DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED *ph*

LA-SUB--94-55

A comprehensive software system for image processing and programming

John Rasure, Stephanie Hallett and Ramiro Jordan

Department of Electrical and Computer Engineering
University of New Mexico
Albuquerque, NM 87131
rasure@wayback.unm.edu

*Final
Report*

MASTER

Abstract

XVision is an example of a comprehensive software system dedicated to the processing of multidimensional scientific data. Because it is comprehensive it is necessarily complex. This design complexity is dealt with by considering XVision as nine overlapping software systems, their components and the required standards. The complexity seen by a user of XVision is minimized by the different interfaces providing access to the image processing routines as well as an interface to ease the incorporation of new routines. The XVision project has stressed the importance of having: 1) interfaces to accommodate users with differing preferences and backgrounds and 2) tools to support the programmer and the scientist. The result is a system that provides a framework for building a powerful research, education and development tool.

1. INTRODUCTION

The following introduction discusses the motivation of the XVision designers and gives an overview of what XVision is and how it can be applied.

1.1 XVision description

XVision is a software system for image processing research and programming. XVision utilizes the X Window System† Version 11, which provides a network transparent windowing environment and software portability. XVision is designed to facilitate:

- data and algorithm exchange of new computer vision/image processing techniques,
- image processing training and education,
- development of turnkey vision solutions for various application areas (automation, medicine, biology, astronomy, etc).

XVision is a comprehensive system because it supports generation of new programs (extensibility), and integration, maintenance, modification and documentation of existing programs;

and it includes:

- three user interfaces; a user configurable form based interface, a quick command line interface that can be customized and a standardized command line interface. A visual programming language, xglyph, is under development.
- tutorials, manual pages, experiments, automated demonstrations and other supplemental documentation,
- an image processing library written in C,
- interactive image display and enhancement, image editing and creation, 2D, 3D, and contour plotting, and data creation/display via user specified functions.

The XVision project started in February of 1987 with its first release in August of 1987. This paper describes the second version which incorporates changes suggested by many of the users (over 30 different institutions) of XVision Version 1.0^{1,2,3}. One of the most important design goals of the XVision project is to provide for easy growth and extensibility. This has been accomplished by clearly defining software levels, software systems and their standard interfaces, and by providing programming tools and a variety of user interfaces.

1.2 Motivation

It is clear that one of the difficulties that face people attempting image processing and computer vision research is the exchange of data and results^{4,5}. Also, there seems to be a lack of "serious experimental research"⁶ that may be caused by the lack the readily available tools. XVision addresses these issues and also addresses the needs of those people that need to assemble image processing applications.

Many image processing software systems available now are primarily made up of a collection of data processing routines. The designers of these systems realize that modern software design paradigms can do much to improve the performance, extensibility and maintainability of the software^{7,8,9} but they have not been able to provide the broad image

† The X Window System is a trademark of M.I.T.

MAR 27 1990

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, make any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

**Portions of this document may be illegible
in electronic image products. Images are
produced from the best available original
document.**

processing community with a common foundation for research and development.

A more recent effort¹⁰ which is currently underway is the Image Kernel System being developed at the University of Lowell. This project is developing a device independent imaging environment and standards for an Applications Program Interface. Another effort targets the needs of the machine vision community by defining language constructs that are needed in their object-oriented language, OLIVE¹¹. XVision can be contrasted to the above systems by considering its comprehensiveness as outlined above, its use of existing standards and specifically the inclusion of programming tools, maintenance software and user interfaces designed for different programming domains.

1.3 Application

Advances in applied image processing have been tied to the evolution of computers. The image processing computer has changed over the years; first as special hardware and displays attached to mainframes, then came turnkey systems designed to solve specific problems and now the high performance workstation. It is the advent of the high performance workstation in a distributed computing environment that makes XVision a viable system for scientific signal processing applications.

The relationship of XVision with a distributed computing system is conceptually simple and is best described by Figure 1.

Figure 1. XVision and its relationship to a distributed computer system.

Because of the X Window System¹² the XVision client can be running on a local workstation while the actual computing is running on a remote host. This maximizes the use of the available computing power, allowing real time vision hardware and supercomputing facilities to be shared. Currently under development by the X Windows Consortium are extensions to X that will provide increased support for imaging applications¹³.

Activity	Software Tool	Description
New Algorithm	interface/program database, xvskel, xvform, xvinstall	The input and output specifications and documentation of the new algorithm are entered into the database and then used by xvskel to generate a program code skeleton. This generates the command line interface, while xvform is used to generate the graphical interface. Once the new algorithm is tested, it is integrated into the XVision system with xvinstall.
Experimentation	XVision image programming system (xvips)	xvips contains many common image processing routines. The graphical interface can be used to obtain documentation, help and examples that guide the user.
Visualization	vdisplay, xprism	vdisplay and xprism are network transparent, portable, window based visualization programs appropriate for 2D and 3D data.
Application	xvips logging system, journal/playback of graphical interface	The logging system builds an executable image processing program pipeline script, i.e. the application becomes a single or set of simple commands. Also, the application can be played back via the high level graphical interface.

Figure 2. XVision's use in building an application.

Given the above environment, how can a scientist or engineer develop an application? The list of activities will at least include: generation of data from physical measurements, the use of common algorithms for experimentation, the research and development of new algorithms, visualization of results and processes, and integration of the above steps to build the application. The use of XVision in the above activities is illustrated in Figure 2.

2. SOFTWARE COMPONENTS

Figure 3 presents XVision as a set of integrated software components that are used to build nine different software systems. A layered design approach and well defined standards are used in the design of the systems. Section 2 discusses the related software components, as described in Figure 3, and then Section 3 discusses in more detail several of the resulting systems.

System	Related Components	Standards Used
Instructional System	graphical interface (xvform), visual interface (xvglyph), tutorials, experiments and automated demonstrations (journal/playback)	interface/program database
Help System	graphical interface playback provides automated tutorials and animated help, xvman provides a manual facility	manual page and usage standard
Documentation System	xvskel, written tutorials, supplemental documentation on XVision design and xvman	manual page and usage standard, interface/program database
Applications Programming System	xvips, journal/playback of graphical interface and program execution log files to create new executable programs	C shell script programming interface/program database
Algorithm Programming System	xvskel	interface/program database, command line standard, XVision Image File Format (XVIFF), XVision Plot File Format (XVPFF), documentation and coding standards
User Interface System	xvips, xvskel, xvform, xvglyph and mkshort	command line standard and the interface/program database
Data Display System	vdisplay and xprism	X Window System V11, XVision Image File Format (XVIFF) and XVision Plot File Format (XVPFF)
Code Maintenance System	xvskel, vmake and revision control system (RCS)	not applicable
Data Processing System	data processing libraries, xprism, vdisplay and the XVision C programs and macros built from script files	XVision Image File Format (XVIFF) and XVision Plot File Format (XVPFF)

Figure 3. XVision's software components.

2.1 The X Window System

The goal of the X Window System project was to create a vendor-independent base window system to facilitate portable workstation software. "X is a network transparent windowing system developed at MIT which runs under UNIX X display servers run on computers with either monochrome or color bitmap terminals. The server distributes user input to, and accepts output requests from various client programs located either on the same machine or elsewhere in your network. Xlib is a C subroutine library that application programs ('clients') use to interface with the window system via a stream connection. While a client normally runs on the same machine as the X server it is talking to, this need not be the case."¹² X consists of Xlib, an interface toolkit and its intrinsics (Xtk), and a window or session manager. The X windows environment has been endorsed by most major computer manufacturers.

2.2 The graphical and command line user interfaces

The full functionality of the data processing programs can be obtained via several different levels of user interaction. All of the different user interfaces, existing and planned, are built upon a well defined program command line interface.

2.2.1 Command line interface. The command line interface is not intended for user interaction because of its length, but it is available to a user. For example, the command used to add two images is:

```
vadd -i1 image1.xv -i2 image2.xv -m imagemask.xv -o result.xv
```

The -flags are referred to as command line options or switches. Besides the options listed on the vadd command above, there are several other options that are available and that are common to all programs:

- -U prints out the usage for vadd and is also used to build the data processing programs interface database,
- -a [file1] attempts to find command line options in an answer file,
- -P prompts the user for command line options,
- and finally, a -A [file2] creates an answer file that when invoked with a -a [file1] will execute the command in the same way as when the answer file was created.

This command line interface is standardized across all of XVision because of the use of xvskel to build the programs. There are also Unix† shell variables that affect the interpretation of the standardized commands. vadd will print out a warning message and a default value for each option that is not used if the "-v" (verbose) flag is set in the environment variable "XVISION2_ENV". Also, vadd will log its execution occurrence into a command log file if the "-l file_name" option is specified in the environment variable "XVISION2_ENV".

Because the standard command line interface can often be too long or cumbersome, a shorthand user interface is also provided. The vadd command can be made to look like:

```
+ 1 xv 2.xv result.xv
```

The XVision user can use a program, mkshort, to automatically build and customize shorthand commands to fit his/her specific requirements. Mkshort parses the usage statement created by xvskel to obtain the form of the command. It then allows the name of the command to be changed and the non-optional arguments to be given without the switches. Up to one optional argument can be used without a switch, others will still need switches. Since some of the switches are no longer used, the order of the arguments is important. Mkshort creates a usage statement to define the order of the expected arguments.

With the use of mkshort and a working knowledge of the Unix c shell, an XVision user has an interface that is similar to an interpreted language for image processing. The XVision designers have never attempted to explicitly build an interpreted command line language for image processing because of its limited potential. The visual language presented later has much more potential.

2.2.2 Graphical interface. The previous XVision system ¹⁴ depended on xvmenu for all window based user interaction. The menu system produced an interface that was:

- highly functional (the menus took the appearance of simple forms that reflected past actions and input from the user)
- reconfigurable by changing the menu interface database (this could be done either with a text editor or at runtime with the metamenu program)
- and capable of recording a session to produce automated tutorials.

However, xvmenu was very difficult for the programmer to use in developing non image processing programs like vdisplay and xprism. Also, testing showed that the menus constrain an expert and perform slowly for a new user. Because of this, a new interface system, xvform, has been developed that has the following attributes: 1) journal/ playback, 2) automatically built from a database, 3) input and output error checking, 4) consistency in look and feel, 5) modifiable default settings, 6) localized and global help, 7) visual cues, 8) non-hierarchical access to functions (versus multi-level menus), 9) clear organization of data and input/output requirements and 10) multiple forms available at any instant.

Figure 4. The xvform user interface system.

The xvform system is actually a set of programs that are used to automatically generate a graphical interface from a high level specification in a database. The xvform system is best discussed first with an overview of the system relative to

† Unix is a trademark of AT&T Bell Laboratories.

Figure 4, and then by giving a short example. The interface/program database contains the information necessary to produce both a program skeleton (see section 2.5) and a graphical interface. Specifically, the database contains the following information:

- dialogue flow,
- user interaction type (example: toggle buttons, text entry, etc),
- interface display geometry (a CRT screen is assumed),
- program variables and an English description of them,
- default actions,
- and valid data ranges.

An example database for the *vadd* image processing program is in Figure 5. The accompanying graphical interface was created by following the *xvform* procedure outlined in Figure 6. The *vadd* pane corresponds to the database, while the rest of the form was created by extending the database to include several arithmetic routines.

```
#vadd - Add two images pixel by pixel.
-P 1 1 80x38+20+2 +1+1 'Addition Options' addpane
  -l 1 0 0 1 40x1+2+2+3+0 ' ' 'Input #1'
    'first input image' -i1 infile1
  -l 1 0 0 1 40x1+2+3+3+0 ' ' 'Input #2'
    'second input image' -i2 infile2
  -O 1 0 0 1 40x1+2+4+3+0 ' ' 'Output'
    'resulting output image' -o outfile
  -l 1 0 1 0 40x1+2+6+3+0 ' ' 'Mask'
    'operation gating image' -m mask
  -H 1 15x2+30+7+1+1 'Help' 'vadd man page' xvman vadd
  -a 1 0 15x2+30+9+1+1 'DO add' 'execute vadd' a x_add
-E
```

arithmetic	
<input type="button" value="add"/>	Addition Options
<input type="button" value="sub"/>	Input #1 <input type="text" value="infile1"/>
<input type="button" value="mul"/>	Input #2 <input type="text" value="infile2"/>
<input type="button" value="div"/>	Output <input type="text" value="outfile"/>
<input type="button" value="Help"/>	☐ Mask <input type="text" value="mask"/>
<input type="button" value="Quit"/>	<input type="button" value="Help"/> <input type="button" value="DO add"/>

Figure 5. A simple database and the graphical interface produced by *xvform*.

Procedure to add a new program to *xvips*.

Procedure to generate a custom X Windows application.

Figure 6. Programmers view of the *xvform* system.

2.2.3 A visual language for programming. The above discussion has summarized research and development that enhances programmer productivity and produces a superior interface. However, it has not addressed the problem of how to make an untrained user productive in a complex image processing and programming system. In the *XVision* system, a user is faced with literally thousands of possible actions. A visual programming environment, *xglyph*, is currently in the specification stage that will take advantage of the greater level of abstraction of a visual interface to simplify the users task. Key

components of the specification are:

- context specific visual representations,
- dynamic views of the interface,
- and hierarchical representation.

The visual primitives, referred to as glyphs, of xvglyph will represent the data processing programs and the macros built from execution log files. The glyphs will have several representations: a graphical interface (xvform), an English word/description, an icon, and a representative image showing the results of the particular process. The glyph set will be built from the interface/program database. The glyphs will be available to the user on a pallet. There will also be glyphs representing input devices/data and output devices/data. An applications solution is designed by connecting glyphs with pipes and tees.

Figure 7. Example of the appearance of the xvglyph user interface.

Xvglyph will allow the major steps of the image processing task to be put together and then the more detailed information about each step to be filled out. This is more of a top-down approach to a problem rather than the linear approach used by the menus or forms. Also this interface will allow fine tuning to be done easily by just changing a few steps (glyphs) and then allowing the whole string of routines to be run again without having the user set up and run each routine again.

2.2.4 Interface Summary. XVision has several interfaces to its image processing routines so that each user can find a comfortable interface and be encouraged to experiment. The best learning can be done when an interface is easy to use and does not intimidate the user. By minimizing the effort needed to put a set of routines together more time is left to experiment.

2.3 Display capability

The software components that provide visualization of data are vdisplay and xprism. Vdisplay runs on monochrome, grey level and color displays. The capabilities of the program are:

- image display (pan, scroll and zoom)
- pseudo-coloring via either color bars, color pallet, interactive look up table design or predefined look up tables,
- image tiling (insertion and extraction),
- image pixel value editing and display,

- overlay display and creation,
- image (mask) creation via a drawing facility (text, lines, boxes and fill).

Vdisplay is designed to maximize the human ability of image interpretation. It also has facilities to do interactive image enhancement through altering the colormap or region-of-interest operations. The colormap can be manipulated through a series of mouse clicks allowing a piece-wise-linear specification of the colormap that would otherwise have to be a very complex mathematical equation. The change in the image is immediately visible after each entry to provide quick feedback to the user. Vdisplay was designed to provide detailed information about the image that the human eye cannot detect. This information, when combined with the user's interpretation of the image, can aid in decisions for further processing of the image.

2.4 Plotting capability

The 2D/3D plotting program, xprism, is integrated with XVision to enhance the data analysis and creation capabilities. These capabilities include:

- 2D, contour, 3D, mesh and 3D scatter plot types;
- input data can come from image files, plot files, ascii data files, keyboard entry and user defined functions (this allows the user to generate an arbitrary function specified by an equation, plot it, and then convert it to an image);
- rotation, variable perspective and scaling of plots;
- multiple plots in a single window.

Xprism is useful for designing convolution kernels, examining image details, viewing color map data and viewing histogram data as well as providing a general plotting capability. Xprism can be executed as a single command with restricted functionality or as a user interactive program with full capability. By using xprism, the user can quickly scale, rotate, or even change from one type of plot to another with the same data set, allowing multiple views to obtain the full value of the information present. Also, the user can arithmetically combine any of the plots to produce new data.

2.5 Programming tools

XVision is designed to be a dynamic system which can grow or be divided to solve scientific data processing problems. This attribute requires sophisticated but simple to use tools. The current tools address the needs of two user domains:

- a user who has a knowledge of data processing algorithms, a familiarity with Unix, an average programming capability and a desire to add new capabilities to XVision
- a user unfamiliar with data processing theory but capable of using the applications programming system (see section 3.x).

xvskel and xvinstall provide an automatic programming and installation capability. xvskel builds a data processing program skeleton from a database specification. The skeleton program includes the code to implement the standard command line interface, input/output error checking, usage information and the program data structure. The user is responsible for coding the actual algorithm or combining existing libraries in a new way. xvskel also automates the production of the necessary make files, the include files and the documentation outlines.

xvinstall takes the debugged program and installs it into the XVision system by updating the data processing library, installing the completed documentation and installing the executable program. Then, vmake is used by the XVision maintainer to incorporate bug fixes into the multi-vendor environment.

xvglyph will be used by an applications programmer who wants to use and combine the existing data processing capabilities to perform experiments and then build turnkey solutions. xvglyph should be thought of as a programming language that produces complex data processing programs from primitive programs.

3. SOFTWARE SYSTEMS

The previous section presented XVision at the component level. The following discussion presents four of the nine resulting integrated systems and how they are used to develop new algorithms and applications, see Figure 8. The integration of the software components is made consistent by enforcing a documentation, data structure, user interface, or programming standard.

3.1 Instructional and learning system

The acceptance of XVision as a research and development tool depends on how easy it is to learn and on how easy it is to produce the desired results. This depends on a carefully implemented learning and instructional system that should be diverse in learning situations and styles, and be rich in support materials and documentation¹⁵. XVision provides both *guided exploration* and *active learning*.

Active learning is possible with both the form based interface and the glyph based interface. The user can compose solutions to image processing problems and immediately see the results. Xvglyph goes one step further to make the novice more successful by hiding the details of the data processing options.

Guided exploration is possible because of the design of xvform's journal/playback capability. Simple tutorials can be automatically performed by the xvips system by using the playback of an expert's solution to a problem, i.e. animated help. The expert's actions are recorded in the journal. The journal differs significantly from the log script file that is also built during the expert's session. The journal must record interface actions (selections) while the log file records the list of programs executed. The journal can be used as a live example of how to use the graphical interface and thus shorten the learning curve involved in using a new interface.

Figure 8. Structural view of XVision's systems and components.

3.2 Applications programming system and xvips

The building of an application with XVision is outlined in Figure 2. It is the xvips, XVision image programming system, that facilitates the application programming activities by providing the user with an interface to the primitive data processing operations needed to build a particular application. These primitives are C programs that perform functions like convolution, Fourier transforms and dynamic thresholding.

The primitives can be combined in two ways:

- Execution of all of the primitives via any of the interfaces (graphical or command line) is logged into a file in c shell format. A sequence of executions produces an executable macro or program. This program can then be added to the xvips database as a new program.
- Actions taken with the graphical interface can be recorded into a file and played back within xvips.

xvips provides the user with a palette of forms similar to the one in Figure 5. Currently there are over 100 primitive operations organized into about 20 forms. The xvips environment is specified with the interface database. The data processing power of the applications programmer is limited by the number of image processing primitives or algorithms available. The data processing capabilities of XVision can be extended by using the algorithm programming system.

3.3 Algorithm programming and data processing system

The algorithm programming system is targeted at the individual which is doing research or development in the image processing field. This programming domain assumes a working knowledge of C programming and UNIX. The algorithm programming system is designed to support redesign and reuse of software^{16,17}. The attributes of the algorithm programming system that allows reuse and redesign are:

defined software levels; the C function library, the C programs built from the library function, the standardized command line interface, the program/interface database and the data file formats, support tools; xvskel, xvinstall and xvform,

the thesis that a complex tool is warranted only if it eases the use of the system.

Currently, the data processing system contains a C library of image processing functions, xv2lib.a, and a set of about 100 C image processing programs built from these functions. As outlined earlier, the algorithm programmer can use xvskel to build the documentation and program outlines needed to incorporate new C functions into the data processing system.

4. CONCLUSION

XVision represents a research effort that is attempting to develop a very productive scientific work environment. This is being done by utilizing existing standards, developing new standards, developing tools and researching user interfaces. The initial efforts have produced a workstation based system that is currently being used by many institutions to develop applications and algorithms. It will be the evolution of the existing tools and the development of a visual programming language that will produce the next generation scientific research and development tool.

ACKNOWLEDGMENTS

The XVision system has been realized through the direct efforts of the following people: Danielle Argiro, Eric Engquist, Rick Ferreri, Stephanie Hallett, Ramiro Jordan, Ron Neher, Tom Sauer, Marcelo Teran, Scott Wilson, Cliff Wood and Mark Young.

REFERENCES

1. John Rasure, *Interactive Image Processing Using X Windows*, Electronic System Design, December 1987, pp. 32-33.
2. Richard Krukar, *Estimating Kernels from Amplitude Spectrum*, Electronic System Design, December 1987, pp. 35-36.
3. Scott Wilson, *Image Restoration Techniques*, Electronic System Design, December 1987, pp. 39-42.
4. Robert Haralick, *Computer Vision Theory: The Lack Thereof*, Proceedings of the Third Workshop on Computer Vision: Representation and Control, October 13, 1985. pp 113-121.
5. Keith Price, *I've seen your demo; so what?*, Proceedings of the Third Workshop on Computer Vision: Representation and Control, October 13, 1985. pp 122-124.
6. Jorge Sanz, *A letter to A. Rosenfeld*, IEEE PAMI technical committee newsletter, October, 1988.
7. Alan Paeth, *The IM Raster Toolkit: Design, Implementation and Use*, University of Waterloo, Computer Science Department, December 1986.
8. Peterson, Bogart and Thomas, *The Utah Raster Toolkit*, Third Usenix Workshop on Graphics, Monterey California, November 1986.
9. Landy, Cohen and Sperling, *HIPS: A Unix-Based Image Processing System*, Computer Vision, Graphics, and Image Processing 25, 331-347, 1984.
10. Coelho, Drake and Miner, *The Imaging Kernel System (IKS): A Device Independent Imaging Environment*, University of Lowell Centre for Productivity Enhancement, 1987.
11. Lavin and Flickner, *An Object-oriented Language for Image and Vision Execution (OLIVE)*, IBM Almaden Research Center, 1988.
12. Gettys, Newman and Schiefler, *Xlib - C Language X Interface Protocol Version 11*, Massachusetts Institute of Technology and Digital Equipment Corporation, 1988.
13. Joseph Mauro, *X Image Extension Concepts*, Digital Equipment Corporation, Merrimack, New Hampshire, rev 2.4, June, 1988.
14. John Rasure, et. al., *XVision: A Comprehensive Software System For Image Processing Research, Education and Applications*, ACM SIGGRAPH User Interface Software Symposium, Banff, Alberta, Canada, October, 1988.
15. John Carroll, *Minimalist Design for Active Users*, Human-Computer Interaction: A Multidisciplinary Approach, 1987. pp. 607-626.
16. Will Tracz, *Reusability Comes of Age*, IEEE Software, July 1987. pp 6-8.
17. Gerhard Fischer, *Cognitive View of Reuse and Redesign*, IEEE Software, July 1987. pp 60-72.