XIL White Paper

2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

SunSoft
A Sun Microsystems, Inc. Business

The XIL Imaging Library

Introduction to the XIL Imaging Library

The Solaris XIL™ Imaging Library provides a basic set of functions for imaging and video applications. The XIL library provides the imaging and video component of the Solaris® Graphics Architecture, a strategy for providing low-level software interfaces known as foundation libraries; application and API developers can port their code to such foundation libraries. The XIL library is complemented by the XGL™ Graphics Library, which addresses application and API requirements for geometry-based graphics.

Solaris Graphics Architecture

The XIL foundation library is an integral part of the Solaris Graphics Architecture. The Solaris 2.x software, using loadable drivers, enables display devices using the Solaris Graphics Architecture to be easily installed and used, without requiring kernel modifications. The Solaris Graphics Architecture, through the XIL, XGL, and OpenWindows™ software, provides a means for third-party hardware and software vendors to develop applications with the knowledge that their investment will see long-term benefits, including access to a range of computing platforms and complete integration into the Solaris environment.

Figure 1 illustrates how the XIL foundation library fits into the Solaris Graphics Architecture.

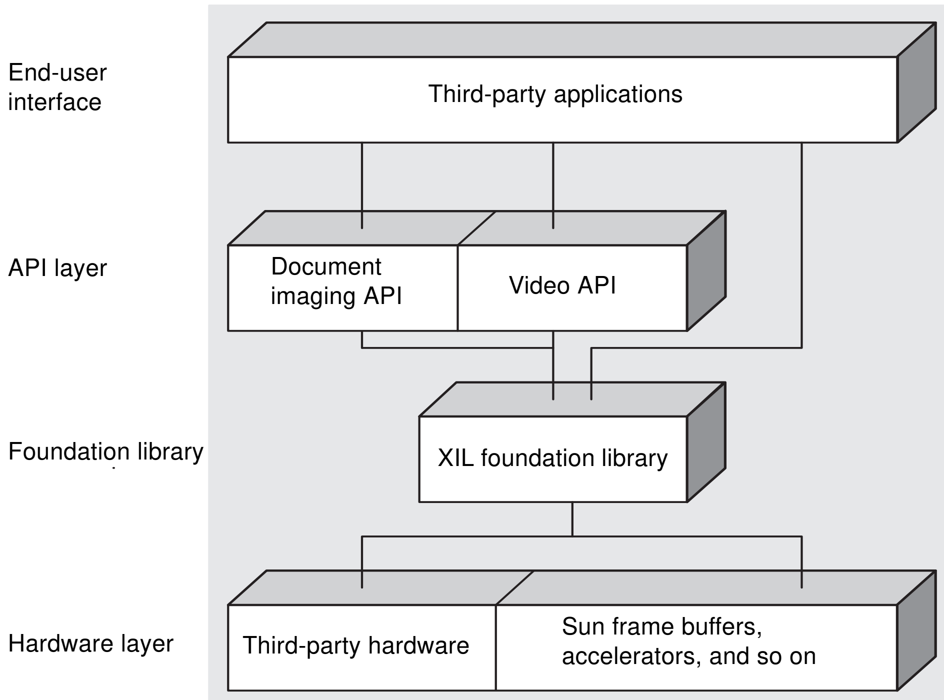

Figure 1 Foundation Libraries and Application Programming Interfaces

The XIL software contains three primary components:

- A C programming interface specification for basic imaging and video functionality.
- A high-performance implementation of the specification tuned for specific Solaris platforms.
- A C++ standard hardware interface specification, which enables third-party hardware developers to readily support the XIL library and applications built on it.

XIL Applications

The XIL software does not address any one set of customers, but rather provides the basic imaging and video capabilities required by a number of markets, including:

- Markets that require digital video technology
- Commercial document imaging
- Technical document imaging
- Desktop publishing
- Color prepress
- Graphics arts

Markets Requiring Video Technology

Developers working with digital video technology are interested in writing applications that broadcast movies (for educational purposes or business training), in video conferencing, in multimedia mail, and so on. The XIL software provides these developers with the functions they need to capture video, compress and decompress it, and finally display it. The XIL software supports JPEG (baseline sequential and lossless), MPEG 1 (decompression only), H.261 (decompression only), and Sun's Cell and CellB compression formats.

Commercial Document Imaging

Banks and other financial institutions, airlines, and insurance companies process enormous numbers of paper documents in a typical day. Developers addressing this market must provide applications that read digital versions of documents from a scanning device, compress the images into CCITT Group 3 or 4 format, index them, and store them. Later, it may be necessary to retrieve the documents in a rapid manner, or to perform simple image processing, rescaling, and sharpening operations.

Developers of applications or APIs for this market can take advantage of the following XIL functions:

- Processing of single-bit images
- CCITT Group 3 and Group 4 compression of images for storage and transmission
- Rapid access to images and rapid image display
- Rescaling, translation, rotation, and convolution operations

Technical Document Imaging

This market handles fewer images than the commercial market, although the images are generally much larger. Potential applications for the XIL library in the technical document imaging market include textile design, cartography, and architecture. The XIL software enables applications in this market to process very large images with a large number of bands. The width, height, and number of bands in an XIL image can each be as great as 2^{32}- 1 (limited by virtual memory).

Desktop Publishing

Desktop publishing products must be able to capture or import images. The XIL software enables developers of desktop publishing products to use imaging accelerators to resize images, and to filter and convert them from one data type to another. The software's advanced imaging functions make it a better choice for managing images in desktop publishing applications than other pixel-oriented interfaces such as the Xlib software.

Color Prepress

A typical customer in this field wants to scan in color photographs and prepare them for printing on a color printer. The first step in this preparation might be to perform simple image processing operations on the images, such as sharpening, blurring, or arbitrary rotation. The images may also require cropping. Finally, the images must be converted to the CMYK colorspace, which provides the digital equivalent of photographic color separation. The individual bands in the resulting CMYK image might then be dilated to avoid registration problems.

Graphic Arts

Applications addressing the graphic arts market would use the XIL software for two principal tasks: painting and retouching images. Graphics arts applications can utilize the software's imaging capabilities for compositing, color correction, image enhancement, and scaling.

XIL Data Objects

The XIL library defines a number of data objects that you can create and manipulate using XIL functions. These objects are shown in Figure 2.

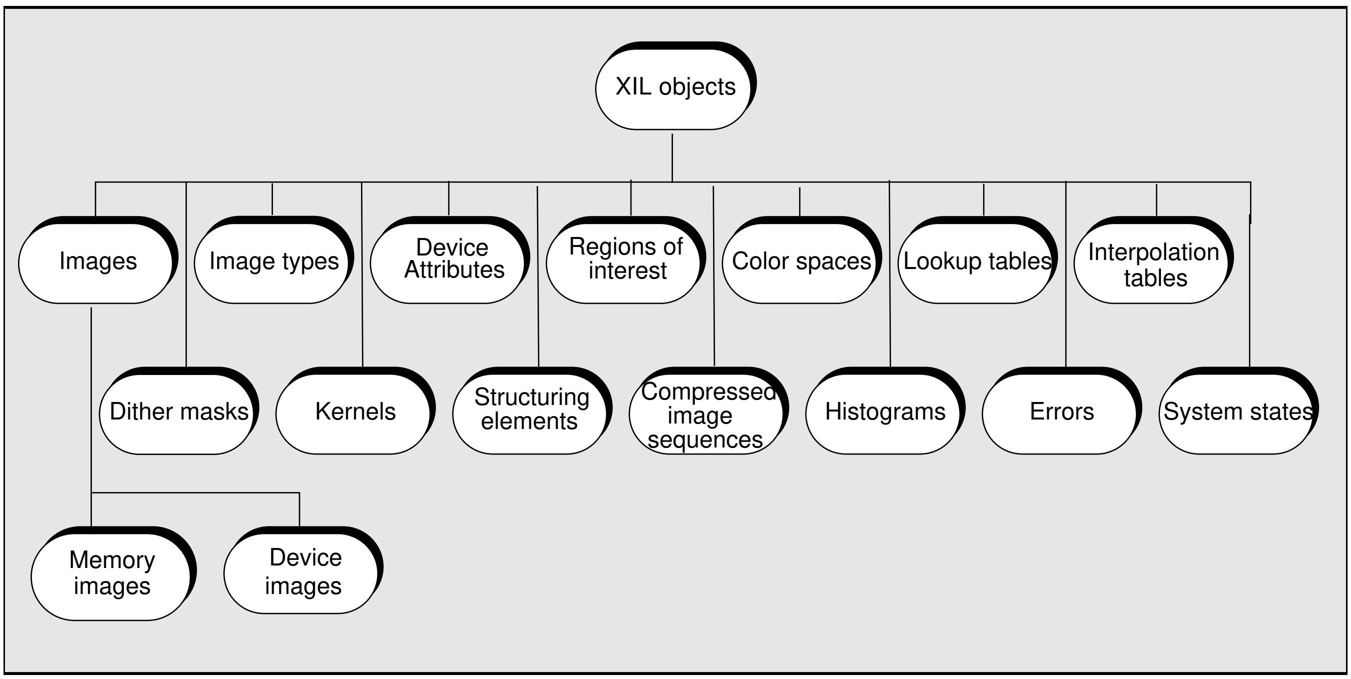

Figure 2 XIL Data Objects

All of these data objects are opaque. You read and set an object's data members using XIL functions designed for those purposes.

Images

The most important XIL data object is the image. This object stores a set of pixel values and values for a number of image attributes, such as width, height, number of bands, and data type. The library supports images of virtually any size: an image's width, height, and number of bands can each be as great as 2^{32}-1. In addition, the library supports three types of data: 1-bit data, 8-bit unsigned data, and 16-bit signed data.

Note – The XIL library does not currently support multiband images in which different bands have different dimensions. Thus, there is no direct support of 4:1:1 or 4:2:2 data.

When an application creates an image, the application receives an opaque handle to the image and, therefore, must process the image using XIL functions exclusively. However, the library enables applications that need to manipulate pixel values directly to *export* images. When an application exports an image, it gains access to information about how the image is stored in memory, including the address at which the image's pixel values are stored. The pointer in which this address is stored remains valid until the application *imports* the image back into XIL space.

Typically, when an XIL image is created, both the image structure that describes the image and the image's pixel values are stored in system memory. This type of image is called a *memory image*. All operations that do not read or write a device use memory images exclusively.

In addition to memory images, the XIL library supports *device images*. These images represent such devices as displays, cameras, frame grabbers, scanners, and printers. For each device image, the structure describing the image is stored in system memory, but the image's pixel values are read from or written to the device. A device image can be used as either the source or destination image for an XIL operation.

Display images are a special subclass of device images. A display image is associated with an X window and enables an XIL application to write directly to a screen.

Image Types

For every XIL image, there is a subset of attributes that constitutes its image type. These attributes are its width, height, number of bands, data type, and color space. An application might get an image's image type in preparation for creating an X window in which to display the image.

Device Attributes

Typically when working with device images, an application creates the device image, then sets its attributes. Some device types, however, require that the device attributes be defined when the device image is created. For these device types, the application can create and associate a *device object* with the device type, and set the device attributes on the object. The device object can then be passed as an argument on the function that creates the device image, thereby creating the device image and simultaneously setting its required attributes.

Regions of Interest

In the XIL scheme, a region of interest is attached to an image rather than to an operation. This region of interest is a single-bit mask that determines either which pixels in a source image can be read or which pixels in a destination image can be written. You build this mask by adding rectangles to an empty region of interest or by converting an XIL image or an X region to a region of interest.

The library calculates the region of interest for an operation by taking the intersection of the regions of interest of the images involved in the operation.

Color Spaces

A color space data object stores information about an image's color space. The color spaces supported by the XIL library include:

- CCIR Rec. 709 RGB
- A linear version of CCIR Rec. 709 RGB
- CCIR Rec. 709 YCC
- A luminance-only space derived from CCIR Rec. 709 YCC
- A linear version of the luminance-only space mentioned above
- CCIR Rec. 601 YCC

- A luminance-only space derived from CCIR Rec. 601 YCC
- A YCC space defined by Eastman Kodak Company for its Photo CD™ product
- A linear CMY
- A linear CMYK

An application generally attaches color-space objects to images in preparation for converting an image from one color space to another.

Lookup Tables

Lookup tables are used in two ways in XIL applications. First, they are used for general image modification. A single-band image of any data type can be converted to a single-band or multiband image of any data type; or, a multiband image of any data type can be converted to a multiband image of any data type. The correspondence between pixel values in the source and destination images is user defined. Second, lookup tables are used in dithering operations. In a dithering operation, a single-band or multiband image of any data type is converted to a single-band image of any data type.

Interpolation Tables

For geometric transformations that can result in non-integer pixel values, the XIL library supports the following methods for interpolating pixel values in a destination image: nearest-neighbor, bilinear, bicubic, and general. For each method except general, the XIL library provides the filter for interpolating the values; for a general interpolation, the application defines the filter.

For defining the general-interpolation filter, the XIL library provides an *interpolation table*, which is an array of 1xn kernels that represents the interpolation filter in either the horizontal or vertical direction. Interpolation tables let an application determine how many pixels are used to calculate interpolated pixel values, weight the pixels used in the calculations, and divide the space between adjacent pixels into multiple areas so a destination pixel's value can be interpolated differently, depending on which area it falls in. The interpolation tables are set on the system-state object (see "System States" on page 12), and the tables affect all general interpolation operations using images created from that system state.

Dither Masks

A dither-mask object is one of the required parameters to the XIL library's ordered-dither operator. The mask is a three-dimensional array whose depth matches the number of bands in the image being dithered. The values in the mask act as threshold values that help eliminate contouring in the dithered image.

Kernels

A kernel is one of the required parameters to the convolution operator. The kernel is a two-dimensional array of single-precision floating-point values, one of which is designated as the kernel's key value. When the library performs a convolution operation, it aligns the kernel's key value with the source-image pixel it is currently processing.

The values in the kernel determine the effect that the convolution operation will have. For example, the operation may sharpen an image, smooth an image, or highlight the edges in an image.

Structuring Elements

A structuring element serves as a parameter to the erosion and dilation operators and defines the neighborhood of pixels in the source image to be used in calculating the value of each pixel in the destination image. This structuring element looks like a convolution kernel except that its member values are Boolean.

Compressed Image Sequences

A compressed image sequence is a container for compressed images. These images may represent a series of still images, frames from a movie, or pages from a document, and they may have been compressed in one of several formats:

- Cell or Cell B (formats developed at Sun)
- JPEG baseline sequential or lossless
- MPEG I
- CCITT Group 3 or Group 4
- H.261

Histograms

When an XIL application calculates an image's histogram, it writes its results to a histogram object. This object contains a set of bins that store information about gray or color levels in the image.

Errors

When an XIL error occurs, the library generates an error object. This object contains such information as an error ID and a string containing a description of the error. The XIL library includes a default error handler that gets values from this object and prints a message to `stderr`. It also contains a set of error-handling functions that get information from this type of object. You can use these functions to construct a custom error handler.

System States

A system-state object is created when an application initializes the library. Among other things, this object is used to maintain a list of the data objects the application has created, and to keep track of the current error handler.

XIL Functions

The XIL library provides these functions to operate on the data objects described in the last section.

Image Processing Functions

The image processing functions in the XIL library can be broadly grouped under the following headings: arithmetic and logical functions, geometric functions, and miscellaneous functions.

Arithmetic, Relational, and Logical Functions

The functions in this class can be used to lighten or darken an image, to increase or decrease contrast in an image, or to produce the negative of an image. Some of these functions require two source images (see Table 1); some require a source image and a constant (see Table 2); and others require only a single source image (see Table 3).

Table 1 Arithmetic, Relational, and Logical Functions Involving Two Source Images

Function Name	Description
`xil_add()`	Adds two images
`xil_subtract()`	Subtracts one image from another
`xil_multiply()`	Multiplies two images
`xil_divide()`	Divides one image into another
`xil_max()`	Finds the larger of pixels in two images
`xil_min()`	Finds the lesser of pixels in two images
`xil_and()`	Finds the logical AND of two images
`xil_or()`	Finds the logical OR of two images
`xil_xor()`	Finds the exclusive OR of two images

Table 2 Arithmetic and Logical Functions Involving a Source Image and a Constant

Function Name	Description
`xil_add_const()`	Adds an image and a constant
`xil_subtract_const()`	Subtracts a constant from an image
`xil_subtract_from_const()`	Subtracts an image from a constant
`xil_multiply_const()`	Multiplies an image by a constant
`xil_divide_by_const()`	Divides an image by a constant
`xil_divide_into_const()`	Divides an image into a constant

Table 2 Arithmetic and Logical Functions Involving a Source Image and a Constant

Function Name	Description
`xil_and_const()`	Finds the logical AND of an image and a constant
`xil_or_const()`	Finds the logical OR of an image and a constant
`xil_xor_const()`	Finds the exclusive OR of an image and a constant

Table 3 Arithmetic, Relational, and Logical Functions Involving One Source Image

Function Name	Description
xil_not()	Takes the one's complement of the image's values
xil_absolute()	Finds the absolute value of pixels in an image

Geometric Functions

Table 4 lists the XIL library's geometric functions. The functions that interpolate destination-image pixel values support the following types of interpolation: nearest neighbor, bilinear, bicubic, and general.

Table 4 Geometric Functions

Function Name	Description
`xil_translate()`	Moves an image up or down and/or left or right
`xil_scale()`	Increases or decreases the width and/or height of an image
`xil_subsample_adaptive()`	Reduces the size of an image using an algorithm that takes into consideration the value of each pixel in the source image
`xil_subsample_binary_to_gray()`	Scales down a 1-bit image, making up for a loss in spatial resolution with an increase in grayscale resolution
`xil_rotate()`	Rotates an image

Table 4 Geometric Functions

Function Name	Description
`xil_affine()`	Performs affine transforms, including scaling, rotation, translation, and shearing
`xil_transpose()`	Reflects an image in one of several directions, or rotates it in multiples of 90 degrees
`xil_tablewarp()`	Warps an image by displacing the *x* and *y* coordinates of its pixels

Miscellaneous Image Processing Functions

The library also contains several functions that do not readily fall into any grouping. These functions are shown in Table 5.

Table 5 Miscellaneous Image Processing Functions

Function Name	Description
`xil_extrema()`	Finds the minimum and maximum values in an image
`xil_histogram()`	Produces a histogram for an image
`xil_threshold()`	Thresholds an image
`xil_fill()`	Performs a boundary fill
`xil_soft_fill()`	Performs a soft fill
`xil_convolve()`	Filters an image
`xil_edge_detection()`	Detects an image's edges
`xil_dilate()`	Dilates an image
`xil_erode()`	Erodes an image
`xil_lookup()`	Passes an image through a lookup table
`xil_blend()`	Blends images
`xil_paint()`	Paints on an image
`xil_set_pixel()`	Sets the value of a single pixel
`xil_get_pixel()`	Reads the value of a single pixel

Table 5 Miscellaneous Image Processing Functions

Function Name	Description
`xil_set_value()`	Sets all the pixels in an image to a constant
`xil_copy_pattern()`	Copies a pattern to an image
`xil_band_combine`	Performs the arbitrary interband linear combination of an image using a specified matrix

Presentation Functions

Table 6 lists a group of functions you are likely to use in displaying or printing image.

Table 6 Presentation Functions

Function Name	Description
`xil_copy()`	Copies an image (for example, to a display or a printer)
`xil_copy_with_planemask()`	Copies the image planes defined by a plane mask
`xil_rescale()`	Maps the values in each band of an image from one range to another
`xil_cast()`	Casts an image from one data type to another
`xil_nearest_color()`	Dithers an image by finding the nearest match between colors in the source image and colors in a lookup table
`xil_error_diffusion()`	Performs error diffusion dithering
`xil_ordered_dither()`	Performs an ordered dither
`xil_color_convert()`	Converts an image from one color space to another
`xil_black_generation()`	Performs undercolor removal and black generation on CMYK images

Image Compression Functions

Table 7 lists the XIL functions you use to operate on compressed image sequences.

Table 7 Compressed Image Sequence Functions

Function Name	Description
`xil_compress()`	Compresses an XIL image and writes the compressed data to a compressed image sequence (CIS)
`xil_cis_put_bits()`	Copies compressed data into a CIS
`xil_cis_put_bits_ptr()`	Give a CIS a pointer to a compressed datastream
`xil_decompress()`	Decompresses a compressed image
`xil_cis_get_bits_ptr()`	Retrieves a pointer to a CIS's compressed data
`xil_cis_has_frame()`	Determines whether a CIS contains at least one frame of compressed data
`xil_cis_has_data()`	Returns the number of unread bytes in a CIS
`xil_cis_number_of_frames()`	Returns the number of unread frames in a CIS
`xil_cis_seek()`	Seeks to a particular frame of compressed data in a CIS
`xil_cis_reset()`	Clears all existing data from a CIS
`xil_cis_attempt_recovery()`	Attempts to recover from a datastream error

Error Handling Functions

The XIL library enables you to write a custom error handler that is called when XIL errors occur. Table 8 lists the functions you can use in writing this error handler.

Table 8 Error Handling Functions

Function Name	Description
`xil_error_get_string()`	Returns a string that describes the error that just occurred
`xil_error_get_id()`	Returns the error ID of the last error to occur
`xil_error_get_category()`	Returns a constant indicating the type of error that occurred. Possible error types include resource errors, arithmetic errors, and user errors, among others.
`xil_error_get_category_string()`	Returns a string indicating the type of error that occurred.
`xil_error_get_primary()`	Indicates whether an error is the primary cause of a problem you are seeing (or a secondary error)
`xil_error_get_location()`	Returns encrypted information about where in the XIL library an error was generated
`xil_error_get_object()`	Returns a handle to the XIL data object affected by the error

XIL Glossary

This section contains definitions for many of the terms used in this paper.

affine transform

Affine transforms include such operations as scaling, rotation, translation, and shearing. What these operations have in common is that they can change the lengths and angles of lines, but not their parallelism. The XIL library contains a function `xil_affine()` that in one operation can scale, rotate, and translate an image.

baseline sequential codec

A sequential encoder-decoder defined by the JPEG standard. It is designed to handle images with 8-bit samples and uses Huffman coding for its entropy coding.

bicubic interpolation

Bicubic interpolation is the most time consuming of the XIL library's interpolation methods, but produces the best results. When this type of interpolation is requested, the library calculates the value of a pixel in the destination image by determining the point in the source image to which that pixel maps and then examining the values of the sixteen pixels closest to that point.

bilinear interpolation

When the XIL library uses bilinear (or first-order) interpolation to determine the value of an image at noninteger coordinates, it calculates the value by looking at the values of the four pixels surrounding the point of interest and then using a bilinear equation. This type of interpolation yields better results than nearest neighbor interpolation, but can itself have an undesirable smoothing effect on an image. To alleviate this problem, you can use bicubic interpolation.

CCITT Group 3

A standard that specifies how facsimile machines must compress and decompress image data. The compression method relies heavily on run-length encoding. Runs of white pixels and runs of black pixels are represented with codes from a Huffman table.

CCITT Group 4

A standard for compressing document images. This compression technique takes advantage of a characteristic of document images called vertical coherence. This means that transitions from black to white or vice versa generally occur in almost the same place on adjacent scanlines. The Group 4 encoder compresses an image by recording information about the relative locations of these transitions.

Cell encoding

A video compression algorithm, which was developed by Sun. In Cell encoding, a 4-by-4 region of pixels is represented by two colors and a 16-bit mask that indicates which of the two colors to place at each of the 16 pixel positions. The colors and mask are chosen to preserve the mean and variance of the luminance and the average chrominance for the 4-by-4 block.

Cell decoding takes advantage of the fonting hardware commonly found in bitmapped displays.

CellB encoding

A Sun video-compression algorithm, derived from the Cell algorithm. Like a Cell encoder, a CellB encoder performs intraframe compression by coding 4-by-4 regions of pixels in four bytes. Two of the bytes are used to specify colors, and the other two represent a 16-bit mask that indicates which of the two colors to place at each of the 16 pixel positions. The CellB encoder achieves interframe encoding using skip codes. These codes allow the encoder to skip a series of one or more 4-by-4 cells in the current image if the cells match their counterparts in the preceding image in the sequence within a certain tolerance.

CellB encoding and decoding are intended primarily for use in video-conferencing applications.

CIS

Compressed image sequence. The XIL library's compressors store (generally related) compressed images in structures called CIS buffers. The images may represent frames in a movie, pages in a document, and so on.

convolution

An image-processing operation frequently used to sharpen an image, blur an image, or highlight the edges in an image. The operation calculates the values of pixels in the destination image using the values of a neighborhood of pixels in the source image and the values in a special filter called a convolution kernel.

convolution kernel

A two-dimensional array of weighted values used in a convolution operation. In the XIL library, a kernel is a data structure of data type `XilKernel`.

display image

The XIL library treats displays as special images, and operations allow display images to serve as destinations. This strategy enables an operation to draw on a screen directly, without requiring an intermediate copy. In some cases, display images can also serve as source images.

dithering

The XIL library includes a table-lookup function that enables you to convert a single-band image of any data type to a single-band or multiband image of any data type. Dithering can be thought of as an *inverse* lookup operation; that is, dithering enables you to convert a single-band or multiband image of any data type to a single-band image of any data type. The most common dithering operations convert 3-band, 8-bit images to 1-band, 8-bit images and 1-band, 8-bit images to 1-band, 1-bit images.

error diffusion

A technique for removing some of the artifacts produced during the dithering process. In its most common form, Floyd-Steinberg error diffusion, this technique involves (1) determining the amount of error produced in dithering a particular pixel and (2) distributing fractions of that error to the pixels to the right of and below the pixel just dithered.

frame

A single image in a movie.

geometric operation

An image-processing operation that changes the size, shape, or orientation of objects in an image. XIL geometric operations include scaling, rotation, and general affine transforms.

H.261

A video compression standard developed by the CCITT for use in encoding video to be transferred over an ISDN (Integrated Service Digital Network). In this compression method, data is compressed so that the output bit rate is p x 64 Kbits per second, where *p* can range from 1 to 30 depending on the number of ISDN channels used. This standard was developed primarily to support video phone and video-conferencing applications.

handle

An identifier that refers to a unique object. In the XIL library, an identifier of type `XilImage` is a handle to an actual image.

histogram

A collection of information about how frequently certain gray levels or color levels appear in an image. The XIL library contains a data structure of type `XilHistogram` to hold this type of information.

image

A two-dimensional array of pixels that represents an object.

interpolation

A way of calculating a value that falls between other, known values. In image processing, interpolation frequently plays a part in geometric operations such as rotation. After that type of spatial transformation, pixel locations in the output image will correspond to noninteger coordinates in the input image. Therefore, the pixel values in the output must be calculated by looking at the values of the pixels surrounding the point of interest in the input. The XIL library supports several types of interpolation, including nearest neighbor, bilinear, and bicubic interpolation.

JPEG

Joint Photographic Experts Group. A joint venture of the CCITT and ISO that has developed a standard for compressing grayscale or color *still* images. Actually, the standard defines a number of methods for compressing images. Several of these are lossy methods based on the Discrete Cosine Transform (DCT), but one method is lossless and is based on a predictive coding technique.

linear interband combination

An operation used in conversion between arbitrary color spaces. A matrix multiply is performed between the bands of a source image and a matrix, which yields a destination image.

lookup table

Lookup tables are used for general image modification. Each entry in an XIL lookup table contains an index—a value that may appear in the source image—and a value or set of values to be written to the destination image. For each pixel in the source, a table-lookup function finds the pixel's value on the index side of the table and then writes the output value or values for that entry to the corresponding pixel in the destination.

MPEG

Moving Picture Experts Group. This group has developed standards for compressing moving pictures and audio data and for synchronizing video and audio datastreams. The XIL library includes a decompression module that can decode MPEG video bitstreams.

The MPEG video-compression standard is similar to the H.261 standard developed by CCITT, but places less emphasis on low bit rates. By accepting a higher bit rate—up to 1.5 Mbits per second—an MPEG codec is able to recreate very high-quality pictures and to produce a bitstream that is easily editable.

The rate of 1.5 Mbits/s makes the MPEG codec especially viable in applications that read compressed data from CD-ROMs. For example, putting an MPEG bitstream on a CD-ROM is an effective way to distribute movies, business presentations, and training videos.

nearest neighbor interpolation

One of the methods that the XIL library uses to determine the value of an input image at noninteger coordinates. When using this method, the XIL library takes the value it is looking for to be the value of the pixel closest to the point of interest. This type of interpolation is the fastest type, but can introduce artifacts in the output image; for example, smooth lines in the input image may show up as jagged lines in the output.

Nearest neighbor interpolation is sometimes called zero-order interpolation.

rescaling

Sometimes called linear remapping, rescaling involves changing the values in an image using a linear equation. Each value in the image is multiplied by a constant; then a second constant is added to the product. Rescaling is useful for mapping the values in an image from one range to another.

ROI

A region of interest. In the XIL scheme, an ROI is a 1-bit mask that can be specified as an attribute of an image. This mask determines what part of a source image can be read or what part of a destination image can modified during an image-processing operation. The ROI for a given operation is the intersection of the ROIs of all the images involved in the operation.

structuring element

A two-dimensional array of Boolean values used as a parameter to the XIL dilation and erosion functions.

thresholding

An image-processing technique used to convert a single-band image to a bivalued image. Specifically, the operation sets all pixels in the input image that fall between a low threshold and a high threshold to 1 in the output image. All other pixels in the output image are set to 0. This technique is useful for creating images that can serve as regions of interest in later operations.

translation

A geometric operation in which an image is moved up or down and/or left or right.

transposition

A quasi-geometric transformation that involves rotating an image by a multiple of 90 degrees or flipping pixels in an image across a line that passes through the center of the image. The XIL library's transposition function allows for flips across a horizontal or vertical line passing through an image's center and across the main diagonal or antidiagonal.

XIL library

The XIL library is a foundation library for imaging and video support. It provides an implementation of imaging functionality that is common to multiple higher-level interfaces, provides imaging capabilities that are not currently available, and provides a way for ISVs to access low-level and hardware functionality.

Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
415 960-1300
FAX 415 969-9131

For U.S. Sales Office locations, call:
800 821-4643
In California:
800 821-4642

Australia: (02) 413 2666
Belgium: 32-2-759 5925
Canada: 416 477-6745
Finland: 358-0-502 27 00
France: (1) 30 67 50 00
Germany: (0) 89-46 00 8-0
Hong Kong: 852 802 4188
Italy: 039 60551
Japan: (03) 221-7021
Korea: 822-563-8700
Latin America: 415 688-9464
The Netherlands: 033 501234
New Zealand: (04) 499 2344
Nordic Countries: +46 (0) 8 623 90 00
PRC: 861-831-5568
Singapore: 224 3388
Spain: (91) 5551648
Switzerland: (1) 825 71 11
Taiwan: 2-514-0567
UK: 0276 20444

Elsewhere in the world,
call Corporate Headquarters:
415 960-1300
Intercontinental Sales: 415 688-9000