an image, and also to modify the type and size of the structuring element. The Morphology VI allows the following structuring element types: Row, Column, Square, Cross, and X-Shape.

*3.3 Histogram and point operations (Grayscale)*

We developed VIs that perform linear (offset, scaling and full-scale contrast stretch) and non-linear (logarithmic range compression) image point operations. The VI for a linear point operation is shown in *Fig 4a*. The user has the option to perform the offset and scaling operations or to perform a full scale contrast stretch operation on an input image. The histograms of the input image and the image after the linear point operation are also displayed in the front panel in the Histogram and New Histogram Waveform Graph indicators. Effects of linear point operations on an image are shown in *Fig 4b.*

*3.4 Image analysis (Frequency interpretations)*

3.4.1 Discrete Fourier Transform (DFT)

The Fourier transform is fundamental to image filtering and spectral theory, yet is a difficult concept for many users to correctly grasp. We therefore begin by teaching the concept of digital frequency and digital sinusoidal gratings (2-D sine waves) as the basis functions for the DFT. We have also constructed the FFT (Fast Fourier Transform) demonstration VI that calculates and displays the magnitude and the phase of the DFT for gray level images. The front panel of this VI is shown in *Fig 5a* Usually, the DFT is displayed with its centered coordinates (u, v) = (0, 0) at the center of the image. This way, the lower frequency information is clustered together near the origin at the centered of the display. The Center/ Uncenter input parameter of the front panel specifies whether the DFT will be displayed

with its low frequencies clustered together at the center of the image or not. It is also the case that the low-amplitude frequencies in the magnitude of the DFT will be hard to see, thus it is best to logarithmically compress the DFT magnitude prior to display. An option is given again for displaying the log compressed or uncompressed magnitude of the DFT. This is specified by the Compr/Uncompr input parameter of the front panel. *Fig 5b* shows an image and the magnitude and phase of the DFT.

3.4.2 Directional DFTs

When the DFT of an image is brighter along a specific orientation, the image contains highly oriented components in that direction. Suppose that we define several oriented zero-one images. Masking the DFT with these images will produce IDFT images with only highly-oriented frequencies remaining. To demonstrate this effect, we implemented the DFT Direction VI. The input parameters, Theta 1 and Theta 2, to this VI, are used as the starting and final angles (in degrees) respectively, of the black region in the mask images as it can be seen from the "Circle" illustration in the front panel in *Fig 6a.* Results of DFT masking are shown in *Fig 6b.*

The other VI s developed includes linear and non-linear filtering, image compression schemes and a large number of edge detectors. We have presented the simplest ones here for the purpose of illustration.

## 4 Conclusions

In this paper, we gave an overview of Digital Image Processing education at UT-Austin. We describe the use of powerful visualization tools developed using LabVIEW for image processing. An overview of LabVIEW and a few of the demonstrations developed were provided.

## 5 References


1. Lisa K. Wells and Jeffrey Travis, "*LabVIEW for Everyone, Graphical Programming made even easier*," Prentice Hall, 1997.
2. ___,"*Labview User Manual*," National Instruments, 1996.
3. ___, "*BridgeVIEW and LabVIEW IMAQ Vision for G reference manual*," National Instruments, 1996.
4. George C Panayi, "*Implementation of Digital Image Processing functions using LabVIEW*," Master's thesis, UT-Austin 1999.


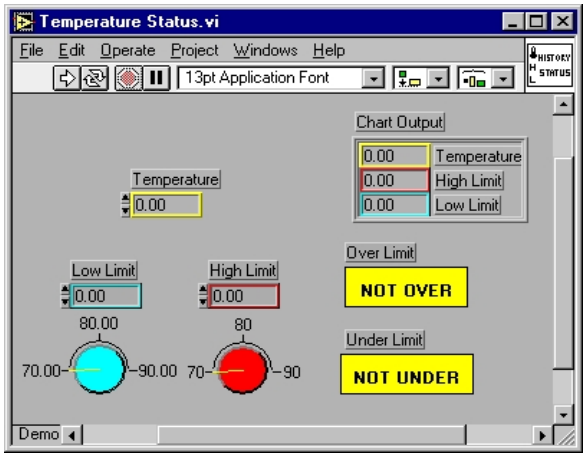


***Fig 1a: Typical front panel***

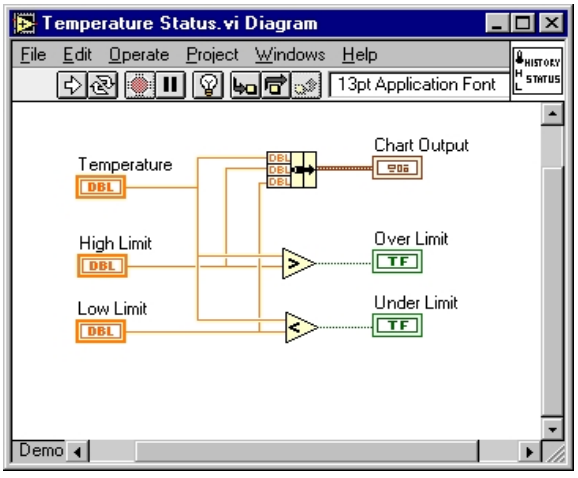


***Fig 1b: Block diagram for front panel***

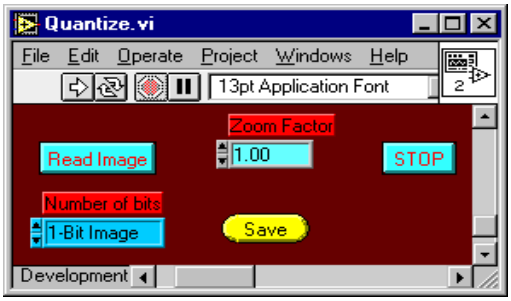


*Fig 2a: Front panel for Quantization*

*Fig 2b: Original (left), Quantized(right)*

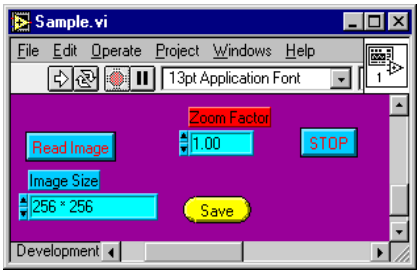


*Fig 3a: Front panel for sampling*

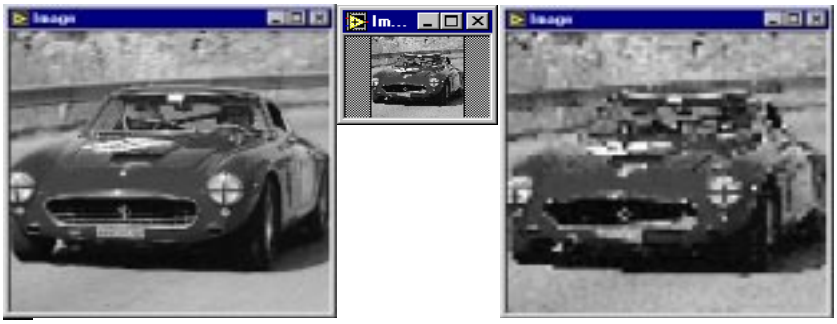

*Fig 3b: Original(left), sampled(middle), interpolated (right)*

*Fig 4a: Front panel for linear point operations*

*Fig 4b: Original(left), Offset and Scaling(middle), Full Scale stretch(right)*

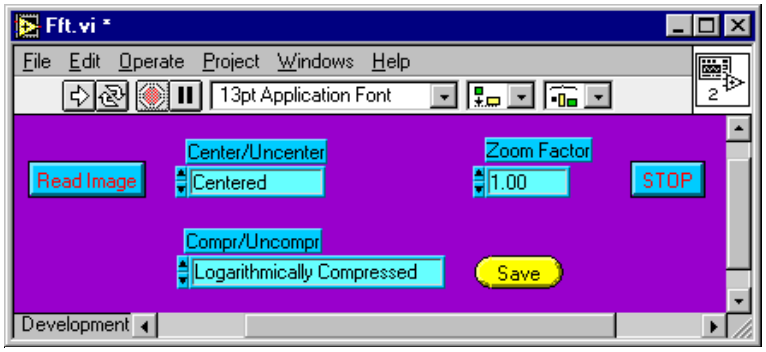


*Fig 5a: Front panel for DFT*

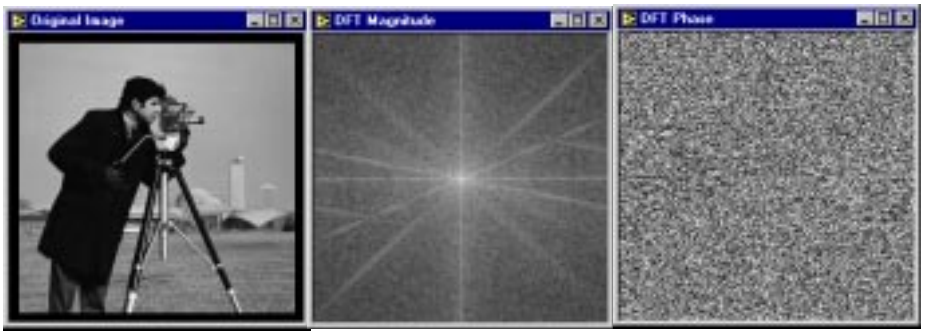


*Fig 5b: Original(left), DFT Magnitude(middle), DFT Phase(right)*

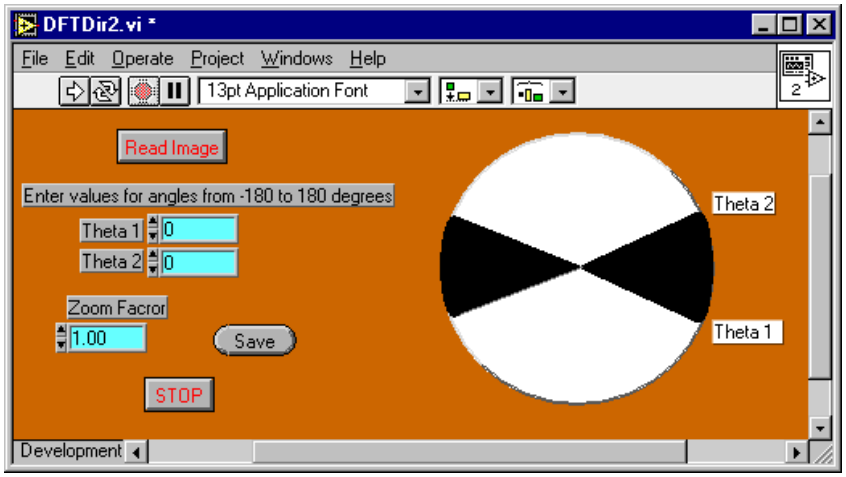


*Fig 6a: Front Panel for directional DFT*

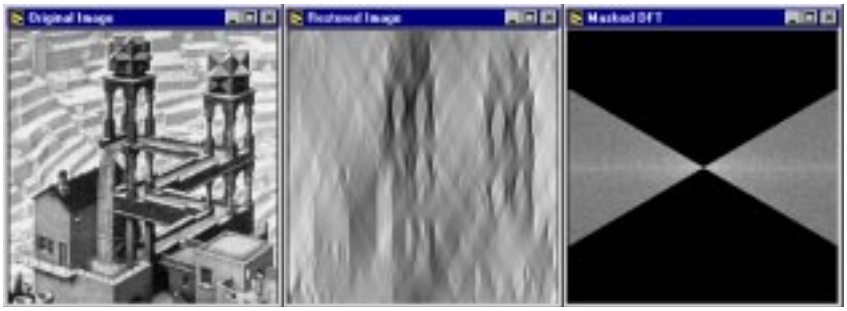


*Fig 6b: Original (left), Result of directional DFT(middle), regions selected(Right)*