2.4. IMAGE PROCESSING EDUCATION

Umesh Rajashekar & Alan C. Bovik
The University of Texas at Austin, USA

Daniel Sage & Michael Unser
Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland

Lina J. Karam
Arizona State University, USA

R. (Inald) L. Lagendijk
Delft University of Technology, The Netherlands

Contents

List of Figures

1 Introduction

The principles of digital image processing (DIP) have found applications in an amazing diversity of areas, such as astronomy, genetics, remote sensing, video communications, and biomedicine to name a few. Clearly, image processing education clearly needs to cater to a wide spectrum of people from different educational backgrounds. Although well rooted in advanced mathematics (probably unfamiliar and often superfluous to a majority of the general image processing audience), the theory of DIP needs to be made *'accessible'* to practitioners from diverse backgrounds. Complicating this is the highly multidisciplinary nature of DIP - the field draws upon a great variety of areas such as mathematics, computer graphics, computer vision, visual psychophysics, optics, and computer science. Presenting such an inter-disciplinary topic with perspicuity to a heterogeneous audience is challenging. With the recent trend of signal and image percolating lower down the curriculum to even high school, as in the Infinity project [1] and the importance of designing signal processing as a first course in Electrical and Computer Engineering [2], the tools and techniques to present signal and image processing with minimal math to a non-technical audience is a desideratum.

Image processing is a subject that lends itself to a rigorous, mathematical treatment and which, depending on how it is taught, is often perceived as being rather theoretical. Two-dimensional digital signal processing concepts are particularly challenging to students who are mainly exposed to one-dimensional signals and single-variable functions throughout their high-school and undergraduate education. In contrast, we are immersed in a spatio-temporal "multi-dimensional" world and we are constantly exposed on a daily basis in our life to multi-dimensional signals in the form of images, video, three-dimensional audio and visual signals, for example. Since engineering or applied science students are often more interested in application than in pure theory, it is strongly advisable for instructors to complement their courses with hands-on experimentation with real images [3], eventually encouraging the students to program simple algorithms. It is proven that students gain a lot in their understanding when they can actively manipulate and process images [4] [5]. To reinforce concrete fundamental concepts, most introductory courses assign computer-based exercises. However, more often than not, the learning curve involved in becoming familiarized with the software detracts the student from assimilating the concept. More effective techniques to uncover the intuition behind 'murky' equations, are visualization tools that facilitate for aural and visual consumption of information. A ready-to-use set of demonstrations illustrating the concepts that the instructor deems important can, therefore, help the student to begin experimenting and assimilating immediately without having to bother about programming intricacies. This situation also encourages students to experiment with their desired inputs at their leisure. Further, such tools bolster the success of distance learning by facilitating interactive education and are the topic of discussion in this chapter.

There have been many significant contributions to the general area of signal processing

educational tools, developed primarily with MATLAB [6–10], LabVIEW [11, 12], Java [13–15], and Mathematica [16]. Common Gateway Interface has also been used for handling signal-processing routines, with Java used for the user interface [17]. See Section 4.19 of this handbook for an overview of various other software for image and video processing. In general, the ease of programmability, cost of software, availability of inexpensive student versions of the software , flexibility of extending the education tools to an online community, execution speed (especially for video processing), and operating system independence are some of the desired factors to be considered in the development of education tools.

This chapter describes tools and techniques that facilitate a gentle introduction to fascinating concepts in digital image processing from four leading universities. Equipped with informative visualizations and a user-friendly interfaces, these modules are currently being used effectively in a classroom environment for teaching DIP at many universities across the world. In particular we present:

- IPLab - A java-based plug-in to the popular ImageJ software from the Swiss Federal Institute of Technology Lausanne, Switzerland
- ALMOT 2D DSP and 2D J-DSP - Two java-based education tools from Arizona State University, USA
- SIVA - A LabView-based education tool from the University of Texas at Austin, USA
- VcDemo - A Microsoft Windows-based interactive video and image compression tool from the Delft University of Technology, The Netherlands

2 IP-LAB: A tool for teaching image-processing programming in Java using ImageJ

Daniel Sage and Michael Unser, Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland

This section is devoted to the description of IP-LAB, a didactic tool for teaching the practical and computational aspects of image processing (IP). IP-LAB is a software suite and a collection of computer laboratories (assignments) that were designed to complement the basic lectures of an IP course.

IP-LAB provides a framework where the students can experiment themselves with image-processing algorithms, change the parameters and get immediate visual feedback, and also write their own code down to the pixel level. IP-LAB is based on a general-purpose image-processing software called ImageJ. ImageJ is written in Java and freely available. It has an open architecture that allows the addition of plug-ins. IP-LAB takes advantage of this functionality; it adds to it a programmer-friendly software layer that greatly facilitates the encoding of image processing algorithms in Java. Even though a basic knowledge of the Java syntax is required to implement standard IP algorithms, this knowledge comes at essentially no effort from the part of the student. Thanks to the interface layer (ImageAccess) that has been developed and to the learning-by-example strategy, the laboratories require little

programming skills; they also take full advantage of the user-friendly interface of ImageJ.

2.1 Digital Image Processing Education (SMT, SIN, EDNE) at the Swiss Federal Institute of Technology Lausanne (EPFL)

IP-LAB was developed by the Biomedical Imaging Group (BIG) at the Swiss Federal Institute of Technology in Lausanne (EPFL). IP-LAB covers the standard image-processing methods. It is freely available at http://bigwww.epfl.ch/teaching/iplabsite/.

Up to now, three courses at the Swiss Federal Institute of Technology Lausanne use IP-LAB as a complement to the basic lectures:

- Image processing I and II (EPFL Master in Micro-Engineering, SMT)
- Computer vision (EPFL Master in Computer Science, SIN)
- Image acquisition and processing in biology (EPFL Doctoral School in Neurosciences, EDNE)

Students who choose to complete a practical semester or diploma project with EPFL's Biomedical Imaging Group also heavily rely on the IP-LAB concept. They use the environment to develop their new image-processing algorithms for real-world applications. The IP-LAB framework can also be used to produce demonstration applets on the Internet. The IP-LAB software is probably also used at other institutions since it is available over the web: over 350 potentially interested instructors have subscribed so far (as of June 2004) to obtain full access to the system.

2.2 Description of IP-LAB

2.2.1 Image-processing hands-on and image-processing programming

The learning process is greatly facilitated when the students participate in hands-on experimentation. This is the first level of involvement: applying algorithms to real images and observing the results. This can be achieved easily by providing a user-friendly computer environment that allows trying out different algorithms and visualizing their results. Such experimentation is feasible on virtually any standard image-processing software.

The second level of involvement is more ambitious. It is to have the students take part in the programming itself and have them translate formulas into algorithms and code. Here, the education requirements are the following:

- The best way to understand an algorithm is obviously to code it and to test it. Students should get the opportunity to implement the most representative algorithms.
- The exercises should be accessible to inexperienced programmers. The assignment should concentrate on image-processing issues alone and not on the programming skills.
- The code should be as generic as possible, close to the formulas.

- The programming should be simple, robust and convivial (graphical user interface and input/output task).
- The edit-compile-execute programming cycle should be short to see immediate effects on the images when modifying the code.

These features are not easy to achieve with standard software or programming languages. IP-LAB is a contribution to the initiatives of providing pedagogical tools that fulfill these requirements.

2.2.2 Foundation of IP-LAB

The IP-LAB framework proposes a computing environment where the students can implement the algorithms literally as they are seen in the course [18]. The system is based on the Java language, on ImageJ, and on a specific student-friendly interface layer called ImageAccess (Section 2.2.3) that simplifies the access to pixel data without having to worry about technicalities and the interfacing with ImageJ. The justifications for these choices are as follows.

Java Language

- Java is free, platform-neutral, well-adapted to the diversity of the student community.
- Java is an object-oriented language; it is ready for signal and image processing applications.
- Java is robust with a good handling of errors and garbage collection; this eliminates the main source of bugs.
- Java is syntactically close to C, a well-known language.
- Java is reasonably fast: applying a 3*3 filter takes only a fraction of a second on a 512*512 pixels image.
- Java is attractive for students: Java plays a major role on the Web, it is a modern and fashionable language.
- Java has a mechanism for loading classes dynamically after each modification of the code; this functionality offers a fast and comfortable way to edit-compile-execute a program.

ImageJ

- ImageJ is a powerful, full-featured image-processing program developed at the NIH [19]; it is freely available.
- ImageJ has a graphical user-interface which provides convivial interaction with images.
- ImageJ has an open architecture that allows extensibility by addition of plug-ins.

2.2.3 ImageAccess: the foolproof interface layer

Since the programming of ImageJ plug-ins was not originally meant for beginners, we have developed a Java class called ImageAccess which is the key component of IP-LAB. Using this

class, an image becomes simply an instance of the ImageAccess class, providing high-level, robust functions to access to the pixels. In this context, three kinds of abstraction were defined to help producing pure source code that is close to the original textbook algorithm.

1. Type abstraction: the pixel data is always retrieved and stored in the 64-bits floating-point format (double in Java), independently of the underlying image type (often 8-bit). In this way, students do not have to worry about rounding, truncation, or conversion of pixel data and results are always high-precision
2. Spatial abstraction: the pixel data can be accessed anywhere through the use of consistent mirror symmetric boundary conditions. For example, when a student wants to retrieve a 3*3 block of an image centered on the upper left corner (0,0), the interface layer provides a full block with outside pixel values that are correctly extrapolated. This frees the student from having to worry about what happens at the boundaries and results in more pleasant output images without border artifacts.
3. Read/write pixel abstraction: the typical way to program is to retrieve an image block by using a method that begins with get...(). The block is processed and the result is written in the output image using a put...() method. The block can be a single pixel, a row, a column, a neighborhood window. It is useful for implementing filters that need to access the data in the neighborhood of pixel.

Conceptually, there is a clear pedagogical advantage in separating the image-processing code from the access to the pixels as much as possible. However, this is not the approach taken in ImageJ because is has a computational cost associated with it. As a result, the typical image-processing routines in ImageJ are faster than their IP-LAB counterparts, but they are also significantly more complicated. We consider the overhead an acceptable price to pay for the substantial simplifications in algorithm transcription. Figure 1 shows the overhead time for a standard function of 2D FIR filtering. It compares the built-in function of ImageJ and a convolution routine written using ImageAccess (non-separable implementation) for various lengths of the filter. Note that the ImageAccess routine includes data conversion, pixel access, and implementation of the boundary conditions.

In the case of a separable algorithm where rows and columns are processed in succession, the cost of the access is fixed (about 25 ms), irrespective of the kernel size.

2.2.4 IP-LAB sessions

A laboratory session, which is usually three-hours long, is typically devoted to one chapter of the course. It contains a programming part and an experimental part. Currently, the following sessions are available:

1. Pointwise operations for (a) engineers and (b) biologists
2. Morphology for (a) engineers and (b) biologists
3. Digital filtering for (a) engineers and (b) biologists

4. Fourier transform
5. Edge detection
6. Hough transform
7. Interpolation and geometric transformation
8. Wavelet transform
9. Tomography and backprojection
10. Image Restoration and deconvolution

For the programming part, an example of a Java method that does an operation that is similar to the assignment is always provided. In addition, the student code is structured by providing empty templates that need to be completed. This means that a good portion of the assignment can usually be implemented by simple modifications of the example.

2.3 Examples

2.3.1 Maximum intensity projection

This is an example of a function that produces the maximum intensity projection (MIP) image of a z-stack of images (see Listing 2.3.1). It illustrates how easy it is to access a single pixel using the ImageAccess methods: getPixel() and putPixel().

Listing 1: Maximum intensity projection

```
public ImageAccess MIP(ImageAccess[] zstack) {
        int nx = zstack[0].getWidth();
        int ny = zstack[0].getHeight();
        int nz = zstack.length;
        ImageAccess output = new ImageAccess(nx, ny);
        double max = 0.0;
        for (int x=0; x<nx; x++)
        for (int y=0; y<ny; y++) {
            max = zstack[0].getPixel(x, y);
            for (int z=1; z<nz; z++) {
                if (max < zstack[z].getPixel(x, y)) {
                    max = zstack[z].getPixel(x, y);
                }
            }
            output.putPixel(x, y, max);
        }
        return output;
}
```

2.3.2 Moving-average 5*5

The next example is a moving-average filter that produces a smoothed output image. We compare two implementations of this digital filter using a non-separable approach (see Listing 2.3.2) and a separable algorithm (see Listing 2.3.2). This code illustrates the access to a block of pixels of an image: methods getNeighborhood(), getRow() and getColumn(). The code is relatively straightforward; it is essentially a literal translation of the textbook version of the algorithm.

Listing 2: Non-separable filtering

```
public ImageAccess MA5x5_NonSeparable(ImageAccess in) {
    int nx = in.getWidth();
    int ny = in.getHeight();
    double arr[][] = new double[5][5];
    double sum=0.0;
    ImageAccess out = new ImageAccess(nx, ny);
    for (int x = 0; x < nx; x++)
    for (int y = 0; y < ny; y++) {
        in.getNeighborhood(x, y, arr);
        sum = 0.0;
        for (int k = 0; k < 5; k++) {
            for (int l = 0; l < 5; l++) {
                sum = sum + arr[k][l];
            }
        }
        sum = sum / 25.0;
        out.putPixel(x, y, sum);
    }
    return out;
}
```

The separable implementation offers many advantages in terms of computation time and modularity. The code, which is generic for the most of part, clearly shows the two loops: the first one, which scans the rows, and the second one, which scans the columns. The only specific part is the 1D routine average5(), which can easily be modified to yield other separable filters.

Listing 3: Separable filtering

```
public ImageAccess MA5x5_Separable(ImageAccess in) {
    int nx = in.getWidth();
```

```
    int ny = in.getHeight();
    ImageAccess out = new ImageAccess(nx, ny);
    double rowin[]  = new double[nx];
    double rowout[] = new double[nx];
    for (int y=0; y<ny; y++) {
        in.getRow(y, rowin);
        average5(rowin, rowout);
        out.putRow(y,rowout);
    }
    double colin[]  = new double[ny];
    double colout[] = new double[ny];
    for (int x=0; x<nx; x++) {
        out.getColumn(x, colin);
        average5(colin, colout);
        out.putColumn(x,colout);
    }
    return out;
}

private void average5(double vin[], double vout[]) {
    int n = vin.length;
    vout[0] = (2.0*vin[1]+2.0*vin[2]+vin[0])/5.0;
    vout[1] = vout[0]+(vin[3]-vin[2])/5.0;
    vout[2] = vout[1]+(vin[4]-vin[1])/5.0;
    for (int k=3; k<n-2; k++) {
        vout[k] = vout[k-1]+(vin[k+2]-vin[k-3])/5.0;
    }
    vout[n-2] = vout[n-3]+(vin[n-2]-vin[n-5])/5.0;
    vout[n-1] = vout[n-2]+(vin[n-3]-vin[n-4])/5.0;
}
```

2.3.3 Canny edge detector

Figure 2 shows a typical example of the application of the well-known Canny edge detector [20]. Here, the students can dissociate the various modules and see the step-by-step evolution of the image during the various phases of the algorithm.

2.3.4 Wavelet transforms

Another interesting example is the implementation of a separable wavelet transform of an image (see Figure 3). The figure shows the higher level part of the code for the analysis, which

is primarily a loop over the scale (number of iterations). The methods getSubImage() and putSubImage() of ImageAccess are useful for manipulating the data (wavelet coefficients) in the various subband of the wavelet decomposition

2.4 Conclusion

The IP-LAB computer laboratories have received extremely positive feedback from the students. It helps students become more active participants in the image-processing course. They get more interested in processing images when they see the results right away, and they also enjoy implementing image-processing algorithms. The proposed computer sessions reinforce the learning process and the motivation of the students. Based on our experience, we believe that IP-LAB computer laboratories are a perfect complement to a theoretical course on image processing. The IP-LAB sessions take advantage of the robustness and the platform independence of Java. They make good use of the friendly graphical user interface and the domain-public licensing of ImageJ. IP-LAB makes image-processing programming accessible to inexperienced programmers thanks to the ImageAccess layer and a learning-by-example strategy. Finally, the available system offers an attractive and professional-level software and the more motivated students can continue to work with this environment. The framework is able to solve relevant imaging problems inspired from real-world applications.

3 Java-based Educational Software for Image and Two-Dimensional Signal Processing

Lina J. Karam, Arizona State University, USA

This section describes interactive, platform-independent, educational software and on-line laboratories that were developed and are used at Arizona State University for teaching the theory of two-dimensional signals and systems and its application to processing digital imagery.

Two java-based educational 2D DSP software tools are described. The first tool, called ALMOT 2D DSP, is based on Active Learning through MOdular Testing (ALMOT), and is targeted toward users with no a-priori knowledge or background in the signals and systems area, such as high-school students. The second tool, called 2D J-DSP, is based on a data-flow object-based visual environment and is targeted toward undergraduate- and graduate-level engineering students who want to gain a deeper understanding of the 2D DSP theory through on-line laboratories and experimentations.

The ALMOT 2D DSP software has been used in several courses at Arizona State University including the freshman-level *Introduction to Engineering (ECE100)* course, the junior-level *Signals and Systems (EEE 303)* course, and the graduate-level *Multi-dimensional Signal Processing (EEE 507)* and *Digital Image Processing and Compression (EEE 508)* courses. In addition, the ALMOT 2D DSP software has been used by first-year high-school students. Section 3.2 provides more details about the ALMOT 2D DSP software.

The 2D J-DSP software has been used and evaluated as part of the graduate-level *Multidimensional Signal Processing* (EEE507) course. More details on the 2D J-DSP software are provided in Section 3.3.

3.1 Digital Image and Video Processing Education at Arizona State University

Currently, two graduate-level courses *EEE507: Multi-D Signal Processing* and *EEE508: Digital Image Processing and Compression* are offered in the Electrical Engineering Department at Arizona State University and are important to understanding the area of digital image processing.

The EEE507 course is concerned with understanding signals of more than one variable and with systems for processing them. These signals include images, video, in addition to other signals such as multivariate data, sonar, and radar signals. This course covers the fundamental concepts and theory in the area of multi-dimensional signal processing including multi-D sampling, multi-D transforms, multi-D filter design and implementations, processing of propagating space-time signals, multi-D signal restoration and reconstruction.

The EEE508 course focuses on the area of image and video processing and is concerned with understanding the fundamentals of digital image perception, representation, processing, and compression.

In addition to on-campus students with majors in electrical engineering, bioengineering, and computer science, these courses attract off-campus "on-line" students who are typically working professionals. In order to cater to the needs of the distance learners, the lectures of these courses are streamed on-line and all course material including lectures, assignments and demonstrations are available to students over the world wide web; Blackboard is also used to post the course material and announcements.

As indicated earlier and as discussed in Section 3.2, image processing concepts and applications have also been introduced to freshman and junior-level students at ASU in addition to local high-school students.

3.2 ALMOT 2D DSP

The ALMOT 2D DSP software was developed using HTML and Java to provide convenient platform-independent ubiquitous learning. In its current form, the software consists of an interactive tutorial on 2D convolution and how it is used to implement 2-D linear systems with emphasis on image processing applications.

ALMOT 2D DSP includes several important features that provide effective interactive learning. These features include a user-friendly, point-and-click, graphical user interface with context-sensitive help at each level of the tutorial, visual animated illustrations and, in particular, the incorporation of Active Learning through the use of MOdular Testing (ALMOT).

Active learning, in an interactive environment which provides feedback to learners based on the results of their actions, has been shown by science education research as an essential component for the development of effective teaching tools [21]. Also, studies have shown that learners retain 25% of what thet hear, 45% of what they see and hear, and approximately 70% when they actively participate in the process [22].

Active learning and modular testing are implemented in ALMOT 2D DSP by dividing the tutorial into a series of modules (also referred to as levels). These modules are arranged in the order of increasing level of interactivity, flexibility, and experimentation, similar to computer games. This would motivate the students to complete each level in order to move into the subsequent more interesting (and more fun) levels. In addition, the students have to answer correctly one or more questions for each module before they can go to the next one. The software provides the students with immediate feedback after answering a question or making a selection, and with helpful explanation if they could not answer the questions correctly. So, the students cannot passively scan through the electronic tutorial pages, but have to demonstrate understanding of the illustrated concepts in the current module, answer the questions correctly in order to move on to the next module. Figs. 4–5 present excerpts from the developed 2D DSP tutorial.

As indicated earlier, the web-based ALMOT 2D DSP software have been used and tested in freshman-, junior-, and senior-level courses at Arizona State University as well as by a class of first-year high-school students at Carl Hayden Community High School in West Phoenix. All the participating students were asked to complete pre-evaluation and post-evaluation forms consisting of questions and feedback sections. The evaluation results were very positive [23]. All the partcipating students indicated that they were interested in learning more about image processing after completing the on-line tutorial. Approximately 95% of the students indicated that the proposed web-based ALMOT 2D DSP was better than learning from a textbook. The remaining 5% indicated that it was as good as learning from a book.

The software can be accessed at http://www.fulton.asu.edu/ image/almot2dsp/.

3.3 2D J-DSP

Java Digital Signal Processing (J-DSP) [24, 25] is a java-based object-oriented signal processing educational software and programming environment that was developed at Arizona State University (ASU) for use in undergraduate- and graduate-level engineering classes and for distance learning. J-DSP resides on the web as a platform-independent java applet, and is thereby easily accessible using a web browser.

J-DSP provides a complete simulation environment and doesn't require any programming experience. Its user friendly environment helps students to establish and execute simulations easily and without the need to spend time and efforts on programming details. These simulations are well-illustrated using graphics to understand the complex mathematics that

describe fundamental and advanced signal processing concepts.

2D J-DSP [26] was developed as a software extension to J-DSP in order to incorporate educational tools and on-line laboratories for two-dimensional digital signal processing. Two-dimensional DSP capabilities in 2D J-DSP include 2D signal generation, 2D FIR filter design and implementation, and 2D transforms. In addition, the developed 2D J-DSP environment offers image processing capabilities including image restoration and enhancement. In order to illustrate 2D concepts graphically, contour (2D) and perspective (3D) plots have also been incorporated into 2D J-DSP. On-line laboratory exercises have been developed based on 2D J-DSP for use in the graduate-level Multidimensional Signal Processing (EEE 507) and Image Processing courses (EEE 508) at ASU. The 2D J-DSP simulation environment enables students to establish and execute 2D DSP and image processing simulations. These simulations are well-illustrated using graphical tools and can help students understand advanced 2D digital signal processing concepts.

Figure 6 shows the 2D J-DSP editor frame. It is divided into three main areas. The first is the white colored area, called the graph panel or working area where the simulations are performed; the second is the Error/Warning area that shows general messages related to the 2D J-DSP functionality; and the third area is where the 2D J-DSP blocks are shelved. There are different categories of these blocks. The blocks placed in a horizontal line on top of the editor frame are depicted in a yellow color and are changeable, while the blocks placed in a vertical line on the left-hand side of the editor frame are shown in green color and are called fixed blocks. The yellow-colored blocks are selected from a drop-down menu list called block-sets, which can be seen on the top left corner of the editor frame. These blocks are grouped together according to their functionalities and placed under one block-set as shown in Figure 6. For example, the 2D Transforms block-set contains 2D FFT, 2D DCT and 2D Wavelet forward and inverse transform blocks.

A number of 2D blocks have been developed, in 2D J-DSP, that cover various areas of 2D signal and image processing. The blocks are grouped together according to their functionality, and are called block-sets. The following block-sets are available: 1) Basic Blocks, 2) 2D Filters, 3) 2D Transforms and, 4) Filter Banks. The Basic block-set covers blocks for 2D signal generation, display, and manipulation, 2D output and 2D arithmetics. The 2D Filters block-set covers 2D FIR Filter Design and Implementation blocks. The 2D Transforms block-set covers some of the popular 2D transforms, and the Filter Bank block-set covers 1D blocks that operate on the 2D images for filter bank analysis including row/column filtering and up/down sampling. A detailed description of the 2D J-DSP blocks can be found in [27].

The 2D J-DSP software provides a number of ways to view 2D signals and to view the output and characteristics of 2D systems. These include sample view, image view, contour plot and perspective plot. Figure 7 shows the perspective plot for a 2D FIR filter that is designed using a non-separable Hamming window. The perspective plot is generated as a 3D mesh using Java 3D [28]. A color scheme is used to display the frequency response, ranging

from red to blue. Red color shows the maximum frequency response, while the blue color shows the minimum frequency response of the 2D LSI systems. A View window enables the user to view the numerical values of the frequency response of the designed 2D FIR filter.

An example of a 2D J-DSP simulation is shown in Figure 8. Fig. 8 also shows the dialog windows for the 2D Images block and for the 2D plot block, which are the input and output of the system, respectively. In this simulation, the 2D Images block is used as a 2D input signal, the 2D FIR Coef block provides the 2D FIR filter coefficients, the 2D Filter block performs the 2D filtering operation and the 2D Plot block is used to view the 2D filtered output.

The 2D Images block is double clicked with the left mouse button and the desired image is selected from the drop-down menu and the [Update] button is pressed; this will start the simulation. The output can be viewed using the 2D Plot block that is connected to the 2D Filter block. The blurred output image illustrates the low-pass (LP) filtering effect of the system.

A number of lab exercises have been developed and assigned in the graduate-level *Multi-Dimensional Signal Processing (EEE 507)* course in order to help the students understand 2D DSP concepts that are hard to visualize in a traditional classroom environment. These on-line laboratories mainly cover fundamentals of 2-D signals and image processing concepts, implementation of 2D linear systems, 2D FIR filter design and 2D transforms.

Statistical and qualitative evaluations that assess the learning experiences of the participating students have been carried out. General assessment includes providing feedback on the developed 2-D J-DSP java tools and lab environment, while specific assessment focuses on the developed laboratory exercises by posing questions to determine whether the students have effectively learned the intended concepts using these simulations. Table 1 lists some of the general assessment statistics for the developed 2D J-DSP environment. Overall, the response is very promising. More than 90% of the users appreciated the idea of an internet-based simulation tool.

The 2D J-DSP tools, on-line laboratories, and evaluation forms are available through the web site http://jdsp.asu.edu.

4 SIVA - The Signal, Image, and Video Audio-Visualization Gallery

Umesh Rajashekar and Alan C. Bovik, The University of Texas at Austin, USA

This section describes tools for image processing education from SIVA [29] - The Signal, Image, and Video Audio-visualization gallery developed at the University of Texas at Austin (UT-Austin). The image processing gallery in SIVA consists of a suite of special-purpose Labview-based programs (known as Virtual Instruments or VIs) designed to serve as powerful visualization modules that are rich in fundamental image and video processing concepts. With its powerful point-and-click graphical user interface, SIVA is ideal for an in-class or online mode of instruction. Though all the programs in SIVA are useful as stand-alone

Table 1: General Assessment

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Establishing/connecting blocks are easy.	53%	39%	7%	1%	0%
The graphical user interface of 2D J-DSP is intuitive and user-friendly.	31%	63%	5%	1%	0%
Setting up the required lab simulations was easy.	40%	52%	8%	0%	0%
Blocks are elaborative and sufficient for visualizing 2D signals.	20%	70%	10%	0%	0%
Lab exercises improve 2D DSP concepts.	20%	70%	10%	0%	0%

image/video processing tools, they were, however, specifically created for communicating concepts while teaching Image and Video Processing, and therefore most effective in a didactic setting. At UT-Austin, SIVA is used in an undergraduate image processing course as an in-class demonstration tool to provide live examples that illustrate the image/video processing concept or algorithm under discussion. SIVA is available as a free download from [30].

4.1 Digital Image and Video Processing Education at The University of Texas at Austin

EE371R:Digital Image and Video Processing is a true multimedia, hands-on, web-intensive, demonstration-rich course on Digital Image and Video Processing offered in the Department of Electrical and Computer Engineering at UT-Austin. Though intended for an Electrical and Computer Engineering (ECE) curriculum, this junior/senior course attracts students (graduate and undergraduate alike) from various other majors such as Psychology; Geology; Astronomy; Mechanical, Chemical, and Aerospace Engineering; and Computer science. This popular course regularly attracts over 80 students and is often the largest upper-division undergraduate class in ECE. There is also a steady enrollment of professionals from the local industries at Austin, Texas. In line with the academic variety of the audience, the main objective of this courses is *'to make image processing accessible to an audience with heterogenous backgrounds by augmenting theory with numerous visual examples.'* Introductory material covered in this course includes binary image processing, image analysis, and image enhancement, while the more advanced material covers topics like Hough transforms, edge detection and video processing. The outline of topics covered in this class is listed below.

- Module 1 - Course Introduction
- Module 2 - Binary Image Processing

- Module 3 - Histogram and Point Operations
- Module 4 - Discrete Fourier Transform, Sampling Theorem
- Module 5 - Linear Filtering, Enhancement, and Restoration
- Module 6 - Nonlinear Image Filtering
- Module 7 - Digital Image Coding and Compression
- Module 8 - Template Matching, Edge Detection
- Module 9 - Hough Transform, B-Splines, Stereo Imaging
- Module 10 - Digital Video Processing

Along with course material designed specifically to keep the level of math accessible to the audience, this handbook is used as the optional textbook to provide students with an invaluable reference to the state-of-the-art in image processing. Simple, but intuitive MATLAB-based assignments are designed to reinforce the concepts without overwhelming students with programming intricacies.

To encourage a web-based educational system, Blackboard [31] and WebCT [32] are used as an instruction medium to make all course material and demonstrations available to students over the world wide web. Blackboard and WebCT are powerful tools to build online courses, requiring minimal technical expertise on the part of the instructor. They offer an impressive collection of online tools such as bulletin boards for course announcements; discussion boards, chat rooms and graphical tools for students to discuss their course-related issues with the instructor, teaching assistants, or amongst themselves. Timed online quizzes and grade tools are few of the other features offered by these online course tools. The discussion board, in particular, has proven to be a very useful tool to get students involved in discussions, which otherwise are infeasible to promote in-class due to time restrictions.

The course also has a project component where students are encouraged to investigate image processing applications in their respective fields. Students are provided with ample imaging hardware such as digital camcorders and firewire cards to download digital video onto computers, webcams, state-of-the-art video conferencing equipment and video editing software to develop their projects. Towards the end of the semester, students present their work and are graded by the entire class on creativity, difficulty of the problem, and presentation. In our experience, and from student feedback, we have found that the project component has proven to be extremely successful in motivating and sustaining the students' interests in the general area of image processing. Students enjoy discovering image processing applications in their respective areas and often, to our pleasant surprise, winners of the 'Ram's Horn Best Project Award' are students from non-engineering backgrounds![1]

However, the most unique feature that make this course very popular among students is the multimedia experience of image and video processing that is given throughout the course using live Labview demos in SIVA. To assist visual interpretation of ideas discussed,

[1]The term "Ram's Horn" refers to the flourishing check-mark grading symbol used by Dean T. U. Taylor at UT-Austin early in the twentieth century, and is a symbol of excellence at UT-Austin.

in-class demonstrations using SIVA are used to complement lectures. The demos have also been seamlessly integrated into the class notes to provide contextual illustrations of the principles under discussion. We will now describe the software framework used by the SIVA demonstration gallery and illustrate some of the image and video processing demos in SIVA.

4.2 LabVIEW development environment

LabVIEW [33] is a graphical programing language used as a powerful and flexible instrumentation and analysis software system in industry and academia. LabVIEW uses a graphical programing language, G, to create programs called Virtual Instruments or VIs in a pictorial form, eliminating much of the syntactical details of other text-based programing languages like C and MATLAB. LabVIEW includes many tools for data acquisition, analysis, and display of results. LabVIEW is available for all the major platforms and is easily portable across platforms. LabVIEW also provides IMAQ Vision, which includes more than 400 imaging functions and interactive imaging windows and utilities for displaying and building imaging systems, giving designers the opportunity to create examples for many important concepts in image processing, and using them for educational purposes. An excellent introduction to LabVIEW is provided in [34, 35]. Simple programability, impressive integrated graphical functions, availability for a wide variety of platforms, and extensibility to a web-based education system, such as that used at UT-Austin, made LabVIEW an attractive choice for developing SIVA.

Each VI consists of two main parts:

- The front panel contains the user interface control inputs, such as knobs, sliders, and push buttons, and output indicators to produce items such as charts and graphs. Inputs can be fed into the system using the mouse or the keyboard. A typical front panel (provided in LabView) is shown in Fig. 9(a).

- The block diagram shown in Fig. 9(b) is the equivalent of a 'source code' for the VI. The blocks are interconnected, using wires to indicate the dataflow. Front panel indicators pass data from the user to their corresponding terminals on the block diagram. The results of the operation are then passed back to the front panel indicators.

4.3 Examples of SIVA demos

The image processing gallery of SIVA contains over 40 VIs (see Table. 2) that can be used in conjunction with class lectures on image processing. Since most of the images in Sections I, II and Section 3.1 of this handbook are, in fact, the outputs from the SIVA demonstration modules, we will illustrate only a few VIs here. The Front Panels for each of these VIs is also shown to highlight the simple and intuitive user interface. The reader is invited to explore and download the other DIP demos in SIVA from the website mentioned in [30].

Table 2: List of Digital Image and Video Processing Demos in the SIVA Gallery

Image Quantization	Image Additive Noise
Image Sampling	Image Deblurring Inverse Filter
Image Histogram	Wiener Filter
Image Thresholding	Gray-Level Morphological Filters
Image Complementation	Median Filtering
Binary Morphological Filters	Trimmed Mean Filters
Image Skeletonization	Peak and Valley Detection
Histogram Shaping	Homomorphic Filters
Linear Point Operations	Block Truncation Image Coding
Image Interpolation	Entropy Reduction Via DPCM
Digital 2-D Sinusoids	JPEG Coding
Discrete Fourier Transform (DFT)	Gradient-Based Edge Detection
Bandpass Images	Template Matching
Orientation-Selective Images	Laplacian-of-Gaussian Edge Detection
DFTs of Important 2-D Functions	Canny Edge Detection
Low-Pass, Bandpass and Highpass Filtering	Hough Transform
Average Filtering	Video Frame Differencing
Ideal Low-Pass Filtering	Motion Compensation
Gaussian Filtering	Optical Flow Calculation
Double Thresholding and Contour Thresholding	Block Motion Estimation

4.3.1 Binary image processing

Binary images have only two possible 'gray levels' and are therefore represented using only one bit per pixel. Besides simple VIs used for thresholding gray-scale images to binary, SIVA has other VIs that demonstrate the effects of various morphological operations on binary images such as Median, Dilation, Erosion, Open, Close, Open-Clos, Clos-Open and other binary operations such as skeletonization. In these VIs, the user has the option to vary the shape and the size of the structuring element. The interface for the Morphology VI with a processed binary image is shown in Fig. 10.

4.3.2 Histogram and point operations (Gray-scale)

To demonstrate the effects of elementary gray-scale image processing, VIs that perform linear (offset, scaling, and full-scale contrast stretch) and non-linear (logarithmic range compression) point operations on images are included in SIVA. A more advanced VI shown in Fig. 11 demonstrates the effects of histogram shaping. The histograms of the input image and the resulting image after the linear point operation are also displayed on the front panel to verify the results of the shaping (e.g. the histogram in Fig. 11(d) is inverse Gaussian-like).

4.3.3 Image analysis using the Discrete Fourier Transform

A lucid understanding of the Discrete Fourier Transform (DFT) is important to appreciate the more advanced topics of image filtering and spectral theory. SIVA has many VIs to

provide an intuitive understanding of the DFT by first introducing the concept of spatial frequency using images of 2-D digital sinusoidal gratings. The DFT VI can then be used to compute and display the magnitude and the phase of the DFT for gray-level images. The user has the option of displaying the DFT with its low frequencies clustered together at the center of the image or distributed at the periphery. An option is also provided to display the logarithmically compressed, full-scale contrast-stretched version of the magnitude spectrum to reveal low contrast values. Masking sections of the DFT using zero-ones masks of different shapes and then performing inverse DFT is a very intuitive way of understanding the granularity and directionality of the DFT (see Section 2.3 of this handbook). To demonstrate the directionality of the DFT, the VI shown in Fig. 12 was implemented. As shown on the front panel in Fig. 12(a), the input parameters, Theta 1 and Theta 2, are used to control the angle of the wedge-like zero-one mask. It is instructive to note that zeroing out some of the oriented components in the DFT results in the disappearance of the one of the tripod legs in the 'Cameraman' image in Fig. 12(e).

4.3.4 Linear and Non-linear Image Filtering

SIVA includes many demos to illustrate the use of linear and non-linear filters for image enhancement and image restoration. The use of low-pass filters for noise-smoothing and inverse and pseudo-inverse filters for deconvolving images that have been blurred are examples of some demos for linear image enhancement in SIVA. SIVA also includes demos to illustrate the power of non-linear filters over their linear counterparts. Fig. 13 demonstrates the result of filtering a noisy image corrupted with 'Salt & Pepper noise' with a linear filter (average) and with a non-linear (median) filter.

4.3.5 Image Compression

Block Truncation Coding (see Section 5.2 of this book) is a very simple yet powerful method for image compression. The VI on Block Truncation Coding (BTC) shown in Fig. 14 provides an introduction to lossy image compression. The user can select the number of bits, $B1$, used to represent the mean of each block in BTC and $B2$ for the block variance. The compression ratio is computed and displayed on the front panel in the CR indicator in Fig. 14(a).

As listed in Table. 2, SIVA has many other advanced VIs that include many linear and non-linear filtering for image enhancement, other lossy and lossless image compression schemes, a large number of edge detectors for image feature analysis and Hough transforms.

4.3.6 Video Processing Demos

The video processing demos in SIVA use the LabVIEW environment and IMAQ Vision to demonstrate key concepts like motion estimation and compensation, motion compensated filtering, video compression and reconstruction, the effects of noise, and pre-processing. The

results from these demos can be compared both objectively (peak signal to noise ratio for example) and subjectively for assessment of video reconstruction quality. Within many demos, users have the option to vary computation methods for additional comparison. Fig. 15 for example illustrates the estimation of optical flow between two scenes of a video. To illustrate a practical optical flow estimating technique, SIVA contains a motion estimation demo that allows the use of different search methods, such as the 3-step search, cross search or brute force, and different match methods, such as the minimum mean square error, maximum absolute difference, maximum matching pixel count, or maximum correlation. Additionally, many demos will operate on live video acquired in class directly into LabVIEW from a digital camcorder and a laptop IEEE 1394/firewire port. These demos will add flexibility and intrigue to the learning process and help emphasize both the simplicities and the intricacies of video processing that may be overlooked when using a predefined set of videos that have controlled acquisition environments.

4.4 Conclusions for SIVA section

The interactive image/video processing demonstrations in SIVA have been integral to the success of the EE371R Digital Image & Video Processing course at UT-Austin. The course has become amazingly popular, typically attracting more than 80 students per class in recent semesters. The visual intuition provided by the demos has proved to be very useful in introducing image processing to an audience with very diverse academic backgrounds. The SIVA demonstration gallery is gaining popularity and is being widely used by instructors in many educational institutions over the world for teaching their signal, image and video processing courses, and by many individuals in industry for testing their image processing algorithms. To date, there are around 150 users from over 40 countries using SIVA. SIVA is available as a free download from [30].

5 VcDemo – The Image and Video Compression Learning Tool

R. (Inald) L. Lagendijk, Delft University of Technology, The Netherlands

VcDemo is a software tool intended to assist students at graduate level and (postgraduate) professionals with sufficient training in stochastic processes and signal processing in studying lossy compression techniques [36]. Whereas in textbooks the emphasis is typically on explaining compression methods from the perspective of information theory and algorithm structure, VcDemo emphasizes studying compression techniques from observed performance (e.g. signal-to-noise ratio and visual compression artifacts) under different parameters settings (such as selected bit rate or bit-error-rate of the channel over which compressed data is sent). The software tool comprises nearly twenty image and video compression modules. The operation of the software compression modules is fully controlled by menus and push buttons: no programming by students is required. The modules are self-explaining, yet basic

help functions are available on-line (F1 button). VcDemo is very suitable for self-study, for student homework exercises, and for in-class demonstrations.

5.1 ET4269: Digital Signal Coding at Delft University of Technology

The graduate course ET429 (Digital Signal Coding) [37] is a compulsory course of 6 ECTS credits for students in the *Media and Knowledge Engineering* masters program at Delft University of Technology [38]. The course also attracts students from various other M.Sc. programs, such as Telecommunications and Computer Engineering. The yearly enrollment is between 80 and 120 graduate students. Several courses for professional engineers from industry have been derived from ET4269. Two examples are the course "Video Compression" lectured within the curriculum of the dutch association for postgraduate education (PATO) [39], and a yearly course with the same name being lectured in-house at Philips Research, The Netherlands. Especially the courses for professionals, who often have widely varying backgrounds and interests, are build-up around hands-on exercises using VcDemo. The VcDemo tool then serves to stimulate self-exploration of compression algorithms as well as the interaction with teaching assistants on compression techniques of interest to the professionals's work.

The student's background assumed in ET4269, is signal processing and transformations (in particular linear systems, Discrete (time) Fourier Transform, and spectral representations), and stochastic processes (in particular the theory and interpretation of the autocorrelation function). The course ET4269 begins with motivating the need for signal compression. Right from the start, image compression examples are shown using VcDemo. For example, the students are given an outlook on what compression algorithms can achieve by simultaneously showing a pulse-coded modulation (PCM) image at 1 bit per pixel and a JPEG compressed image at the same bit rate. Although the student cannot yet explain the cause(s) of the different artifacts, from such a simple example the inherent quality loss and trade-offs that are made by different compression techniques are immediately clear.

The courses then continues with three main topics, spread out over one quarter (7 weeks of teaching) with two lectures of two hours each per week. The subjects are:

- fundamentals of compression, including information theoretical background, lossless source coding, rate-distortion theory, and quantization theory,
- audio and image compression algorithms, covering PCM, differential PCM, vector quantization, transform coding (including DCT), subband coding, the JPEG image compression standards, and the mp3/AAC audio compression standards,
- video compression and motion estimation, the MPEG-compression standards, and an outlook to advanced compression techniques such as scalable and error-resilient compression. The latter subjects are often of most interest to professionals from industry.

The VcDemo tool greatly supports the lectures on the last two subjects. Live demonstrations are seamlessly integrated into the lectures, or illustrations in (power point) presentations are taken directly from VcDemo's output (screen dumps). Students are encouraged to download the VcDemo software and practice at home with the compression module that have been shown in class. Further, the opportunity is offered to carry out the exercises under supervision of teaching assistants during the last two weeks of the quarter. Exams regularly include questions using the output of VcDemo; examples of (English) exams – as well as all other lecture materials – can be downloaded from Delft University of Technology's blackboard [37] using the course search key "ET4269".

5.2 VcDemo Compression Environment

5.2.1 Balance between Versatility and Efficiency

In developing software that supports learning processes, the primary choice is always how much programming the student should do in order to obtain a certain functionality. For basic image processing operations, such as linear filtering, the complexity of implementation – and hence the time a student spends on getting the algorithm to function properly – is manageable, especially if a well-structured programming environment is offered. Such environment should at least take away the burden of writing image I/O and display functions. For more complicated image operations, pre-developed parametrized building blocks are desirable in order to avoid students spending too much time on debugging. Clearly, this takes away a lot of the insight that students obtain by implementing a particular algorithm. For image and video compression, the underlying algorithms are often so complex, that, first, pre-developed building blocks are needed, and that, second, few choices remain for how to combine different building blocks.

For that reason we decided already in the early days of VcDemo (around 1990) that a somewhat rigid menu and push-button driven structure was favorable over a more versatile programmable structure, at the advantage of transparency and especially time-efficiency for students. We have stayed with this choice during the different usage and development stages of VcDemo: from the VAX/VMS context, to Unix and Linux, to the current VcDemo Version 5 for the Windows platform. Considering the complexity of some of the algorithms included in VcDemo – such as the JPEG-2000 standard, and an MPEG-2 and H.264 video encoder – we believe that VcDemo indeed strikes the right balance between versatility of operation and the learning curve of the students.

5.2.2 Workspace

VcDemo provides the workspace as illustrated in Figure 16a. The workspace consists of three parts. The top part contains the pull-down menus and the push-buttons that start particular compression modules. The right hand side of the workspace contains the module interfaces, stacked on top of each other. Depending on which image and module is active,

the correct module interface is displayed on top. In this way it is very easy to work on multiple images at the same time, and have different modules active in order to compare different compression results. The top part of the module interface contains tabs with which the parameters can be set that control the compression module. The bottom part shows numerical output such as image variance, signal-to-noise ratio, bit rate, and bit allocation. Finally, the middle/left hand side of the module interface displays original and compressed images, as well as intermediate results, for instance a subband decomposition.

The compression interfaces control the operation of the underlying compression modules. In the current version of VcDemo, many of the compression parameters are selectable from a discrete set of values. Although this is certainly not a requirement of the underlying software code, we have found that limiting the parameter choices helps the student to find his/her way in doing the most interesting experiments. For instance, many of the image compression modules operate with the bit rate set $\{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1.00, 1.5, 2.0, 3.0\}$ bit per pixel, indicating the "interesting" bit rate settings for most compression techniques.

The workspace itself and all compression modules come with on-line help (pressing F1). Figure 16b gives an example of the help information provided for the PCM module. The help information does *not* explain the underlying compression algorithm, only how the input parameters and output images and numerical results relate to the compression algorithms. A second form of help is the information provided about compressed images. Since compressed images are kept until the user either closes that particular image window or the window of the related original image (in which case all derived images and compression modules will be closed), many different image windows may be visible in the workspace. In order to find back which parameters settings were used and what the resulting compression performance was, clicking the right mouse button gives access to a context menu from which "image information" can be selected, as illustrated in Figure 16a.

5.2.3 Supporting Functions

In addition to the compression modules itself – of which some examples are given in Section 5.3 – VcDemo contains several supporting functions. The four most important supporting functions are described here.

First, VcDemo allows for saving *any* output image or decompressed video sequence, either in JPEG or TIFF format. This makes it very easy for students and instructors to include interesting (intermediate) results into lecture materials or home work reports.

Second, in VcDemo any two images originating from the same original image can be subtracted. This allows the visual comparison of compression artifacts for different compression techniques. Numerical information (average values, variance, and range) of the difference image is provided.

Third, series of images and video sequences of YUV-format can be played in real time. Video sequences are assumed to be stored in raw YUV 4:2:0 format. A separate ASCII header file provides information about, for instance, the frame dimensions and number of

video frames.

And finally, the student can let VcDemo store all (bit-rate, signal-to-noise ratio) pairs of the experiments carried out. These pairs can then be used for plotting rate-distortion ($R(D)$) curves that numerically show and compare the performance of different compression algorithms on the same image, and/or the performance of the same compression algorithm under different parameter settings. Figure 17 shows an example of such a comparison.

5.3 Examples of VcDemo Compression Modules

VcDemo encompasses a wide range of compression modules. The reader is invited to download and install VcDemo from the website [36], and explore the different compression modules. We here give three examples, namely differential pulse-coded modulation (DPCM), subband coding (SBC), and MPEG video compression.

5.3.1 Differential Pulse-Coded Modulation (DPCM)

The DPCM module implements one of the simplest compression algorithm. It combines a spatial prediction loop with a non-uniform quantizer of which the representation levels are entropy (Huffman) encoded. Four different spatial prediction models can be selected. The output shows – in addition to the compressed image – the prediction error image (top midddle image in Figure 18). Figure 18 shows the rate-distortion performance of DPCM on the image shown for two different prediction models in comparison with PCM. Students are encourage to relate the observed performance to theoretical concepts such as high-rate performance (6 dB/bit) and DPCM prediction gain. Further, the enlarged image shows the effect of randomly inserted bit error into the bit stream on the decompressed image. Students should be able to explain these particular artifacts from the algorithm structure of a DCPM-encoder/decoder.

5.3.2 Subband Coding (SBC)

The subband coding module implements a spatial subband decomposition (with different selectable subband structures) in combination with the PCM/DPCM encoding of the subbands. An important aspect of subband coding is the allocation of bits over the subbands. The implemented optimal bit allocation algorithm has been described in [40], and uses predesigned LLoyd-Max or uniform threshold quantizers. The numerical output window (part of the module interface) lists the variances of the subbands, and the resulting bit allocation and quantization error variance. Students should be able to explain the observed pattern in bit allocation, and relate this to the underlying information theory. Figure 19 shows the workspace with the activated SBC module. In addition to the (intermediate) result of the bit allocation and subband quantization of the "Lena" image, the figure also shows the subband decomposition of a zone-plate (2-D frequency sweep) image. Students should be able to explain the content of the subbands of this image in terms of 2-D (subband) filters.

5.3.3 MPEG Video Encoding and Decoding

Obviously, video encoders are among the most complicated image/video processing operations. VcDemo currently includes an MPEG 1 and 2 video encoder and decoder, as well as an H.264 video encoder and decoder[2]. The MPEG-encoder creates an MPEG video file that is compatible with most MPEG players. The student can set parameters such as bit rate, group-of-pictures structure, and motion search length. The numerical output window gives the student information about the performance of the MPEG encoder on the current video frame, as well as the type of frame (I, P, B) and macroblock being encoded.

The MPEG decoder can – in addition to normal decoding of any MPEG-compressed video sequence – also display the motion vectors per macro block. Further, the MPEG decoder can also display the *video information encoded in the bit stream*, such as macro block differences or intra-encoded macroblocks. Figure 20 illustrates the GUI of the MPEG compression module. It also shows the temporal prediction difference and motion vectors which represent the actual video information encoded in the MPEG bit stream.

5.4 Conclusions

The structure of compression algorithms and their rationale are often not difficult to understand. However, implementing compression algorithms in order to evaluate their performance and typical artifacts is far too complicated for a compression course, even at graduate level. VcDemo offers a solution to this implementation hurdle, and makes it possible for students to quickly focus on questions that are important, such as "can I explain what I see from the theory and algorithms learned in class?". The VcDemo software has been used in classes at Delft University of Technology for nearly 15 years now, and it has been available for public download since February 2000. As of May 2004, the VcDemo web page [36] has received around 20,000 hits. Based on the VcDemo concept, the SIPL Group at Technion has started the development of a similar signal processing tool [41]. Thanks to the many positive reactions and suggestions from lecturers, professionals, and students worldwide, we plan to continue the further enhancement and extension of VcDemo in the near future.

6 Chapter Conclusions

The importance of stressing practical applications in an image processing course has been prevalent for a long time now. Making image processing accessible to an ever growing non-expert audience is highly desirable. In this chapter, a library of demonstrations built, and in successful use at various universities around the world, intuitively introduce the concepts of digital image and video processing to a disparate audience consisting of graduate and

[2]Most VcDemo code has been developed by the Information and Communication Theory Group at Delft University of Technology. Exceptions are compression code of EZW (contributed by Purdue University), SPIHT (contributed by PrimaComp, Inc.), JPEG/JPEG2000, MPEG, and H.264 (public domain software). The latter module has been incorporated into VcDemo by the SIPL group at Technion [41].

undergraduate students from a variety of backgrounds. The inertia to develop such demos is understandable since a high investment of time and effort is required. It is hope of the authors that this chapter will succeed in attracting the attention of many more instructors to the availability of wonderful didactic tools for image processing education. The authors are optimistic that these libraries will be instrumental in adding value to the way digital video processing and wavelets will be taught at many other universities around the world.

References

[1] http://www.infinity-project.org/home.html. (Last viewed: Feb. 6, 2002).

[2] T. Barnwell and B. Evans, "DSP as a first course." First Signal Processing Education Workshop, Hunt, TX, Oct 2000. http://spib.ece.rice.edu/DSP2000/program.html#dspcourse, (Last viewed: Feb. 6, 2002).

[3] G. S. K. Bowyer and L. Stark, "Themes for improved teaching of image computation," *IEEE Trans. Education*, vol. 43, pp. 221–223, 2000.

[4] E. L. D. M. Sonka and S. M. Collins, "Image systems engineering education in an electronic classroom," *IEEE Trans. Education*, vol. 41, pp. 263–272, 1998.

[5] D. E. G. Bebis and M. Shah, "Review of computer vision education," *IEEE Trans. Education*, vol. 46, pp. 2–21, 2003.

[6] U. Rajashekar and A. C. Bovik, "Interactive DSP education using MATLAB demos," *Proc. First Signal Processing Education Workshop*, Oct 2000. Available online at http://spib.ece.rice.edu/DSP2000/, (Last viewed: Feb. 6, 2002).

[7] G. C. Orsak and D. M. Etter, "Collaborative DSP education using the internet and MATLAB," *IEEE Signal Processing Magazine*, vol. 12, pp. 23–32, Nov 1995.

[8] R. Radke and S. Kulkarni, "An integrated MATLAB suite for introductory DSP education," *Proc. First Signal Processing Education Workshop*, Oct 2000. Available online at http://spib.ece.rice.edu/DSP2000/, (Last viewed: Feb. 6, 2002).

[9] M. W. Joe Williams and G. C. Orsak, "Peruna and pony express: Two MATLAB-based educational software packages for signal processing and communications," *Proc. First Signal Processing Education Workshop*, Oct 2000. Available online at http://spib.ece.rice.edu/DSP2000/, (Last viewed: Feb. 6, 2002).

[10] J. Rosenthal and J. H. McClellan, "Animations and guis for Introductory Engineering Courses," *Proc. of the Internation Conf. on Electrical Engineering*, pp. 6E411–6E416, August 2001. Available online at http://fie.engrng.pitt.edu/icee/ Last viewed: Feb 7,2002.

[11] G. C. Panayi, U. Rajashekar, and A. C. Bovik, "Image processing for everyone," *Proc. First Signal Processing Education Workshop*, Oct 2000. http://spib.ece.rice.edu/DSP2000/, (Last viewed: Feb. 6, 2002).

[12] "National Instruments LabVIEW Player VI Gallery." Available online at http://zone.ni.com/devzone/
explprog.nsf/webLabVIEWenabled, (Last viewed: Feb. 6, 2002).

[13] Y. Cheneval, L. Balmelli, P. Prandoni, J. Kovacevic, and M. Vetterli, "Interactive DSP education using Java," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 3, pp. 1905–1908, 1998.

[14] A. Spanias, A. Constantinou, J. Foutz, and F. Bizuneh, "An online signal processing laboratory," *Proc. First Signal Processing Education Workshop*, Oct 2000. Available online at http://spib.ece.rice.edu/DSP2000/program.html, (Last viewed: Feb. 6, 2002).

[15] J. Shaffer, J. Hamaker, and J. Picone, "Visualization of signal processing concepts," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 3, pp. 1853 –1856, 1998.

[16] B. L. Evans, L. J. Karam, K. A. West, and J. H. McClellan, "Learning signals and systems with Mathematica," *IEEE Trans. on Education*, vol. 36, pp. 72 –78, Feb 1993.

[17] R. Martti and K. Matti, "An interactive DSP tutorial on the web," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 3, pp. 2253 –2256, 1997.

[18] D. Sage and M. Unser, "Teaching image-processing programming in java," *IEEE Signal Processing Magazine*, vol. 20, pp. 43–52, 2003.

[19] W. Rasband, "Imagej." [Online] Available: http://rsb.info.nih.gov/ij/ (visited in 2004). National Institute of Health, Bethesda, Maryland, USA,.

[20] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, pp. 679–714, 1986.

[21] J. Brown et al., "Situated cognition and the culture of learning," *Educational Researcher*, pp. 32–42, Jan. 1989.

[22] D. Myers, "Interactive video: A chance to plug the literacy leak," *Industry Week*, pp. 15–18, Apr. 1990.

[23] L. Karam and D. Rice, "Teaching image processing to high-school students," in *First Signal Processing Education Workshop*, Oct. 2000. Available on-line at http://spib.ece.rice.edu/DSP2000/.

[24] A. Spanias et al, "Development and evaluation of a web-based signal and speech processing laboratory for distance learning," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. VI:3534–3537, June 2000.

[25] A. Clausen et al, "J-dsp: an internet-based educational tool for digital filter experiments," in *IEEE Symposium on Advances in Digital Filtering and Signal Processing*, pp. 57–61, June 1998.

[26] M. Yasin, L. Karam, and A. Spanias, "On-line laboratories for image and two-dimensional signal processing using 2d j-dsp," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. III:785–788, 2003.

[27] M. Yasin, "Web-based two-dimensional signal processing," Master's thesis, Arizona State University, Tempe, AZ, Dec. 2003.

[28] A. E. Walsh and D. Gehringer, *Java 3D API Jump-Start*. New Jersey: Prentice Hall, 2003.

[29] U. Rajashekar, G. C. Panayi, F. P. Baumgartner, and A. C. Bovik, "The SIVA demonstration gallery for signal, image, and video processing education," *IEEE Transactions on Education*, vol. 45, pp. 323 – 335, November 2002.

[30] http://live.ece.utexas.edu/class/siva. Last accessed July 21, 2004.

[31] "Blackboard Homepage." http://www.blackboard.com/, (Last viewed: Jul. 20, 2004).

[32] "WebCT Homepage." http://www.webct.com/, (Last viewed: Jul. 20, 2004).

[33] http://www.ni.com/labview, (Last viewed: Feb. 6, 2002).

[34] L. K. Wells and J. Travis, *LabVIEW for Everyone*. Upper Saddle River, NJ 07458: Prentice Hall PTR, first ed., 1997.

[35] R. H. Bishop, *LabVIEW Student Edition 6i*. Upper Saddle River, NJ 07458: Prentice Hall, first ed., 2001.

[36] VcDemo software: http://www-ict.ewi.tudelft.nl/vcdemo.

[37] Enter as a guest (called "preview mode")at Delft University of Technology's blackboard system: http://blackboard.tudelft.nl.

[38] M.Sc. programs at Delft University of Technology:http://new.ewi.tudelft.nl/index.php?taal=uk.

[39] Association for postgraduate education (PATO): http://www.pato.nl.

[40] B. J. B. D. Westerink, P.H., "An optimal bit allocation algorithm for subband coding," *Proc. Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 757–760, 1988.

[41] http://www-sipl.technion.ac.il.

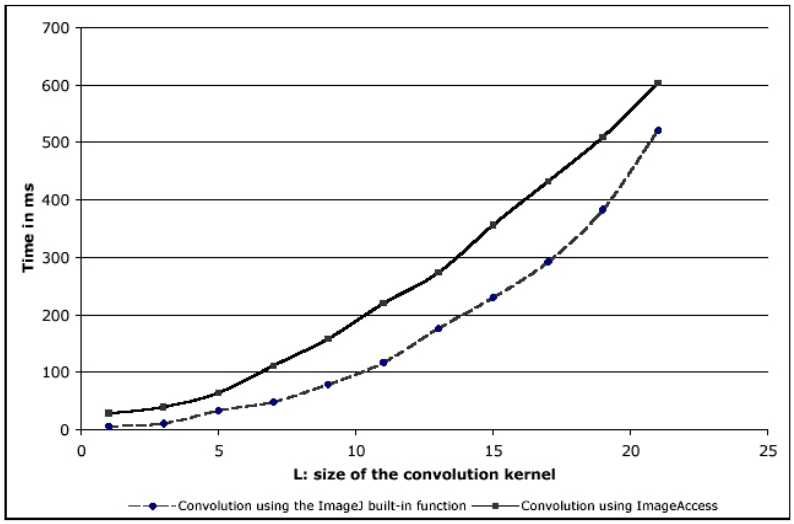
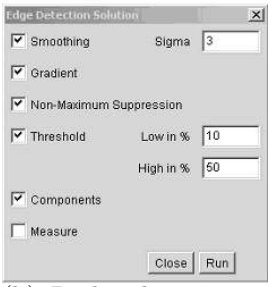


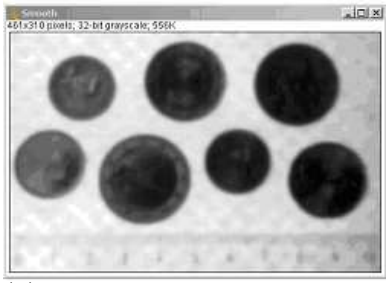
Figure 1: Overhead of the ImageAccess class. Convolution of a 256*256 image with a kernel of size L*L on a Apple PowerMac 2.0 GHz

(a) ImageJ running on Microsoft Windows

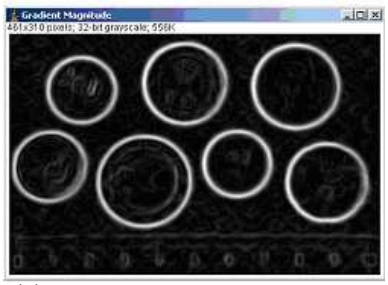


(b) Dialog box to set the parameters of the operations

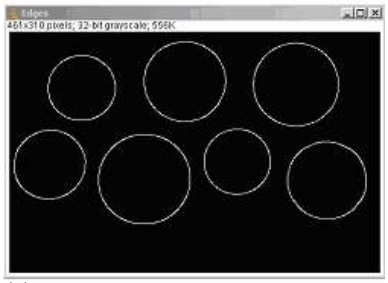
(c) Original image



(d) Image after a Gaussian Smoothing (s=3.0)



(e) Image after a gradient filter



(f) Image after non-maximum suppression and hyteresis thresholding

Figure 2: Example of an ImageJ plug-in that implements the various steps of the Canny edge detector

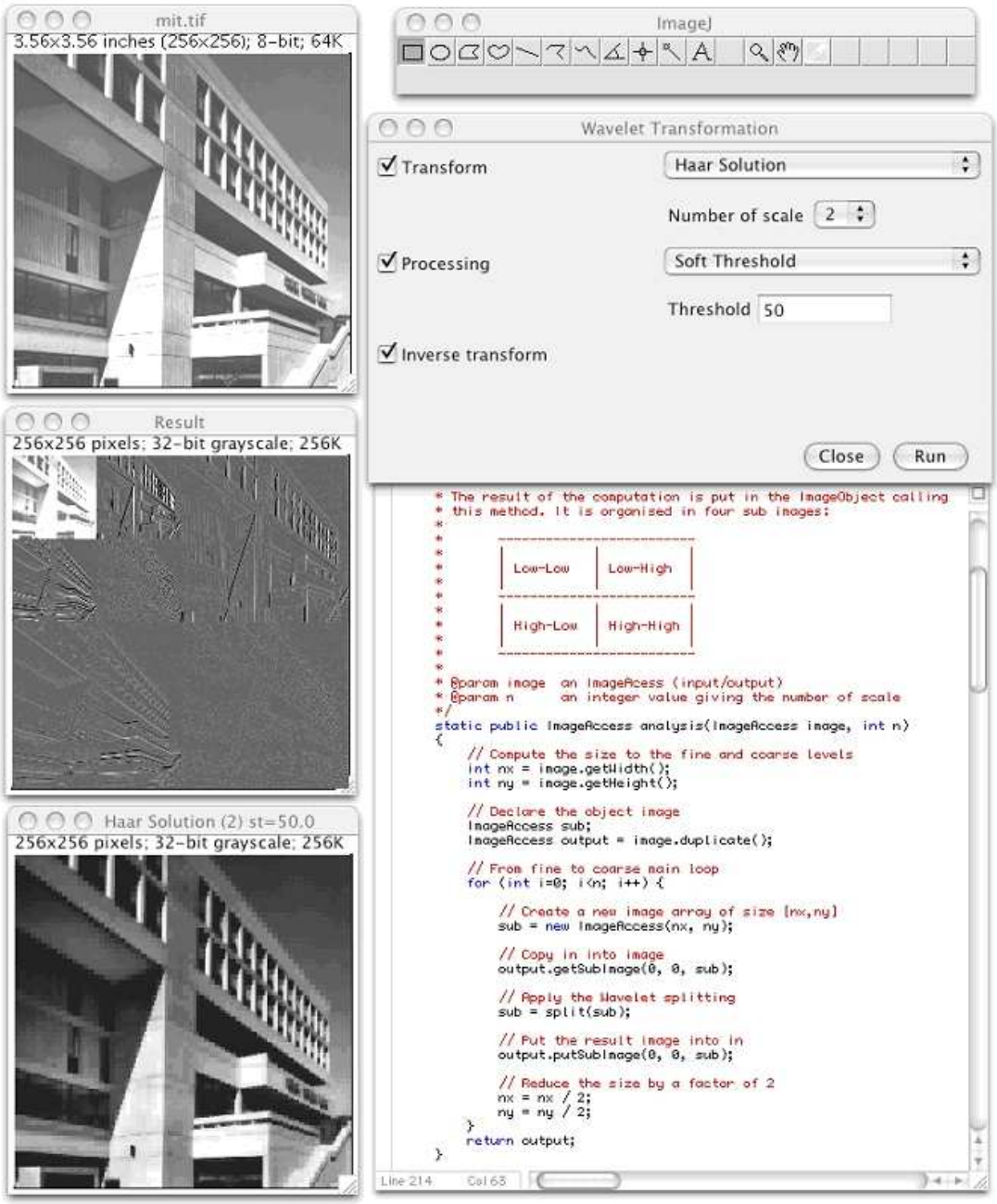


Figure 3: Illustration of the wavelet transform plug-in running on an Apple Macintosh machine. The original mit.tif image is analyzed with the 2D Haar wavelet. A soft threshold is then applied to the wavelet coefficients, and the image is finally reconstructed by inverse transformation.

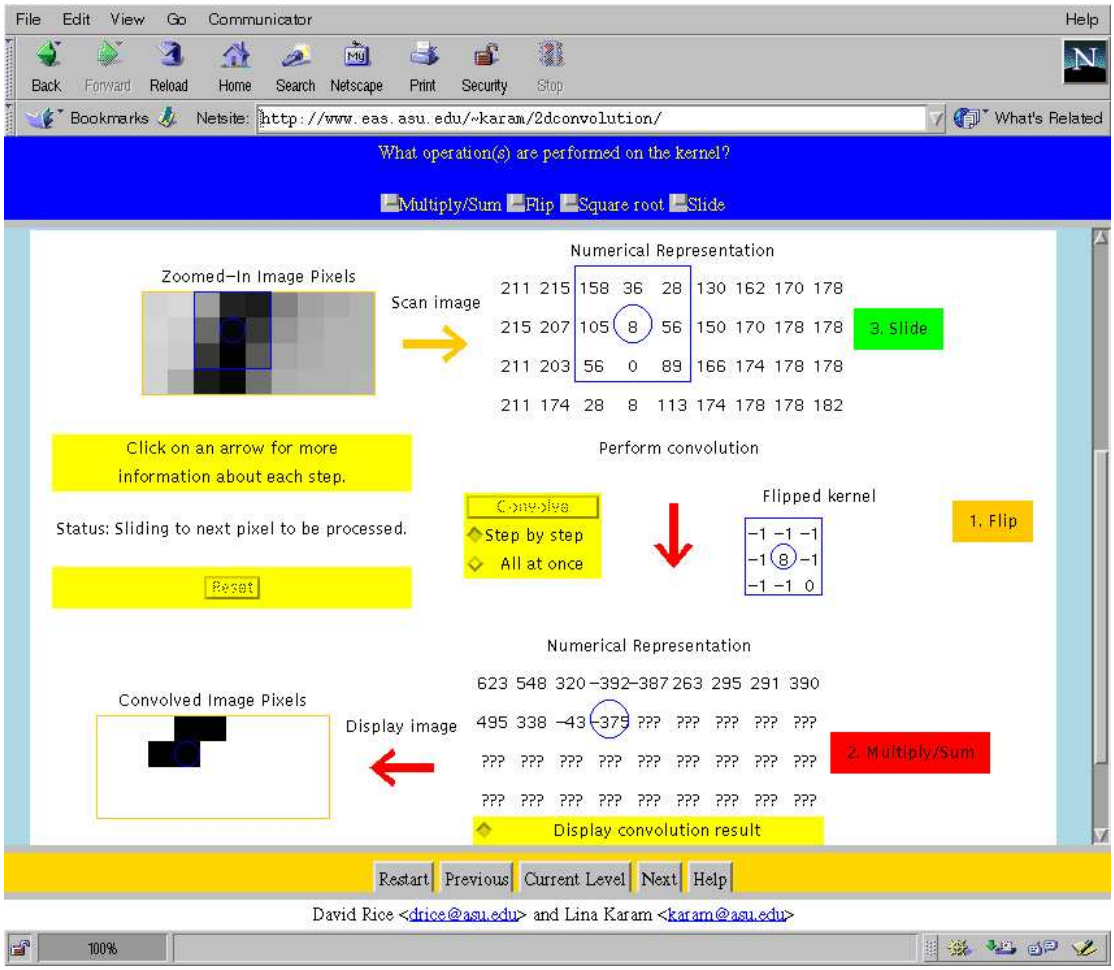
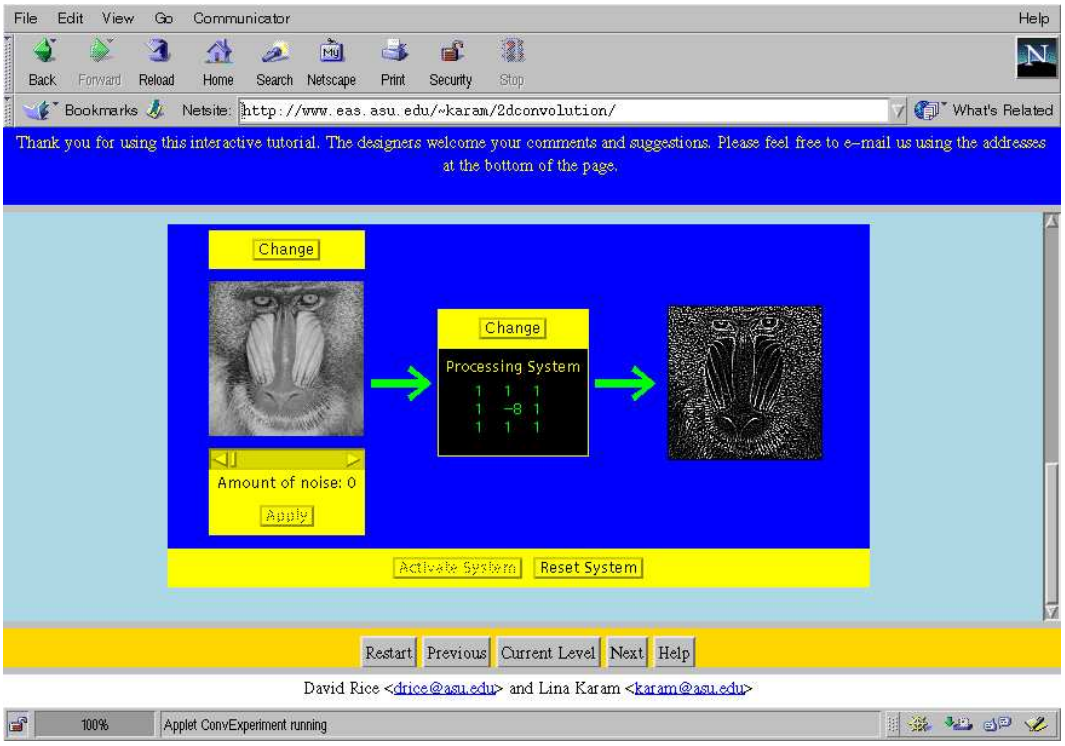
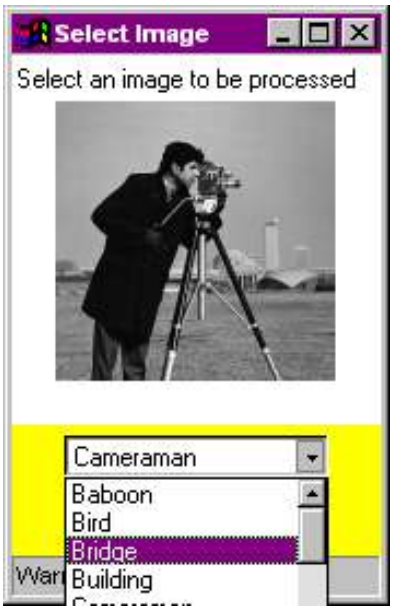


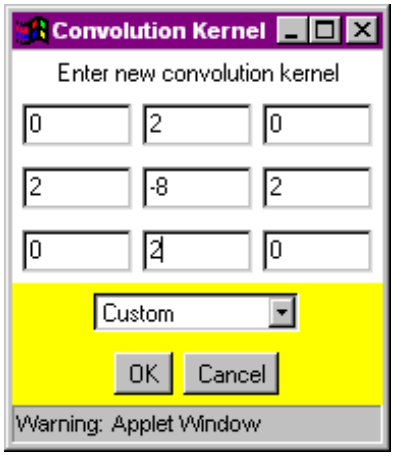
Figure 4: Interactive and animated 2-D convolution demo illustrating the 3 main 2-D convolution steps and image processing. Context sensitive help is available for each step by clicking on the corresponding arrow. Students need to answer correctly the question at the top of the page in order to access the subsequent page of the tutorial.



(a)



(b)



(c)

Figure 5: Interactive 2-D linear system and lab experiments. (a) Example of the effect of applying a highpass system to the Cameraman image. (b) User can change the image and add noise to it interactively. (c) User can change the system interactively by choosing from provided standard systems or by entering a custom 2-D system impulse response (kernel).

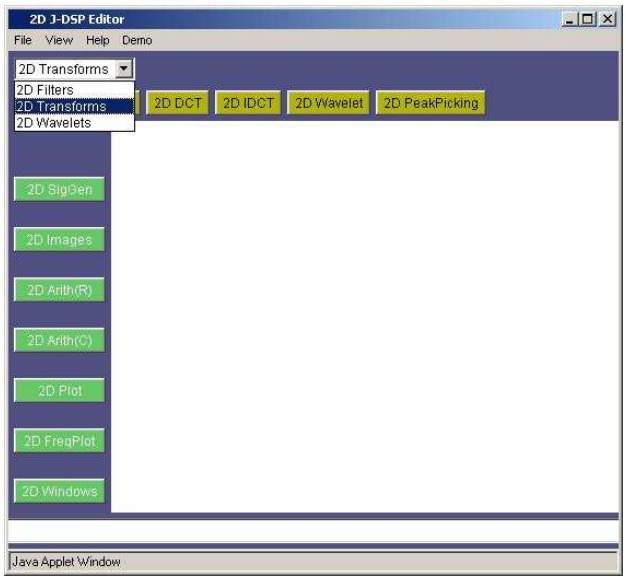


Figure 6: 2D J-DSP Editor.

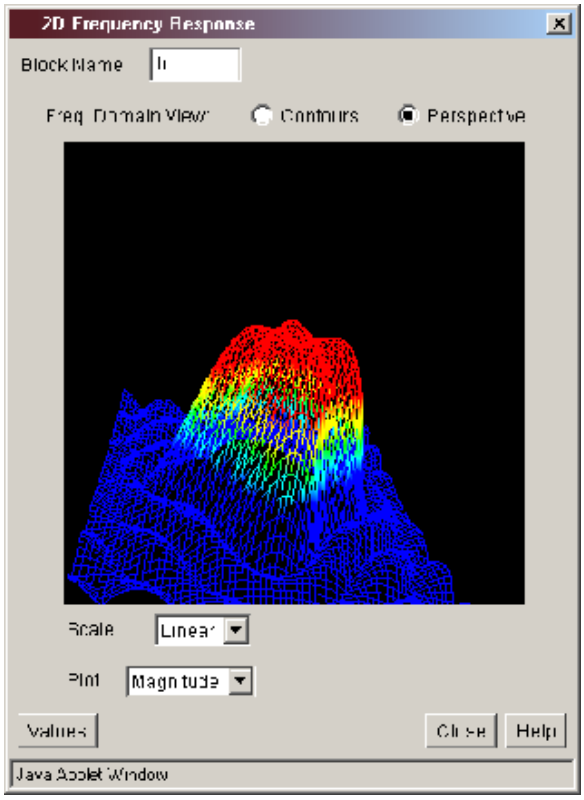
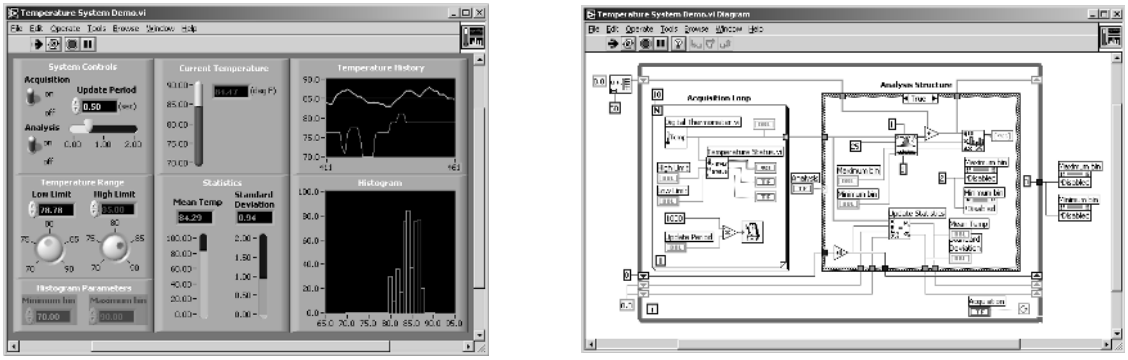


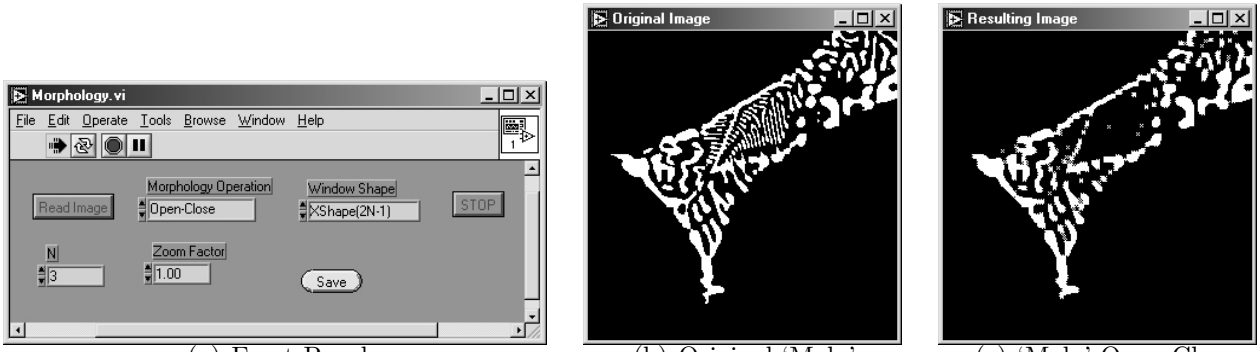
Figure 7: 3D Mesh Plot of Filter Frequency Response.

Figure 8: Example of a 2D J-DSP simulation, and Dialog windows for the 2D Images and 2D Plot blocks.



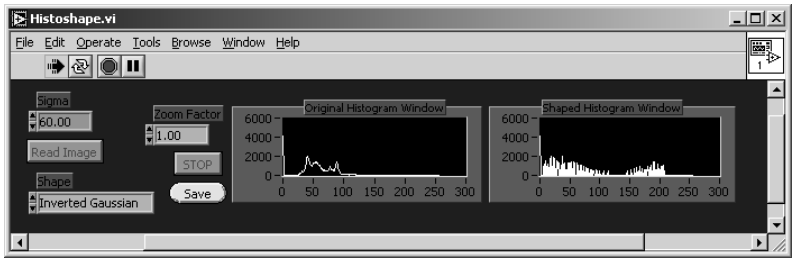
(a) FrontPanel (b) Block Diagram

Figure 9: Typical GUI development environment in LabVIEW



(a) Front Panel (b) Original 'Moly' (c) 'Moly' Open-Clos

Figure 10: Binary image morphology



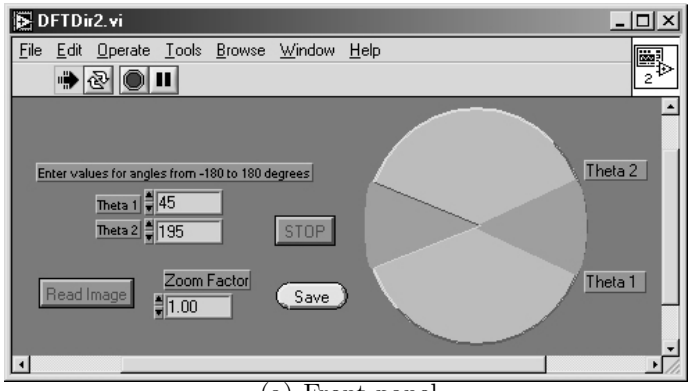
(a) Front Panel

(b) Original 'Books'

(c) Histogram-Flat

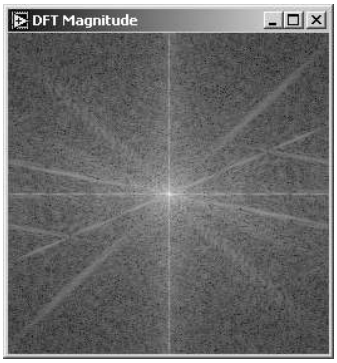
(d) Histogram-Inverted Gaussian

Figure 11: Histogram Shaping

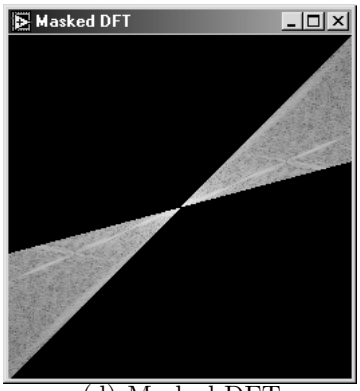


(a) Front panel

(b) Original 'Cameraman'



(c) DFT magnitude



(d) Masked DFT

(e) Inverse DFT after masking

Figure 12: Directional DFT

(a) Front Panel

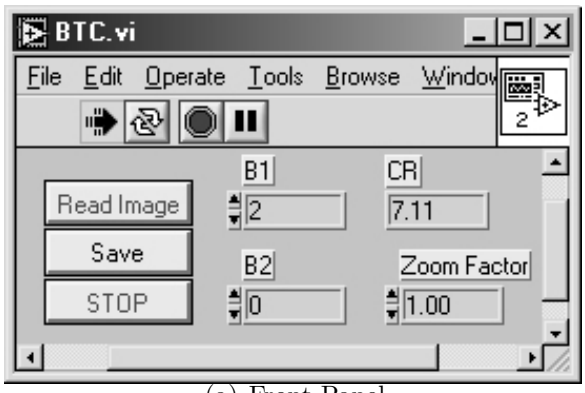
(b) Original 'Dhivya'

(c) Salt & Pepper Noise

(d) Average filtering

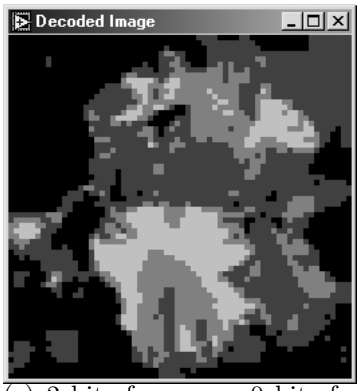
(e) Median filtering

Figure 13: Non-linear filtering



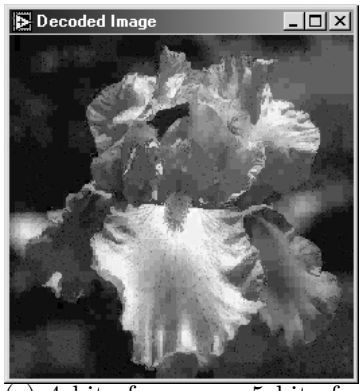
(a) Front Panel

(b) Original 'Iris'



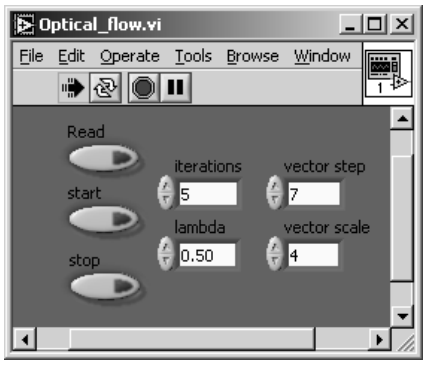
(c) 2 bits for mean; 0 bits for variance

(d) 4 bits for mean; 0 bits for variance

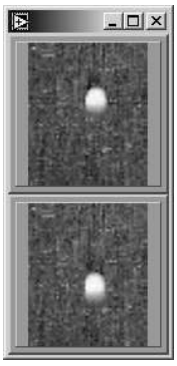


(e) 4 bits for mean; 5 bits for variance

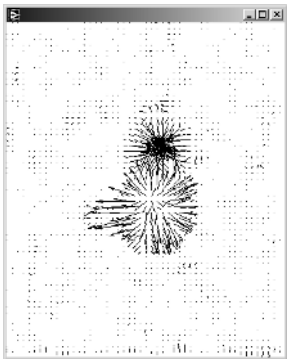
Figure 14: Block Truncation Coding



(a) Front Panel

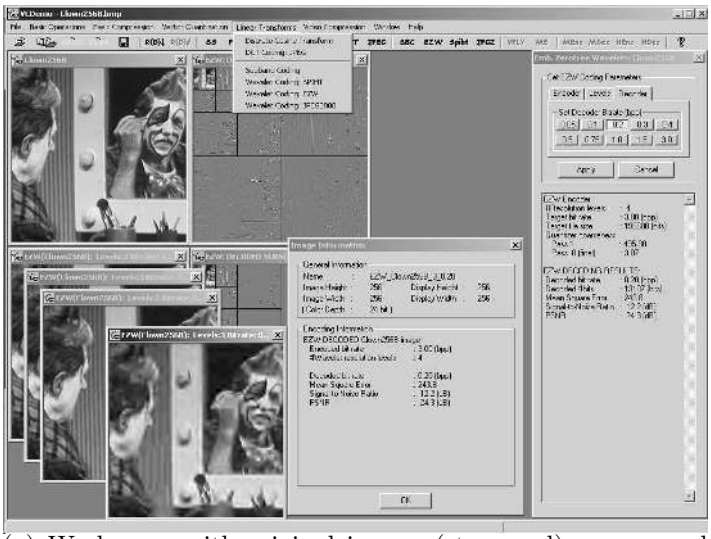


(b) Successive video frames

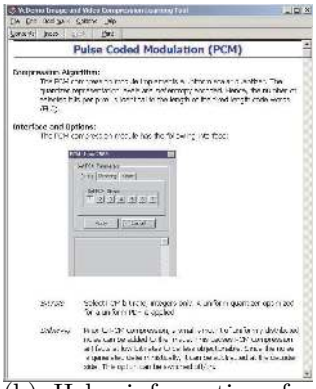


(c) Estimated Optical Flow

Figure 15: Optical flow



(a) Workspace with original image, (staggered) compressed images, intermediate results, and compression module interface.



(b) Help information for the PCM module.

Figure 16: Typical GUI in VcDemo.

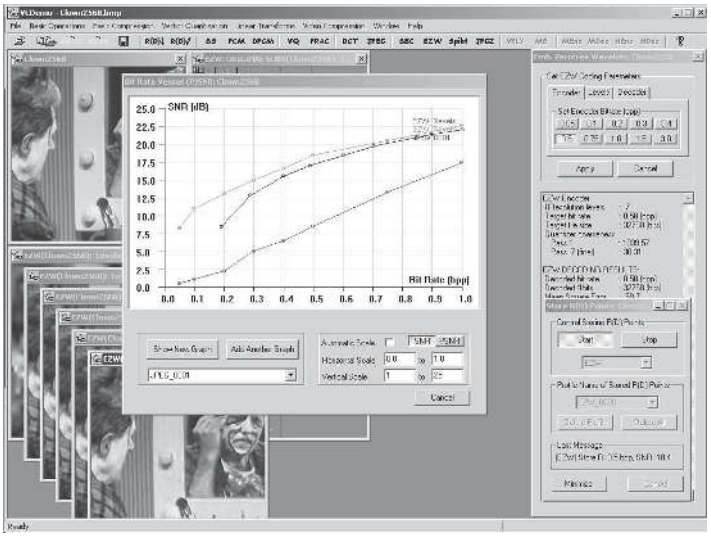


Figure 17: Automatic plotting of R(D) curves in VcDemo.

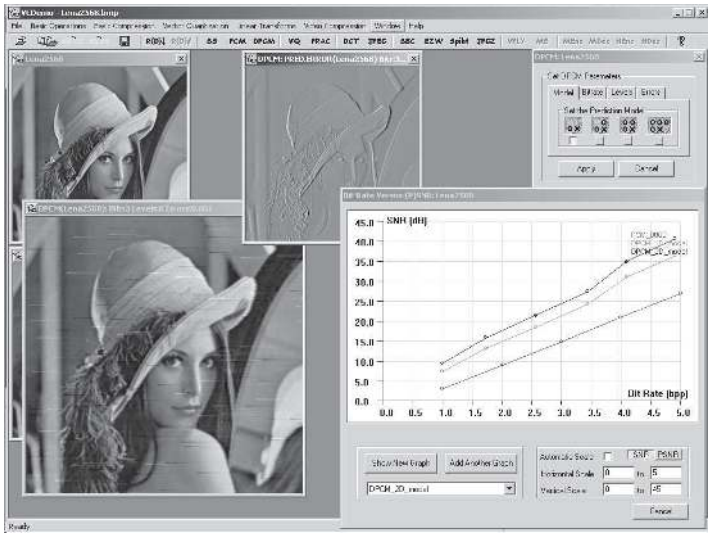


Figure 18: GUI of the DPCM compression module. The enlarged image shows the effect of randomly inserted bit errors into the bit stream.

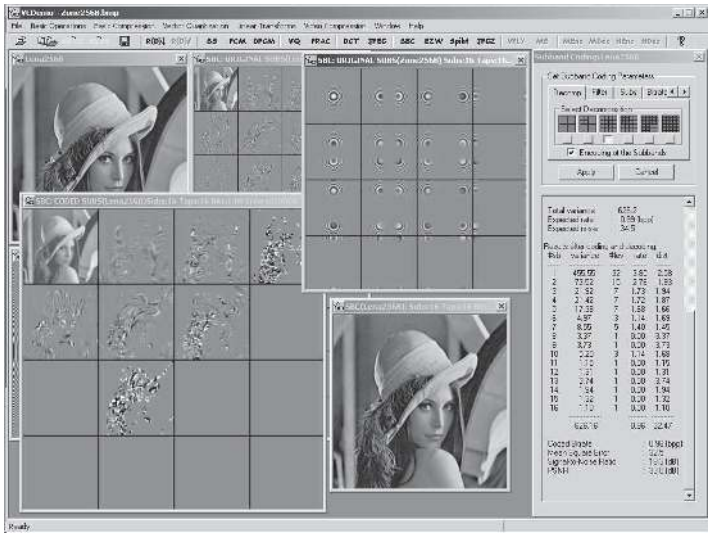


Figure 19: GUI of the SBC compression module. The enlarged image shows an intermediate results: the quantized subbands. Also shown is the subband decomposition prior to quantization of a zone-plate image.

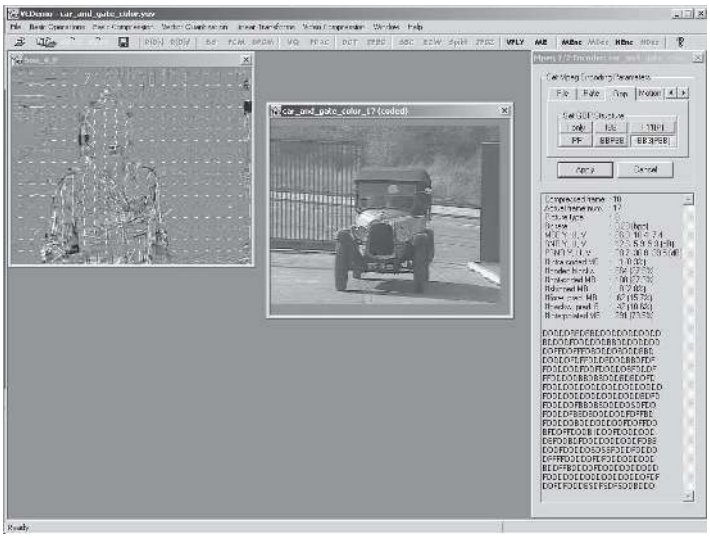


Figure 20: GUI of the MPEG encoder module. The video frame on the left hand side shows the temporal prediction difference frame plus motion vectors used to build the motion-compensated prediction. The window on the right hand side shows the video frame currently being compressed, with numerical information shown in lower part of the MPEG module interface.