

---

---

**Information technology — Computer  
graphics and image processing — Image  
Processing and Interchange (IPI) —  
Functional specification —**

**Part 2:**

Programmer's imaging kernel system  
application program interface

*Technologies de l'information — Infographie et traitement de l'image —  
Traitement de l'image et échange (IPI) — Spécification fonctionnelle —  
Partie 2: Interface de programme d'application PIKS*



## Contents

|                                                                                    |      |
|------------------------------------------------------------------------------------|------|
| Foreword                                                                           | viii |
| 1 Scope                                                                            | 1    |
| 2 Normative references                                                             | 3    |
| 3 Symbols and abbreviations                                                        | 5    |
| 4 Programmer's Imaging Kernel System specification                                 | 11   |
| 4.1 PIKS imaging model                                                             | 11   |
| 4.1.1 Image data objects                                                           | 12   |
| 4.1.2 Non-image data objects                                                       | 14   |
| 4.1.3 Data object creation                                                         | 15   |
| 4.2 PIKS operators, tools, data object repository utilities, and system mechanisms | 15   |
| 4.2.1 Operators                                                                    | 15   |
| 4.2.2 Tools                                                                        | 16   |
| 4.2.3 Data object repository                                                       | 16   |
| 4.2.3.1 Impulse response function arrays                                           | 17   |
| 4.2.3.2 Dither arrays                                                              | 17   |
| 4.2.3.3 Colour conversion matrices                                                 | 17   |
| 4.2.4 Utilities                                                                    | 18   |
| 4.2.5 System mechanisms                                                            | 18   |

© ISO/IEC 1994

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

|                                                                         |    |
|-------------------------------------------------------------------------|----|
| 4.3 PIKS operator model                                                 | 30 |
| 4.3.1 Non-image to non-image operators                                  | 30 |
| 4.3.2 Image to non-image operators                                      | 30 |
| 4.3.3 Image to image operators                                          | 31 |
| 4.3.4 Neighbourhood operators                                           | 31 |
| 4.3.5 Operator index assignment                                         | 33 |
| 4.4 PIKS system mechanisms                                              | 35 |
| 4.4.1 Data object allocation                                            | 35 |
| 4.4.2 Match point control                                               | 35 |
| 4.4.3 ROI control                                                       | 36 |
| 4.4.4 ROI data object creation and manipulation                         | 40 |
| 4.4.5 Asynchronous control                                              | 41 |
| 4.4.6 Element chaining                                                  | 43 |
| 4.4.6.1 Chain construction                                              | 44 |
| 4.4.6.2 Chain execution                                                 | 45 |
| 4.4.7 Virtual register control                                          | 47 |
| 4.4.7.1 Virtual registers for storage of PIKS temporary variables       | 48 |
| 4.4.7.2 Virtual registers for asynchronous control                      | 48 |
| 4.4.7.3 Virtual registers for chain iteration and conditional execution | 48 |
| 4.4.7.4 Virtual registers for auditing asynchronous state               | 49 |
| 4.4.8 Global element control                                            | 49 |
| 4.4.9 Composite image management                                        | 50 |
| 4.4.9.1 Composite image identifier arrays                               | 51 |
| 4.4.9.2 Composite image identifier lists                                | 51 |
| 4.4.9.3 Composite image identifier records                              | 52 |
| 4.4.10 PIKS error handling                                              | 53 |
| 4.4.11 PIKS operational states                                          | 55 |
| 4.5 PIKS utilities                                                      | 57 |
| 4.5.1 Inquiry                                                           | 57 |
| 4.5.2 Import and export                                                 | 57 |
| 4.5.2.1 Data object import and export utilities                         | 58 |
| 4.5.2.2 PIKS to application data type conversion                        | 58 |

|          |                                               |           |
|----------|-----------------------------------------------|-----------|
| <b>5</b> | <b>PIKS conformance and extension methods</b> | <b>61</b> |
| 5.1      | Foundation profile                            | 61        |
| 5.2      | Application profiles                          | 61        |
| 5.3      | Full profile                                  | 61        |
| 5.4      | IIF capability profiles                       | 61        |
| 5.5      | PIKS profile definitions                      | 61        |
| 5.6      | Extension methods                             | 74        |
| <b>6</b> | <b>PIKS element specification template</b>    | <b>75</b> |
| <b>7</b> | <b>PIKS element specifications</b>            | <b>83</b> |

## Annexes:

|          |                                                                      |            |
|----------|----------------------------------------------------------------------|------------|
| <b>A</b> | <b>Definitions of mathematical functions</b>                         | <b>817</b> |
| A.1      | Conventional mathematical symbols                                    | 817        |
| A.2      | Operational symbols                                                  | 817        |
| A.3      | Mathematical functions                                               | 819        |
| <b>B</b> | <b>PIKS element support - source image structure</b>                 | <b>825</b> |
| <b>C</b> | <b>PIKS element support - destination image structure</b>            | <b>835</b> |
| <b>D</b> | <b>PIKS element support - source and destination image data type</b> | <b>845</b> |
| <b>E</b> | <b>PIKS operator support - operator order</b>                        | <b>855</b> |
| <b>F</b> | <b>PIKS element functionality</b>                                    | <b>861</b> |
| <b>G</b> | <b>PIKS data object repository</b>                                   | <b>871</b> |
| G.1      | Impulse response function arrays                                     | 872        |
| G.2      | Dither arrays                                                        | 888        |
| G.3      | Colour conversion matrices                                           | 889        |
| <b>H</b> | <b>PIKS image resampling</b>                                         | <b>907</b> |
| <b>J</b> | <b>PIKS error codes</b>                                              | <b>917</b> |
| <b>K</b> | <b>Bibliography of image processing books</b>                        | <b>925</b> |
| <b>L</b> | <b>Alphabetical listings of PIKS elements by profile</b>             | <b>927</b> |
| L.1      | Alphabetical listing of PIKS elements in the Foundation profile      | 928        |
| L.2      | Alphabetical listing of PIKS elements in the Technical profile       | 932        |
| L.3      | Alphabetical listing of PIKS elements in the Scientific profile      | 938        |
| L.4      | Alphabetical listing of PIKS elements in the Full profile            | 946        |

**List of figures**

|    |                                                          |    |
|----|----------------------------------------------------------|----|
| 1  | PIKS imaging model                                       | 11 |
| 2  | PIKS operator model: non-image to non-image operators    | 30 |
| 3  | PIKS operator model: image to non-image operators        | 31 |
| 4  | PIKS operator model: image to image operators            | 32 |
| 5  | Operator index assignment                                | 33 |
| 6  | Example of match point translation for image subtraction | 37 |
| 7  | Examples of ROI operation                                | 39 |
| 8  | Example of the relationship between a ROI and an image   | 41 |
| 9  | Examples of PIKS element chains                          | 45 |
| 10 | PIKS operational state diagram                           | 55 |
| 11 | PIKS interface to the IIF gateway and an application     | 57 |
| 12 | PIKS to application interface                            | 59 |

**List of tables**

|    |                                                                                       |    |
|----|---------------------------------------------------------------------------------------|----|
| 1  | PIKS image objects                                                                    | 13 |
| 2  | PIKS operators listed by functional class                                             | 19 |
| 3  | PIKS tools listed by functional class                                                 | 24 |
| 4  | PIKS utilities listed by functional class                                             | 26 |
| 5  | PIKS system mechanisms listed by functional class                                     | 28 |
| 6  | PIKS data type codes                                                                  | 60 |
| 7  | External physical data types supported by PIKS                                        | 60 |
| 8  | Data types of PIKS non-image data objects supported in the Foundation profile         | 62 |
| 9  | PIKS conformance profiles                                                             | 63 |
| 10 | Number of operators, tools, utilities, mechanisms, and total elements in each profile | 64 |
| 11 | PIKS elements in each profile                                                         | 65 |

## Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 12087-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 24, *Computer graphics and image processing*.

ISO/IEC 12087 consists of the following parts, under the general title *Information technology — Computer graphics and image processing — Image processing and interchange (IPI) — Functional specification*:

- *Part 1: Common architecture for imaging*
- *Part 2: Programmer's imaging kernel system application program interface*
- *Part 3: Image Interchange Facility*

Annexes A, G, H and J form an integral part of this part of ISO/IEC 12087. Annexes B, C, D, E, F, K and L are for information only.

# Information technology — Computer graphics and image processing — Image Processing and Interchange (IPI) — Functional specification —

## Part 2:

## Programmer's imaging kernel system application program interface

### 1 Scope

This part of ISO/IEC 12087 establishes the specification of the application program interface (API), called the Programmer's Imaging Kernel System (PIKS). ISO/IEC 12087-1 establishes the conceptual and architectural definitions of the Common Architecture for Imaging (CAI). ISO/IEC 12087-3 establishes the specification of the Image Interchange Facility (IIF).

PIKS is intended to provide a rich set of both low-level and high-level services on image and image-derived data objects. These services can be used as building blocks for a broad range of common imaging applications.

A conscious effort has been made by the developers of PIKS to create a standard that does not favor any particular computing system. Implementations of PIKS should be possible on computing systems ranging in architecture from general purpose computers to specialised hardware accelerators, ranging in size from personal computers to mainframe supercomputers, and ranging in connectivity from stand-alone machines to distributed computing networks.

Where applicable, PIKS relies on other APIs and data format standards to provide capabilities that are not unique to imaging. The following lists contain a summary of technological capabilities provided by PIKS and not provided by PIKS. However, it should be noted that PIKS functionality may be useful as a pre-processor or co-processor for many of the technologies in the "Not provided by PIKS" list.

**Scope**

**Provided by PIKS**

- image analysis
- image classification (basic)
- image enhancement
- image interchange between PIKS and an application
- image interchange between PIKS and the IIF
- image manipulation primitives
- image processing data object generation tools (e.g., image filter functions)
- image restoration
- image visualization (basic)
- standard colour models

**Not provided by PIKS**

- audio
- computer graphics
- device control
- image acquisition
- image communication
- image compression and decompression
- image display
- image transport between applications
- image understanding
- multimedia
- pattern recognition
- specific implementations
- video
- window systems

NOTE – The Image Interchange Facility of ISO/IEC 12087-3 specifies image compression and decompression functionality and image transport between applications and between an application and PIKS.

## 2 Normative references

The following standards contain provisions which, through references in this text, constitute provisions of this part of the ISO/IEC 12087. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 12087 are encouraged to investigate the possibility of applying the most recent standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 12087-1:—<sup>1)</sup>, *Information technology — Computer graphics and image processing — Image Processing and Interchange (IPI) — Functional specification — Part 1: Common Architecture for Imaging.*

ISO/IEC 12087-3:—<sup>1)</sup>, *Information technology — Computer graphics and image processing — Image Processing and Interchange (IPI) — Functional specification — Part 3: Image Interchange Facility.*

1) to be published.

*This page intentionally blank.*

### 3 Symbols and abbreviations

The following are symbols and abbreviations utilized in this standard. The mathematical functions are defined in Annex A.

|        |                                                                    |
|--------|--------------------------------------------------------------------|
| 0D     | zero-dimensional                                                   |
| 1D     | one-dimensional                                                    |
| 2D     | two-dimensional                                                    |
| 3D     | three-dimensional                                                  |
| 4D     | four-dimensional                                                   |
| 5D     | five-dimensional                                                   |
| ACOS   | arc cosine function                                                |
| AND    | logical AND operator                                               |
| API    | Application Program Interface                                      |
| ASIN   | arc sine function                                                  |
| ASSIGN | index assignment function between image and operator input indices |
| ASN.1  | abstract syntax notation one                                       |
| ATAN   | arc tangent function                                               |
| ATAN2  | arc tangent ratio function                                         |
| b      | spectral band index of an image                                    |
| B      | image spectral band size                                           |
| BD     | Boolean data type (PIKS internal)                                  |
| BER    | basic encoding rules                                               |
| BI     | external Boolean pixel data type                                   |
| BITEXT | bit extraction function                                            |
| BITINS | bit insertion function                                             |
| BP     | Boolean data type (PIKS parameter)                                 |
| CAI    | Common Architecture for Imaging                                    |
| CAS    | Hartley cas function                                               |
| CCIR   | Comite Consultatif International des Radiocommunications           |
| CD     | complex arithmetic data type (PIKS internal)                       |
| CF     | external complex floating point pixel data type                    |
| CHAIN  | chain data object                                                  |
| CHOICE | choice function                                                    |

|                  |                                                    |
|------------------|----------------------------------------------------|
| CIE              | Commission Internationale de l'Eclairage           |
| COLR             | colour image                                       |
| CONJ             | complex conjugate function                         |
| COS              | cosine function                                    |
| CP               | complex arithmetic data type (PIKS parameter)      |
| CS               | character string data type                         |
| D                | single operator output image                       |
| DET              | determinant of matrix argument                     |
| DOB              | destination non-image object                       |
| D <sub>q</sub>   | q-th operator output image, $1 \leq q \leq Q$      |
| DST              | single destination image                           |
| DST <sub>q</sub> | q-th destination image, $1 \leq q \leq Q$          |
| EBU              | European Broadcasting Union                        |
| EP               | enumerated parameter                               |
| ERF              | Gaussian error function                            |
| EXP              | exponential function                               |
| EXT              | external data type                                 |
| GEN              | generic image                                      |
| HIST             | histogram data object                              |
| i                | square root of minus one, $i = \sqrt{-1}$          |
| ID               | data object identifier (PIKS internal)             |
| ID_ARRAY         | composite image identifier array                   |
| ID_LIST          | composite image identifier list                    |
| ID_RECORD        | composite image identifier record                  |
| IEC              | International Electrotechnical Commission          |
| IEEE             | Institute Electrical and Electronic Engineers      |
| IIF              | Image Interchange Facility                         |
| IMAG             | imaginary part of complex number argument function |
| in               | input parameter                                    |
| IP               | data object identifier (PIKS parameter)            |
| IPI              | Image Processing and Interchange                   |
| ISO              | International Organisation for Standardisation     |

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <b>j</b>            | first operator index                            |
| <b>J</b>            | first operator index size                       |
| <b>k</b>            | second operator index                           |
| <b>K</b>            | second operator index size                      |
| <b>l</b>            | third operator index                            |
| <b>L</b>            | third operator index size                       |
| <b>LIV</b>          | lowest integer value of argument function       |
| <b>LOG</b>          | base <i>e</i> natural logarithm function        |
| <b>LOOK</b>         | lookup table function                           |
| <b>LUT</b>          | lookup table data object                        |
| <b>m</b>            | fourth operator index                           |
| <b>M</b>            | fourth operator index size                      |
| <b>MATRIX</b>       | matrix data object                              |
| <b>MAX</b>          | maximum of argument sequence function           |
| <b>MIN</b>          | minimum of argument sequence function           |
| <b>MOD</b>          | modulus function                                |
| <b>MON</b>          | monochrome image                                |
| <b>n</b>            | fifth operator index                            |
| <b>N</b>            | fifth operator index size                       |
| <b>NA</b>           | not applicable                                  |
| <b>NAND</b>         | logical NAND operator                           |
| <b>NBHOOD_ARRAY</b> | neighbourhood array data object                 |
| <b>ND</b>           | non-negative integer data type (PIKS internal)  |
| <b>NI</b>           | external non-negative integer pixel data type   |
| <b>NIV</b>          | nearest integer value of argument function      |
| <b>NOR</b>          | logical NOR operator                            |
| <b>NOT</b>          | logical NOT operator                            |
| <b>NP</b>           | non-negative integer data type (PIKS parameter) |
| <b>NTSC</b>         | National Television Systems Committee           |
| <b>NULL</b>         | null data type                                  |
| <b>OR</b>           | logical OR operator                             |
| <b>out</b>          | output parameter                                |

|                   |                                                                       |
|-------------------|-----------------------------------------------------------------------|
| PIKS              | Programmer's Imaging Kernel System                                    |
| PIXEL_RECORD      | pixel record data object                                              |
| POW               | power law function of two arguments                                   |
| RD                | real arithmetic data type (PIKS internal)                             |
| REAL              | real part of complex number argument function                         |
| REASSIGN          | index reassignment function between operator output and image indices |
| RES <sub>1</sub>  | one-dimensional resampling function                                   |
| RES <sub>2</sub>  | two-dimensional resampling function                                   |
| RES <sub>3</sub>  | three-dimensional resampling function                                 |
| RES <sub>4</sub>  | four-dimensional resampling function                                  |
| RES <sub>5</sub>  | five-dimensional resampling function                                  |
| RF                | external real floating point pixel data type                          |
| ROI               | Region-of-Interest                                                    |
| ROI_ARRAY         | array Region-of-Interest data object                                  |
| ROI_COORD         | coordinate collection Region-of-Interest data object                  |
| ROI_ELLIP         | elliptical Region-of-Interest data object                             |
| ROI_GEN           | generic Region-of-Interest data object                                |
| ROI_POLY          | polygon Region-of-Interest data object                                |
| ROI_RECT          | rectangular Region-of-Interest data object                            |
| ROID              | Region-of-interest of destination image                               |
| ROID <sub>q</sub> | Region-of-interest of q-th destination image, $1 \leq q \leq Q$       |
| ROIS              | Region-of-interest of source image                                    |
| ROIS <sub>p</sub> | Region-of-interest of p-th source image, $1 \leq p \leq P$            |
| RP                | real arithmetic data type (PIKS parameter)                            |
| S                 | single operator input image                                           |
| SD                | signed integer data type (PIKS internal)                              |
| SI                | signed integer number                                                 |
| SIN               | sine function                                                         |
| SMPTE             | Society of Motion Picture and Television Engineers                    |
| SOB               | source non-image object                                               |
| SORT              | sort function                                                         |
| Sp                | p-th operator input image, $1 \leq p \leq P$                          |

|                  |                                                          |
|------------------|----------------------------------------------------------|
| SP               | signed integer data type (PIKS parameter)                |
| SPE              | spectral image                                           |
| SRC              | single source image                                      |
| SRC <sub>p</sub> | p-th source image, $1 \leq p \leq P$                     |
| STATIC_ARRAY     | static array data object                                 |
| t                | temporal index of an image                               |
| T                | image temporal size                                      |
| TAN              | tangent function                                         |
| TEM              | temporal image                                           |
| TC               | temporal-color image                                     |
| TI               | two's complement fixed point integer number              |
| TS               | temporal-spectral image                                  |
| TUPLE            | tuple data object                                        |
| VALUE_BOUNDS     | value bounds collection data object                      |
| VOL              | volume image                                             |
| VC               | volume-colour image                                      |
| VREG             | virtual register data object                             |
| VS               | volume-spectral image                                    |
| VT               | volume-temporal image                                    |
| VTC              | volume-temporal-colour image                             |
| VTs              | volume-temporal-spectral image                           |
| x                | horizontal index (along rows) of an image                |
| X                | image horizontal size (image width)                      |
| XOR              | exclusive OR operator                                    |
| y                | vertical index (along columns) of an image               |
| Y                | image vertical size (image height)                       |
| z                | depth index of an image                                  |
| Z                | image depth size                                         |
| $\cap$           | boolean intersection symbol                              |
| $\cup$           | boolean union symbol                                     |
| $\overline{[ ]}$ | overbar indicates complement of boolean argument         |
| $ [.] $          | vertical lines indicate magnitude of arithmetic argument |

|         |                                          |
|---------|------------------------------------------|
| $[.]^T$ | superscript T indicates matrix transpose |
| +       | plus or positive symbol                  |
| -       | minus or negative symbol                 |
| ×       | multiply symbol                          |
| /       | arithmetic divide symbol                 |
| ÷       | integer divide symbol                    |
| ⊕       | logical AND symbol                       |
| ⊖       | logical OR symbol                        |
| ⊗       | logical XOR symbol                       |
| ⊛       | convolution symbol                       |
| ⊞       | morphological dilation symbol            |
| ⊟       | morphological erosion symbol             |
| Σ       | sequence summation symbol                |
| Π       | sequence product symbol                  |
| M       | sequence maxima symbol                   |
| N       | sequence minima symbol                   |
| ∪       | sequence union symbol                    |
| ∩       | sequence intersection symbol             |

## 4 Programmer's Imaging Kernel System specification

This clause establishes the specification of the Programmer's Imaging Kernel System (PIKS) application program interface.

### 4.1 PIKS imaging model

Figure 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:

- data objects
- operators, tools, and utilities
- system mechanisms
- import and export

The PIKS data objects are specified in ISO/IEC 12087-1. They include both image and 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 Image Interchange Facility (IIF) specified in ISO/IEC 12087-3. Also, PIKS can receive data objects in its internal format, which have been supplied by the IIF. PIKS can operate independently of the IIF.

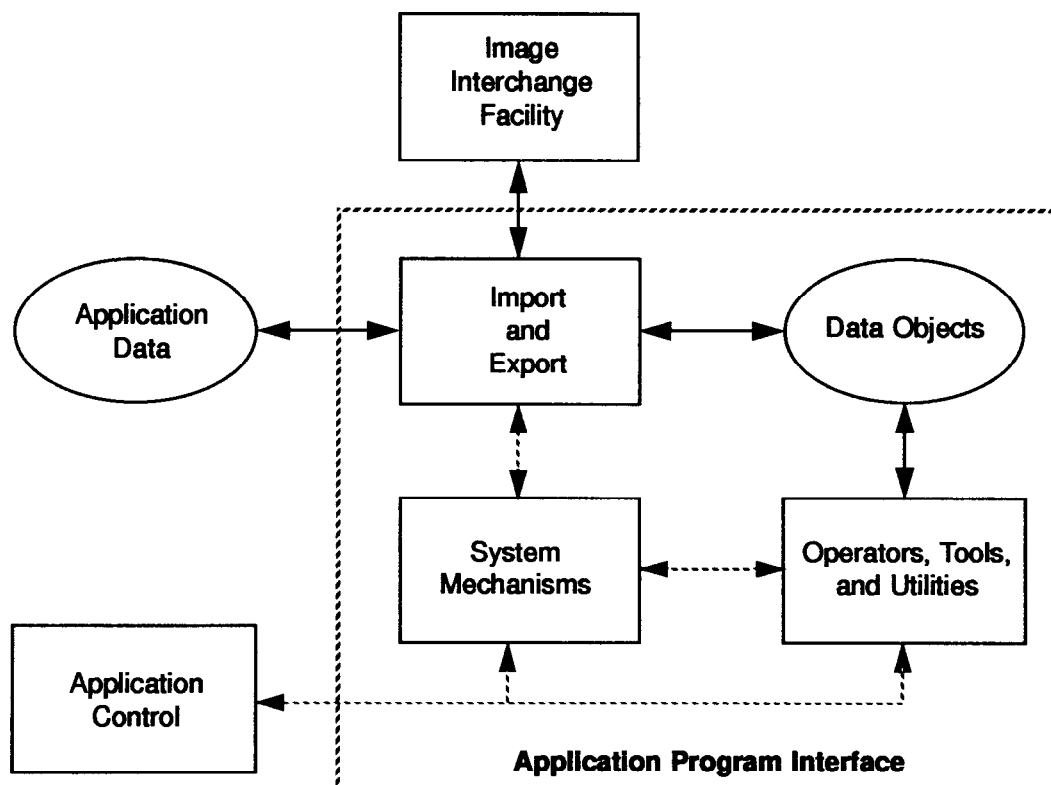


Figure 1 – PIKS imaging model

#### 4.1.1 Image data objects

ISO/IEC 12087-1 describes the image data objects supported by PIKS. This subclause summarizes this information and establishes notation used in this part of ISO/IEC 12087.

As specified in ISO/IEC 12087-1, PIKS supports a general five-dimensional image object with ordered indices

$$0 \leq x \leq X - 1$$

$$0 \leq y \leq Y - 1$$

$$0 \leq z \leq Z - 1$$

$$0 \leq t \leq T - 1$$

$$0 \leq b \leq B - 1$$

where  $x$  is the horizontal index,  $y$  is the vertical index,  $z$  is the depth index,  $t$  is the time index, and  $b$  is the band index. The characters  $X$ ,  $Y$ ,  $Z$ ,  $T$ ,  $B$  represent the maximum sizes of the image coordinates  $x$ ,  $y$ ,  $z$ ,  $t$ ,  $b$ , respectively.

NOTE – Some of the image dimensions may be unpopulated, e.g., for a monochrome image,  $Z = 1$ ,  $T = 1$  and  $B = 1$ .

A two-dimensional pixel array involving only two indices is called a pixel plane. A pixel plane involving the  $x$ - $y$  indices is called a  $x$ - $y$  pixel plane. Conceptually, a five-dimensional image object can be considered to be organized as a set of depth, time, and band  $x$ - $y$  pixel planes.

NOTE – PIKS operators are able to access and process arbitrary pixel planes.

In this part of ISO/IEC 12087, a single source image is expressed notationally as a five-dimensional array

$$\text{SRC}(x, y, z, t, b)$$

and the  $p$ -th source image in a set of  $P$  images is expressed notationally as

$$\text{SRC}_p(x, y, z, t, b)$$

where  $1 \leq p \leq P$ .

NOTE – In general, source images can be of different sizes.

In this part of ISO/IEC 12087, a single destination image is expressed notationally as a five-dimensional array

$$\text{DST}(x, y, z, t, b)$$

and the  $q$ -th destination image in a set of  $Q$  images is expressed notationally as

$$\text{DST}_q(x, y, z, t, b)$$

where  $1 \leq q \leq Q$ .

NOTE – In general, source and destination images can be of different sizes. However, some PIKS operators require that source and destination images be of the same size, e.g., the Fourier transform operator.

PIKS gives semantic meaning to certain dimensional subsets of a general five-dimensional image object. These are listed in Table 1. Definitions of these image objects are presented in ISO/IEC 12087-1.

Table 1 – PIKS image objects

| Semantic description     | image indices | code |
|--------------------------|---------------|------|
| monochrome               | x, y, 0, 0, 0 | MON  |
| volume                   | x, y, z, 0, 0 | VOL  |
| temporal                 | x, y, 0, t, 0 | TEM  |
| colour                   | x, y, 0, 0, b | COLR |
| spectral                 | x, y, 0, 0, b | SPE  |
| volume-temporal          | x, y, z, t, 0 | VT   |
| volume-colour            | x, y, z, 0, b | VC   |
| volume-spectral          | x, y, z, 0, b | VS   |
| temporal-colour          | x, y, 0, t, b | TC   |
| temporal-spectral        | x, y, 0, t, b | TS   |
| volume-temporal-colour   | x, y, z, t, b | VTC  |
| volume-temporal-spectral | x, y, z, t, b | VTs  |
| generic                  | x, y, z, t, b | GEN  |

NOTE – The generic category represents images whose indices do not have any semantic meaning. For notational consistency, the generic image uses the same index notation as the semantic categories.

As specified in ISO/IEC 12087-1, PIKS supports the following abstract data types for the internal representation of image pixels.

| <u>Code</u> | <u>Data type</u>     |
|-------------|----------------------|
| BD          | Boolean              |
| ND          | non-negative integer |
| SD          | signed integer       |
| RD          | real arithmetic      |
| CD          | complex arithmetic   |

All image pixels accessed by the x, y, z, t indices shall be of the same pixel data type. The pixel data type may differ between bands. An image whose pixels are of different data types across bands is called a heterogeneous band image.

A PIKS implementation shall be free to provide any form of physical representation of the image pixel data types provided that the physical representation satisfies the abstract data type definitions, as specified in ISO/IEC 12087-1. The minimum levels of storage precision for the ND, SD, RD, and CD pixel types shall be

|    |                                                                  |
|----|------------------------------------------------------------------|
| ND | 8 bits per pixel                                                 |
| SD | 16 bits per pixel                                                |
| RD | 32 bits per pixel                                                |
| CD | 32 bits per real component and per imaginary component per pixel |

#### 4.1.2 Non-image data objects

ISO/IEC 12087-1 describes the non-image data objects supported by PIKS. This subclause summarizes this information and establishes notation used in ISO/IEC 12087-2.

PIKS supports the following internal, non-image data objects, which are defined in ISO/IEC 12087-1:

| <u>Code</u>  | <u>Name</u>                               |
|--------------|-------------------------------------------|
| CHAIN        | chain                                     |
| HIST         | histogram                                 |
| ID_ARRAY     | composite identifier array                |
| ID_LIST      | composite identifier list                 |
| ID_RECORD    | composite identifier record               |
| LUT          | lookup table                              |
| MATRIX       | matrix                                    |
| NBHOOD_ARRAY | neighbourhood array                       |
| PIXEL_RECORD | pixel record                              |
| ROI_ARRAY    | region-of-interest, array                 |
| ROI_COORD    | region-of-interest, coordinate collection |
| ROI_ELLIP    | region-of-interest, elliptical            |
| ROI_GEN      | region-of-interest, generic               |
| ROI_POLY     | region-of-interest, polygon               |
| ROI_RECT     | region-of-interest, rectangular           |
| STATIC_ARRAY | static array                              |
| TUPLE        | tuple                                     |
| VALUE_BOUNDS | value bounds collection                   |
| VREG         | virtual register                          |

Non-image data objects that are created by PIKS elements and returned directly to an application are not regarded as PIKS non-image data objects. The data structures of directly-returned non-image data objects are defined by the elements providing the return. Collectively, such data objects are called external data objects, and are referenced by the code EXT.

In this part of ISO/IEC 12087, a single source non-image data object is expressed notationally as SOB, and the p-th source non-image data object in a set of P objects is expressed as SOB<sub>p</sub> where  $1 \leq p \leq P$ . In this standard, single destination non-image data object is expressed notationally as DOB, and the q-th destination non-image data object in a set of Q objects is expressed as DOB<sub>q</sub> where  $1 \leq q \leq Q$ .

4.1.3 Data object creation

PIKS source and destination data objects that are referenced by an operator, tool, or utility shall be allocated by a system management allocation mechanism, as described in 4.4, prior to the invocation of an operator, tool, or utility. There are three classes of allocated data objects:

- Class A - identified with object attributes and data;
- Class B - identified with object attributes only;
- Class C - identified only

When an image is allocated, an identifier is assigned to the image data object by the PIKS implementation. The image allocation mechanism allows an application to specify certain image attributes, such as size, structure, and colour space, upon allocation, or to leave the attributes unspecified. When a class C data object is specified as a destination, many of the PIKS operators, tools, and utilities will supply the “missing” data object attributes.

NOTE – A PIKS implementation may reserve data storage for a Class B data object at the time that the data object is allocated, or the implementation may defer data storage reservation until the time that the data for a data object is first available.

4.2 PIKS operators, tools, data object repository, utilities, and system mechanisms

This subclause establishes the general specification of the PIKS operators, tools, data object repository, utilities, and system mechanisms.

4.2.1 Operators

PIKS operators are functional elements that perform manipulations of images or of data objects extracted from images in order to enhance images, restore images, or to assist in the extraction of information from images. Operators can be classified as being primitive or high-level operators. Primitive operators are a small set of low-level operators, fundamental to image processing, which can be used to implement high-level operators. Examples are image convolution and image histogram generation. High-level operators are operators of greater complexity. Examples are adaptive histogram equalization and Hough transform.

NOTE – There is no requirement in the PIKS specification that high-level operators shall be constructed from primitive operators.

Another form of operator classification is by functionality, e.g., operators that perform geometric manipulations such as rotate or operators that detect image edges. Table 2 contains a list of the PIKS operators arranged by functionality class. These operators are specified in clause 7.

Still another form of operator classification, as listed below, is by the types of source and destination objects supported by an operator.

| <u>Source</u> | <u>Destination</u> |
|---------------|--------------------|
| non-image     | non-image          |
| image         | non-image          |
| image         | image              |

NOTE – PIKS does not provide any non-image to image operators. PIKS does provide several image generation tools.

Table 2 indicates the type of source and destination objects supported by each functional class of operators.

PIKS operators that have an image as a source data object can also be classified by their neighbourhood processing dimensionality as defined below.

- 0D A zero-dimensional operator is one that operates on each pixel of a source image independently of its neighbouring pixels. An example is taking the logarithm of a spectral image.
- 1D A one-dimensional operator is one that operates on a group of pixels along one dimension of an image. An example is a one-dimensional moving window average along the rows of a monochrome image. Another example is a one-dimensional Fourier transform along the columns of a monochrome image.
- 2D A two-dimensional operator is one that operates on a group of pixels along two dimensions of an image. An example is erosion of a monochrome image with a two-dimensional morphological structuring element.
- 3D A three-dimensional operator is one that operates on a group of pixels along three dimensions of an image. An example is convolution of a volume image with a three-dimensional impulse response function array.
- 4D A four-dimensional operator is one that operates on a group of pixels along four dimensions of an image. An example is computation of the four-dimensional Fourier transform of a volume-temporal image.
- 5D A five-dimensional operator is one that operates on a group of pixels along five dimensions of an image. An example is computation of the extrema pixels of a five-dimensional image.

#### NOTES

1 – By this form of operator classification, the neighbourhood need not be compact. Also, the neighbourhood may contain all of the pixels along an image dimension.

2 – In subsequent clauses, an “rD” operator is called an operator of order *r*. The words “dimensional” and “dimension” are reserved, whenever possible, for image descriptors.

Annexes B to F contain alphabetical listings of PIKS elements with indications of the supported image structure, image data type, operator order, and element functionality.

### 4.2.2 Tools

PIKS tools are elements that create data objects to be used by PIKS operators. Examples are the generation of test images, the creation of impulse response function arrays, and the generation of Region-of-Interest objects.

Table 3 contains a list of PIKS tools. The PIKS tools are specified in clause 7.

### 4.2.3 Data object repository

Many data objects are routinely used by operators. An example is the set of Sobel edge detection horizontal and vertical edge gradient impulse response function arrays. PIKS provides a repository of commonly used data objects. Annex G contains a specification of the PIKS data object repository.

The data object repository contains three types of data:

- impulse response function arrays
- dither arrays
- colour conversion matrices

#### 4.2.3.1 Impulse response function arrays

Impulse response function arrays in the data object repository are commonly-used instantiations of two-dimensional neighbourhood array data objects. In the repository, each array is specified according to the following general form:

$$\frac{1}{S} \begin{bmatrix} H(0, 0) & H(1, 0) & \dots H(C-1, 0) \\ \vdots & \vdots & \vdots \\ H(0, r) & H(1, r) & \dots H(C-1, r) \\ \vdots & \vdots & \vdots \\ H(0, R-1) & H(1, R-1) & \dots H(C-1, R-1) \end{bmatrix}$$

where  $S$  is a scale factor.

#### NOTES

- 1 – The impulse response function arrays are specified with signed integer scale factors and signed integer array terms. The physical storage data format is implementation dependent.
- 2 – The impulse response array indexing is the same as the indexing of a two-dimensional image array.

#### 4.2.3.2 Dither arrays

Dither arrays in the data object repository are commonly-used instantiations of two-dimensional neighbourhood array data objects. In the repository, each dither array is specified according to the same general form as an impulse response function array.

#### 4.2.3.3 Colour conversion matrices

Colour conversion matrices in the data object repository are commonly-used instantiations of matrix data objects. In the repository, each 3x3 matrix is specified according to the following general form:

$$\begin{bmatrix} T(1, 1) & T(1, 2) & T(1, 3) \\ T(2, 1) & T(2, 2) & T(2, 3) \\ T(3, 1) & T(3, 2) & T(3, 3) \end{bmatrix}$$

All colour conversion matrices are specified for the transformation of an input column vector to an output column vector.

#### NOTES

- 1 – ISO/IEC 12087-1 defines the colour spaces and associated chromaticity coordinates and white points used to generate each colour conversion matrix.
- 2 – In order to minimize quantization error in colour component conversion, the colour conversion matrices are specified in terms of six decimal digits of precision, even though the original definitions specify fewer digits of precision.
- 3 – The colour conversion matrix names specify the proper input and output colour spaces for colour component conversion.

#### **4.2.4 Utilities**

PIKS utilities are elements that perform basic mechanical image manipulation tasks such as extracting a pixel from an image and composing a volumetric image from a set of monochromatic images. Table 4 contains a list of PIKS utilities. These utilities are specified in clause 7.

#### **4.2.5 System mechanisms**

PIKS system mechanisms are elements that perform control and management tasks such as allocating data storage references for PIKS data objects, opening and closing PIKS sessions, error handling, and element chaining. Table 5 contains a list of PIKS mechanisms.

**Table 2 – PIKS operators listed by functional class****ANALYSIS OPERATORS**

Class of 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 OPERATORS**

Class of image to image operators that classifies each pixel of a multispectral image into one of a specified number of classes based upon the amplitudes of pixels across image bands.

- classifier, Bayes
- classifier, nearest neighbour

**COLOUR OPERATORS**

Class of image to image operators that convert a colour image from one colour space to another.

- colour conversion, linear
- colour conversion, nonlinear
- colour conversion, subtractive
- colour lookup, interpolated
- luminance generation

**COMPLEX IMAGE OPERATORS**

Class of 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 OPERATORS**

Class of image to non-image operators that compute a correlation array of a pair of images.

- cross-correlation
- template match

**Table 2 – PIKS operators listed by functional class (*continued*)**

#### **EDGE DETECTION OPERATORS**

Class of 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 OPERATORS**

Class of 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 colour
- histogram modification
- outlier removal
- pseudocolour
- unsharp mask
- Wallis statistical differencing

#### **ENSEMBLE OPERATORS**

Class of image to image operators that perform arithmetic (add, subtract, multiply), extremal (maximum, minimum), and logical (and, or) combination of a pair of images.

- alpha blend, constant
- alpha blend, variable
- dyadic, arithmetic
- dyadic, complex
- dyadic, logical
- dyadic, predicate
- split image
- Z merge

#### **FEATURE EXTRACTION OPERATORS**

Class of 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 OPERATORS**

Class of 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

**Table 2 – PIKS operators listed by functional class (*continued*)****FILTERING OPERATORS (continued)**

- filtering, median
- filtering, pseudomedian
- filtering, rank order

**GEOMETRIC OPERATORS**

Class of image to image operators that perform geometric manipulations such as minification, magnification, rotation, and rubber sheet warping.

- 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 OPERATORS**

Class of 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 OPERATORS**

Class of image to image operators that perform morphological operations such as erosion, dilation, and skeletonization 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 OPERATORS**

Class of image to image operators that modify an image by pixel drawing or painting.

- draw pixels
- paint pixels

**Table 2 – PIKS operators listed by functional class (*continued*)**

#### **POINT OPERATORS**

Class of image to image operators that perform point manipulation such as magnitude, logarithm, and threshold 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 OPERATORS**

Class of image to image operators that prepare an image for display.

- diffuse
- dither

#### **SHAPE OPERATORS**

Class of 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 OPERATORS**

Class of image to image operators that perform multi-dimensional forward and inverse unitary transforms of an image.

- transform, cosine
- transform, Fourier
- transform, Hadamard
- transform, Hartley

**Table 2 – PIKS operators listed by functional class (*concluded*)**

**3D SPECIFIC OPERATORS**

Class of image to image operators that perform manipulations of three-dimensional image data.

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

**Table 3 – PIKS tools listed by functional class**

**IMAGE GENERATION**

Class of 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**

Class of 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

**LOOKUP TABLE GENERATION**

Class of tools that create entries of a lookup table data object.

- array to LUT

**MATRIX GENERATION**

Class of tools that create matrix data objects.

- colour conversion matrix

**Table 3 – PIKS tools listed by functional class (*concluded*)****REGION-OF-INTEREST GENERATION**

Class of 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**

Class of 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 process power spectrum
- windowing function

**Table 4 – PIKS utilities listed by functional class**

#### **EXPORT FROM PIKS**

Class of utilities that export image and non-image data objects from PIKS to an application or to the IIF.

- export histogram
- export image
- export LUT
- export matrix
- export neighbourhood array
- export ROI array
- export static array
- export tuple
- export value bounds
- get colour pixel
- get pixel
- get pixel array
- get pixel record
- output object

#### **IMPORT TO PIKS**

Class of utilities that import image and non-image data objects to PIKS from an application or from the IIF.

- import histogram
- import image
- import LUT
- import matrix
- import neighbourhood array
- import ROI array
- import static array
- import tuple
- import value bounds
- input object
- put colour pixel
- put pixel
- put pixel array
- put pixel record

#### **INQUIRY**

Class of 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

**Table 4 – PIKS utilities listed by functional class (*concluded*)**

**INTERNAL**

Class of 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

**Table 5 – PIKS system mechanisms listed by functional class**

## **CHAINING**

Class of system 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**

Class of system 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**

Class of system mechanisms that control the basic operational functionality of PIKS.

- abort asynchronous execution
- close PIKS
- close PIKS, emergency
- open PIKS
- synchronize

## **ERROR**

Class of system mechanisms that provide means of reporting operational errors.

- error handler
- error logger
- error test

**Table 5 – PIKS system mechanisms listed by functional class (*concluded*)****SYSTEM MANAGEMENT**

Class of system 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 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**

Class of system 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

### 4.3 PIKS operator model

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

- non-image to non-image
- image to non-image
- image to image

Class A source data objects (identified with object attributes and data) shall exist prior to the invocation of an operator. Destination data objects may be Class A or Class B (identified with object attributes only). In the case of a Class A destination data object, the previously stored data will be overwritten unless there is inhibition invoked by a PIKS system mechanism, as described in 4.4.1. Some operators support Class C destination data objects (identified only) or Class B destination data objects with partial object attributes. Such operators shall supply the necessary object attributes. If the attributes of a source data object are mismatched to the attributes expected by an operator, or if the attributes of a destination data object are mismatched to the attributes expected by an operator, that operator shall generate an error indication at the time of execution.

#### 4.3.1 Non-image to non-image operators

Figure 2 shows the PIKS operator model for the transformation of non-image data objects by an operator to produce other non-image data objects. An example of such a transformation is the generation of shape features from an image histogram. In this operator model, there is no explicit operator processing control; the transformation is completely defined by the specification of each operator, as presented in clause 7.

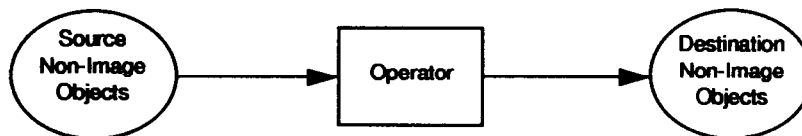


Figure 2 - PIKS operator model: non-image to non-image operators

#### 4.3.2 Image to non-image operators

Figure 3 shows the PIKS operator model for the transformation of image data objects by an operator to produce non-image data objects. An example of such a transformation is the computation of the sum-squared difference between a pair of images. In this operator model, processing is subject to two control mechanisms: source match point translation and Region-of-Interest (ROI) source selection. As shown in Figure 3, source match point translation occurs first, followed by ROI source selection. This is followed by index assignment within the operator, as defined in Figure 5. These control mechanisms are defined in 4.4.2 and 4.4.3, respectively. The dashed line in Figure 3 indicates the transfer of control information. The dotted line indicates the binding of source ROI objects to source image objects.

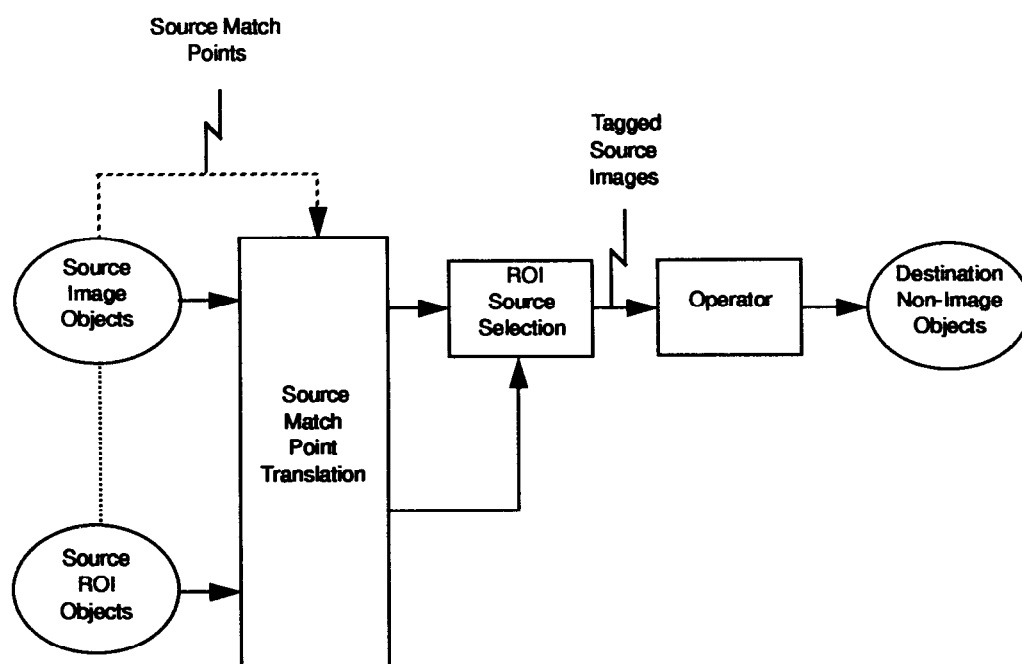


Figure 3 – PIKS operator model: image to non-image operators

NOTE – The concept of tagged source images is described in 4.4.2.

#### 4.3.3 Image to image operators

Figure 4 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 a transformation is the unsharp masking operator that 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. As shown in Figure 4, source match point translation occurs first, followed by ROI source selection. This is followed by index assignment within the operator, as defined in Figure 5. The destination match point translation and ROI destination selection control mechanisms are defined in 4.4.2 and 4.4.3, respectively.

NOTE – Some PIKS operators are restricted to certain semantic types of image objects. For example, the colour processing operators are applicable only to colour images.

Certain PIKS operators are capable of accepting source images with pixels of non-negative, or signed integer data type, and automatically converting the source images to the real arithmetic data type for processing. The source promotion conversion process is implementation-dependent. The Element Functionality list associated with each PIKS operator indicates by “yes” or “no” if the operator supports source promotion.

#### 4.3.4 Neighbourhood operators

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 source pixel to produce a single destination image pixel. For such operators, the neighbourhood is specified by a neighbourhood array non-image data object as defined in ISO/IEC 12087-1. 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, the “missing” source pixel values are assumed to be of zero value for arithmetic combinations or of FALSE value for Boolean combinations. This is the default condition. Some operators override the default condition by specifying the boundary conditions differently. For example, the *convolve\_2d* operator has an option for which the “missing” neighbourhood pixels are replaced by pixels obtained by reflections about the boundaries of pixels within the source image.

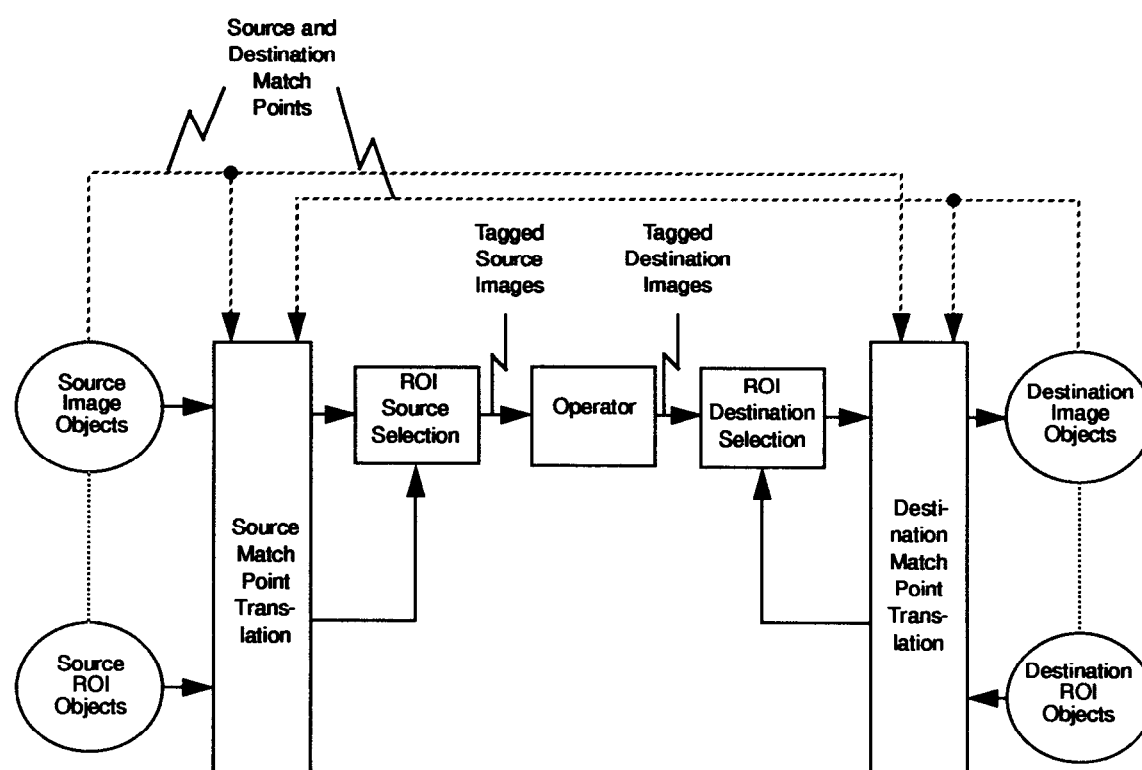


Figure 4 – PIKS operator model: image to image operators

### 4.3.5 Operator index assignment

Some PIKS image to non-image and image to image operators shall have the capability of assigning operator indices to image indices. This capability permits operators that are inherently  $r$ -th order, where  $r > 0$ , to be applied to five-dimensional images in a flexible manner.

Figure 5 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 ( $j, k, l, m, n$ ).

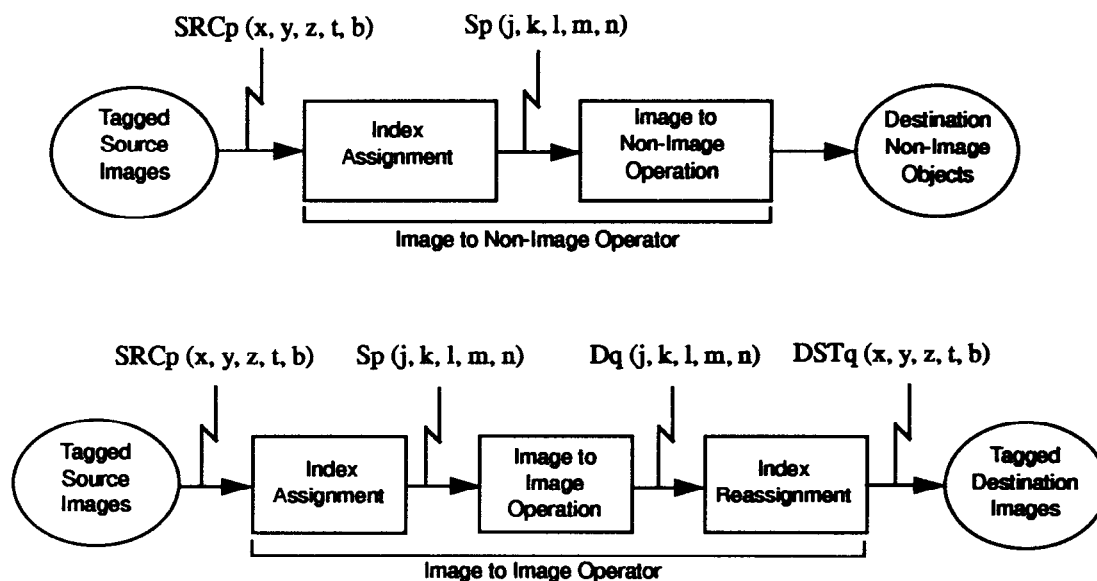


Figure 5 - Operator index assignment

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, as shown below.

| Operator Index | Image Index           |
|----------------|-----------------------|
| j              | x or y or z or t or b |
| k              | x or y or z or t or b |
| l              | x or y or z or t or b |
| m              | x or y or z or t or b |
| n              | x or y or z or t or b |

The character assignment  $[x, y, z, t, b]$  must be unique, i.e., if  $x$  is assigned to the  $j$  operator index, it cannot be assigned to any other operator index. Operator index assignment is not permitted for an  $rD$  operator if the band index is to be reassigned to one of the first  $r$  indices, and the source image does not have bands of the same data type.

The assignment process defined above is expressed functionally as

$$Sp(j, k, l, m, n) = SRCp(ASSIGN\{x, y, z, t, b\})$$

The destination image index reassignment process from an operator output image  $Dq(j, k, l, m, n)$  to a destination image  $DSTq(x, y, z, t, b)$  is a direct inverse of the assignment process. The reassignment process is expressed functionally as

$$DSTq(x, y, z, t, b) = Dq(REASSIGN\{j, k, l, m, n\})$$

NOTE – The index assignment and index reassignment processes are described in the *index\_assignment* system management mechanism described in clause 7.

As an example, consider a temporal source image  $SRC(x, y, 0, t, 0)$ , which is input to a 3D convolution operator to perform image smoothing in x-y-t image space. The following index assignment table could be formed.

| <u>Operator Index</u> | <u>Image Index</u> |
|-----------------------|--------------------|
| j                     | x                  |
| k                     | y                  |
| l                     | t                  |
| m                     | z                  |
| n                     | b                  |

For an rD operator, the assignment sequence of the (r+1)-st and higher indices is immaterial. In the example above, b could have been assigned to m, and z could have been assigned to n.

For most operators, if the operator is of order less than the image dimension, then the operator is executed independently over the pixel planes not directly involved in the operator. As an example, consider a temporal-colour source image  $SRC(x, y, 0, t, b)$ , which is input to a 2D convolution operator to perform image smoothing in each x-y pixel plane. The following index assignment table could be formed.

| <u>Operator Index</u> | <u>Image Index</u> |
|-----------------------|--------------------|
| j                     | x                  |
| k                     | y                  |
| l                     | z                  |
| m                     | t                  |
| n                     | b                  |

The 2D convolution operation will be executed for all T temporal planes and all B bands. A 2D operator need not be restricted to operate on the x-y pixel planes of a source image. It could, for example, operate on the t-x pixel planes of a temporal image by assignment of t to j and x to k.

#### NOTES

1 – A PIKS implementation may choose to physically reorder the indices of an image or choose to pass to an operator the index assignment information.

2 – The index assignment technique applies to some utilities and to some image generation and ROI generation tools.

## 4.4 PIKS system mechanisms

This subclause describes the PIKS system mechanisms. These mechanisms are specified in clause 7.

### 4.4.1 Data object allocation

PIKS shall provide automatic memory management that permits an application to specify image and non-image data objects by object identifiers, and not be concerned with physical storage mechanisms. Image data object identifiers are created by the *allocate\_image* system management mechanism. Non-image data object identifiers are created by an allocate system management mechanism corresponding to the non-image data object name. Image and non-image attribute information is accessed via the object identifier by inquiry mechanisms.

Any temporary images that may be required for storage of intermediate results in an implementation of a PIKS operator shall be automatically generated. A PIKS application does not have access to temporary images through PIKS.

PIKS does not specify the internal physical storage format of an implementation. PIKS implementations may choose to store images as conventional pixel planes, tiled planes, interleaved pixels, or any combination of storage means. The physical storage mechanism is hidden from the PIKS application.

Each PIKS implementation shall establish image size and capacity limits, and shall make this information available to the PIKS system inquire utility.

### 4.4.2 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. Notes associated with each operator indicate if the operator is or is not subject to match point control. In order to simplify the mathematical definitions of the operators, they are expressed as if there were no match point control.

NOTE – Match points need not be integer valued.

The generic effect of match point control for an operator that creates a single destination image from a single source image is to independently translate each dimension of the source image such that its match point is aligned with the corresponding match point of the destination image prior to processing. This generic operation is mathematically defined as follows. Let

$$\text{SRC}(x, y, z, t, b) = X_S x Y_S y Z_S z T_S t B_S b \text{ source image}$$

$$\text{DST}(x, y, z, t, b) = X_D x Y_D y Z_D z T_D t B_D b \text{ destination image}$$

$$(x_{sm}, y_{sm}, z_{sm}, t_{sm}, b_{sm}) = \text{source match point}$$

$$(x_{dm}, y_{dm}, z_{dm}, t_{dm}, b_{dm}) = \text{destination match point}$$

$$O\{\cdot\} = \text{general operator}$$

Then

$$\text{DST}(x, y, z, t, b) = O\{\text{RESr}\{\text{SRC}, x-x_{\text{dm}}+x_{\text{sm}}, y-y_{\text{dm}}+y_{\text{sm}}, z-z_{\text{dm}}+z_{\text{sm}}, t-t_{\text{dm}}+t_{\text{sm}}, b-b_{\text{dm}}+b_{\text{sm}}\}\}$$

for the destination image pixels  $x, y, z, t, b$  lying in the range

$$\text{MAX}\{0, x_{\text{dm}}-x_{\text{sm}}\} \leq x \leq \text{MIN}\{X_{\text{D}}-1, X_{\text{S}}-1+x_{\text{dm}}-x_{\text{sm}}\}$$

$$\text{MAX}\{0, y_{\text{dm}}-y_{\text{sm}}\} \leq y \leq \text{MIN}\{Y_{\text{D}}-1, Y_{\text{S}}-1+y_{\text{dm}}-y_{\text{sm}}\}$$

$$\text{MAX}\{0, z_{\text{dm}}-z_{\text{sm}}\} \leq z \leq \text{MIN}\{Z_{\text{D}}-1, Z_{\text{S}}-1+z_{\text{dm}}-z_{\text{sm}}\}$$

$$\text{MAX}\{0, t_{\text{dm}}-t_{\text{sm}}\} \leq t \leq \text{MIN}\{T_{\text{D}}-1, T_{\text{S}}-1+t_{\text{dm}}-t_{\text{sm}}\}$$

$$\text{MAX}\{0, b_{\text{dm}}-b_{\text{sm}}\} \leq b \leq \text{MIN}\{B_{\text{D}}-1, B_{\text{S}}-1+b_{\text{dm}}-b_{\text{sm}}\}$$

where  $\text{RESr}\{.\}$ , for  $1 \leq r \leq 5$ , denotes a resampling function, as defined in Annex H, whereby destination image pixels are generated from interpolated source image pixels.

The destination image  $\text{DST}(x, y, z, t, b)$  is only modified by operators within the bounds specified above. All other portions of the destination image remain unchanged if it is an existent Class A image. If the destination image is a Class B or Class C image (void of image data) prior to the operation, pixels in all other portions of the destination image are set to zero value if the destination image is an arithmetic data type or set to the FALSE state if the destination image is a boolean data type. As indicated by the generic definition, the source and destination images are aligned by their match points prior to an image operation. Match point alignment occurs prior to ROI control and image to operator index assignment.

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 destination image, such that the match points of these images are aligned with the match point of the first destination image prior to processing. Figure 6 provides an example of match point translation for the subtraction of a pair of monochrome images. In the example, the difference region is shown cross-hatched.

A ROI that is bound to an image is spatially translated in register with the image to which it is bound. Each translated image and bound ROI control object is resampled according to the global resampling options specified in 4.4.8.

NOTE – The match point control mechanism can be used to extract a lower dimensional subset of an image for processing. For example, by properly choosing the source and destination match points, and by properly selecting the spatial size of the destination image, it is possible to process only a spatial window of each pixel plane of a source image.

#### 4.4.3 ROI control

The PIKS region-of-interest (ROI) control mechanism shall adhere to the following definition.

NOTE – This ROI definition conceptually describes the semantics of the PIKS ROI functionality. There is no requirement that the functionality shall be implemented in the conceptual manner described.

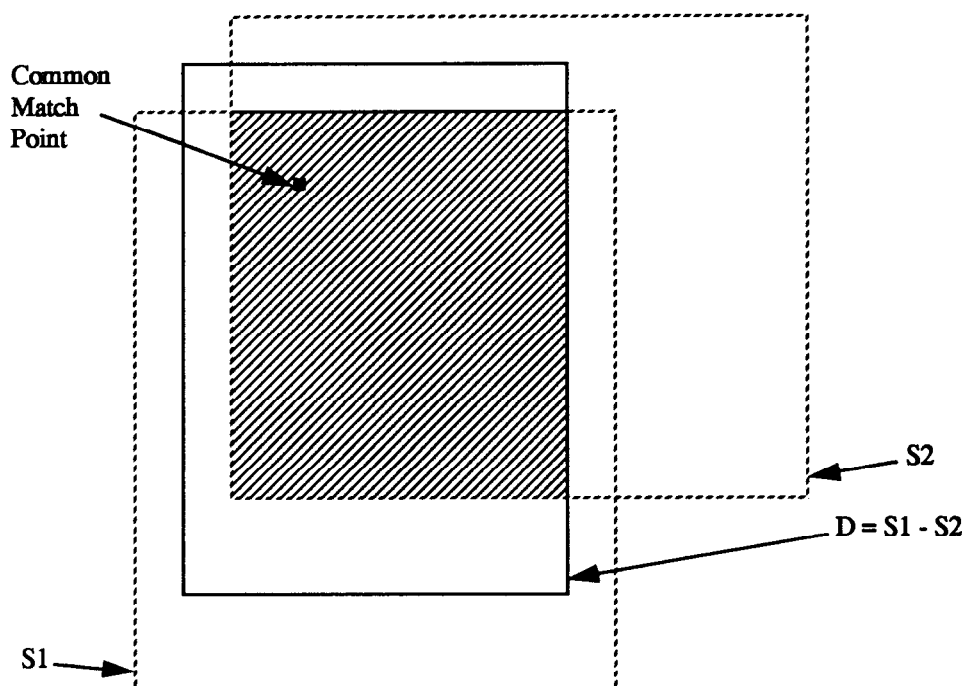


Figure 6 - Example of match point translation for image subtraction

Region-of-interest is a mechanism for controlling which pixels within an image will be processed by an operator, for specifying how they will be processed (for some, but not all operators), and for specifying how the processed pixels will be recorded in a destination image. Conceptually, a ROI control object is a Boolean-valued, five-dimensional array that is associated with each source and destination image. It is of the same size as its associated image, and is in spatial register with respect to the image origin. The ROI control object array assumes one of two logical states at each pixel. If a ROI pixel is in the logical TRUE state, the corresponding image pixel is said to be inside the ROI. Conversely, an image pixel is outside the ROI if its ROI pixel is in the logical FALSE state.

A ROI control object is created, conceptually, by a ROI binding mechanism that binds a ROI data object to an image. The ROI binding mechanism creates a ROI data object identifier reference in the image attribute list associated with each image. ROI data object creation and binding is described in 4.4.4.

If an image attribute list makes no reference to a ROI, a default ROI control object is assumed to exist and to be logically TRUE for each pixel of the image.

NOTE – The ROI control object defined in this clause is used as a means of explaining the operation of ROI control in the PIKS operator model. The ROI control object is not a standardized PIKS data object.

In the PIKS operator model, the source ROI control object is a selector that chooses which source pixels are to be processed. The destination ROI control object is a spatial selector that chooses which processed source pixels are to be recorded in a destination image. The selection operations are defined only at integer pixel coordinates, i.e., PIKS does not support subpixel ROI selection.

The image to operator index assignment operation specified in 4.3.4 applies to a ROI control object bound to an image as well as to the image itself. For example, if the y index of an image is associated with the first operator input index, then the y index of the associated ROI control object is also associated with the first operator input index.

Figure 4 contains a diagram that conceptually describes the ROI operation in the PIKS operator model for image to image operators. In the special case of the processing of multiple source images by a point processing operator to produce a single destination image, let

$\text{SRCp}(x, y, z, t, b) = p\text{-th source image for } 1 \leq p \leq P$

$\text{DST}(x, y, z, t, b) = \text{destination image}$

$\text{P}(x, y, z, t, b) = \text{operator output image without ROI control}$

$\text{O}(x, y, z, t, b) = \text{operator output image when ROI is invoked}$

$\text{ROISp}(x, y, z, t, b) = p\text{-th source ROI control object}$

$\text{ROIS}(x, y, z, t, b) = \text{composite source ROI control object}$

$\text{ROID}(x, y, z, t, b) = \text{destination ROI control object}$

A composite source ROI control object,  $\text{ROIS}(x, y, z, t, b)$ , is conceptually formed by taking the logical intersection of all of the source ROI control objects. Thus

$$\text{ROIS}(x, y, z, t, b) = \text{ROIS1}(x, y, z, t, b) \cap \text{ROIS2}(x, y, z, t, b) \cap \dots \cap \text{ROISP}(x, y, z, t, b)$$

Conceptually, the composite source ROI control object tags each pixel in the source image as being inside or outside the ROI, and conceptually toggles a source control switch within the operator such that

$$\text{O}(x, y, z, t, b) = \text{P}(x, y, z, t, b)$$

if  $\text{ROIS}(x, y, z, t, b)$  is TRUE. There is no operator output if  $\text{ROIS}(x, y, z, t, b)$  is FALSE.

Conceptually, the destination ROI control object toggles a destination control switch such that

$$\text{DST}(x, y, z, t, b) = \text{O}(x, y, z, t, b)$$

if  $\text{ROIS}(x, y, z, t, b)$  is TRUE and if  $\text{ROID}(x, y, z, t, b)$  is TRUE. The destination image remains unchanged if  $\text{ROID}(x, y, z, t, b)$  is FALSE or if  $\text{ROIS}(x, y, z, t, b)$  is FALSE and  $\text{ROID}(x, y, z, t, b)$  is TRUE.

Figure 7 illustrates three cases of ROI functionality for point processing of a monochrome image. In Case 1, the destination ROI is logically TRUE over the full image extent, and the source ROI is TRUE over a 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 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 extends 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.

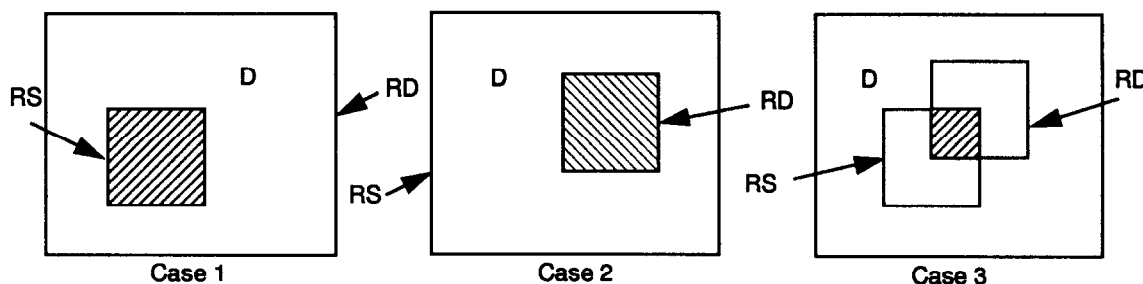


Figure 7 – Examples of ROI operation

The ROI concept applies to neighbourhood operators as well as point operators. Each neighbourhood processing object, such as an impulse response array or a morphological structuring element, 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.

Some PIKS operators are under match point control. For such operators, the source and destination images and their registered ROI control objects are translated such that their match points are in alignment prior to ROI processing. Image processing operations are valid only in the intersection region of the translated source and destination images and their corresponding ROIs.

When a ROI control object is translated under match point control along with the image to which it is bound, the ROI resampling option invoked is applied to the ROI control object. For resampling, the ROI control object is conceptually treated as a bi-level image in which the TRUE state pixels are represented as value 1.0 and FALSE state pixels are represented as value 0.0. After the resampling interpolation operation is completed, the interpolated ROI control object is thresholded at a level of 0.5 for which the equal or above threshold value is set TRUE and the below threshold value is set FALSE. ROI resampling is defined mathematically in Annex H.

Some PIKS operators, such as the unitary transform operators, are not subject to ROI control because they are only defined over rectangular extent images. It is noted in the specification of an operator if it is or is not subject to ROI control.

In order to simplify the mathematical definition of the operators in clause 7, they are expressed as if there were no ROI control.

Some PIKS operators, e.g., rotate, can perform the same processing operation on a source ROI control object as on a source image. For such operators, the existing destination ROI control object can be replaced by the processed source ROI control object. The Element Functionality list associated with each PIKS element indicates by “yes” or “no” if the element supports ROI processing. The processing of a source ROI control object and subsequent replacement of the existing destination ROI control object can be suppressed by the *set\_globals* system management mechanism described in 4.4.8.

NOTE – In PIKS, ROI control is only defined conceptually. Some implementations may choose to implement ROIs as physical bitmaps. Other implementations may choose to implement ROIs as physical coordinate lists or as run length code generators.

#### 4.4.4 ROI data object creation and manipulation

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

| Name                  | Code      |
|-----------------------|-----------|
| array                 | ROI_ARRAY |
| coordinate collection | ROI_COORD |
| elliptical shape      | ROI_ELLIP |
| polygonal shape       | ROI_POLY  |
| rectangular shape     | ROI_RECT  |

These tools, together with the ROI binding mechanism, 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 conceptually replicated over the other pixel planes of a volume image to obtain a cylinder-shaped ROI. The specific functionality of the ROI tools is defined in clause 7.

Conceptually, a ROI data object is a Boolean-valued, five-dimensional array. The ROI data object array assumes one of two logical states (TRUE or FALSE) at each point in the array. Figure 8 presents an example of the spatial cross-section of an elliptical ROI data object. The TRUE state region is shown cross-hatched.

In ISO/IEC 12087-1, a ROI data object is expressed, notationally, as a five-dimensional array:

ROI(x, y, z, t, b)

The PIKS ROI data objects are specified in ISO/IEC 12087-1. Associated with each image is a key point 5-tuple (xk, yk, zk, tk, bk) 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 8 shows an example of the spatial relationship between an ROI data object and an image to which it is bound.

NOTE – The key point may lie outside the extent of the ROI data object and the image.

PIKS provides utilities for converting a Boolean image to a ROI array data object and for converting a ROI array data object to a Boolean image.

PIKS provides operators for the processing of ROI data objects. For example, a rectangular ROI data object can be resized to produce another rectangular-shaped ROI data object. Such operators can accept as a source ROI any of the ROI data object structures, e.g., ROI\_ARRAY, ROI\_RECT, etc., but the operator output is specified only as being a generic ROI structure. The ROI\_GEN structure is implementation-dependent.

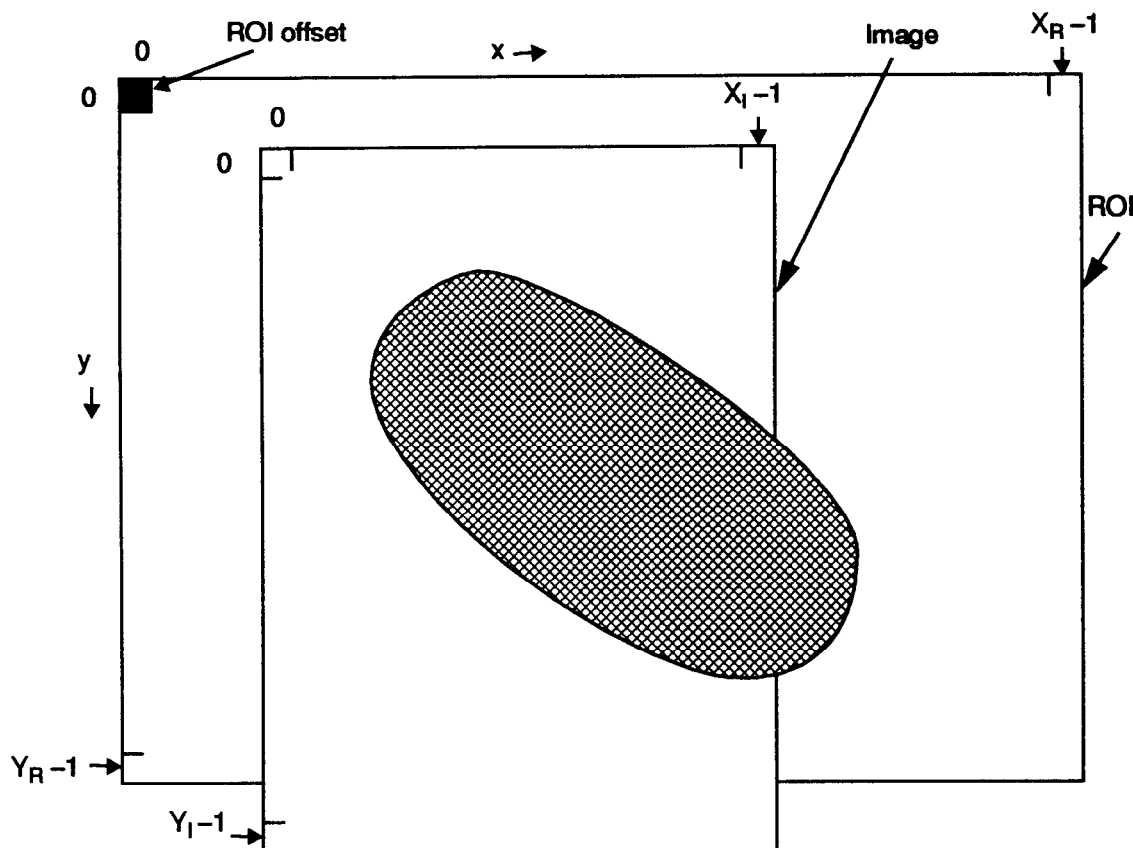


Figure 8 - Example of the relationship between a ROI and an image

#### 4.4.5 Asynchronous control

PIKS provides two global modes in which certain of its elements (operators, tools, utilities, or mechanisms) may be executed: synchronous or asynchronous. The global PIKS mode is initially set to Synchronous by the *open\_piks* mechanism and remains in that state until changed to Asynchronous by invocation of the *set\_globals* mechanism. The Element Functionality list associated with each element indicates by “yes” or “no” if the element supports asynchronous execution.

The following control mechanisms relating to asynchronous execution are available:

*set\_asynchronous* (via the *set\_globals* mechanism)  
*set\_synchronous* (via the *set\_globals* mechanism)  
*synchronize*  
*abort\_asynchronous\_execution*

When PIKS elements are executed synchronously, all processing and error reporting is completed before the implementation returns control to the calling application. This is the least sophisticated mode of operation, and shall be supported by all implementations.

Asynchronous execution permits implementations to use multiple processors or external accelerator devices to take advantage of possible parallel or distributed execution resources. It also allows for some optimization of the execution of sequences of PIKS elements because execution need not begin prior to returning to the application. The implementation may defer execution of a sequence of PIKS elements in

order to combine consecutive PIKS elements into single macro operations or to execute a sequence of operations in a way that reduces loop control overhead, lessens the temporary storage space needed or better allocates available processing facilities to the deferred set of operations. The methods provided to enhance asynchronous operations are strictly internal to the implementation, outside the scope of this standard, and need not be visible to the application.

Asynchronous execution also permits the application to perform other activities, such as numerically intensive processing, the reading and writing of files, input/output operations, or other time consuming processing, while a PIKS implementation is performing the enqueued imaging operations.

The degree of optimization supplied in the asynchronous mode is strictly implementation-dependent. In fact, all conformance profiles allow an implementation to support the asynchronous mode with synchronous execution, without violating conformance to the standard. In this case, all operations in the asynchronous mode would actually be performed in the synchronous mode. An application may perform an *inquire\_piks\_implementation* to verify whether asynchronous operations are supported by a particular PIKS profile.

In the asynchronous mode, each PIKS element is added to the end of a queue of all pending asynchronous PIKS elements. This mechanism is referred to as the Asynchronous Execution Queue. The parameters associated with each enqueued PIKS element are also fetched and saved to prevent subsequent changes by the application from modifying the function to be performed. The queue keeps the elements in the same sequence that they were called by the application. Control may be immediately returned to the application, prior to initiation of execution, without any error checking or reporting. Although the timing of the error detection in parameters of asynchronous calls is implementation-dependent, the resulting error detection and error handling produce the standard-specified results via the PIKS error mechanisms.

A synchronization control mechanism, called *synchronize*, is provided to allow an application to request a wait until all pending asynchronous PIKS elements have been completed. While the *synchronize* function causes the application to wait, it does not affect the execution of asynchronous elements.

Some PIKS elements cause an implicit synchronization. For example, PIKS elements that must return a value of a non-PIKS data type to the application must wait until processing is completed before they can return a result, and then return control to the calling program. Most PIKS operations do not cause implicit synchronization. Because PIKS data types are referenced by identifiers — not values — their values are automatically updated by an asynchronous element, and are available to asynchronous elements invoked later. An invocation of the *set\_globals* mechanism, when the synchronicity mode is set to SYNCHRONOUS, causes an implicit synchronization.

The *close\_piks* mechanism does an implicit synchronization to assure that all operations have been completed prior to the closing actions.

Execution of the enqueued asynchronous operations is controlled by the specific scheduling mechanism provided by the implementation. It must begin and complete execution of all enqueued elements prior to giving a return to any explicit or implicit *synchronize* call. In most cases, PIKS elements will not be executed as fast as they can be added to the queue. In the event that the asynchronous execution control does complete all enqueued elements, it simply waits until additional elements are entered to the queue to resume asynchronous processing.

All PIKS elements which do not cause a synchronization prior to their execution are placed on the Asynchronous Execution Queue. This includes operations that simply change a global state, such as the resampling mode. This will ensure that all operations are executed in the order they were invoked, and that earlier operations properly affect later operations avoiding the side-effects that could be caused by changes in the global state during the execution of asynchronous elements invoked by prior calls.