COMPUTER VISION RESEARCH AS A TEACHING TOOL IN CS1

Dwight Egbert[1], *George Bebis*[2], *Meggin McIntosh*[3], *Nancy LaTourrette*[4] *and Aniruddha Mitra*[5]

Abstract ¾ *We have developed a computer vision teaching module consisting of materials for two or three lectures and a final project for use in a CS1 programming course. The final project is to write an image processing program with applications in computer vision. The program will read in a two dimensional array of data from a file that represents a black and white photographic image, perform one or more transformations on the data and write the transformed data to a new file. A simple image viewer program is used to display the before and after images. In addition to learning more about programming it is the intent of the project that students also have some fun with images. Most students did indeed enjoy the visual nature of the project and were surprised that they could write a program to accomplish so much after just one programming course. A few students wrote very creative transformation functions. This CS1 module is one of several developed as part of a CRCD project, sponsored by NSF. The modules are available for free use or adaptation by other instructors and institutions. http://www.cs.unr.edu/CRCD/*

Index Terms ¾ *Computer Vision, Image Processing, Programming Projects, Teaching Modules*

INTRODUCTION AND BACKGROUND

The use of computer graphics and image processing programs as teaching and motivational tools is becoming common at all levels of education. The motivation for introducing Computer Vision and Image Processing into the high school (and even middle and primary school) curriculum is vividly described by Thomas et al [1].

> "Vision is the sense through and by which we perceive and understand our world. … Learned eye-body coordination makes it possible for us to act and/or react smoothly and efficiently in all sorts of vision-guided situations. … It is also a powerful medium for communicating complex scientific ideas, especially those involving scientific processes. … We have never seen a technology so appealing to students of all ages as scientific visualization."

Thomas describes visiting a one-room elementary school in Montana and showing students images of Mars taken from the Viking spacecraft. The students became involved in using the images to answer questions about Mars such as the size of craters and characteristics of other geologic features. He states "Over the next hour the class took first an interest, then ownership, then pride in their investigation of Mars."

Several publications by Greenberg et al spanning almost ten years show a similar experience using image processing for upper elementary and secondary teaching [2] – [3]. Their project, "Image Processing for Teaching" (IPT) started as a response to the 1988 solicitation by NSF for Projects to Promote the Effective Use of Technology in the Teaching of Science and Mathematics. Part of the motivation for the IPT project derives from a 1983 survey of teachers in which nearly 85% of the respondents noted that their preferred style for both teaching and learning was visual.

Likewise, image processing has been used in general engineering education at the college level at several institutions. For example, Shultz describes the use of digital signal processing and image processing research experiences in the undergraduate electrical engineering curriculum [4]. Jankowski also describes the use of *Mathematica* for digital image processing teaching modules in electrical engineering education [5]. Jimenez-Peris et al have described their approach to adding depth to CS1 and CS2 courses through the use of several interactive programming projects including games, image processing, and other applications [6]. Andrews et al have also included image processing projects in CS1 through the use of class libraries which include graphics primitives [7].

These representative examples are a few of the many ways to include visually stimulating projects into the curriculum at a variety of levels. Our project emphasizes computer vision research and the teaching modules are one component which integrates computer vision with several basic topics covered in the computer science curriculum.

INTEGRATING COMPUTER VISION RESEARCH INTO THE CS CURRICULUM

Computer vision is an ideal area for integrating research with teaching. As described above, computer vision has an immediate appeal to most students due to their intimate

[1] Dwight Egbert, Computer Science Department, University of Nevada, Reno, egbert@cs.unr.edu
[2] George Bebis, Computer Science Department, University of Nevada, Reno, bebis@cs.unr.edu
[3] Meggin McIntosh, Director- Excellence in Teaching Program, University of Nevada, Reno, mcintosh@.unr.edu
[4] Nancy LaTourette, Computer Science Department, University of Nevada, Reno, latour@cs.unr.edu
[5] Aniruddha Mitra, Western Nevada Community College, mitra@wncc.nevada.edu

relationship to visual experience and people's fascination with this sense. Students have the opportunity to literally "see" the results of applying theory to solve practical problems. We believe that using computer vision research results can provide a high level of motivation to students. It is also an excellent learning tool for teaching students to integrate and use their acquired knowledge.

The successful integration of teaching and research is a challenging issue that requires sensitivity and careful planning. We believe that integrating research seamlessly into and throughout the curriculum and quantifying its effectiveness are the key factors to a successful research curriculum development program. Our philosophy is that students should be introduced to research as soon as possible in the undergraduate program. In contrast to traditional approaches which tend only to add senior level research courses, our approach for immersing students into research involves the systematic engagement of students into research through well-structured research activities which start from their sophomore year and continue until their graduation.

In order to introduce students to research as soon as possible, we are integrating computer vision results into traditional "core" courses. Not only does this integrated strategy expose the students to computer vision related research problems at an early stage of the curriculum but it also imparts cohesiveness to the course concepts and brings them closer to real life than is the usual practice. Students are introduced to the learning process by seeking answers to “what if” types of questions from them, encouraging them to predict the outcome of certain experiments and then asking them to explain their results. It should be emphasized that computer vision research results can be naturally integrated in these courses without detracting from the original teaching goals. We are developing research components for each of the following four courses:

CS201 – Intro. to Computer Science I (sophomore level)
CS/EE 236 – Intro. to Computer Engineering (soph. level)
CS30S - Data Structures (junior level)
CS/EE 336 - Microprocessor Engineering (junior level)

The integration of computer vision research results into these course is being done through self-contained modules in such as way as to make the integration easily transferable to similar courses. The modules include lecture notes, example student design projects, recommended reading materials, a prerequisite list, etc. The list of prerequisite knowledge can be used by instructors at other institutions to aid in deciding whether or not the research results are of an appropriate level for their course. Furthermore, the results of this project will be more easily disseminated, and the same or similar research results can be easily integrated into similar courses at other institutions.

COMPUTER VISION IN CS1

The Introduction to Computer Science (CS 201) course is a typical CSAB CS1 course, the first programming course for most students. The introduction of computer vision research in this course comes primarily at the end of the semester with two or three lectures and an image processing term project. One lecture covers research principles and computer vision basics followed by a second lecture dealing with questions about the project. Usually the project also brings out questions and discussion during parts of several more class periods as students progress through it. Image data are introduced as an example of two dimensional arrays. Basic concepts such as edges in images being changes in brightness are explained so that students can understand the processing they will perform. The project as implemented the first time consists of a menu driven image processing program with several functions including file read and write, negative, rotate, threshold, and basic filter. The necessary programming does not involve anything beyond what is normally covered in the course.

At the beginning of the project students are given a function to read a black and white PGM file and use this as a model to write their own function to write data back to a file. All other functions involve user input/output and simple array manipulations. One of the image processing functions is an operation of the student’s choice. Many students were very creative in their choice of operations and most students enjoyed the project. The visual feedback made them feel they could actually accomplish a lot with only one programming course.

During the introduction to computer vision lecture the principles of scientific research are reviewed so that students are reminded of the scientific method before beginning their exploration into image processing. The concept of images as two dimensional arrays is discussed at length. The relationship between our visual experience and the array element properties which must relate to that experience are demonstrated. For example, an “edge” in an image is a result of a transition between dark and light pixels along either side of a line. Likewise, thresholding a gray scale image to get a binary black & white image can be used to demonstrate regions and histogram analysis.

An effective introduction to computer vision is to ask students to raise their hands if they recognize the object shown in Figure 1.

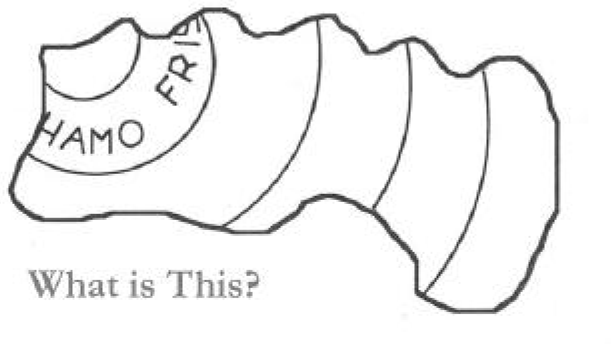


FIGURE. 1
IMAGE OF A WHAMO FRISBY USED TO INTRODUCE OBJECT RECOGNITION. (ADAPTED FROM [8])

After a few moments the object is identified as a chewed up Frisbee, common enough on college campuses. Then students are asked to explain how they identified the display as a Frisbee. This first glimpse of the difficulties involved in object recognition and computer vision demonstrates the difference between what we learn to do as humans and what we must do to extract information from arrays of numbers.

After the introduction of object recognition we examine a small portion of the image and compare the visual effect with the data array and the histogram. Figure 2 shows the "M" in Whamo after it is re-sampled to 16 by 18 pixels together with a histogram of the image.

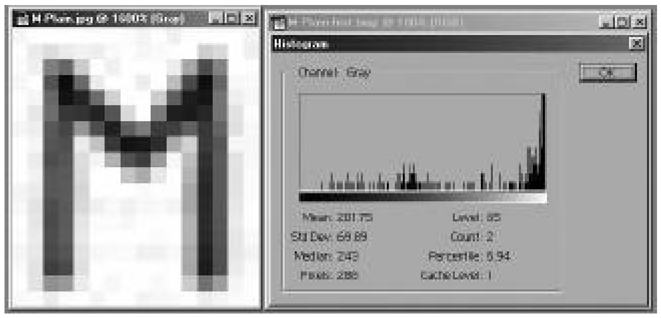
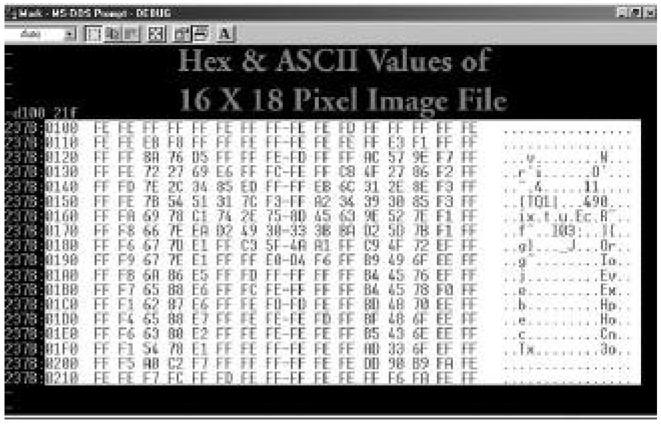


FIGURE. 2
"M" FROM WHAMO AND HISTOGRAM AFTER RESAMPLING TO 16X18 PIXELS.

This small image file can easily be examined with DOS Debug or other file viewers. Figure 3 shows a Debug display of the hex and ASCII pixel values for the "M" image file. The file display shows the different numeric values of array element for areas within and without the "M" demonstrating the relationship between intensity and array element value.

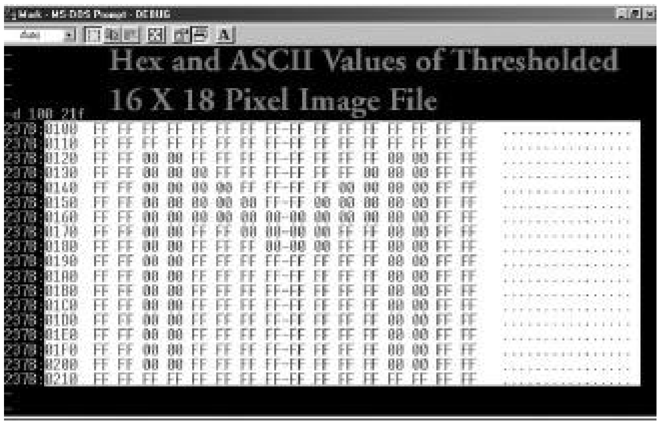


Hex & ASCII Values of
16 X 18 Pixel Image File

```
-d100 21f
2378:0100  FE FE FF FF FF FE FF FF-FE FE FD FF FF FF FF FE   ................
2378:0110  FE FE E8 F0 FF FF FE FF-FE FE FE FF E3 F1 FF FF   ................
2378:0120  FF FF 8A 76 D5 FF FF FE-FD FF FF AC 57 9E F7 FF   ...v........W...
2378:0130  FF FE 72 27 69 E6 FF FC-FE FF C8 4F 27 06 F2 FF   ..r'i......O'...
2378:0140  FF FD 7E 2C 34 85 ED FF-FF EB 6C 31 2E 8E F3 FF   ..~,4.....l1....
2378:0150  FF FE 7B 54 51 31 7C F3-FF A2 34 39 30 85 F3 FF   ..{TQ1|...490...
2378:0160  FF FA 69 78 C1 74 2E 75-8D 45 63 9E 52 7E F1 FF   ..ix.t.u.Ec.R~..
2378:0170  FF F8 66 7E EA D2 49 30-33 3B 8A D2 5D 7B F1 FF   ..f~..I03;..]{..
2378:0180  FF F6 67 7D E1 FE C3 5F-4A A1 FE C9 4F 72 EF FF   ..g}..._J...Or..
2378:0190  FF F9 67 7E E1 FF FF E0-D4 F6 FF B9 49 6F EE FF   ..g~........Io..
2378:01A0  FF F8 6A 86 E5 FF FD FF-FF FF FF B4 45 76 EF FF   ..j.........Ev..
2378:01B0  FF F7 65 88 E6 FF FC FE-FF FF FF B4 45 78 F0 FF   ..e.........Ex..
2378:01C0  FF F1 62 87 E6 FF FE FD-FD FE FF 8D 48 70 EE FF   ..b.........Hp..
2378:01D0  FF F4 65 88 E7 FF FE FE-FE FD FF 8F 48 6F EE FF   ..e.........Ho..
2378:01E0  FF F6 63 80 E2 FF FE FE-FE FE FF B5 43 6E EE FF   ..c.........Cn..
2378:01F0  FF F1 54 78 E1 FF FE FF-FE FE FF A0 33 6F EF FF   ..Tx........3o..
2378:0200  FF F5 A8 C2 F7 FF FF FF-FF FE FE DD 98 B9 FA FE   ................
2378:0210  FE FE F7 FC FF FD FE FF-FF FE FE FF F6 FA FE FE   ................
```

FIGURE. 3
DEBUG DISPLAY OF IMAGE FILE FOR "M" FROM WHAMO .

A threshold value to be applied to the image is then chosen based upon the histogram display. The shape of the histogram is described and the meanings of peaks and valleys are discussed. Figure 4 shows the Debug display of the thresholded "M".



Hex and ASCII Values of Thresholded
16 X 18 Pixel Image File

```
-d 100 21f
2378:0100  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF   ................
2378:0110  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF   ................
2378:0120  FF FF 00 00 FF FF FF FF-FF FF FF FF 00 00 FF FF   ................
2378:0130  FF FF 00 00 00 FF FF FF-FF FF FF 00 00 00 FF FF   ................
2378:0140  FF FF 00 00 00 00 FF FF-FF FF 00 00 00 00 FF FF   ................
2378:0150  FF FF 00 00 00 00 00 FF-FF 00 00 00 00 00 FF FF   ................
2378:0160  FF FF 00 00 00 00 00 00-00 00 00 00 00 00 FF FF   ................
2378:0170  FF FF 00 00 FF FF 00 00-00 00 FF FF 00 00 FF FF   ................
2378:0180  FF FF 00 00 FF FF FF 00-00 00 FF FF 00 00 FF FF   ................
2378:0190  FF FF 00 00 FF FF FF FF-FF FF FF FF 00 00 FF FF   ................
2378:01A0  FF FF 00 00 FF FF FF FF-FF FF FF FF 00 00 FF FF   ................
2378:01B0  FF FF 00 00 FF FF FF FF-FF FF FF FF 00 00 FF FF   ................
2378:01C0  FF FF 00 00 FF FF FF FF-FF FF FF FF 00 00 FF FF   ................
2378:01D0  FF FF 00 00 FF FF FF FF-FF FF FF FF 00 00 FF FF   ................
2378:01E0  FF FF 00 00 FF FF FF FF-FF FF FF FF 00 00 FF FF   ................
2378:01F0  FF FF 00 00 FF FF FF FF-FF FF FF FF 00 00 FF FF   ................
2378:0200  FF FF 00 00 FF FF FF FF-FF FF FF FF 00 00 FF FF   ................
2378:0210  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF   ................
```

FIGURE. 4
DEBUG DISPLAY OF THRESHOLDED IMAGE FILE FOR "M" FROM WHAMO.

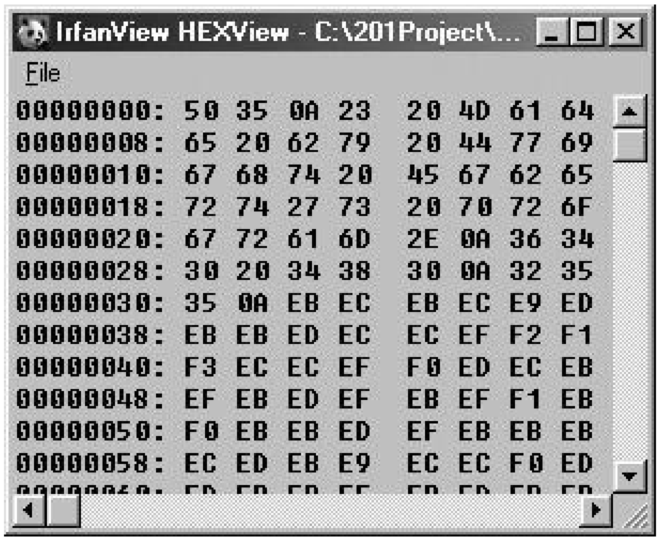
The result of the thresholding is now immediately obvious in the array element values which are either 0 or 255 (FF hex). The concept of edge detection is now discussed and the relative difficulties involved with detecting edges on raw and thresholded images can be demonstrated.

These and other concepts of computer vision, object recognition, and image processing are related to the programming principles and constructs which the students have been learning in the CS1 course. The images in particular give meaning to arrays and make the two dimensional arrays come alive.

THE PROGRAMMING PROJECT

The objective of the project is to write an image processing program with applications in Computer Vision which will read in a two dimensional array of data from a file that represents a black and white photographic image, perform one or more transformation operations on the data and write the transformed data to a new file.

A sample executable program and image files are available on the laboratory server for students to test. Students can view the image data files both before and after performing transformations on them using a freeware program *IrfanView* [9] which can be downloaded from: http://www.irfanview.com. *IrfanView* also allows students to read in images in a variety of different formats and save them in the format which we will be using for the project (portable graymap file format or PGM). An additional advantage of *IrfanView* is that it provides a hexadecimal display of the image file contents. This allows students to examine their image files for correctness and relate image contents to the hex displays used in lecture. An example hex dump from *IrfanView* is shown in Figure 5. The simple image header format for PGM can be read from the first few bytes. For example the basic code for PGM binary files is P5linefeed (ASCII values 50, 35, 0A in hex) followed by, comment lines which start with # (ASCII 23 hex), image size parameters and finally by raw image data.



```
00000000: 50 35 0A 23  20 4D 61 64
00000008: 65 20 62 79  20 44 77 69
00000010: 67 68 74 20  45 67 62 65
00000018: 72 74 27 73  20 70 72 6F
00000020: 67 72 61 6D  2E 0A 36 34
00000028: 30 20 34 38  30 0A 32 35
00000030: 35 0A EB EC  EB EC E9 ED
00000038: EB EB ED EC  EC EF F2 F1
00000040: F3 EC EC EF  F0 ED EC EB
00000048: EF EB ED EF  EB EF F1 EB
00000050: F0 EB EB ED  EF EB EB EB
00000058: EC ED EB E9  EC EC F0 ED
```

FIGURE. 5
IRFANVIEW HEX DISPLAY FOR SAMPLE PGM IMAGE .

Students may use one or more of the sample image files to test their program, they may use one of the digital cameras during a lab period to go out and take their own photos, or they may bring in one of their own photographic prints, slides or negatives to be scanned and converted to a data file. In addition to learning more about programming it is the intent of the project that students also have some fun with images.

Students are provided with the specifications for the PGM file format and a sample C++ function which will read such a file into a two dimensional array. The maximum array size is fixed and the read function has been written so that it does not contain any programming constructs which have not been covered in the CS1 course. The project is written in standard C++ and does not require special Windows functions. In our course we use Visual C++ and the laboratory machines use thin client stations with the server running Windows 2000. However, the programming project could just as easily be implemented on UNIX or LINUX systems. *IrfanView* is a Windows image viewer that is very easy and fast to use. However, GIMP or any one of several other viewers could be used.

Each semester the project transformations, format and design can be varied. During the first semester we used this project we defined ten transformations and functions which each student must implement using a menu structure in their main program. An example menu listing of the ten operations is shown in Figure 6.

All image transformations are performed on a "current image array" so that multiple transformations can be accommodated. This is consistent with many graphics and CAD systems which use a "current position" and/or a "current object". An example screen display is shown in Figure 7 which shows the main menu running in a DOS window together with three different images being displayed by three different instances of *IrfanView*.

```
Welcome to My Computer Vision Software (rev. 1.4)

Please Select the Operation You Want to Perform.

1.  Load current image array from file.
2.  Reload current image array from same file.
3.  Write current image array to file.
4.  Create a negative of the current image array.
5.  Specify filter array contents.
6.  Filter current image array.
7.  Threshold current image array.
8.  Rotate current image array 90 deg. clockwise.
9.  My own transformation of current image array.
10. Exit program.

Enter Menu Item >
```

FIGURE. 6
EXAMPLE MAIN MENU DISPLAY FOR STUDENT PROGRAMMING PROJECT.

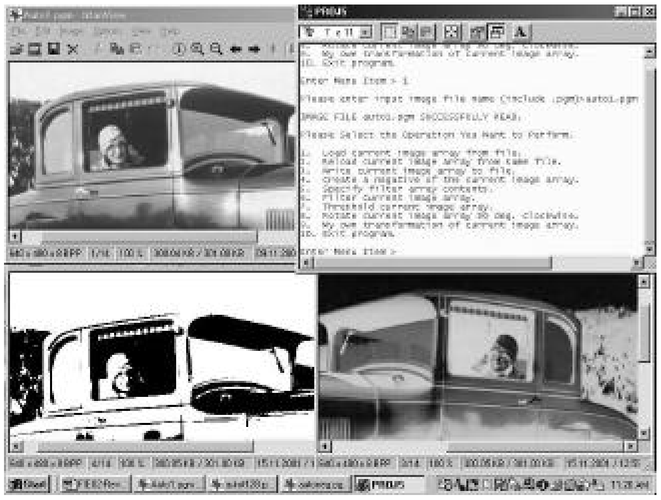


FIGURE. 7
EXAMPLE SCREEN DISPLAY FOR STUDENT PROGRAMMING PROJECT .

The top left image in Figure 7 is the original file, the bottom left is a thresholded image and the bottom right is a negative image that has also been flipped horizontally.

This is representative of the kind of user interface that can be achieved with readily available image viewer programs today. This does not impose the burden of direct Windows or graphics programming on students in a first programming course. Instead, they can concentrate on the programming principles they have been studying during the semester and still have the advantages of dynamic and fun image displays. They can get direct visual feedback showing the effects of their programming functions. In fact, at least one student was able to use the image displays to find and correct a programming bug which caused the output images to be distorted.

RESULTS

An integral part of this project is the assessment of results. One of several instruments we use is a survey of students in each course after the computer vision modules have been completed. The CS1 module was used in nine lab (three lecture) sections of CS201 during the Fall 2001 semester with a total of 105 students responding to the survey. During the Spring 2002 semester the module was used in eight lab (four lecture) sections and 73 students responded. The responses to the survey are included in Table 1. Students were asked to respond to each question on a scale of 1 (strongly disagree) to 10 (strongly agree) with 5 being neutral.[6]

TABLE. 1
SURVEY RESULTS FOR CS1 COMPUTER VISION MODULE .

QUESTION	Fall 2001*	Spring 2002*	Adjust. Avg.*
I had good knowledge of the course material before taking this course.	3.9/4.2	2.4/2.6	3.6
The CV module in this course improved the learning process and my understanding of the course material.	6.5/6.9	5.9/6.8	6.8
The CV module in this course improved my software design and programming skills.	7.1/7.4	6.5/7.4	7.4
The CV module in this course improved my skills to collaborate with others.	4.9/5.3	4.8/5.0	5.2
The CV module in this course improved my critical thinking skills.	7.2/7.8	6.2/6.6	7.3
The CV module in this course improved my writing and documentation skills.	4.9/5.4	4.4/4.7	5.1
Would you recommend this course to others?	6.5/7.0	4.8/5.5	6.4
The CV module in this course improved my general perception of CV.	6.4/6.7	6.4/7.3	7.0
The CV module made me feel like I should take more CV courses in the future, even if I don't pursue a career in CV.	5.4/6.1	3.9/4.5	5.4
The CV module in this course made me feel like I should pursue graduate studies even though they may not be in CV.	4.9/5.4	3.8/4.0	4.9
The CV module in this course mede me feel like I should pursue a research career even though it may not be in CV.	4.6/5.1	3.6/3.9	4.7
I was generally very attracted to the CV area prior to taking this course.	4.1/4.5	3.6/4.2	4.4
*Semester values are for: (all sections) / (all except anamolous section). Adjusted average values are for all except anamolous sections.			

[6] The original scale on the surveys was –2 to +2 with 0 being neutral. We have changed the scale for clarity of analysis.

At this point we are pleased with the integration of computer vision research into the CS1 course even though we will continue to make improvements. Over the two semester period presented there were seven different lecture sections. One unexpected finding is that the survey results for one of the sections each semester are notably different from the others. The overall means on every question are lower for these two sections by an average of five percent. Also, the distributions are skewed differently. There is not room here to fully explore this issue, nor is it central to this paper. Future work will examine variations with both lecture and lab sections as well as several other factors. However, for simplicity the last column in Table 1 shows only the adjusted average response for both semesters without the two anamolous sections. The adjusted average represents 120 out of the total of 178 reponses. The general trend in student survey responses is consistent. The main difference is that the trend would be dampened slightly if the anomolous sections were included.

The survey results show that most students thought the computer vision module helped them with the basic course content which was one of our main goals.

- Improved understanding of course material (6.8)
- improved programming skills (7.4)
- improved critical thinking skills (7.3)
- improved perception of computer vision (7.0).

Alternatively, students were neutral about the computer vision module=s ability to improve either their collaborative or writing and documentation skills. Students in our CS1 courses historically perceive that collaboration is discouraged because group-written code is not accepted on class assignments. All students are required to write their own code. Often at this level they are still learning the distinction between collaboration and plagiarism. Although students are required to submit program design documents, they do not always grasp the logical relationship between documentation and programming skills.

It is not surprising that the survey results show a neutral to slightly negative response towards a career in computer vision or research. More than half of the CS1 students are not computer science majors. Also, for many freshman level students it is a new experience to imagine themselves either in a research career or in graduate school. Likewise, the entry level CS1 course is demanding for most students and by the end of the semester when the survey is taken the prospect of extended graduate studies in CS must seem overwhelming.

When considered as a whole these results for CS1 are encouraging. The computer vision module is well received and students think it helps with the basic course content. Individual student responses, however, vary considerably. Many students went out of their way to let us know how much they liked the project and approximately ten percent of the students responded that they were very interested in computer vision and research as a career. In addition to the survey results in Table 1 some anecdotal and unsolicited comments on the final project which we received on the end of semester student evaluations are included for general inrterest.

- "The final project was outstanding."
- "The final project was ludicrous."
- "Final project was awesome."
- "The final project was a lot of fun while letting me excersise everything I had learned."
- "Final project should more closely reflect course material."
- "Final project – make more visual demos than verbal. Use computer to show what stuff does."

In general our student responses have been more consistent in the junior level courses than in this CS1 course which, although it is a sophomore course has a large freshman and non-major enrollment. We have found the experience challenging and rewarding.

CONCLUSIONS

Computer vision systems are already becoming commonplace, and vision technology will soon be applied across a broad range of business and consumer products. This means that there will be strong industry demand for computer vision scientists and engineers, for people who understand computer vision technology and know how to apply it in real-world problems. As a result of our integrating computer vision research experiences throughout our curriculum, many students may consider pursuing careers in computer vision.

For the majority of the students who will not pursue a career in computer vision, such a background will prove helpful in other areas such as pattern recognition, graphics, robotics, multimedia, virtual reality and medical imaging. We have also demonstrated that even the first programming course can support the introduction of computer vis ion research in a way that is meaningful to the course content. Computer vision and image processing provide valuable and interesting implementations for demonstrating basic programming concepts.

Teaching modules and support material can be downloaded from links found at our project website: http://www.cs.unr.edu/CRCD/.

ACKNOWLEDGMENTS

The work reported here has been funded in part by a grant from the National Science Foundation, Combined Research and Curriculum Development, (#0088086).

REFERENCES

[1] Thomas, D. A., K. Johnson, and S. Stevenson, "Integrated Mathematics, Science, and Technology: an Introduction to Scientific Visualization", *Journal of Computers in Mathematics and Sciecne Teaching*, Vol. 15, No. 3, 1996, 267-94.

[2] Greenberg, R., R Kolvoord, M. Magisos, R. Strom, and S. Croft, "Image Processing for Teaching", *Journal of Science Education and Teaching*, Vol. 2, No. 3, 1993, 469-80.

[3] Greenburg, R., "Image Processing for Teaching: Transforming a Scientific Research Tool Into an Educational Technology", *Jl of Computers in Mathematics and Sciece Teaching*, Vol. 17, No. 2, 1998, 149-60.

[4] Shultz, R. R., "Experience in the Integration of Digital Signal and Image Processing Research into the Undergraduate Electrical Engineering Curriculum", *Proceedings of the XXXX American Society for Engineering Education Annual Conference & Exposition*, Session 2632.

[5] Jankowski, M., "New Courseware Modules and Software for Digital Image Processing", *Proceedings of the 2001 American Society for Engineering Education Annual Conference & Exposition*, Session 1320.

[6] Jimenez-Peris, R., S. Khuri, and M. Patino-Martinez, "Adding Breadth to CS1 and CS2 Courses Through Visual and Interactive Programming Projects", *Proceedings of the 1999 ACM Special Interest Group for Computer Science Education*, 252-56.

[7] Andrews, P., D. Broline, W. Slough, and N. Van Cleave, "A Set of CS1 Labs Utilizing Graphical Objects and Inheritance", *Proceedings of the 2001 ASEE/IEEE Frontiers in Education Conference*, T3C-10-14.

[8] Hecht-Nielson, R., *Course Notes: Hecht-Nielson Neurocomputer Application Course*, San Diego, CA, January 1988.

[9] Skiljan, Irfan, *IRFANVIEW Freeware Image Viewer*, http://www.irfanview.com, (last accessed May 24, 2001).