7

Process Elements

Overview

Process elements have one or more **Phototag** sources and have one output connection. Each process element applies a specific operation to image data within an ***Active*** **Photoflo**. Some process elements perform their operation on a single source of image data (for example, **Geometry**). Others, operate on either a pair of image sources or an image and a constant (for example, **Logical**). **BandCombine** associates bands from three ***SingleBand*** image sources to form a ***TripleBand*** image. **PasteUp** can combine multiple images (tiles) to form a composite image.

The **ConvertToIndex** element can generate an event to notify the client about color allocation results.

A process element will not be involved in the execution of a **Photoflo** if there is no eventual consumer of its output (that is, no terminating export element).

Limiting the Process

Processing may be limited to a subset of the bands present and a subset of pixels within the processed bands. Data falling outside the selection criteria are passed through unaffected by the element's *process*.

Process Selected Bands

Some process elements accept a *band-mask* to restrict processing to a subset of the source data bands. Only bands selected by the *band-mask* are subject to processing. Other bands present in the image are passed through to the output. For example: a *band-mask* of 001_2 indicates that only the "least significant band" would be *processed* (see **Orientation** for a definition of band ordering).

Process Intersecting Pixels

Dyadic process elements can accept a pair of data sources. These sources are spatially registered at their origins (upper left corners). Their *class* and *type* must match, but their dimensions (*width* and *height*) need not match. The dimensions of the output are determined by *src-1*. Processing is spatially restricted to the intersection of the sources on a per-band basis. Data from *src-1* that do not intersect with *src-2* are passed through to the output.

Process Selected Pixels

Some process elements accept a *processing domain* (**ProcessDomain**) to restrict processing to a subset of the source data pixels. The processing domain represents either a *list-of-rectangles* or a bi-level ***SingleBand*** *control-plane*.

A processing domain contains an *x,y* coordinate pair to spatially position the overall processing domain relative to the origin of the source(s). It also contains a **Phototag** that references the import or process element that supplies the processing domain data. The element referenced by the **Phototag** may provide the following:

- ***list-of-rectangles*** — Imported using **ImportROI** or **ImportClientROI**. Processing is restricted to the intersection of the data source(s) and any rectangle within the list.
- ***control-plane*** — Imported using **ImportDrawable**, **ImportDrawablePlane**, **ImportPhotomap**, or **ImportClientPhoto**; or the output of an upstream process element (for example, **Compare**). Processing is restricted to the intersection of the data source(s) and nonzero locations within the *control-plane*.

Processing is not restricted if **Phototag** zero (0) is specified as the process domain's data source.

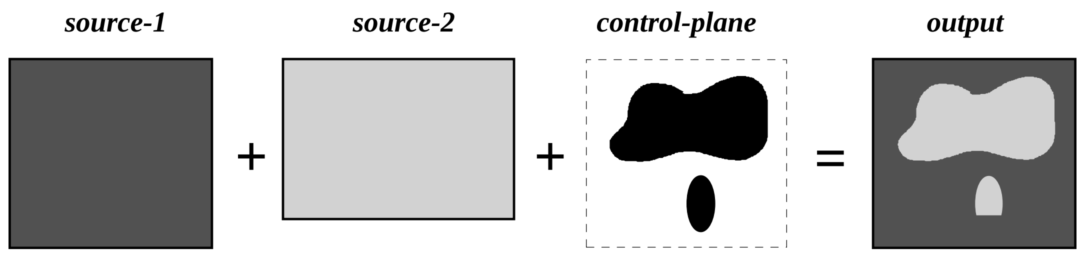

Figure 7-1 Combining two sources using a control plane – Logical: *Copy*

Data Types

In XIE, an image consists of one or three arrays of pixels, each of which is measured in terms of its width, height, and depth. Each array, which represents a chromatic band of image data, is dimensioned independently. The depth component of each array is expressed in quantization *levels* or the potential dynamic range of the pixels. All process elements within XIE are capable of operating on image data that have a fixed number of quantization levels. These data are said to be ***Constrained*** such that the minimum possible intensity value of a pixel is zero (0), and the maximum possible intensity level is the number of quantization levels minus one.

With the exception of operations that specifically change the number of quantization levels (for example, **Dither**) all other operations that are performed on ***Constrained*** data will maintain the *levels* attribute from input to output. An operation that computes a negative pixel value will set the resulting pixel to zero. Likewise, an operation that computes a pixel value that would exceed the *levels* attribute of the data will saturate the resulting pixel at its maximum allowable value (that is, *levels*-1). This method of constraining image data is equivalent to the ***HardClip*** **Constrain** technique.

For some operations (for example, **Arithmetic** operations), it is desirable to allow pixel values to stray beyond their ***Constrained*** values. Such data are said to be ***Unconstrained*** and may contain negative values, fractional values, or other values that might be beyond the normal display capabilities of a frame buffer. Performing a sequence of image operations on ***Unconstrained*** image data allows the maximum degree of precision to be maintained throughout the process (although there may be a cost in performance, depending on the server's capabilities). Several XIE process elements are capable of operating on either ***Constrained*** or ***Unconstrained*** data. These elements always output the same **DataType** as they are fed at their inputs. An exception to this rule is that certain colorspace conversion techniques can accept either **DataType** but are only capable of outputting ***Unconstrained*** data.

XIE provides a complementary pair of elements for converting image data between these two processing styles. The **Unconstrain** element simply removes the normal *levels* restrictions from the data. The **Constrain** element provides a means of bringing the data back into a ***Constrained*** range. A variety of techniques are available for this purpose that allow the pixel values to be level-shifted, scaled, clipped, rounded, and so on

Process Categories

- Area elements
  - **Convolve** — Specifies an area of input pixels is involved in producing each output pixel (controlled by a convolution kernel).
- Format conversion elements
  - **ConvertFromIndex** — Specifies convert *index* data to *achromatic* or *trichromatic* data.
  - **ConvertFromRGB** — Specifies convert *RGB* data to another *trichromatic* colorspace.
  - **ConvertToIndex** — Specifies convert *achromatic* or *trichromatic* data to *index* data.
  - **ConvertToRGB** — Specifies convert non-RGB *trichromatic* data to *RGB* data.
- Dyadic elements: process either two images or an image and a constant
  - **Arithmetic** — Specifies an arithmetic operation is performed between two images or an image and a constant.
  - **Blend** — Specifies combine two images or an image and a constant.
  - **Compare** — Specifies two images or an image and a constant are compared to generate a Boolean bitmap result.
  - **Logical** — Specifies perform logical bit-wise operations on an image, between two images, or between an image and a constant.
- Geometric elements
  - **Geometry** — Specifies general affine transform (provides crop, mirror, scale, shear, rotate, translate, and combinations thereof).
  - **PasteUp** — Specifies an N-input translate operation that reconstructs an image from various source tiles.
- Point elements
  - **Math** — Specifies apply a mathematical operation to each pixel.
  - **Point** — Specifies remap each pixel value through a lookup table.
- Radiometric elements
  - **BandCombine** — Specifies merge three ***SingleBand*** images.
  - **BandExtract** — Specifies extract ***SingleBand*** data from a trichromatic source.
  - **BandSelect** — Specifies select a single band of data from a trichromatic source.
  - **Constrain** — Specifies constrain an image to a fixed number of quantization levels.
  - **Dither** — Specifies reduce image quantization levels through an area technique.
  - **MatchHistogram** — Specifies transform an image to have a specific histogram shape.
  - **Unconstrain** — Specifies remove the normal *levels* restrictions from the image data.

Process Definitions

The complete list of *process* **PhotoElement**s follows in alphabetical order.

Arithmetic

XieFloArithmetic

src-1: XieTypPhototag
src-2: XieTypPhototag
domain: XieTypProcessDomain
constant: XieTypConstant
operator: XieTypArithmeticOp
band-mask: CARD8

Errors: ***FloAlloc, FloSource, FloDomain, FloMatch, FloOperator, FloValue***
Events: none

Attribute	Value
class	Same as *src-1*
type	Same as *src-1*
width	Same as *src-1*
height	Same as *src-1*
levels	Same as *src-1*

Arithmetic produces output data by performing an addition, subtraction, minimum, or maximum operation between two data sources or between a single data source and a constant. Furthermore, multiplication, division, or gamma correction may by applied to a single data source.

When two sources are involved, *src-1* and *src-2* are the **Phototag**s of the elements supplying source data (*constant* is ignored). If the operation is to involve a constant, *src-1* is one operand, *src-2* must be zero, and *constant* is used as the other operand. *Domain* may control the subset of source data that will be operated upon. *Operator* is the arithmetic operation to be performed. *Band-mask* specifies which bands are to be operated upon.

When two sources are involved, all attributes, other than *width* and *height*, must match; all output attributes are inherited from *src*-1.

Pixel computations that would lead to errors will yield valid server-dependent values (for example, dividing by a ***Constrained*** pixel value of zero might result in a value of *levels*-1) . Using *band-mask* to select source data that have two (2) or less *levels* is not permitted.

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src-1* or *src-2* *src-2* has been specified with a monadic *operator* (for example, ***Div*** or ***Gamma***)
FloDomain	Invalid *domain*
FloMatch	*class*, *type*, or *levels* differs between *src-1* and *src-2* Selected source data are bitonal (that is, 2 or less *levels*)
FloOperator	Invalid *operator*

BandCombine

XieFloBandCombine

src-1: XieTypPhototag
src-2: XieTypPhototag
src-3: XieTypPhototag

Errors: ***FloAlloc, FloSource, FloMatch***
Events: none

Attribute	Value
class	***TripleBand***
type	Same as *src-1*
width	Same as *src*s
height	Same as *src*s
levels	Same as *src*s

BandCombine merges three ***SingleBand*** data sources to produce a ***TripleBand*** result.

Src-1, *src-2*, and *src-3* are the **Phototag**s of the elements supplying source data. All three sources must be of the same *type*, and each source must be ***SingleBand***. Other attributes that are taken from the individual sources may differ. The output will be ***TripleBand***.

No subsetting by **ProcessDomain** is provided.

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src-1*, *src-2*, or *src-3*
FloMatch	A source has more than one band *type* differs between sources

BandExtract

XieFloBandExtract

src: XieTypPhototag
coefficients: XieTypConstant
bias: XieTypFloat
levels: CARD32

Errors: ***FloAlloc, FloSource, FloMatch***
Events: none

Attribute	Value
class	***SingleBand***
type	Same as *src*
width	Same as *src*
height	Same as src
levels	*levels* or unknown (see text)

BandExtract produces ***SingleBand*** output data from a ***TripleBand*** source by multiplying a pixel value from each source band by its corresponding *coefficient* and then summing the results with the *bias* value.

Src is the **Phototag** of the element supplying source data. *Coefficients* is a three (3) element array that determines the proportion of each source band pixel that is used to form the output. The *bias* value is added to each output pixel. *Levels* is used as the *levels* attribute of the output data if the *src* data are ***Constrained***; otherwise, it is ignored.

The source data must be ***TripleBand***, and all bands must have equal dimensions. The output data will be ***SingleBand*** with *levels* taken from the *levels* parameter if *type* is ***Constrained***. All other attributes are inherited from *src*.

No subsetting by **ProcessDomain** is provided (the entire image is processed).

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src*
FloMatch	*src* is not ***TripleBand*** Unequal inter-band dimensions (*width* or *height*)

BandExtract can be used to extract a band of luminance data from an RGB image. It can also be used to compute `COLORMAP` indices in cases where the contents of the `COLORMAP` is well ordered (for example, a `Standard-COLORMAP` or a `TrueColor visual`).

BandSelect

XieFloBandSelect

src: XieTypPhototag
band-number: CARD8

Errors: ***FloAlloc, FloSource, FloMatch, FloValue***
Events: none

Attribute	Value
class	***SingleBand***
type	Same as *src*
width	Same as band selected from *src*
height	Same as band selected from *src*
levels	Same as band selected from *src*

BandSelect produces ***SingleBand*** output data by selecting a single band from a ***TripleBand*** source.

Src is the **Phototag** of the element supplying source data. *Band-number* specifies which *src* band is to be selected to provide the output data.

No subsetting by **ProcessDomain** is provided.

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src*
FloMatch	*src* is not ***TripleBand***
FloValue	Invalid *band-number*

Blend

XieFloBlend

src-1: XieTypPhototag
src-2: XieTypPhototag
alpha: XieTypPhototag
constant: XieTypConstant
alpha-const: XieTypFloat
domain: XieTypProcessDomain
band-mask: CARD8

Errors: ***FloAlloc, FloSource, FloDomain, FloMatch, FloValue***
Events: none

Attribute	Value
class	Same as *src-1*
type	Same as *src-1*
width	Same as *src-1*
height	Same as *src-1*
levels	Same as *src-1*

Blend produces output data from two data sources or a single data source and a constant. Each output pixel is a percentage combination of the source values, as controlled by the alpha input or the alpha constant.

When two sources are involved, *src-1* and *src-2* are the **Phototag**s of the elements supplying source data (*constant* is ignored). If the operation is to involve a constant, *src-1* is one operand, *src-2* must be zero, and *constant* is used as the other operand. If *alpha* is nonzero, it controls the blend proportion for each pixel that is processed; otherwise, *alpha-const* provides this control. *Domain* may control the subset of source data that will be operated upon. *Band-mask* specifies which bands are to be operated upon.

When two sources are involved, all attributes, other that *width* and *height*, must match. If *alpha* is nonzero, it must be a source of ***Constrained*** data. Using *band-mask* to select source data that have two (2) or less *levels* is not permitted.

Within the intersection of the source(s) and *domain*, each output pixel is a blend of the corresponding pixels from *src-1* and *src-2* (or *src-1* pixels blended with *constant*). The degree of blend is determined by the corresponding value taken from *alpha* or the value of *alpha-const*. If *alpha* is nonzero, the proportion of blend is further scaled by *alpha-const*:

output = *src*-1 * (1 - *alpha* / *alpha-const*) + *src*-2 * (*alpha* / *alpha-const*)

(where *alpha-const* is greater than 0.0).

If *alpha* is zero:

output = *src*-1 * (1 - *alpha-const*) + *src*-2 * *alpha-const*

(where *alpha-const* is in the range [0.0, 1.0]).

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src-1*, *src-2*, or *alpha*
FloDomain	Invalid *domain*
FloMatch	Incompatible attributes between *src-1* and *src-2*
	alpha is ***Unconstrained***
	Selected source data are bitonal (that is, 2 or less *levels*)
FloValue	*alpha* is zero and *alpha-const* is outside the range [0.0, 1.0]
	alpha is nonzero and *alpha-const* is zero or negative

Compare

XieFloCompare

src-1: XieTypPhototag
src-2: XieTypPhototag
domain: XieTypProcessDomain
constant: XieTypConstant
operator: XieTypCompareOp
combine: BOOL
band-mask: CARD8

Errors: ***FloAlloc, FloSource, FloDomain, FloMatch, FloOperator***
Events: none

Attribute	Value
class	See text
type	***Constrained***
width	Same as *src-1*
height	Same as *src-1*
levels	2 per band (see text)

Compare takes two data sources or a single data source and a constant and generates a Boolean bitmap output that reflects the results of a point-wise comparison. The output data has a value of one (1) wherever the comparison is true, and a value of zero (0) everywhere else (that is, comparison false or comparison not performed). **Compare** may be requested on a per-band basis, or for all bands taken together.

When two sources are involved, *src-1* and *src-2* are the **Phototag**s of the elements supplying source data (*constant* is ignored). If the operation is to involve a constant, *src-1* is one operand, *src-2* must be zero, and *constant* is used as the other operand. *Domain* may control the subset of source data that will be compared. *Operator* is the logical predicate operator used in the comparison. *Combine* is a Boolean that determines if the comparison should be done on a per-band basis or on an all bands basis. *Band-mask* specifies which bands are to be involved.

If *combine* is true or *src-1* is ***SingleBand***, the output data will form a single Boolean bitmap. If *src-1* is ***TripleBand*** and *combine* is false, the output data will yield three separate Boolean bitmaps (for this case *band-mask* must specify all bands).

If *src-1* is ***TripleBand*** and *combine* is true, only the ***EQ*** and ***NE*** *operator*s are allowed; equality is established for each band selected by *band-mask*, and then the result is logically *AND*ed to derive equality (inequality is a logical *NOT* of this result). For this case, *width* and *height* must match for all bands selected by *band-mask*.

combine	**input *class***	***band-mask***	**output *class***
True	***SingleBand***	n/a	***SingleBand***
	TripleBand	selected bands	***SingleBand***
False	***SingleBand***	n/a	***SingleBand***
	TripleBand	all bands	***TripleBand***

Table 7-1 Compare parameter and DataClass dependencies

When two sources are involved, all attributes, other than *width* and *height*, must match. The comparison is performed over the intersect of the source(s) as restricted by *domain*. All points not compared are given the value zero (0).

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src-1* or *src-2*
FloDomain	Invalid *domain*
FloOperator	Invalid *operator*
FloMatch	Class differs between *src-1* and src-2 Invalid combination of *operator* and *combine* ***TripleBand***, and *combine* is false, and *band-mask* incomplete

Compare does not support inter-band distance metrics therefore, *combine*d ***LT***, ***LE***, ***GT***, and ***GE*** *operator*s are not supported. Clients may implement these using other elements in concert with **Compare**.

Typical uses for **Compare** include: creating data to be used as a *control-plane* for a **ProcessDomain** and converting index data into bitonal image data.

Constrain

XieFloConstrain

src: XieTypPhototag
levels: XieTypLevels
constrain: XieTypConstrainTechnique
constrain-params: <technique-dependent>

Errors:*: FloAlloc, FloSource, FloTechnique*
Events: none

Attribute	Value
class	Same as *src*
type	***Constrained***
width	Same as *src*
height	Same as *src*
levels	*levels*

Constrain applies a quantization model to the image data to convert the data to a fixed number of quantization levels. Application of the quantization model may involve steps such as range shifting, scaling, clipping, and rounding.

Src is the **Phototag** of the element supplying source data. *Level*s is the number of quantization levels desired in the output data. *Constrain* specifies the **ConstrainTechnique** to be used when constraining the data. *Constrain-params* is the list of additional parameters required by *constrain*.

If the input image is already ***Constrained*** the data will be re***-Constrained***.

No subsetting by band mask or **ProcessDomain** is provided (the entire image is ***Constrained***).

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src*
FloTechnique	Invalid *constrain* technique or *constrain-params*

ConvertFromIndex

XieFloConvertFromIndex

src: XieTypPhototag
colormap: `COLORMAP`
class: XieTypDataClass
precision: `CARD8`

Errors: ***FloAlloc, FloSource, FloColormap, FloMatch, FloValue***
Events: none

Attribute	Value
class	*class*
type	***Constrained***
width	Same as *src*
height	Same as *src*
levels	$2^{precision}$ (per band)

ConvertFromIndex converts `COLORMAP` index data into achromatic or trichromatic data.

Src is the **Phototag** of the element supplying source data. *Colormap* is the `COLORMAP` from which to obtain the value for each output pixel. *Class* specifies the **DataClass** of the desired output data. *Precision* specifies how many bits (most significant) are to be used from the red, green, and blue values found in *colormap*.

If *class* is ***SingleBand*** and a trichromatic *colormap* is specified (*`StaticColor`*, *`PseudoColor`*, *`TrueColor`*, or *`DirectColor`*), the gray shade for each pixel is taken from the *red* values in *colormap*. If *class* is ***TripleBand*** and an achromatic *colormap* is specified (*`StaticGray`* or *`GrayScale`*), the *red* band will be replicated to populate the *green* and *blue* output bands.

The depth of *colormap* must match the *levels* attribute of *src* (that is, 2^{depth} must equal *levels*).

No subsetting by band mask or **ProcessDomain** is provided (the entire image is converted).

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src*
FloColormap	Invalid *colormap*
FloMatch	*levels* of *src* do not match depth of *colormap*
FloValue	Invalid *class* or *precision*

To achieve predictable results, the client should not modify *colormap* cells referenced by the source data while the **Photoflo** is ***Active*** (see **Await** for synchronization help).

A value of *bitsPerRGBValue* or less (defined in the `VISUAL` of *colormap*) is usually appropriate for *precision*.

Compare can be used to convert index data into bitonal data. Use **ConvertFromIndex** followed by **BandExtract** to convert *colored* index data into full fidelity achromatic data.

ConvertFromRGB

XieFloConvertFromRGB

src: XieTypPhototag
convert: XieTypConvertFromRGBTechnique
convert-params: <technique-dependent>

Errors:*: FloAlloc, FloSource, FloMatch, FloTechnique*
Events: none

Attribute	Value
class	***TripleBand***
type	*convert* technique dependent
width	Same as *src*
height	Same as *src*
levels	*convert* technique dependent

ConvertFromRGB converts RGB source data to an alternate colorspace.

Src is the **Phototag** of the element supplying source data (RGB assumed). *Convert* is the **ConvertFromRGBTechnique** that will be used in the conversion to the destination colorspace. *Convert-params* is the list of additional parameters required by *convert*.

In addition to colorspace conversion, some techniques allow for adjusting the white point of the output data (see Appendix A, "**Convert From RGB Technique,**" for more information on conversion techniques).

The source data must be ***TripleBand***, and all bands must have equal dimensions. The *type* and *levels* of the output data are determined by *convert* technique parameters. All other attributes are inherited from *src*.

No subsetting by **ProcessDomain** is provided (the entire image is processed).

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src*
FloMatch	*src* is not ***TripleBand*** Unequal inter-band dimensions (*width* or *height*)
FloTechnique	Invalid *convert* or *convert-params*

ConvertToIndex

XieFloConvertToIndex

notify: `BOOL`
src: XieTypPhototag
colormap: `COLORMAP`
color-list: XieTypColorList
color-alloc: XieTypColorAllocTechnique
color-alloc-params: <technique-dependent>

Errors: ***FloAlloc, FloSource, FloColormap, FloColorList, FloMatch, FloValue***
Events: none

Attribute	Value
class	***SingleBand***
type	***Constrained***
width	Same as *src*
height	Same as *src*
levels	2^{depth} (that is, *colormap* depth)

ConvertToIndex allocates and/or matches colors or gray shades, as required, from a `COLORMAP`. It produces pixel indices as output data, and records indices that it allocates in a **ColorList**. The specified *color-alloc* technique may generate a **ColorAlloc** event to warn the client that results are of lesser fidelity than desired.

Notify allows the client to be notified about inferior results from color allocation or matching. *Src* is the **Phototag** of the element supplying ***Constrained*** source data**.** Colormap is the `COLORMAP` from which colors or gray shades are allocated and/or matched. *Color-list* is the **ColorList** where allocated `COLORMAP` indices are to be stored. *Color-alloc* specifies the desired **ColorAllocTechnique**. *Color-alloc-params* is the list of additional parameters required by *color-alloc*.

Color-list is purged of any colors it already contains when **Photoflo** execution begins. Allocated `COLORMAP` indices can be freed using **PurgeColorList**, **DestroyColorList**, or by making *color-list* the target of an ***Active*** **Photoflo**.

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src*
FloColormap	Invalid *colormap*
FloColorList	Invalid *color-list*
FloAccess	*color-list* is already being used by another ***Active*** **Photoflo**
FloMatch	Unequal inter-band dimensions (*width* or *height*) or ***Unconstrained*** *src* data
FloTechnique	Invalid *color-alloc* technique or *color-alloc-params*

ConvertToIndex is usually a preparatory step for an **ExportDrawable** element. Alternatively, continuous-tone data can be converted to a fixed palette of `COLORMAP` index entries using **Point**. Or **BandExtract** can be used if the `COLORMAP` contains a ramp of colors or gray shades (for example, a `Standard-COLORMAP` or a `TrueColor visual`). Bitonal image data can be presented directly to **ExportDrawablePlane**, allowing the `GCONTEXT` mechanism to translate ones and zeros in the image into `foreground` and `background` colors in the `DRAWABLE`.

If *colormap* is `static`, image colors or gray shades must be matched to those available in *colormap*. Because no cells are actually allocated, *color-list* can be specified as zero (0). Of the three standard techniques defined in this document only, ***ColorAlloc_Match*** is appropriate for `static` `COLORMAP`s. If *colormap* is `dynamic`, image colors or gray shades may be matched or allocated from *colormap* depending on the *color-alloc* technique; *colormap* and the list of cells actually allocated are stored in *color-list*.

Although the three standard techniques defined in this document allocate read-only cells, **ConvertToIndex** could be extended with private techniques that allocate, match, or write into read-write cells.

ConvertToRGB

XieFloConvertToRGB

src: XieTypPhototag
convert: XieTypConvertToRGBTechnique
convert-params: <technique-dependent>

Errors: ***FloAlloc, FloSource, FloTechnique***
Events: none

Attribute	Value
class	***TripleBand***
type	*convert* technique dependent
width	Same as *src*
height	Same as *src*
levels	*convert* technique dependent

ConvertToRGB converts alternate colorspace source data into RGB data.

Src is the **Phototag** of the element supplying source data. *Convert* is the **ConvertFromRGBTechnique** that will be used in the conversion from the source colorspace. *Convert-params* is the list of additional parameters required by *convert*.

In addition to colorspace conversion, some techniques allow for adjusting the white point of the source data prior to conversion and compressing the gamut of the output data (see Appendix A, "**Convert To RGB Technique,**" for more information on conversion techniques).

The source data must be ***TripleBand***, and all bands must have equal dimensions. The *type* and *levels* of the output data are determined by *convert* technique parameters. All other attributes are inherited from *src*.

No subsetting by band mask or **ProcessDomain** is provided (the entire image is converted).

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src*
FloMatch	*src* is not ***TripleBand***
	Unequal inter-band dimensions (*width* or *height*)
FloTechnique	Invalid *convert* or *convert-params*

Arithmetic or **Point** can be used to gamma-correct the resulting data prior to displaying it.

Convolve

XieFloConvolve

src: XieTypPhototag
domain: XieTypProcessDomain
kernel: LISTofXieTypFloat
kernel-size: CARD8
band-mask: CARD8
convolve: XieTypConvolveTechnique
convolve-params: <technique-dependent>

Errors: ***FloAlloc, FloSource, FloDomain, FloMatch, FloValue, FloTechnique***
Events: none

Attribute	Value
class	Same as *src*
type	Same as *src*
width	Same as *src*
height	Same as *src*
levels	same as *src*

Convolve produces output data by convolving each input pixel value (and surrounding area) with the specified convolution kernel.

Src is the **Phototag** of the element supplying source data. *Domain* may control the subset of the image that will be operated upon. *Kernel* contains the coefficients used in the convolution process. *Kernel-size* specifies the dimension of *kernel*. *Band-mask* specifies which bands are to be operated upon. *Convolve* specifies the **ConvolveTechnique** for handling edge conditions (when *kernel* requires data outside of *src*). *Convolve-params* is the list of additional parameters required by *convolve*. *Kernel* represents a square array of float data that has odd dimensions. Thus, a single dimension is used to specify, *kernel-size*.

Using *band-mask* to select source data that have two (2) or less *levels* is not permitted.

All output data attributes are inherited from the source data.

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src*
FloDomain	Invalid *domain*
FloMatch	Selected source data are bitonal (that is, 2 or less *levels*)
FloValue	Invalid *kernel-size* (for example, not odd)
FloTechnique	Invalid *convolve* edge handling technique or *convolve-params*

For ***TripleBand*** data, convolution using the same kernel is permuted over all three bands.

Dither

XieFloDither

src: XieTypPhototag
levels: XieTypLevels
band-mask: CARD8
dither: XieTypDitherTechnique
dither-params: <technique-dependent>

Errors: ***FloAlloc, FloSource, FloValue, FloMatch, FloTechnique***
Events: none

Attribute	Value
class	Same as *src*
type	***Constrained***
width	Same as *src*
height	same as *src*
levels	*levels*

Dither is used to reduce the number of quantization levels in an image. It accomplishes this by affecting adjacent pixels (area affect) to make up for the lack of depth resolution.

Src is the **Phototag** of the element supplying source data. *Levels* is the number of levels desired in the output data. *Band-mask* specifies which bands are to be operated upon. *Dither* specifies the desired **DitherTechnique**. *Dither-params* is the list of additional parameters required by *dither*.

The source data must be ***Constrained***. Using *band-mask* to select source data that have two (2) or less *levels* is not permitted.

No subsetting by band mask or **ProcessDomain** is provided (the entire image is processed).

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src*
FloMatch	***Unconstrained*** *src* data Selected source data are bitonal (that is, 2 or less *levels*)
FloValue	Invalid output *levels*: less than two, or greater than *src levels*
FloTechnique	Invalid *dither* technique or *dither-params*

Geometry

XieFloGeometry

src: XieTypPhototag
width: CARD32
height: CARD32
coefficients: XieTypFloat[6]
constant: XieTypConstant
band-mask: CARD8
sample: XieTypGeometryTechnique
sample-params: <technique-dependent>

Errors: ***FloAlloc, FloSource, FloValue, FloTechnique***
Events: none

Attribute	Value
class	Same as *src*
type	Same as *src*
width	*width*
height	*height*
levels	Same as *src*

Geometry is used to perform geometric transformations on image data. Linear geometric resampling operations are implemented, such as, crop, mirror, scale, shear, rotate, translate, and combinations thereof.

Src is the **Phototag** of the element supplying source data. *Width* and *height* specify the dimensions of the output data. *Coefficients* is a list of values (a,b,c,d,t_x,t_y) that control the geometric transformation. *Constant* is a fill value used for output pixels that do not map back to a *src* pixel. *Band-mask* specifies which bands are to be operated upon. *Sample* is the **GeometryTechnique** to be used for retrospectively resampling *src*. *Sample-params* is the list of additional parameters required by *sample*. Figure 7-2 shows a combined crop and scale for a *src* with dimensions **w** and **h**, and output *width* **w'** and *height* **h'**. The mapping between the coordinate systems is specified from output back to *src*; that is, for each output pixel (**x'**,**y'**), the coordinate location (**x**,**y**) at which to sample *src* is given by:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Geometry can be visualized as stepping through each possible output pixel location in turn and computing the location from which to obtain each input pixel value. Often a given output pixel location (**x'**,**y'**) will not correspond exactly to a single pixel in the input image. The *sample* technique is used to determine how the input data will be interpolated to produce each output pixel value.

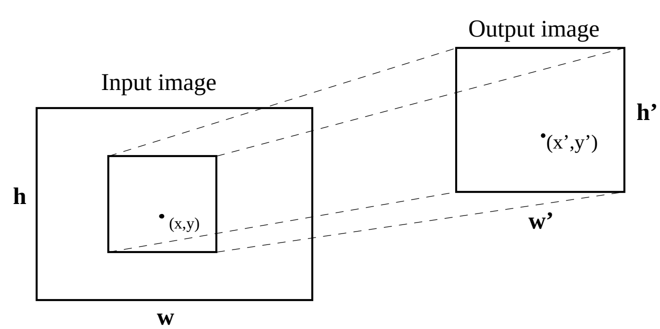

Figure 7-2 A sample geometric transform: crop and scale

The region to be cropped in the input image is *implicitly* defined by the dimensions of the output image and the mapping from output to input coordinates. Depending on the size of the input and output images, the amount of scaling specified, and the amount of translation in the mapping, the "shadow" of the output image may extend beyond the boundaries of the input image, as in Figure 7-3. Pixels in the output image that map off the edge of the input image will be assigned the *constant* value.

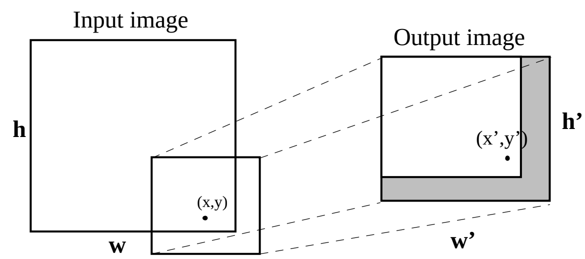

Figure 7-3 Background fill used for pixels beyond the edge

Transformation Coefficients

The coordinate mapping (a,b,c,d,t_x,t_y), together with the output *width* and *height*, fully specify the geometric transformation. To aid in computing the desired transform, one might:

- Draw a picture of the input and output images. Label the corners of the output image A,B,C,D. They will have coordinates x',y' at (0,0), (width-1,0), (0,height-1), and (width-1,height-1),
- Next compute or deduce the coordinates (x,y) of A,B,C,D in the input coordinate space,
- For three corners (preferably those with a 0 for either x' or y'), write down the basic mapping equation with known (x,y) and (x',y') specified. Each corner will give two equations, and simultaneously solving the six equations for the three corners will allow (a,b,c,d,t_x,t_y) to be deduced. The remaining corner may be used to check that these parameters have been derived correctly.

The following briefly (and approximately) summarizes the intuitive role of each parameter:

a,d	Scaling parameters. Increasing a and d will make the output image appear smaller, whereas decreasing them will make the output pixels appear larger.
b,c	Rotation/skew parameters. If b and c are zero, the output image will be a rectangular scaling of the input image. If a and d are both zero, b is one and c is negative one, the image will appear rotated. The magnitude of b and c will affect scaling as well, if a and d are zero. If more than two of (a,b,c,d) are nonzero, the effect is complicated. The image may appear skewed *and* scaled.
t_x,t_y	Translation parameters. Used to specify the offset between input and output coordinate systems.
width,*height*	These specify the output image dimensions of the selected band(s). Note that increasing the output image *height* and *width* over the input image size will not by itself cause magnification — if a and d are one (1) and b and c are zero (0), the output image will have the same appearance as the input, except that the borders will shrink or expand (as determined by *width* and *height*) and new areas of the image will be filled with *constant*.

Trichromatic image bands can be operate individually, all together or in any combination using *band-mask*. Because applying the same (a,b,c,d,t_x,t_y) mapping to inputs with diverse sizes will specify different transformations, operating on all bands in unison (*band-mask* of 111_2) is most appropriate if the dimensions of all bands are equal. No subsetting by **ProcessDomain** is provided (selected bands are processed in their entirety).

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src*
FloValue	Invalid *coefficients, width,* or *height*
FloTechnique	Invalid *sample* technique or *sample-params*

Logical

XieFloLogical

src-1: XiePhototag
src-2: XieTypPhototag
domain: XieTypProcessDomain
constant: XieTypConstant
operator: `GCfunction*`
band-mask: `CARD8`

Errors: ***FloAlloc, FloSource, FloDomain, FloMatch, FloOperator, FloValue***
Events: none

Attribute	Value
class	Same as *src-1*
type	***Constrained***
width	Same as *src-1*
height	Same as *src-1*
levels	Same as *src-1*

Logical performs per-pixel bit-wise operations on a single data source, or between two data sources, or between a single data source and a constant.

When two sources are involved, *src-1* and *src-2* are the **Phototag**s of the elements supplying source data (*constant* is ignored). If the operation is to involve a constant, *src-1* is one operand, *src-2* must be zero, and *constant* is used as the other operand. *Domain* may control the subset of source data that will be operated upon. *Operator* is the logical operator to be used. *Band-mask* specifies which bands are to be operated upon.

The value of *operator* matches the `GC-function`[1] values defined by the core protocol specification for `CreateGC`. The output of **Logical** is determined by the number of data sources and *operator*:

GC-function[1]	monadic operation	dyadic operation
Clear	0	0
And	*constant* AND *src-1*	*src-2* AND *src-1*
AndReverse	*constant* AND (NOT *src-1*)	*src-2* AND (NOT *src-1*)
Copy	*constant*	*src-2*
AndInverted	(NOT *constant*) AND *src-1*	(NOT *src-2*) AND *src-1*
NoOp	*src-1*	*src-1*
Xor	*constant* XOR *src-1*	*src-2* XOR *src-1*
Or	*constant* OR *src-1*	*src-2* OR *src-1*
Nor	(NOT *constant*) AND (NOT *src-1*)	(NOT *src-2*) AND (NOT *src-1*)
Equiv	(NOT *constant*) XOR *src-1*	(NOT *src-2*) XOR *src-1*
Invert	NOT *src-1*	NOT *src-1*
OrReverse	*constant* OR (NOT *src-1*)	*src-2* OR (NOT *src-1*)
CopyInverted	NOT *constant*	NOT *src-2*
OrInverted	(NOT *constant*) OR *src-1*	(NOT *src-2*) OR *src-1*
Nand	(NOT *constant*) OR (NOT *src-1*)	(NOT *src-2*) OR (NOT *src-1*)
Set	1	1

[1] `GC-function` is not a defined type in the core X protocol documentation. This table is taken from the list of alternative function values defined for the`CreateCG` protocol request.

When two sources are involved, all attributes, other than *width* and *height*, must match. All source data must be ***Constrained***, and *levels* must be a power of two. Output attributes are inherited from *src-1*.

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src-1* or *src-2*
FloDomain	Invalid *domain*
FloOperator	Invalid *operator*
FloMatch	*src-1* or *src-2* is not ***Constrained*** *levels* or *class* differs between *src-1* and *src-2* *levels* is not a power of 2

MatchHistogram

XieFloMatchHistogram

src: XieTypPhototag
domain: XieTypProcessDomain
shape: XieTypHistogramShape
tech-params: <technique-dependent>

Errors: ***FloSource, FloDomain, FloAlloc, FloMatch, FloTechnique***
Events: none

Attribute	Value
class	***SingleBand***
type	***Constrained***
width	Same as *src*
height	Same as *src*
levels	Same as *src*

MatchHistogram produces output data that differs from the source data in terms of its pixel value distribution or histogram. It allows the client to specify the desired state of the resulting data's histogram (algorithmic description of resulting histogram shape).

Src is the **Phototag** of the element supplying source data. *Domain* may control the subset of the source data to operated upon. *Shape* is a **HistogramShape** that specifies the form of the desired output data histogram. *Shape-params* is the list of additional parameters required by *shape*.

The source data must be ***Constrained*** and ***SingleBand***, and it must have three (3) or more *levels*. When a **ProcessDomain** is used, only data that intersects with *domain* is included in the histogram, and only that data will be affected in the result of the histogram matching operation (all other data will pass-through unchanged).

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src*
FloDomain	Invalid *domain*
FloMatch	Invalid *src* data: ***Unconstrained*** or ***TripleBand*** or bitonal
FloTechnique	Invalid histogram *shape* or *shape-params*

The **Point** element can also be used to reshape an image's histogram using an appropriate **LUT**.

Math

XieFloMath

src: XieTypPhototag
domain: XieTypProcessDomain
operator: XieTypMathOp
band-mask: CARD8

Errors: ***FloAlloc, FloSource, FloDomain, FloMatch, FloOperator, FloValue***
Events: none

Attribute	Value
class	Same as *src*
type	Same as *src*
width	Same as *src*
height	Same as *src*
levels	Same as *src*

Math applies a single operand mathematical operation to the source data on a point-wise basis.

Src is the **Phototag** of the element supplying source data. *Domain* may control the subset of the image that will be operated upon. *Operator* is the mathematical operation to be applied. *Band-mask* specifies which bands are to be operated upon.

Pixel computations that would lead to errors will yield valid server-dependent values (for example, the log of a ***Constrained*** pixel value of zero might result in a value of zero). Using *band-mask* to select source data that have two (2) or less *levels* is not permitted.

All output data attributes are inherited from the source data.

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src*
FloDomain	Invalid *domain*
FloMatch	Selected source data are bitonal (that is, 2 or less *levels*)
FloOperator	Invalid *operator*

PasteUp

XieFloPasteUp

tiles: `LISTof`XieTypTile
width: `CARD32`
height: `CARD32`
constant: XieTypConstant

Errors: ***FloAlloc, FloSource, FloMatch***
Events: none

Attribute	Value
class	Same as *tiles*
type	Same as *tiles*
width	*width*
height	*height*
levels	Same as *tiles*

PasteUp is an N-input translate operation that outputs data constructed from various source data tiles or a constant value.

Each of the *tile*s specifies a *src* (the **Phototag** of the element supplying source data) and the coordinates, *dst-x* and *dst-y*, where the tile belongs in the output data. *Width* and *height* specify the full extent of the output data. *Constant* is the fill value to as the output data for any region that is not specified as a tile.

Each region of the output data is defined by a tile's destination coordinates, *dst-x* and *dst-y*, and its *src* dimensions. For output regions where no tile provides input, the value of *constant* is used. If tiles overlap, a stacking order rule defines which pixel value will be output; the last tile (involved in the overlap) in the list of *tiles* will provide the output pixel.

At least one tile must be supplied. Except for *width* and *height*, all attributes of each source tile must match. In addition, for ***TripleBand*** input, inter-band dimensions within each *tile* must match.

No subsetting by band mask or **ProcessDomain** is provided.

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid source *tiles* No *tiles* were specified
FloMatch	Incompatible attributes between *tiles* Unequal inter-band dimensions within a *tile* (*width* or *height*)

Point

XieFloPoint

src: XieTypPhototag
lut: XieTypPhototag
domain: XieTypProcessDomain
band-mask: CARD8

Errors: ***FloAlloc, FloSource, FloDomain, FloMatch***
Events: none

Attribute	Value
class	Same as *lut*
type	***Constrained***
width	Same as *src*
height	Same as *src*
levels	Same as lut

Point maps source pixel values to output pixel values using a lookup table (**LUT**)[2].

Src is the **Phototag** of the element supplying ***Constrained*** source data. *Lut* is the **Phototag** of the element supplying the lookup table array(s). *Domain* may control the subset of source data that will be operated upon. *Band-mask* specifies which bands are to be operated upon (all bands must be specified if *lut* is ***SingleBand*** and *src* is ***TripleBand***).

The output is ***Constrained*** with the *width* and *height* taken from *src* and *class* and *levels* taken from *lut*. When *src* is ***SingleBand*** and *lut* is ***TripleBand***, for the bands that are indicated by *band-mask*, the output bands are remapped through their respective *lut* bands, whereas the other bands are just replications of the single *src* band. If *domain* is used, the *class* and *levels* of *lut* must match those of *src*.

Each *lut* array must contain sufficient entries so that all potential pixel values found in *src* can form a valid index into the array. Generally each *src* pixel value is used directly as an index into a *lut* array. When ***TripleBand*** *src* data are remapped through a ***SingleBand*** *lut;* however, pixel values from all three *src* bands are `combined` to form an array index (for this case, *width* and *height* must match for all bands). Figure 7-4 presents the algorithm for computing `combined` array indices.

LUT *band-order*	**LUT** indexing algorithm for combining pixel values from ***TripleBand*** data
LSFirst	index = value[0] + value[1] x *levels*[0] + value[2] x *levels*[0] x *levels*[1]
MSFirst	index = value[2] + value[1] x *levels*[2] + value[0] x *levels*[2] x *levels*[1]

Figure 7-4 Point's algorithm for computing combined LUT indices

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src* or *lut*
FloDomain	Invalid *domain* (for example, attempting to use *domain* with **ServiceClass** ***DIS***)
FloMatch	***Unconstrained*** *src* data

2 Figure 6-1 illustrates the relationship between **LUT** *class* and image *class* and presents some usage examples.

domain is being used, but *lut class* or *levels* do not match those of *src*
lut does not contain enough entries (that is, *length* less than *src levels*)
lut is ***SingleBand*** and *src* is ***TripleBand***, but *band-mask* is incomplete or inter-band dimensions don't match

Unconstrain

XieFloUnconstrain

src: XieTypPhototag

Errors: ***FloAlloc, FloSource, FloMatch***
Events: none

Attribute	Value
class	Same as *src*
type	***Unconstrained***
width	Same as *src*
height	Same as *src*
levels	Unknown

Unconstrain takes ***Constrained*** source data and produces ***Unconstrained*** data.

Src is the **Phototag** of the element supplying ***Constrained*** source data.

No subsetting by band mask or **ProcessDomain** is provided (the entire image is ***Unconstrained***).

Error	Cause
FloAlloc	Insufficient resources
FloSource	Invalid *src*
FloMatch	***Unconstrained*** *src*