PIKS IMAGE PROCESSING SOFTWARE TUTORIAL

PIKS contains a rich set of software operators that perform manipulations of multidimensional images or of data objects extracted from images in order to enhance, restore or assist in the extraction of information from images. This document presents a functional overview of the PIKS standard and a more detailed definition of a functional subset of the standard called *PIKS Scientific*.

1.1. PIKS FUNCTIONAL OVERVIEW

This section provides a brief functional overview of PIKS. References 1 to 6 provide further information. The PIKS documentation utilizes British spelling conventions, which differ from American spelling conventions for some words (e.g., *colour* instead of *color*). For consistency with the PIKS standard, the British spelling convention has been adopted for this chapter.

1.1.1. PIKS Imaging Model

Figure 1.1-1 describes the PIKS imaging model. The solid lines indicate data flow, and the dashed lines indicate control flow. The PIKS application program interface consists of four major parts:

1. Data objects
2. Operators, tools and utilities
3. System mechanisms
4. Import and export

Abstacted from *Digital Image Processing, Fourth Edition*, by William K. Pratt
Copyright © 2013 by PixelSoft, Inc.

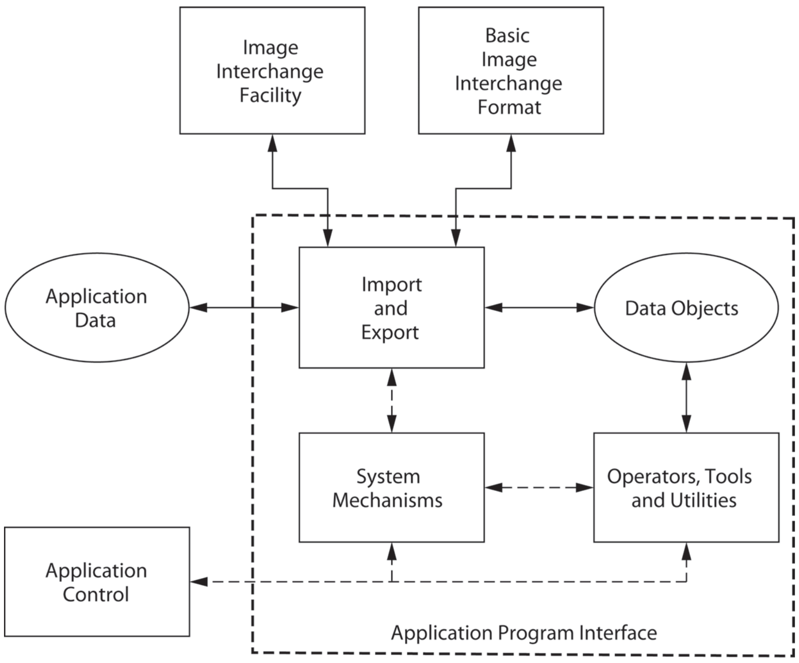
Image Interchange Facility
Basic Image Interchange Format
Application Data
Import and Export
Data Objects
System Mechanisms
Operators, Tools and Utilities
Application Control
Application Program Interface

FIGURE 1.1-1. PIKS imaging model.

The PIKS data objects include both image and image-related, non-image data objects. The operators, tools and utilities are functional elements that are used to process images or data objects extracted from images. The system mechanisms manage and control the processing. PIKS receives information from the application to invoke its system mechanisms, operators, tools and utilities, and returns certain status and error information to the application. The import and export facility provides the means of accepting images and image-related data objects from an application, and for returning processed images and image-related data objects to the application. PIKS can transmit its internal data objects to an external facility through the ISO/IEC standards Image Interchange Facility (IIF) or the Basic Image Interchange Format (BIIF). Also, PIKS can receive data objects in its internal format, which have been supplied by the IIF or the BIIF. References 7 to 9 provide information and specifications of the IIF and BIIF.

1.1.2. PIKS Data Objects

PIKS supports two types of data objects: image data objects and image-related, non-image data objects.

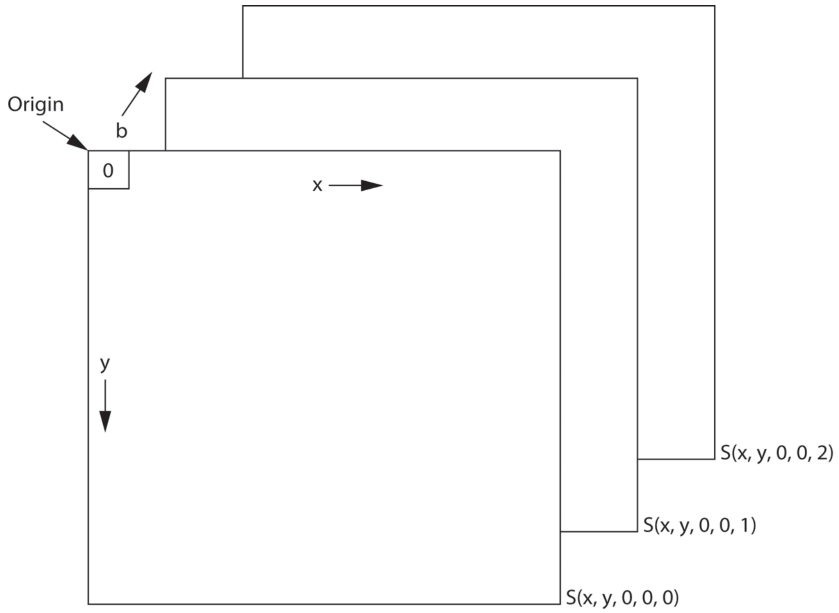

FIGURE 1.1-2. Geometrical representation of a PIKS color image array.

A PIKS image data object is a five-dimensional collection of pixels whose structure is:

x	Horizontal space index, $0 \le x \le X - 1$
y	Vertical space index, $0 \le y \le Y - 1$
z	Depth space index, $0 \le z \le Z - 1$
t	Temporal index, $0 \le t \le T - 1$
b	Color or spectral band index, $0 \le b \le B - 1$

Some of the image dimensions may be unpopulated. For example, as shown in Figure 1.1-2, for a color image, $Z = T = 1$. PIKS gives semantic meaning to certain dimensional subsets of the five-dimensional image object. These are listed in Table 1.1-1.

PIKS utilizes the following pixel data types:

1. Boolean
2. Non-negative integer
3. Signed integer
4. Real arithmetic
5. Complex arithmetic

TABLE 1.1-1. PIKS Image Objects

Semantic Description	Image Indices
Monochrome	$x, y, 0, 0, 0$
Volume	$x, y, z, 0, 0$
Temporal	$x, y, 0, t, 0$
Color	$x, y, 0, 0, b$
Spectral	$x, y, 0, 0, b$
Volume–temporal	$x, y, z, t, 0$
Volume–color	$x, y, z, 0, b$
Volume–spectral	$x, y, z, 0, b$
Temporal–color	$x, y, 0, t, b$
Temporal–spectral	$x, y, 0, t, b$
Volume–temporal–color	x, y, z, t, b
Volume–temporal–spectral	x, y, z, t, b
Generic	x, y, z, t, b

The precision and data storage format of pixel data is implementation dependent.

PIKS supports several image related, non-image data objects. These include:

1. *Chain*: an identifier of a sequence of operators
2. *Composite identifier*: an identifier of a structure of image arrays, lists and records
3. *Histogram*: a construction of the counts of pixels with some particular amplitude value
4. *Lookup table*: a structure that contains pairs of entries in which the first entry is an input value to be matched and the second is an output value
5. *Matrix*: a two-dimensional array of elements that is used in vector-space algebra operations
6. *Neighbourhood array*: a multi-dimensional moving window associated with each pixel of an image (e.g., a convolution impulse response function array)
7. *Pixel record*: a sequence of across-band pixel values
8. *Region-of-interest*: a general mechanism for pixel-by-pixel processing selection
9. *Static array*: an identifier of the same dimension as an image to which it is related (e.g., a Fourier filter transfer function)
10. *Tuple*: a collection of data values of the same elementary data type (e.g., image size 5-tuple)
11. *Value bounds collection*: a collection of pairs of elements in which the first element is a pixel coordinate and the second element is an image measurement (e.g., pixel amplitude)
12. *Virtual register*: an identifier of a storage location for values returned from elements in a chain

1.1.3. PIKS Operators, Tools, Utilities and Mechanisms

PIKS operators are elements that manipulate images or manipulate data objects extracted from images in order to enhance or restore images, or to assist in the extraction of information from images. Exhibit 1.1-1 is a list of PIKS operators categorized by functionality.

PIKS tools are elements that create data objects to be used by PIKS operators. Exhibit 1.1-2 presents a list of PIKS tools functionally classified. PIKS utilities are elements that perform basic mechanical image manipulation tasks. A classification of PIKS utilities is shown in Exhibit 1.1-3. This list contains several file access and display utilities. PIKS mechanisms are elements that perform control and management tasks. Exhibit 1.1-4 provides a functional listing of PIKS mechanisms. In Exhibits 1.1-1 to 1.1-4, the elements not in PIKS Scientific or the PixelSoft implementation of PIKS Scientific are identified by an asterisk. Non-standard elements are identified by a pound sign.

EXHIBIT 1.1-1. PIKS Operators Classification

Analysis: image-to-non-image operators that extract numerical information from an image

Accumulator
Difference measures
Extrema
Histogram, one-dimensional
Histogram, two-dimensional
Hough transform
Line profile
Moments
Value bounds

Classification: image-to-image operators that classify each pixel of a multispectral image into one of a specified number of classes based on the amplitudes of pixels across image bands

Classifier, Bayes
Classifier, nearest neighbour

Color: image-to-image operators that convert a color image from one color space to another

Color conversion, linear
Color conversion, nonlinear
Color conversion, subtractive
Color lookup, interpolated
Luminance generation

Complex image: image-to-image operators that perform basic manipulations of images in real and imaginary or magnitude and phase form

Complex composition
Complex conjugate
Complex decomposition
Complex magnitude

Correlation: image-to-non-image operators that compute a correlation array of a pair of images

Cross-correlation
Template match

Edge detection: image-to-image operators that detect the edge boundary of objects within an image

Edge detection, orthogonal gradient
Edge detection, second derivative
Edge detection, template gradient

Enhancement: image-to-image operators that improve the visual appearance of an image or that convert an image to a form better suited for analysis by a human or a machine

Adaptive histogram equalization
False color
Histogram modification
Outlier removal
Pseudocolour
Unsharp mask
Wallis statistical differencing

Ensemble: image-to-image operators that perform arithmetic, extremal and logical combinations of pixels

Alpha blend, constant
Alpha blend, variable
Dyadic, arithmetic
Dyadic, complex
Dyadic, logical
Dyadic, predicate
Split image
Z merge

Feature extraction: image-to-image operators that compute a set of image features at each pixel of an image

Label objects
Laws texture features
Window statistics

Filtering: image-to-image operators that perform neighbourhood combinations of pixels directly or by Fourier transform domain processing

Convolve, five-dimensional
Convolve, two-dimensional
Filtering, homomorphic
Filtering, linear
Filtering, median
Filtering, pseudomedian
Filtering, rank order

Geometric: image-to-image and ROI-to-ROI operators that perform geometric modifications

Cartesian to polar
Flip, spin, transpose
Polar to cartesian
Rescale
Resize
Rotate
Subsample
Translate
Warp, control point
Warp, lookup table
Warp, polynomial
Zoom

Histogram shape: non-image to non-image operators that generate shape measurements of a pixel amplitude histogram of an image

Histogram shape, one-dimensional
Histogram shape, two-dimensional

Morphological: image-to-image operators that perform morphological operations on boolean and grey scale images

Erosion or dilation, Boolean
Erosion or dilation, grey

Fill region
Hit or miss transformation
Morphic processor
Morphology
Neighbour count
Open and close

Pixel modification: image-to-image operators that modify an image by pixel drawing or painting

Draw pixels
Paint pixels

Point: image-to-image operators that perform point manipulation on a pixel-by-pixel basis

Bit shift
Complement
Error function scaling
Gamma correction
Histogram scaling
Level slice
Lookup
Lookup, interpolated
Monadic, arithmetic
Monadic, complex
Monadic, logical
Noise combination
Power law scaling
Rubber band scaling
Threshold
Unary, integer
Unary, real
Window-level

Presentation: image-to-image operators that prepare an image for display

Diffuse
Dither

Shape: image-to-non-image operators that label objects and perform measurements of the shape of objects within an image

Perimeter code generator
Shape metrics

Spatial moments, invariant
Spatial moments, scaled

Unitary transform: image-to-image operators that perform multi-dimensional forward and inverse unitary transforms of an image

Transform, cosine
Transform, Fourier
Transform, Hadamard
Transform, Hartley

3D Specific: image-to-image operators that perform manipulations of three-dimensional image data

Sequence average
Sequence Karhunen-Loeve transform
Sequence running measures
3D slice

EXHIBIT 1.1-2 PIKS Tools Classification

Image generation: tools that create test images

Image, bar chart
Image, constant
Image, Gaussian image
Image, grey scale image
Image, random number image

Impulse response function array generation: tools that create impulse response function neighbourhood array data objects

Impulse, boxcar
Impulse, derivative of Gaussian
Impulse, difference of Gaussians
Impulse, elliptical
Impulse, Gaussian
Impulse, Laplacian of Gaussian
Impulse, pyramid
Impulse, rectangular
Impulse, sinc function

Look-up table generation: tools that create entries of a look-up table data object

Array to LUT

Matrix generation: tools that create matrix data objects

Color conversion matrix

Region-of-interest generation: tools that create region-of-interest data objects from a mathematical description of the region-of-interest

ROI, coordinate
ROI, elliptical
ROI, polygon
ROI, rectangular

Static array generation: tools that create filter transfer function, power spectrum and windowing function static array data objects

Filter, Butterworth
Filter, Gaussian
Filter, inverse
Filter, matched
Filter, Wiener
Filter, zonal
Markov power spectrum
Windowing function

EXHIBIT 1.1-3. PIKS Utilities Classification

Display: utilities that perform image display functions

#Boolean display
#Close window
#Color display
#Event delay
#Monochrome delay
#Open titled window
#Open window
#Pseudocolor display

Export From PIKS: utilities that export image and non-image data objects from PIKS to an application or to the IIF or BIIF

Export histogram
Export image
Export LUT

Export matrix
Export neighbourhood array
Export ROI array
Export static array
Export tuple
Export value bounds
Get color pixel
Get pixel
Get pixel array
Get pixel record
#Output image file
*Output object
#Put file
Tiled image export

Import to PIKS: utilities that import image and non-image data objects to PIKS from an application or from the IIF or the BIIF

#Get file
Import histogram
Import image
Import LUT
Import matrix
Import neighbourhood array
Import ROI array
Import static array
Import tuple
Import value bounds
#Input image file
*Input object
#Input PhotoCD
Output object
Put color pixel
Put pixel
Put pixel array
Put pixel record
Tiled image import

Inquiry: utilities that return information to the application regarding PIKS data objects, status and implementation

*Inquire chain environment
*Inquire chain status
Inquire elements
Inquire image

Inquire index assignment
Inquire non-image object
Inquire PIKS implementation
Inquire PIKS status
Inquire repository
Inquire resampling

Internal: utilities that perform manipulation and conversion of PIKS internal image and non-image data objects

Constant predicate
Convert array to image
Convert image data type
Convert image to array
Convert image to ROI
Convert ROI to image
Copy window
Create tuple
Equal predicate
Extract pixel plane
Insert pixel plane

EXHIBITS 1.1-4 PIKS Mechanisms Classification

Chaining: mechanisms that manage execution of PIKS elements inserted in chains

*Chain abort
*Chain begin
*Chain delete
*Chain end
*Chain execute
*Chain reload

Composite identifier management: mechanisms that perform manipulation of image identifiers inserted in arrays, lists and records

*Composite identifier array equal
*Composite identifier array get
*Composite identifier array put
*Composite identifier list empty
*Composite identifier list equal
*Composite identifier list get
*Composite identifier list insert

*Composite identifier list remove
*Composite identifier record equal
*Composite identifier record get
*Composite identifier record put

Control: mechanisms that control the basic operational functionality of PIKS

*Abort asynchronous execution
Close PIKS
Close PIKS, emergency
Open PIKS
*Synchronize

Error: mechanisms that provide means of reporting operational errors

Error handler
Error logger
Error test

System management: mechanisms that allocate, deallocate, bind and set attributes of data objects and set global variables

*Allocate chain
*Allocate composite identifier array
*Allocate composite identifier list
*Allocate composite identifier record
#Allocate display image
Allocate histogram
Allocate image
Allocate lookup table
Allocate matrix
Allocate neighbourhood array
Allocate pixel record
Allocate ROI
Allocate static array
Allocate tuple
Allocate value bounds collection
*Allocate virtual register
Bind match point
Bind ROI
Deallocate data object
Define sub image
Return repository identifier
Set globals

Set image attributes
Set index assignment

Virtual register: mechanisms that manage the use of virtual registers

*Vreg alter
*Vreg clear
*Vreg conditional
*Vreg copy
*Vreg create
*Vreg delete
*Vreg get
*Vreg set
*Vreg wait

EXHIBITS 1.1-5 PIKS Convenience Functions Classification

Image preparation functions:

Create unbound image
Prepare color image
Prepare monochrome image

ROI creation functions:

Generate 2D rectangular ROI
Generate coordinate ROI
Generate elliptical ROI
Generate polygon ROI
Generate rectangular ROI
Prepare 2D rectangular ROI
Prepare ROI

Tuple generation functions:

Generate ND 1 tuple
Generate ND 3 tuple
Generate ND 4 tuple
Generate ND 5 tuple
Generate RD 3 tuple
Generate RD 4 tuple
Generate RD 5 tuple
Generate SD 1 tuple
Generate SD 3 tuple
Generate SD 4 tuple
Generate SD 5 tuple

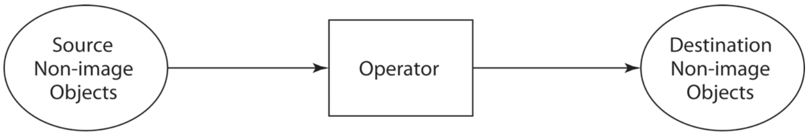

FIGURE 1.1-3. PIKS operator model: non-image to non-image operators.

1.1.4. PIKS Operator Model

The PIKS operator model provides three possible transformations of PIKS data objects by a PIKS operator:

1. Non-image to non-image
2. Image to non-image
3. Image to image

Figure 1.1-3 shows the PIKS operator model for the transformation of non-image data objects to produce destination non-image data objects. An example of such a transformation is the generation of shape features from an image histogram. Another example is the translation of a ROI to produce another ROI.

The operator model for the transformation of image data objects by an operator to produce non-image data objects is shown in Figure 1.1-4. An example of such a transformation is the computation of the least-squares error between a pair of images. In this operator model, processing is subject to two control mechanisms: region-of-interest (ROI) source selection and source match point translation. These

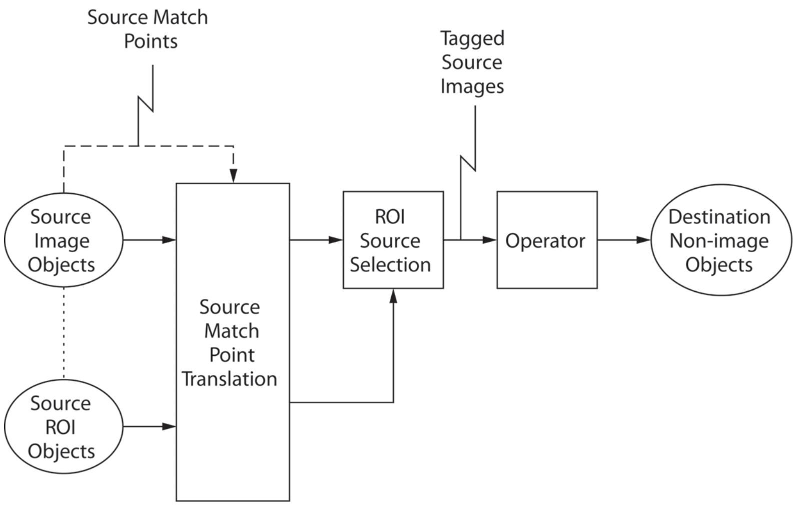

FIGURE 1.1-4. PIKS operator model: image to non-image operators.

control mechanisms are defined later. The dashed line in Figure 1.1-4 indicates the transfer of control information. The dotted line indicates the binding of source ROI objects to source image objects.

Figure 1.1-5 shows the PIKS operator model for the transformation of image data objects by an operator to produce other image data objects. An example of such an operator is the unsharp masking operator, which enhances detail within an image. In this operator model, processing is subject to four control mechanisms: source match point translation, destination match point translation, ROI source selection and ROI destination selection.

Certain PIKS operators are capable of accepting source images with pixels of non-negative or signed integer data type, and automatically promoting the source image to the real arithmetic data type.

Some PIKS operators, e.g. two-dimensional convolution and dilation, perform linear or nonlinear combinations of source image pixels within a neighbourhood of a reference pixel to produce a single destination image pixel. For such operators, the neighbourhood is specified by a neighbourhood array non-image data object. Each neighbourhood array has an associated key pixel, which is, in general, a five-dimensional coordinate offset measured with respect to the origin of the neighbourhood array. In operation, the key pixel is sequentially translated over all source image pixels, the linear or nonlinear neighbourhood combination is computed for each reference source pixel, and the computed result is recorded at the destination pixel corresponding to the reference source pixel that lies “under” the key pixel. In general, if the neighbourhood extends beyond the boundary of the source image,

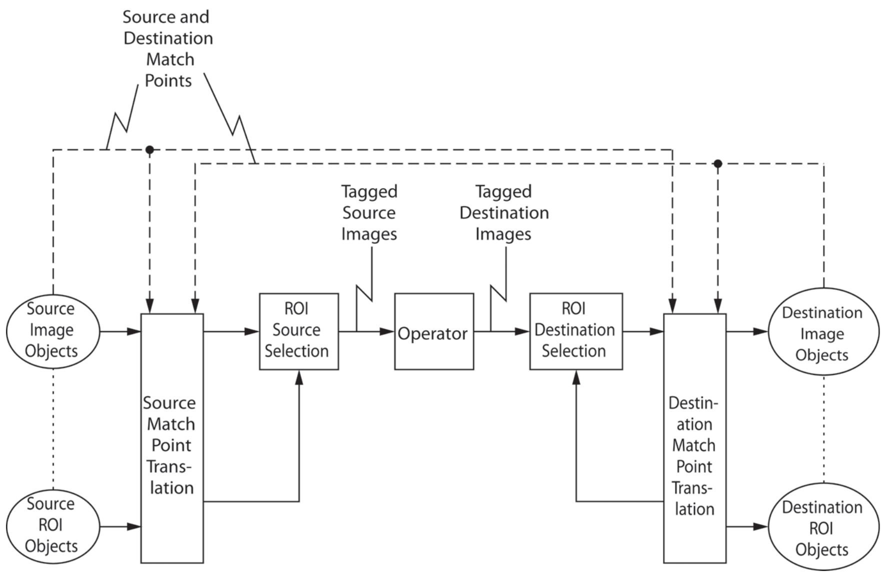
Source and Destination Match Points
Tagged Source Images
Tagged Destination Images
Source Image Objects
Source ROI Objects
Source Match Point Trans-lation
ROI Source Selection
Operator
ROI Destination Selection
Destin-ation Match Point Trans-lation
Destination Image Objects
Destination ROI Objects

FIGURE 1.1-5. PIKS operator model: image to image operators.

the “missing” source pixel values are assumed to be of zero value for arithmetic combinations or of FALSE value for Boolean combinations. Some operators specify the boundary condition differently.

Index Assignment. Some PIKS image to non-image and image to image operators have the capability of assigning operator indices to image indices. This capability permits operators that are inherently *N*th order, where $N < 5$, to be applied to five-dimensional images in a flexible manner. For example, a two-dimensional Fourier transform can be taken of each column slice of a volumetric image using index assignment. Figure 1.1-6 illustrates the PIKS image to non-image and image to image operator index assignment and reassignment process. There is a global index assignment 5-tuple, which relates the ordering of the five image indices (*x, y, z, t, b*) to five generic operator indices (*j, k, l, m, n*). Prior to the execution of an operator, each source image *SRCp*(*x, y, z, t, b*) is logically converted to an operator input image *Sp*(*j, k, l, m, n*) according to a global operator assignment specification table, generated by the *index_assignment* system management mechanism.

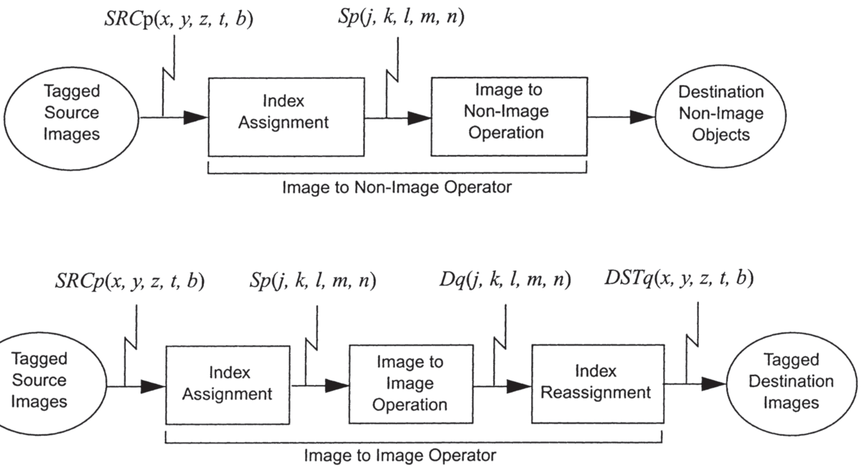

FIGURE 1.1-6. Operator index assignment.

ROI Control. A region-of-interest (ROI) data object can be used to control which pixels within a source image will be processed by an operator and to specify which pixels processed by an operator will be recorded in a destination image. Conceptually, a ROI consists of an array of Boolean value pixels of up to five dimensions. Figure 1.1-7 presents an example of a two-dimensional rectangular ROI. In this example, if the pixels in the cross-hatched region are logically TRUE, the remaining pixels are logically FALSE. Otherwise, if the cross-hatched pixels are set FALSE, the others are TRUE.

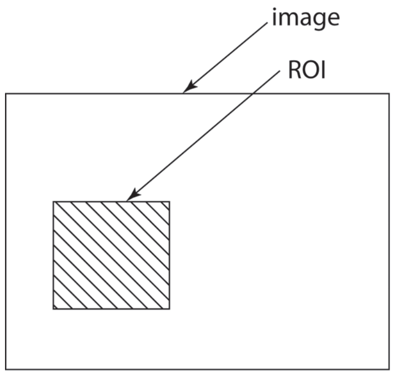

FIGURE 1.1-7. Rectangular ROI bound to an image array.

The size of a ROI need not be the same as the size of an image to which it is associated. When a ROI is to be associated with an image, a binding process occurs in which a ROI control object is generated. If the ROI data object is larger in spatial extent than the image to which it is to be bound, it is clipped to the image size to form the ROI control object. In the opposite case, if the ROI data object is smaller than the image, the ROI control object is set to the FALSE state in the non-overlap region.

Figure 1.1-8 illustrates three cases of ROI functionality for point processing of a monochrome image. In case 1, the destination ROI control object is logically TRUE over the full image extent, and the source ROI control object is TRUE over a cross-hatched rectangular region smaller than the full image. In this case, the destination image consists of the existing destination image with an insert of processed source pixels. For case 2, the source ROI is of full extent, and the destination ROI is of a smaller cross-hatched rectangular extent. The resultant destination image consists of processed pixels inserted into the existing destination image. Functionally, the result is the same as for case 1. The third case shows the destination image when the source and destination ROIs are overlapping rectangles smaller than the image extent. In this case, the processed pixels are recorded only in the overlap area of the source and destination ROIs.

The ROI concept applies to multiple destination images. Each destination image has a separately bound ROI control object, which independently controls recording of pixels in the corresponding destination image. The ROI concept also applies to neighbourhood as well as point operators. Each neighbourhood processing element, such as an impulse response array, has a pre-defined key pixel. If the key pixel lies within a source control ROI, the output pixel is formed by the neighbourhood operator even if any or all neighbourhood elements lie outside the ROI.

PIKS provides tools for generating ROI data objects from higher level specifications. Such supported specifications include:

1. Coordinate list
2. Ellipse
3. Polygon
4. Rectangle

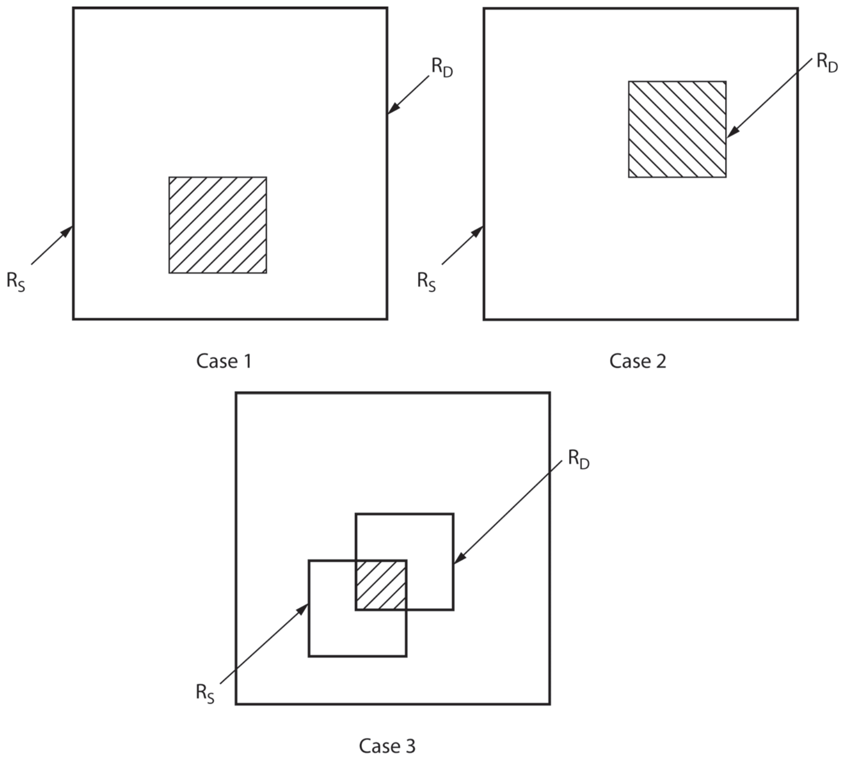

FIGURE 1.1-8. ROI operation.

These tools, together with the ROI binding tool, provide the capability to conceptually generate five-dimensional ROI control objects from lower dimensional descriptions by pixel plane extensions. For example, with the elliptical ROI generation tool, it is possible to generate a circular disk ROI in a spatial pixel plane, and then cause the disk to be replicated over the other pixel planes of a volumetric image to obtain a cylinder-shaped ROI.

A ROI data object is expressed, notationally, as a five-dimensional array $ROI(x, y, z, t, b)$. Associated with each image is a key point 5-tuple $(x_k, y_k, z_k, t_k, b_k)$ consisting of five signed integers, which defines an offset of the ROI origin with respect to the image origin. When a ROI data object is bound to an image, conceptually, the ROI data object is translated in each orthogonal direction such that the origin of the ROI data object is aligned in five-dimensional space with the key point in the image to which the ROI data object is bound. The directional sense (left/right, up/down, etc.) is the same as an image translated by match point alignment. The ROI control object is conceptually formed by clipping the ROI data object to the extent of the image. If the ROI data object is not in the TRUE state at the geometric centre of a particular pixel of the image, the ROI control object is set FALSE at that pixel position. Figure 1.1-9 shows an example of the spatial relationship between a ROI data object and an image to which it is bound.

ROI control can be globally enabled or disabled by the *set_globals* mechanism.

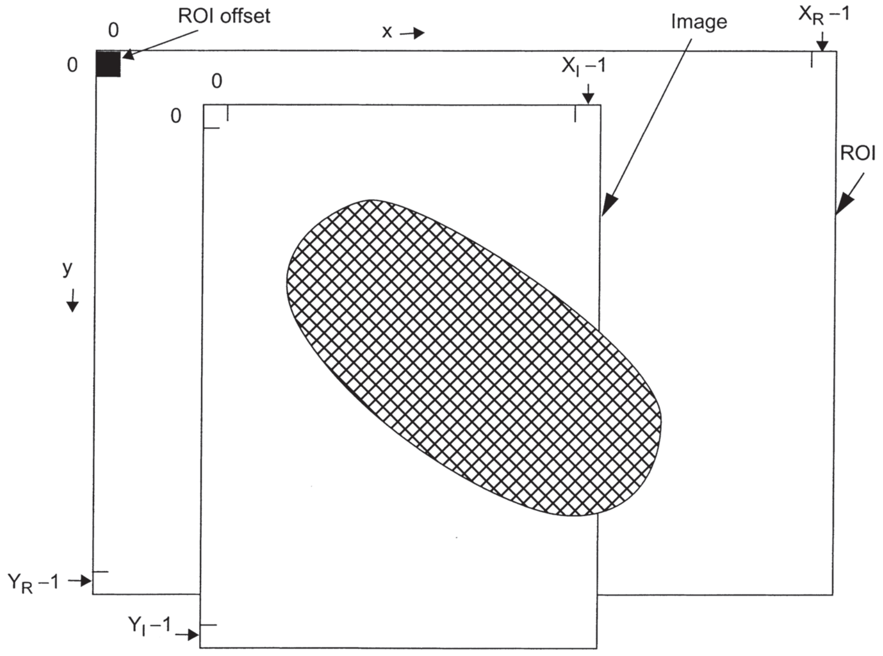

FIGURE 1.1-9. Example of the relationship between a ROI and an image.

Match Point Control. Each PIKS image object has an associated match point coordinate set (x, y, z, t, b), which some PIKS operators utilize to control multi-dimensional translations of images prior to processing by an operator. The generic effect of match point control for an operator that creates multiple destination images from multiple source images is to translate each source image and each destination image, other than the first source image, such that the match points of these images are aligned with the match point of the first source image prior to processing. Processing then occurs on the spatial intersection of all images. Figure 1.1-10 is an example of image subtraction subject to match point control. In the example, the difference image is shown cross-hatched.

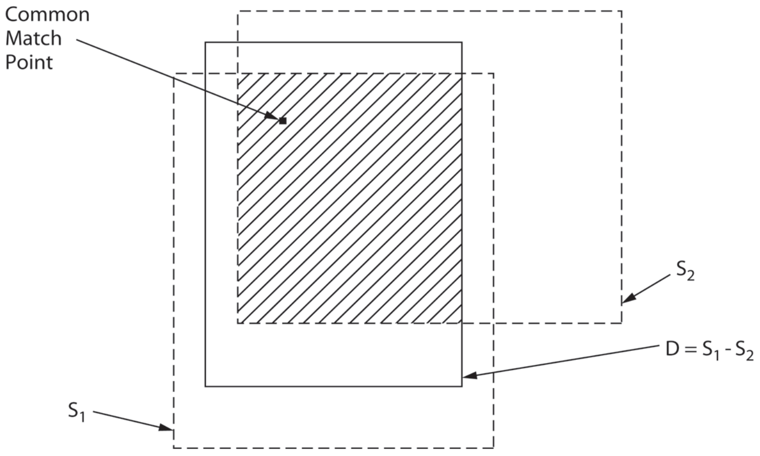

FIGURE 1.1-10. Match point translation for image subtraction.

Other Features. PIKS provides a number of other features to control processing. These include:

1. Processing of ROI objects in concert with image objects
2. Global setting of image and ROI resampling options
3. Global engagement of ROI control and ROI processing
4. Global engagement of index assignment
5. Global engagement of match point control
6. Global engagement of synchronous or asynchronous operation
7. Automatic source promotion, e.g. ND to RD or SD to RD if the destination is RD
8. Heterogeneous bands of dissimilar data types
9. Element chaining
10. Virtual registers to store intermediate results of an operator chain
11. Composite image management of image and non-image objects

The PIKS Functional Specification (2) provides rigorous specifications of these features. PIKS also contains a data object repository of commonly used impulse response arrays, dither arrays and color conversion matrices.

1.1.5. PIKS Application Interface

Figure 1.1-11 describes the PIKS application interface for data interchange for an implementation-specific data pathway. PIKS supports a limited number of physical data types that may exist within an application domain or within the PIKS domain. Such data types represent both input and output parameters of PIKS elements and image and non-image data that are interchanged between PIKS and the application.

PIKS provides notational differentiation between most of the elementary abstract data types used entirely within the PIKS domain (PIKS internal), those that are used to convey parameter data between PIKS and the application (PIKS parameter), and those that are used to convey pixel data between PIKS and the application (external physical image). Table 1.1-2 lists the codes for the PIKS abstract data types. The abstract data types are defined in ISO/IEC 1187-1. PIKS internal and parameter data types are of the same class if they refer to the same basic data type. For example, RP and RD data types are of the same class, but RP and SD data types are of different classes. The external physical data types supported by PIKS for the import and export of image data are also listed in Table 1.1-2. PIKS internal pixel data types and external pixel data types are of the same class if they refer to the same basic data type. For example, ND and NI data types are of the same class, but SI and ND data types are of different classes.

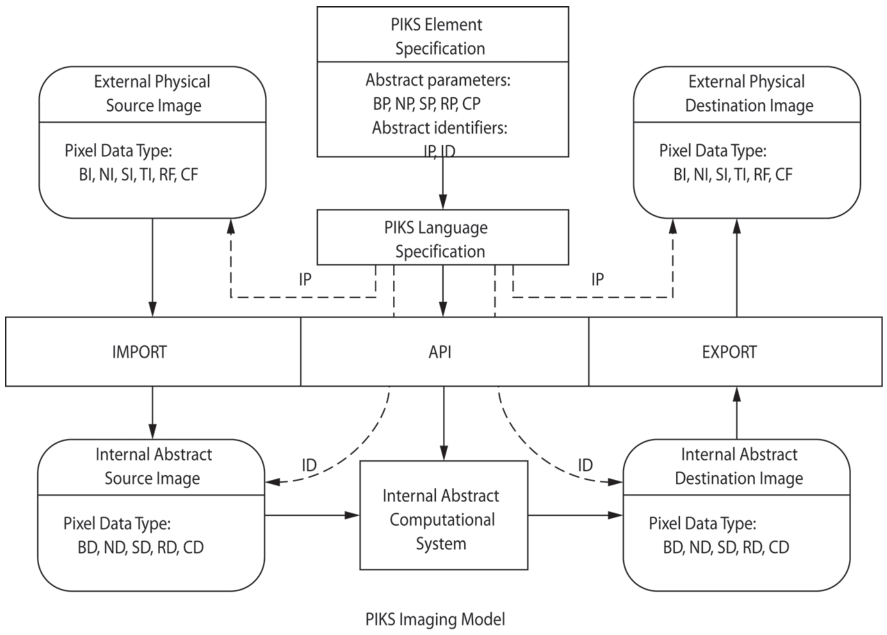
PIKS Element Specification
Abstract parameters: BP, NP, SP, RP, CP
Abstract identifiers: IP, ID
External Physical Source Image
Pixel Data Type: BI, NI, SI, TI, RF, CF
External Physical Destination Image
Pixel Data Type: BI, NI, SI, TI, RF, CF
PIKS Language Specification
IP
IP
IMPORT
API
EXPORT
ID
ID
Internal Abstract Source Image
Pixel Data Type: BD, ND, SD, RD, CD
Internal Abstract Computational System
Internal Abstract Destination Image
Pixel Data Type: BD, ND, SD, RD, CD
PIKS Imaging Model

FIGURE 1.1-11. PIKS application interface.

TABLE 1.1-2. PIKS Datatype Codes

Data Type	PIKS Internal Code	PIKS Parameter Code	Physical Code
Boolean	BD	BP	BI
Non-negative integer	ND	NP	NI
Signed integer	SD	SP	SI
Fixed-point integer	—	—	TI
Real arithmetic	RD	RP	RF
Complex arithmetic	CD	CP	CF
Character string	CS	CS	—
Data object identifier	ID	IP	—
Enumerated	NA	EP	—
Null	NULL	NULL	—

1.1.6. PIKS Conformance Profiles

Because image processing requirements vary considerably across various applications, PIKS functionality has been subdivided into the following five nested sets of functionality called *conformance profiles*:

1. *PIKS Foundation*: basic image processing functionality for monochrome and color images whose pixels are represented as Boolean values or as non-negative or signed integers.
2. *PIKS Core*: intermediate image processing functionality for monochrome and color images whose pixels are represented as Boolean values, non-negative or signed integers, real arithmetic values and complex arithmetic values. PIKS Core is a superset of PIKS Foundation.
3. *PIKS Technical*: expanded image processing functionality for monochrome, color, volume, temporal and spectral images for all pixel data types. PIKS Technical is a superset of PIKS Core.
4. *PIKS Scientific*: complete set of image processing functionality for all image structures and pixel data types. PIKS Scientific is a superset of PIKS Technical functionality.
5. *PIKS Full*: complete set of image processing functionality for all image structures and pixel data types plus the capability to chain together PIKS processing elements and to operate asynchronously. PIKS Full is a superset of PIKS Scientific functionality.

Each PIKS profile may include the capability to interface with the IIF and the BIIF.

Reference 5 is a C programmer's guide for the Foundation profile. Reference 10 describes the functional specification of the PIKS Core profile. This book also contains a compact disk with an implementation of PIKS Scientific and an associated C language programmer's manual.

1.2. PIKS SCIENTIFIC OVERVIEW

The PIKS Scientific profile provides a comprehensive set of image processing functionality to service virtually all image processing applications. It supports all pixel data types and the full five-dimensional PIKS image data object. It provides the following processing features:

1. Index assignment
2. Match point control
3. Support 1, 2, 4 and 8 global resampling interpolation of images and ROIs
4. ROI control for all ROIs
5. ROI processing
6. Heterogeneous bands
7. Composite images
8. Asynchronous processing
9. Automatic source promotion
10. Data object repository

At the time of publication of this book, the PixelSoft implementation of PIKS Scientific is not fully compliant with the standard. The PixelSoft implementation does not support heterogeneous bands, composite images or asynchronous processing. Furthermore, the PixelSoft implementation is limited to nearest neighbour, bilinear and cubic B-spline interpolation in all image dimensions for match point resampling. For geometric operator resampling, the PixelSoft implementation is limited to nearest neighbour, bilinear and bicubic interpolation for the first two operator indices (usually x and y space dimensions) and nearest neighbour for the other three dimensions (usually z for depth, t for temporal and b for band).

The following sections provide details of the data structures for PIKS Scientific non-image and image data objects.

1.2.1. PIKS Scientific Non-image Data Objects

PIKS Scientific supports the non-image data objects listed below. The list contains the PIKS Functional Specific object name code and the definition of each object.

HIST	Histogram
LUT	Lookup table
MATRIX	Matrix

NBHOOD_ARRAY	Neighbourhood array
PIXEL_RECORD	Pixel record
ROI	Region-of-interest
STATIC_ARRAY	Static array
TUPLE	Tuple
VALUE_BOUNDS	Value bounds collection

The tuple object is defined first because it is used to define other non-image and image data objects. Tuples are also widely used in PIKS to specify operator and tool parameters (e.g. the size of a magnified image). Figure 1.2-1 contains the tree structure of a tuple object. It consists of the tuple size, tuple data type and a private identifier to the tuple data values. The tuple size is an unsigned integer that specifies the number of tuple data values. The tuple datatype option is a signed integer from 1 to 6 that specifies one of the six options. The identifier to the tuple data array is private in the sense that it is not available to an application; only the tuple data object itself has a public identifier.

```
Tuple Object
        Tuple data size
                number of tuple data values, e.g. 5
        Tuple data type option
                choice of BD, ND, SD, RD, CD or CS
        Tuple data array
                private identifier
```

FIGURE 1.2-1. Tuple object tree structure.

A PIKS histogram data object is a one-dimensional array of unsigned integers that stores the histogram of an image plus histogram object attributes. Figure 1.2-2 shows the tree structure of a histogram data object. The histogram array size is an unsigned integer that specifies the number of histogram bins. The lower and upper amplitude values are real numbers that specify the pixel amplitude range of the histogram.

```
Histogram Object
        Histogram array size
                number of histogram bins, e.g. 512
        Lower amplitude value
                lower amplitude value of histogram range, e.g. 0.1
        Upper amplitude value
                upper amplitude value of histogram range, e.g. 0.9
        Histogram data array
                private identifier
```

FIGURE 1.2-2. Histogram object tree structure.

A PIKS lookup table data object, as shown in Figure 1.2-3, is a two-dimensional array that stores the lookup table data plus a collection of lookup table attributes. The two-dimensional array has the general form following:

$$\begin{array}{ccccc} T(0,0) & \cdots & T(b,0) & \cdots & T(B-1,0) \\ \cdot & & \cdot & & \cdot \\ T(0,e) & \cdots & T(b,e) & \cdots & T(B-1,e) \\ \cdot & & \cdot & & \cdot \\ T(0,E-1) & \cdots & T(b,E-1) & \cdots & T(B-1,E-1) \end{array}$$

A special, but common case, occurs when the lookup table is one-dimensional and $B = 1$.

A positive integer e is the input row index to the table. It is derived from a source image by the relationship

$$e = S(x, y, z, t, b) \tag{1.2-1}$$

The LUT output is a one-dimensional array

$$a(e) = [T(0, e) \cdots T(b, e) \cdots T(B-1, e)] \tag{1.2-2}$$

- Lookup Table Object
  - Table entries
    - number of table entries, e.g. 512
  - Table bands
    - number of table bands, e.g. 3
  - Table input data type option
    - choice of ND or SD
  - Table output data type option
    - choice of BD, ND, SD, RD or CD
  - Lookup table data array
    - private identifier

FIGURE 1.2-3. Lookup table object tree structure.

There are two types of lookup table usage: (1) the source and destination images are of the same band dimension, or (2) the source image is monochrome and the destination image is color. In the former case,

$$D(x, y, 0, 0, b) = T(0, S(x, y, z, t, b)) \tag{1.2-3}$$

In the latter case,

$$D(x, y, 0, 0, b) = T(b, S(x, y, z, t, 0)) \tag{1.2-4}$$

Figure 1.2-4 shows the tree structure of a matrix data object. The matrix is specified by its number of rows R and columns C and the data type of its constituent terms. The matrix is addressed as follows:

$$\mathbf{M} = \begin{bmatrix} M(1,1) & \cdots & M(1,c) & \cdots & M(1,C) \\ \vdots & & \vdots & & \vdots \\ M(r,1) & \cdots & M(r,c) & \cdots & M(r,C) \\ \vdots & & \vdots & & \vdots \\ M(R,1) & \cdots & M(R,c) & \cdots & M(R,C) \end{bmatrix} \tag{1.2-5}$$

In PIKS, matrices are used primarily for color space conversion.

```
Matrix Object
        Column size
                number of matrix columns, e.g. 4
        Row size
                number of matrix rows, e.g. 3
        Matrix data type option
                choice of ND, SD or RD
        Matrix data array
                private identifier
```

FIGURE 1.2-4. Matrix object tree structure.

A PIKS Scientific neighbourhood array is a two-dimensional array and associated attributes as shown in Figure 1.2-5. The array has J columns and K rows. As shown below, it is indexed in the same manner as a two-dimensional image.

$$H(j,k) = \frac{1}{S}\begin{bmatrix} H(0,0) & \dots & H(j,0) & \dots & H(J-1,0) \\ \vdots & & \vdots & & \vdots \\ H(0,k) & \dots & H(j,k) & \dots & H(J-1,k) \\ \vdots & & \vdots & & \vdots \\ H(0,K-1) & \dots & H(j,K-1) & \dots & H(J-1,K-1) \end{bmatrix} \tag{1.2-6}$$

In Eq. 1.2-6, the scale factor S is unity except for the signed integer data type. For signed integers, the scale factor can be used to realize fractional elements. The key pixel (j_K, k_K) defines the origin of the neighbourhood array. It need not be within the confines of the array. There are five types of neighbourhood arrays, specified by the following structure codes:

GL	Generic array
DL	Dither array
IL	Impulse response array
ML	Mask array
SL	Structuring element array

- Neighbourhood Array Object
  - Neighbourhood size
    - specification of J, K
  - Key pixel
    - specification of j_k, k_k
  - Scale factor
    - integer value
  - Semantic label option
    - choice of GL, DL, IL, ML or SL
  - Neighbourhood data type option
    - choice of BD, ND, SD or RD
  - Neighbourhood data array
    - private identifier

FIGURE 1.2-5. Neighbourhood object tree structure.

Figure 1.2-6 is the tree structure of a pixel record data object. In general, it specifies the band length and the data type of each band.

- Pixel Record Object
  - Record length
    - number of bands, e.g. 3
  - Band data types
    - specification of data types
  - Pixel record data array
    - private identifier

FIGURE 1.2-6. Pixel record object tree structure.

Figure 1.2-7 shows the tree structure of a region-of-interest ROI data object. Conceptually, a PIKS Scientific ROI data object is a five-dimensional array of Boolean value pixels. The actual storage method is implementation dependent.The ROI can be defined to be TRUE or FALSE in its interior.

The ROI can be constructed by one of the following representations:

AR	ROI array
CR	ROI coordinate list
ER	ROI elliptical
GR	ROI generic
PR	ROI polygon
RR	ROI rectangular

- Region-of-Interest Object
  - ROI virtual array size
    - specification of X_R, Y_R, Z_R, T_R, B_R
  - ROI structure option
    - choice of AR, CR, ER, GR, PR or RR
  - Polarity option
    - choice of TRUE or FALSE
  - Conceptual ROI data array
    - private identifier

FIGURE 1.2-7. Region-of-interest object tree structure.

A PIKS Scientific static array is a five-dimensional array as shown in Figure 1.2-8. Following is a list of the types of static arrays supported by PIKS:

GS	Generic static array
HS	Histogram
PS	Power spectrum
TS	Transfer function
WS	Windowing function
XS	Transform

- Static Array Object
  - Static array size
    - specification of X_S, Y_S, Z_S, T_S, B_S
  - Semantic label option
    - choice of GS, HS, PS, TS, WS or XS
  - Data type option
    - choice of BD, ND, SD, RD or CD
  - Static array data array
    - private identifier

FIGURE 1.2-8. Static array object tree structure.

A value bounds collection is a storage mechanism containing the pixel coordinate and pixel values of all pixels whose amplitudes lie within a lower and an upper bound. Figure 1.2-9 is the tree structure of the value bounds collection data object.

1.2.2. PIKS Scientific Image Data Object

A PIKS image object is a tree structure of image attributes, processing control attributes and private identifiers to an image data array of pixels and an associated ROI. Figure 1.2-10 illustrates the tree structure of an image object. The image attributes are created when an image object is allocated. When an image is allocated, there will be no private identifier to the image array data. The private identifier is established automatically when raw image data are imported to a PIKS image object or when a destination image is created by an operator. The processing control attributes are created when a ROI is bound to an image. It should be noted that for the PixelSoft implementation of PIKS Scientific, all bands must be of the same datatype and pixel precision. The pixel precision specification of the PixelSoft implementation of PIKS Scientific is in accordance with the PIKS standard.

- Value Bounds Collection Object
  - Collection size
    - number of collection members
  - Lower amplitude bound
    - value of lower amplitude bound
  - Upper amplitude bound
    - value of upper amplitude bound
  - Pixel data type option
    - choice of ND, SD or RD
  - Value bounds collection data array
    - private identifier

FIGURE 1.2-9. Value bounds collection object tree structure.

1.2.3. PIKS Scientific C Language Binding

The PIKS Functional Specification document (2) establishes the semantic usage of PIKS. The PIKS C language binding document (11) defines the PIKS syntactical usage for the C programming language. At present, there are no other language bindings. Reader familiarity with the C programming language is assumed.

The PIKS C binding has adopted the *Hungarian prototype* naming convention, in which the data types of all entities are specified by prefix codes. Table 1.2-1 lists the datatype prefix codes. The entities in courier font are binding names. Table 1.2-2 gives the relationship between the PIKS Core C binding designators and the PIKS Functional Specification datatypes and data objects.

TABLE 1.2-1. PIKS Datatype Prefix Codes

Prefix	Definition
a	Array
b	Boolean
c	Character
d	Internal data type
e	Enumerated data type
f	Function
i	Integer
m	External image data type
n	Identifier
p	Parameter type
r	Real
s	Structure
t	Pointer
u	Unsigned integer
v	Void
x	Complex
z	Zero terminated string

TABLE 1.2-2. PIKS Core C Binding Designators and Functional Specification Datatypes and Data Objects

Binding	Functional Specification	Description
Imbool	BI	External Boolean datatype
Imuint	NI	External non-negative integer datatype
Imint	SI	External signed integer datatype
Imfixed	TI	External fixed point integer datatype
Imfloat	RF	External floating point datatype
Ipbool	BP	Parameter Boolean datatype
Ipuint	NP	Parameter non-negative integer datatype
Ipint	SP	Parameter signed integer datatype
Ipfloat	RP	Parameter real arithmetic datatype
Idnrepository	IP	External repository identifier
Ipnerror	IP	External error file identifier
Ipsparameter_basic	IP	External tuple data array pointer union
Ipsparameter_numeric	IP	External matrix data array pointer union
Ipsparameter_pixel	IP	External LUT, neighbourhood, pixel data array pointer union
Ipspiks_pixel_types	IP	External image data array pointer union

- Image Object
    - Image attributes
        - Representation
            - Size
                - specification of *X, Y, Z, T, B*
            - Band data type
                - specification of BD, ND, SD, RD or CD
            - Image structure option
                - 13 choices, e.g. MON, VOL, TEM or COLR
        - Channel
            - Band precision
                - specification of band pixel precision
        - Color
            - White point
                - specification of X_0, Y_0, Z_0
            - Color space option
                - 29 choices, e.g. CIE L*a*b* or CMYK
        - Control
            - Match point
                - specification of x_m, y_m, z_m, t_m, b_m
            - ROI
                - private identifier
            - ROI offset
                - specification of x_0, y_0, z_0, t_0, b_0
        - Image data array
            - private identifier

FIGURE 1.2-10. Image object tree structure.

The general structure of the C language binding element prototype is:

```
void IvElementName
```

or

```
I(prefix)ReturnName I(prefix)ElementName
```

As an example, the following is the element C binding prototype for two-dimensional convolution of a source image into a destination image:

```
Idnimage InConvolve2D(    /*  OUT destination image identifier   */
 Idnimage  nSourceImage, /*  source image identifier            */
 Idnimage  nDestImage,   /*  destination image identifier       */
 Idnnbhood nImpulse,     /*  impulse response array identifier */
 Ipint     iOption       /*  convolution 2D option              */
 );
```

In this example, the first two components of the prototype are the identifiers to the source and destination images. Next is the identifier to the impulse response neighbourhood array. The last component is the integer option parameter for the convolution boundary option. The following `#define` convolution options are provided in the `piks.h` header file:

```
ICONVOLVE_UPPER_LEFT    1  /* upper left corner justified   */
ICONVOLVE_ENCLOSED      2  /* enclosed array                */
ICONVOLVE_KEY_ZERO      3  /* key pixel, zero exterior      */
ICONVOLVE_KEY_REFLECTED 4  /* key pixel, reflected exterior */
```

As an example, let `nSrc` and `nDst` be the identifier names assigned to source and destination images, respectively, and let `nImpulse` be the identifier of an impulse response array. In an application program, the two-dimensional convolution operator can be invoked as

```
InConvolve2D(nSrc, nDst, nImpulse, ICONVOLVE_ENCLOSED);
```

or by

```
nDummy = InConvolve2D(nSrc, nDst, nImpulse, ICONVOLVE_ENCLOSED);
```

where `ICONVOLVE_ENCLOSED` is a boundary convolution option. The second formulation is useful for nesting of operator calls.

REFERENCES

1. "Information Technology, Computer Graphics and Image Processing, Image Processing and Interchange, Functional Specification, Part 1: Common Architecture for Imaging," ISO/IEC 1187-1:1995(E).
2. "Information Technology, Computer Graphics and Image Processing, Image Processing and Interchange, Functional Specification, Part 2: Programmer's Imaging Kernel System Application Program Interface," ISO/IEC 1187-2:1994(E).
3. A. F. Clark, "Image Processing and Interchange: The Image Model," *Proc. SPIE/IS&T Conference on Image Processing and Interchange: Implementation and Systems*, San Jose, CA, February 1992, **1659**, SPIE Press, Bellingham, WA, 106–116.
4. W. K. Pratt, "An Overview of the ISO/IEC Programmer's Imaging Kernel System Application Program Interface," *Proc. SPIE/IS&T Conference on Image Processing and Interchange: Implementation and Systems*, San Jose, CA, February 1992, **1659**, SPIE Press, Bellingham, WA, 117–129.
5. W. K. Pratt, *PIKS Foundation C Programmer's Guide*, Manning Publications, Prentice Hall, Upper Saddle River, NJ, 1995.
6. W. K. Pratt, "Overview of the ISO/IEC Image Processing and Interchange Standard," in *Standards for Electronic Imaging Technologies, Devices and Systems,* M. C. Nier, Ed., San Jose, CA, February 1996, **CR61**, SPIE Press, Bellingham, WA, 29–53.
7. "Information Technology, Computer Graphics and Image Processing, Image Processing and Interchange, Functional Specification, Part 3: Image Interchange Facility," ISO/IEC 1187-3:1995(E).
8. C. Blum and G. R. Hoffman, "ISO/IEC's Image Interchange Facility," *Proc. SPIE/IS&T Conf. on Image Processing and Interchange: Implementation and Systems*, San Jose, CA, February 1992, **1659**, SPIE Press, Bellingham, WA, 130–141.

9. "Information Technology, Computer Graphics and Image Processing, Image Processing and Interchange, Functional Specification, Part 5: Basic Image Interchange Format," ISO/IEC 1187-5:1998(E).
10. W. K. Pratt, *Digital Image Processing, Third Edition*, Wiley Interscience, New York, 101.
11. "Information Technology, Computer Graphics and Image Processing, Image Processing and Interchange, Application Program Interface Language Bindings, *Part 4: C*," ISO/IEC 1188-4:1995(E).