SIPTool: The 'Signal and Image Processing Tool' An Engaging Learning Environment

Fred DePiero[1]

Abstract *— The 'Signal and Image Processing Tool' is a multimedia software environment for demonstrating and developing Signal & Image Processing techniques. It has been used at CalPoly for three years. A key feature is extensibility via C/C++ programming. The tool has a minimal learning curve, making it amenable for weekly student projects. The software distribution includes multimedia demonstrations ready for classroom or laboratory use. SIPTool programming assignments strengthen the skills needed for life-long learning by requiring students to translate mathematical expressions into a standard programming language, to create an integrated processing system (as opposed to simply using canned processing routines).*

Index Terms *— Image Processing, Signal Processing, Software Development Environment, Multimedia Teaching Tool.*

INTRODUCTION

The 'Signal and Image Processing Tool' (SIPTool) is a software environment that runs on a Windows™ PC. Two primary uses of the SIPTool are in-class demonstrations and student programming projects. The tool provides on-line sound processing from a microphone or WAV file, image processing from a video camera or BMP file, data plots for signals, graphics, text and display of numeric data. (See Appendix). Programming projects are written using Microsoft Visual C++™ and yield a DLL that is linked into the SIPTool. This architecture helps to isolate students' processing algorithms from the graphical user interface programming – so they can focus on course learning objectives.

LEARNING OBJECTIVES

The SIPTool is not a "toy'. Rather it is meant to be a serious development environment for image and signal processing algorithms. This type of software development provides a long term benefit to students. The programming assignments require students to translate mathematical expressions into a standard language (C). This will be an on-going task for students when they become engineers. Running a canned routine, as in MatLab, doesn't provide the same learning experience.

With the SIPTool, students create an integrated system that includes their processing routine along with image/signal acquisition and display. This integrated system is a very different result than the 'haphazard' line-by-line processing steps that students may or may not successfully stumble through in a MatLab environment, as they follow a given example. The SIPTool-based implementation is much more like a complete, commercial product.

A variety of learning objectives can be readily addressed with the SIPTool including: time/frequency relationships, 1-D and 2-D Fourier transforms, convolution, correlation, filtering, difference equations, and pole/zero relationships [1]. Learning objectives associated with image processing can also be presented, such as: gray scale resolution, pixel resolution, histogram equalization, median filtering and frequency-domain filtering [2]. Demonstrations are provided in the software distribution for all of the above learning objectives.

Processing routines within the SIPTool operate in a sample-by-sample fashion (on sound), and in an image-by-image fashion (on video). These processing styles are consistent with the implementation of real-time systems. Hence SIPTool programming also provides students with an exposure to the kind of development needed for real-time systems.

ENGAGING STUDENTS WITH THE SIPTOOL

Several factors help to engage and motivate students when working on programming projects and when using the SIPTool for demonstrations.

I. Visual Presentation

The SIPTool provides a forum for visualizing complex relationships with on-line display and processing. For example, students can use digital filters to process their own voice and to see results in real-time. This has proven to catch the attention of both students and faculty – each being seen "ooo-ing" and "ahh-ing" into a microphone and studying their own spectrum. Real-time display with a student's own processing makes for very effective demonstrations.

The extensibility of the SIPTool functionality via student programming makes this environment superior to other software packages that offer real-time graphical

[1] Fred DePiero, Cal Poly State University, Electrical Engineering Department, San Luis Obispo, CA 93407 fdepiero@calpoly.edu

display. 'CoolEdit' for example [3] provides input/output and graphical display capabilities, but limits the user to standard processing effects. In the SIPTool, students can investigate their own processing and their own modes of display.

II. Discovery and Problem Solving

The key elements of engineering labroatory work have been defined as: discovery, evaluation and investigation [4]. Students experience this process when they are required to test their SIPTool programs using natural signals of their own choosing. For example, they discover algorithmic limitations, evaluate various conditions and inputs when these problems occur, and investigate remedies through modifications to their processing algorithms or parameters.

A rapid compilation environment, and support for generic microphones and video cameras (thanks to Visual C++ and the SIPTool) make the testing and experimentation process relatively convenient. This promotes student exploration.

III. Builds Confidence and Communication Skills

Students can use the SIPTool to demonstrate their work to friends, family, and (potential) employers. In this setting, students may on occasion describe their work to others having less technical backgrounds. Discussions like this improve communication skills. Also, when a student describes his work to another, this serves to validate the student's own understanding and builds confidence.

The Windows platform was chosen for the SIPTool as it is the most common available. By making the SIPTool free, with unlimited distribution, students can install the program – along with their custom processing routies – on any number of machines. This promotes students' demonstrating their work to others.

III. Pride in Ownership and Accomplishment

During the development of a SIPTool program, students assign string variables with their name and with a brief description of their program. These data appear in the 'About…' dialog for the SIPTool. As such, the name and description are readily identifiable. This defines ownership explicitly, and provides uniqueness for each student's result. It also reinforces the fact that students have engineered something distinct – as opposed to simply using a tool to achieve some result. All these factors promote a pride in ownership and accomplishment for students.

Another advantage to using the SIPTool environment is a reduced learning curve, compared to the use of Microsoft Visual Studio alone. The SIPTool hides an enormous amount of details associated with the Microsoft environment, while still provding sufficient flexibility for student programming projects. This makes the SIPTool feasible for short (week long) assignments. It allows students to attain the sense of accomplishment more readily, with a reasonable amount of effort and time.

IV. Free Dissemination Encourages Varied Use

The SIPTool is shareware [5]. Unlimited downloads and support for a various Windows platforms, permit students to work either on-campus or at home. This wide dissemination is not only a convenience, but it also promotes many different uses, for example: Senior Projects, Master's Theses, Independent Studies, as well as class projects.

MULTIMEDIA CLASSROOM AND INTERACTIVE LECTURES

The SIPTool was used recently in a multimedia classroom with a computer connected to a large screen monitor. The tool brings a number of advantages into lecture, and promotes a highly interactive classroom environment.

The SIPTool presentation capability is superior to chalkboard and overheads, due to sound and video input - along with the real-time processing and display. "What-If" scenarios can be explored in classroom discussions. For example, asking students what effect certain processing parameters will have on a sound or image, and then running the SIPTool to see or hear the results. This promotes a lot of classroom discussion. Also, the simulation and graphical presentation help to reinforce derivations on the chalkboard.

SIPTOOL PROGRAMMING COMPLIMENTS USE OF OTHER TOOLS

The SIPTool is just one software tool used in Junior- Senior- and Graduate-Level courses at Cal Poly. Other tools include MatLab [6], DADiSP[7], and Code Composer Studio™ [8] (by Texas Instruments, for use with their real-time signal processing boards). These software tools are used in courses in discrete signals and systems, Digital Signal Processing, and Image Processing.

The SIPTool compliments learning objectives that can be addressed with other tools. It was not designed necessarily as a substitute. For example, tools such as MatLab and DaDISP are useful for filter design and for signal visualization. Also, many texts include MatLab examples and problems [9]. These are quite helpful for instructors and students.

However, before becoming practicing engineers, it is also valuable for students to implement their own systems, "from the ground, up". This is in contrast to only using the high level processing functions, available in a tool such as MatLab. Programming projects provide better on the job skills, as these are an experience that is much closer to actual system development. The SIPTool was designed as a platform for real-time implementation, and visualization.

Applying mathematics through programming is a useful skill to develop, as it will be a recurring need for many engineers. This experience is not achieved when a student simply runs canned routines, as in MatLab or DaDISP.

Even the best students appear to learn from the experience of implementing common signal processing functions in their own programs. This appears to be true for functions that Graduate students have used since their Junior year, but have not implemented (like correlation or the evaluation of a difference equation).

CONCLUSIONS

The 'Signal and Image Processing Tool' is a software environment for studying, demonstrating, and developing Signal / Image Processing concepts and techniques. It has a number of features that help to engage and motivate students. The SIPTool has been used at Cal Poly for three years, including use in lectures with a multimedia classroom. SIPTool programming assignments strengthen the skills needed for life-long learning by requiring students to translate mathematical expressions into a standard programming language, to create an integrated processing system.

REFERENCES

[1] Proakis, J., G., and Manolakis, D. G., *Digital Signal Processing Principles, Algorithms and Applications*, 3rd ed., Prentice-Hall, New Jersey, 1996.

[2] Gonzalez, R, C, and Woods, R, E, *Digital Image Processing*, Addison-Wesley, New York, 1992.

[3] "Syntrillium Software Corp.", 2001, http://www.syntrillium.com/ (May 28, 2001).

[4] Cooley, W, L, McConnell, R, L, Middleton, N, T, "Matching Laboratory Courses to Engineering Activities", *1994 IEEE Frontiers in Education Conference,* San Jose, CA, Nov. 94, pg. 496.

[5] DePiero, F. W., "SIPTool Home Page", SIPTool Signal and Image Processing Tool, 2000 www.ee.calpoly.edu/~fdepiero/csip_tool/csip_tool.html (March 9, 2001).

[6] The MathWorks Inc., 21 Eliot St., South Natick, MA 01760.

[7] "DSP Development Corporation Home of DADiSP", 2001 http://www.dadisp.com/ (March 9, 2001).

[8] Texas Instruments Inc., PO Box 1443, Houston TX 77251-1443

[9] Dorf, R, C, *Modern Control Systems,* 6th ed., Addison-Wesley Publication Co., 1992.

APPENDIX: SIPTOOL EXAMPLES

Figures 1-8 are windows that appear in a particular SIPTool demonstration. The demo is included in the software distribution, and illustrates concepts of digital filtering, difference equations, frequency response and pole-zero plots. The spectra and time domain plots shown in the actual SIPTool demo are computed based on a microphone input, and are updated in real-time. Signal (and image) plots also include a zoom feature for more flexible viewing.

Signals and processing algorithms appear in a redundant fashion in the demo, represented in different kinds of abstractions. For example, time and frequency domain plots of the input and output are shown to the user. The user can compare the salient features of each to better understand these important signal domains. Another concept that is presented in various representations is the processing of the filter. This is described mathematically (Figure 5) by showing the processing arithmetic. The pole/zero plot shown in Figure 4 is a representation for a mathematical transfer function that also describes the filter. Figure 3 shows the frequency response of the filter.

Filter processing can be adjusted using the controls shown in Figure 7. When the filter specifications are adjusted the student immediately sees changes in all of the graphical representations. The student also sees changes in the output signal. The integrated presentation and on-line processing, together with a rich set of graphics, makes it easier for the student to appreciate the interelationships between the various representations.

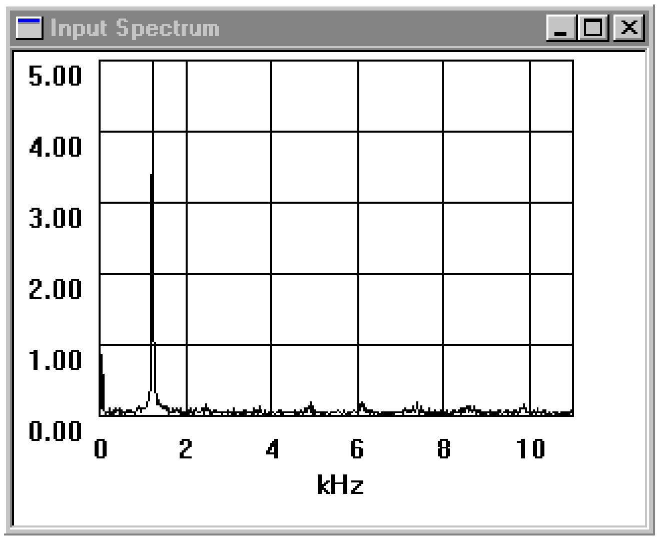


FIGURE 1.
SOUND INPUT IS CAPTURED FROM A MICROPHONE AND PLOTED IN REAL-TIME. THE SPECTRUM OF A WHISTLE IS SHOWN ABOVE.

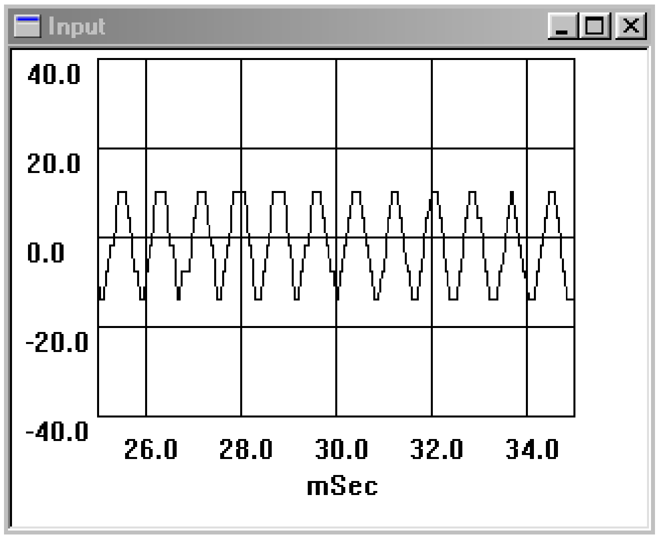


FIGURE 2.
SOUND INPUT IS ALSO DISPLAYED IN REAL-TIME, SHOWN IN THE TIME-DOMAIN. THE SAME WHISTLE FROM FIGURE 1 IS SHOWN ABOVE.

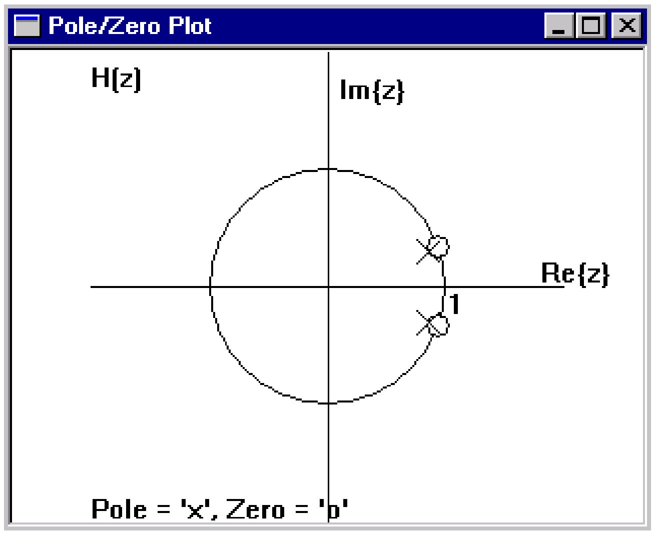


FIGURE 4.
THE FILTERING PROCESS IS DEPICTED IN VARIOUS WAYS. THIS POLE/ZERO PLOT SHOWS THE SALIENT FEATURES OF A MATHEMATICAL FUNCTION (TRANSFER FUNCTION) THAT DESCRIBES THE FILTER.

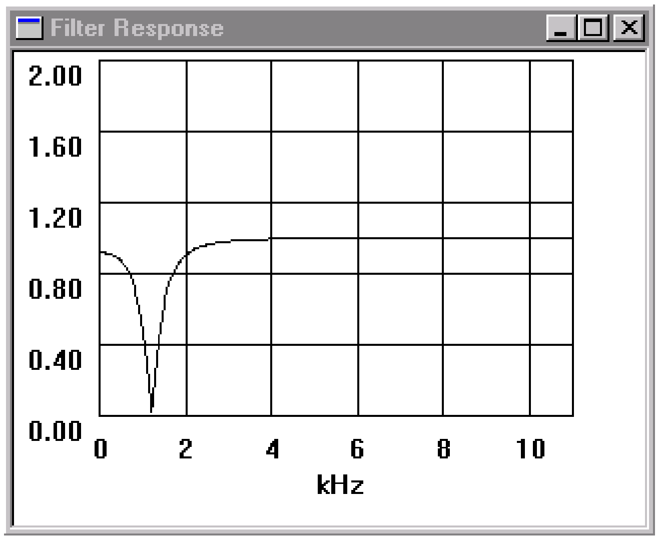
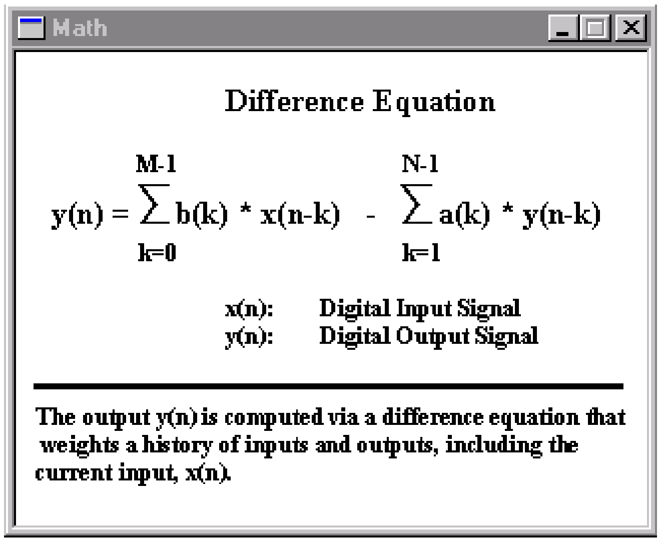


FIGURE 3.
USERS CAN ADJUST FILTERING PARAMETERS AND IMMEDIATELY SEE THE EFFECTS - SUCH AS CHANGES TO THE ABOVE FREQUENCY RESPONSE CURVE.



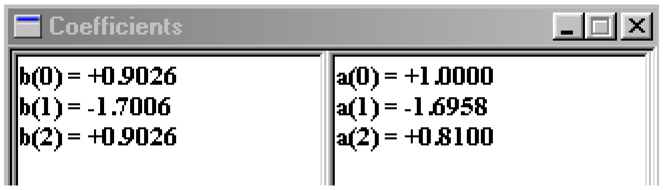
Difference Equation

$$y(n) = \sum_{k=0}^{M-1} b(k) * x(n-k) \; - \; \sum_{k=1}^{N-1} a(k) * y(n-k)$$

x(n):	Digital Input Signal
y(n):	Digital Output Signal

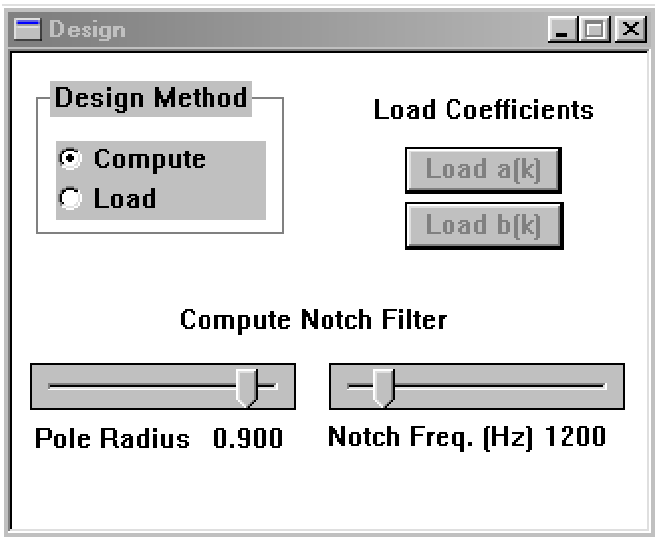
The output y(n) is computed via a difference equation that weights a history of inputs and outputs, including the current input, x(n).

FIGURE 5.
MATHEMATICAL DESCRIPTIONS, TEXT AND DIAGRAMS CAN ALSO BE PRESENTED IN THE SIPTOOL. THIS FIGURE SHOWS THE RELEVENT MATHEMATICAL DESCRIPTION OF A DIGITAL FILTER.



b(0) = +0.9026	a(0) = +1.0000
b(1) = -1.7006	a(1) = -1.6958
b(2) = +0.9026	a(2) = +0.8100

FIGURE 6.
THE SPECIFIC FILTER COEEFICIENTS ASSOCIATED WITH FIGURE 4 ARE ALSO DISPLAYED. THESE ARE UPDATED ANYTIME THE USER ADJUSTS THE FILTER CONTROLS (FIGURE 6).

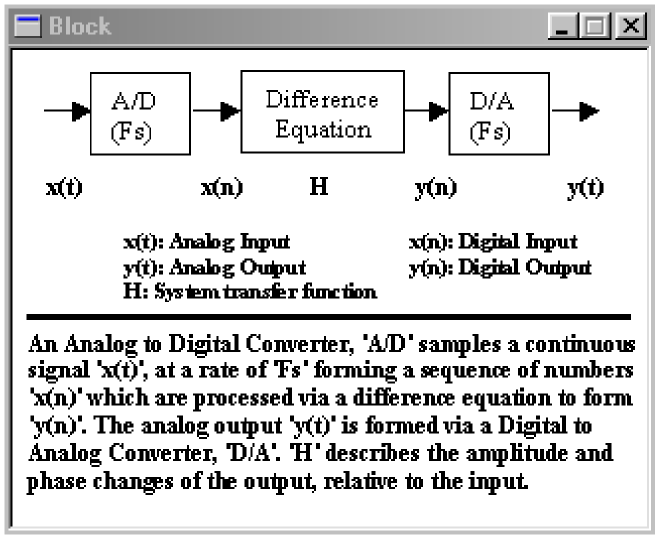


(● Compute ○ Load)

Pole Radius 0.900

Notch Freq. (Hz) 1200

FIGURE 7.
THE USER CAN ADJUST FILTER PROCESSING VIA THE SLIDER CONTROLS. THIS RESULTS IN NOTICEABLE CHANGES IN THE FILTER REPRESENTATIONS DEPICTED IN OTHER FIGURES, AND IN THE PLOTS OF THE OUTPUT SIGNAL (NOT SHOWN HERE).



x(t): Analog Input
y(t): Analog Output
H: System transfer function
x(n): Digital Input
y(n): Digital Output

An Analog to Digital Converter, 'A/D' samples a continuous signal 'x(t)', at a rate of 'Fs' forming a sequence of numbers 'x(n)' which are processed via a difference equation to form 'y(n)'. The analog output 'y(t)' is formed via a Digital to Analog Converter, 'D/A'. 'H' describes the amplitude and phase changes of the output, relative to the input.

FIGURE 8.
FILTER PROCESSING IS ALSO DESCRIBED IN BLOCK DIAGRAM FORM.

*****

Dr. Fred DePiero received his B.S. and M.S. degrees in Electrical Engineering from Michigan State University in 1985 and 1987. He then worked as a Development Associate at Oak Ridge National Laboratory until 1993, and completed his Ph.D. at the University of Tennessee in May 1996. Fred joined the faculty at CalPoly in September of 1996. His teaching areas include VLSI, signal and image processing, and computer engineering.

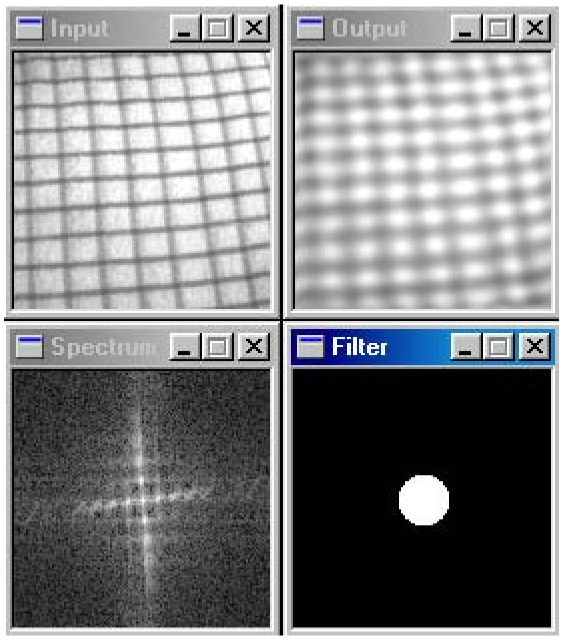


FIGURE 9.
IMAGE PROCESSING EXAMPLE: A LOW PASS FILTER OPERATES ON THE 'INPUT' IMAGE OF THE CLOTH BY REMOVING HIGH SPATIAL FREQUENCY COMPONENTS, RESULTING IN THE BLURRY 'OUTPUT'.

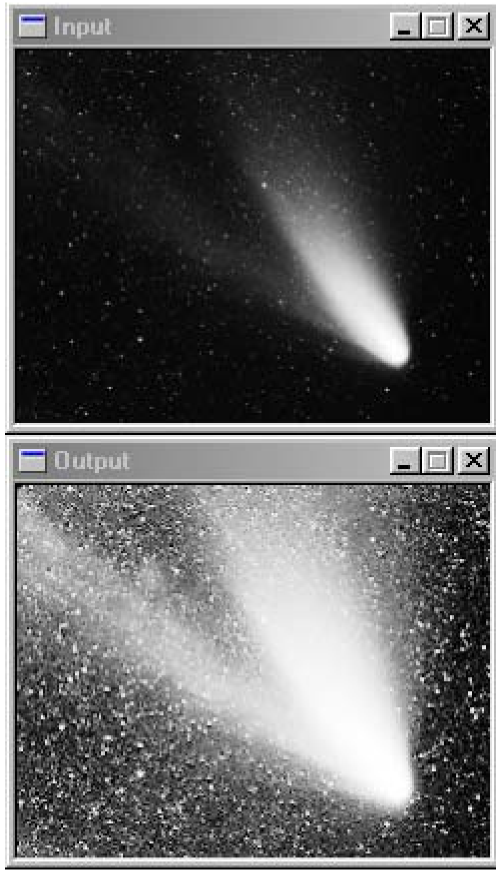


FIGURE 10.
A SECOND TAIL OF THE COMET IS REVEALED VIA IMAGE ENHANCEMENT.