NAME

VImage − image representation

SYNOPSIS

#include <vista/VImage.h>

VImage *image***;**

DESCRIPTION

Introduction

Vista represents an image as one or more two-dimensional arrays of pixel values. Each two-dimensional array, called a *band*, has the same size, which is expressed as a number of rows and a number of columns. Multiple bands may, for example, be used to represent multiple color components, multiple frames of a motion sequence, the real and imaginary components of a complex image, or some combination of these. Besides pixel data, an image may also has an arbitrary list of attributes associated with it.

An image in memory is referred to by the C pointer type **VImage**. In data files it's identified by the type name **image**, and in memory, by the **VRepnKind** code **VImageRepn**. Since it is a standard object type with built-in support in the Vista library, images can be read from data files, written to data files, and manipulated as part of attribute lists by routines such as **VReadFile**(3Vi), **VWriteFile**(3Vi), and **VGetAttr**(3Vi).

Creating and Destroying Images

The following routines create and destroy images.

VImage VCreateImage (int *nbands***, int** *nrows***, int** *ncolumns***, VPixelRepn** *pixel_repn***)**

> **VCreateImage** allocates memory for a new image with the specified numbers of bands, rows, and columns. Its pixel values are not initialized, and it is given an empty attribute list. The *pixel_repn* argument is described below, under ‘‘Pixel Representation’’. (See **VCreateImage**(3Vi) for details.)

void VDestroyImage (VImage *image***)**

> **VDestroyImage** releases the memory occupied by *image*, including that occupied by its pixel data and attribute list.

Image Size

An image's numbers of bands, row, and columns are sometimes abbreviated as **nbands**, **nrows**, and **ncolumns**. The following macros access those numbers.

int VImageNBands (VImage *image***)**

int VImageNRows (VImage *image***)**

int VImageNColumns (VImage *image***)**

> **VImageNBands**, **VImageNRows**, **VImageNColumns** return the number of bands, rows, and columns that *image* has.

int VImageNPixels (VImage *image***)**

> **VImageNPixels** returns the number of pixel values *image* has altogether (i.e., the product **nbands** ∗ **nrows** ∗ **ncolumns**).

Pixel Representation

An image's pixel values may have any of the following types: **VBit**, **VUByte**, **VSByte**, **VShort**, **VLong**, **VFloat**, or **VDouble**. These are described in the ‘‘Numeric Data Types’’ section of the **Vlib**(7Vi) manual page. Variables meant to hold pixel values or pointers to pixel values should be defined using these same types (rather than built-in C types such as **short** and **long**) to ensure software portability.

The following macros return information about an image's pixel representation:

VRepnKind VPixelRepn (VImage *image***)**

VPixelRepn returns a code indicating *image*'s pixel representation. (See **Vlib**(7Vi) for a definition of the **VRepnKind** codes.) If *image* has **VBit** pixels, for example, it returns **VBitRepn**.

size_t VPixelSize (VImage *image***)**

VPixelSize returns the size, in bytes, used for storing a single pixel of *image* (i.e, a pixel within a single band, row, and column). This value depends on machine architecture.

int VPixelPrecision (VImage *image***)**

VPixelPrecision returns the minimum size, in bits, needed for representing a single pixel of *image*. This value does not depend on machine architecture, but only on the kind of pixel value representation used by *image*. If *image* has **VBit** pixels, for example, this macro returns 1.

VStringConst VPixelRepnName (VImage *image***)**

VPixelRepnName returns a string describing the type of pixel value representation used by *image*. For example, if *image* has **VBit** pixels, the string ‘‘bit’’ is returned.

VDouble VPixelMaxValue (VImage *image***)**

VPixelMaxValue returns the maximum value that a pixel of *image* may have. For example, if *image* has **VSByte** pixels, it returns 127.0.

VDouble VPixelMinValue (VImage *image***)**

VPixelMinValue returns the minimum value that a pixel of *image* may have. For example, if *image* has **VSByte** pixels, it returns -128.0.

size_t VImageSize (VImage *image***)**

VImageSize returns the size, in bytes, of *image*'s block of pixel data, which contains the values of all pixels in all bands. This value depends on machine architecture.

Pixel Values

A particular pixel value is indexed by a band number, a row number, and a column number. Bands, rows, and columns are numbered consecutively from zero. Rows number down and columns number to the right so that row 0, column 0 is at the upper-left corner of the image.

type **VPixel (VImage** *image***, int** *band***, int** *row***, int** *column***,** *type***)**

VPixel accesses the pixel at a specified band, row and column in an image whose pixels are of a specified type. The pixel type, *type*, must be one of **VBit**, **VUByte**, **VSByte**, **VShort**, **VLong**, **VFloat**, or **VDouble**. The macro may be used either as an rvalue (to fetch the pixel's value) or as an lvalue (to modify it or obtain its address). The ‘‘EXAMPLES’’ section, below, demonstrates its use as an rvalue.

VPointer VPixelPtr (VImage *image***, int** *band***, int** *row***, int** *column***)**

VPixelPtr returns the address of a pixel specified by its band, row, and column coordinates. The pixel at that address can be fetched or modified by first coercing the pointer to the appropriate type for the pixel. For example:

```
pixel = VPixelPtr (image, band, row, column);
if (VPixelRepn (image) == VBitRepn)
        * (VBit *) pixel = new_value;
else ...
```

type *** **VPixelArray (VImage** *image***,** *type***)**

VPixelArray returns a pointer that can be used to access any image pixel by indexing it first with the pixel's band number, then its row number, and then its column number. For example:

VUByte ∗∗∗pixels = VPixelArray (image, VUByte);
pixels[band][row][column] = new_value;

VPointer VImageData (VImage *image***)**

VImageData returns a pointer to *image*'s block of pixel data. The block has the size **VImageSize (***image***)** bytes.

Other Image Attributes

In addition to its pixel values an image may have any number of other attributes; they are represented as an attribute list.

VAttrList VImageAttrList (VImage *image***)**

VImageAttrList is a macro for accessing the attribute list associated with an image. The macro may be used as an rvalue to reach attributes within the list:

VGetAttr (VImageAttrList (image), VNameAttr, ...**)**

or as an lvalue to manipulate the entire list:

VDestroyAttrList (VImageAttrList (image));
VImageAttrList (image) = VCopyAttrList (other_list);

Band Interpretation

The bands of a multi-band image might represent the successive frames of a motion sequence, the left and right halves of a stereo pair, or the red, green, and blue channels of a color image. They might even represent a combination of such dimensions — for example, a stereo pair of color images. Because a great many different uses of the bands are possible, each image includes information describing how its bands are to be interpreted. This information assigns each band an interpretation in terms of four dimensions:

frame covers the various frames of a motion sequence

viewpoint covers, for example, the left and right channels of a stereo pair

color covers, for example, the red, green, and blue channels of a color image

component covers, for example, the real and imaginary components of a complex image

Although these four dimensions do not account for all conceivable ways one might wish to use the bands of a multi-band image, they do cover most cases. The scheme can usually be adapted to cover cases not explicitly handled. For example, a collection of images obtained under various lighting conditions could be represented by using the frame dimension to index the lighting condition.

Each image specifies the sizes of its four band interpretation dimensions. For a stereo pair of color images, for example, the size of the viewpoint dimension is two, the size of the color dimension is three, and the remaining dimensions have sizes of one. The dimension sizes are often abbreviated as **nframes**, **nviewpoints**, **ncolors**, and **ncomponents**. Their product should always equal the number of bands in the image. These macros exist for accessing the sizes:

int VImageNFrames (VImage *image***)**

int VImageNViewpoints (VImage *image***)**

int VImageNColors (VImage *image***)**

int VImageNComponents (VImage *image***)**

Each macro may be used as rvalue to obtain the size of a dimension, or as an lvalue to set it.

The four dimensions are mapped onto the single dimension, band, according to a particular ordering: component varies most rapidly, then color, then viewpoint, and finally frame. A macro is available for computing this mapping:

int VBandIndex (VImage *image***, int** *frame***, int** *viewpoint***, int** *color***, int** *component***)**

VBandIndex returns the index of the band that corresponds to a particular combination of frame, viewpoint, channel, and component indices.

An image's attribute list may include attributes indicating how the image employs the four band interpretation dimensions. One attribute for each dimension provides the first level of description. Its value is an integer code, with some standard codes defined for common uses. Below is a list of the four attributes and their standard values. Shown in are both the symbolic constants that can be used C programs to refer to the integer values, and (in parentheses) the keywords used to represent the values externally, in Vista data files.

frame_interp, which currently has no standard values defined for it

viewpoint_interp, which has this standard value:

VBandInterpStereoPair (**stereo_pair**)
: The two viewpoints represent (respectively) the left and right images of a stereo pair.

color_interp, which has these standard values:

VBandInterpRGB (**rgb**)
: The three colors represent (respectively) red, green, and blue color channels.

component_interp, which has these standard values:

VBandInterpComplex (**complex**)
: The two components represent (respectively) the real and imaginary components of a complex image.

VBandInterpGradient (**gradient**)
: There is a single component representing gradient magnitude, two components representing (respectively) the x (rightward) and y (upward) components of gradient, or three representing the x, y, and z (forward in band sequence) components of gradient.

VBandInterpIntensity (**intensity**)
: There is a single component representing image intensity. For integer pixel representations, pixel values normally span the range between zero and the maximum representable pixel values (e.g., [0,127] for **VSByte** pixels). For floating point pixel representations, pixel values normally span the range [–1,+1].

VBandInterpOrientation (**orientation**)
: There is a single component representing orientation. Pixel values express orientation in radians, counterclockwise from the direction of increasing column number (rightward).

To prevent accidental misspellings of the attribute names you should use these symbolic constants:

```
#define VFrameInterpAttr "frame_interp"
#define VViewpointInterpAttr "viewpoint_interp"
#define VColorInterpAttr "color_interp"
#define VComponentInterpAttr "component_interp"
```

A dictionary, **VBandInterpDict**, defines the mapping between the band interpretation codes (e.g., **VBandInterpComplex**) and their keywords (e.g., **complex**):

```
VDictEntry VBandInterpDict[];
```

For each of the four band interpretation dimensions there is a routine you can use to simultaneously check both the dimension's size and, if present, its interpretation attribute. The **VBandInterp**(3Vi) man page describes the four routines.

Pixel Aspect Ratio

Pixel aspect ratio is defined as the ratio of pixel width to pixel height. An image may include, on its attribute list, an attribute specifying this ratio as a floating point value. The attribute has the name **pixel_aspect_ratio** and a symbolic constant exists for referring to it:

#define VPixelAspectRatioAttr "pixel_aspect_ratio"

When the attribute is absent, a pixel aspect ratio of 1.0 is assumed.

Calling Conventions

A Vista library routine that processes one image to produce another usually takes three or more arguments, like this:

VImage VProcessImage (VImage *src*, **VImage** *dest*, **VBand** *band***)**

The first argument specifies the *source* image to be processed. The second specifies the *destination* image, to contain the result. You generally have three choices for how this destination image is provided:

a) you can have the routine create a destination image. If you pass **NULL** for *dest*, the routine will create and return a destination image that is appropriate for the source image and for the type of operation being performed on it:

```
dest = VProcessImage (src, NULL, band);
```

b) you can supply a destination image provided it has the correct properties (usually, an appropriate size and pixel representation):

```
dest = VCreateImageLike (src);
VProcessImage (src, dest, band);
```

c) you can specify the same image as both source and destination, in which case the source pixel values will be replaced by destination pixel values. This choice is only available for operations where the source and destination images have the same size and pixel representation.

```
VProcessImage (src, src, band);
```

In all cases, the routine will return the destination image if successful, and **NULL** otherwise. Moreover, if **NULL** is returned, the routine will already have called **VWarning**(3Vi) to issue a warning message describing the problem.

The third argument to most image processing routines, *band*, may specify a particular band of the source image to be processed; or it may be the constant **VAllBands** (defined as -1) to indicate that all bands of the source image are to be processed. When a particular band is specified, *band* must be at least 0 and less than the number of bands in the source image; the destination image will usually have a single band to contain the result. When **VAllBands** is specified, the destination image will usually have the same number of bands as the source image.

Normally a routine will copy the source image’s attribute list to the destination image, replacing any attributes the destination image had already. The sizes of the four band interpretation dimensions (**nframes**, **nviewpoints**, etc.) are also copied. When appropriate, however, a routine may delete or modify the attributes and dimensional sizes as it copies them from source to destination. For example, a routine that converts an RGB color image to a grey-scale one (**VRGBImageToGray**(3Vi)) expects a source image with the properties **ncolors: 3** and **color_interp: rgb**; the destination image it produces has **ncolors: 1** and no **color_interp** attribute.

Image Representation in Memory

```
typedef struct {
        int nbands;                 /* number of bands */
        int nrows;                  /* number of rows */
        int ncolumns;               /* number of columns */
        VRepnKind pixel_repn;       /* representation of pixel values */
        unsigned long flags;        /* various flags */
        VAttrList attributes;       /* list of other image attributes */
        VPointer data;              /* array of image pixel values */
        VPointer *row_index;        /* ptr to first pixel of each row */
        VPointer **band_index;      /* ptr to first row of each band */
        int nframes;                /* number of motion frames */
```

```
        int nviewpoints;        /* number of camera viewpoints */
        int ncolors;            /* number of color channels */
        int ncomponents;        /* number of vector components */
} VImageRec, *VImage;
```

An image is represented in memory by a value of type **VImage**, which points to a **VImageRec** structure. Programmers using images will usually not need to access this structure's fields directly from their code since there are Vista library routines and macros available for most purposes. However, when debugging, one may occasionally need to examine a **VImageRec** structure's fields directly. The following paragraphs describe the fields.

The **nbands**, **nrows**, and **ncolumns** fields of the structure describe how many bands, rows, and columns the image has. Its **pixel_repn** field specifies how the image's pixel values are represented; it contains one of the constants **VBitRepn**, **VUByteRepn**, **VSByteRepn**, **VShortRepn**, **VLongRepn**, **VFloatRepn**, or **VDoubleRepn**. Its **attributes** field contains the list of attributes associated with the image.

An image's pixel values are stored in a contiguous block of memory where they are arranged as follows. Within each band pixel values are ordered, beginning with the pixel at row 0, column 0, proceeding to row 0, column 1, etc., and ending with the highest numbered row and column. Each band's array of pixel values occupies a separate block of memory, and these blocks are arranged contiguously. All of band 0's pixel values are first, followed by all of band 1's, etc. In summary, pixel values are ordered by band, then by row within each band, and then by column. The pixel values themselves are of type **VBit, VUByte, ..., or VDouble.**

The **data** field of a **VImageRec** points to the block of memory containing pixel values. The **row_index** field points to a vector of pointers to the first pixel value of each row within each band; this vector is of length (**nbands** ∗ **nrows**) and it is indexed by an expression of the form (*band* ∗ **nrows** + *row*). The **band_index** field points to a vector of pointers to the first **row_index** entry of each band; this vector is of length **nbands** and it is indexed by band number.

Individual bits of a **VImageRec**'s **flags** field are used to denote various things. One bit is currently defined:

VImageSingleAlloc This bit, if nonzero, indicates that the **VImageRec**, the block of pixel values, the row index, and the band index were all obtained with a single call to **VMalloc**(3Vi). If the bit is zero, these things were allocated separately and therefore they must be released separately.

The **nframes**, **nviewpoints**, **ncolors**, and **ncomponents** fields describe the number of motion frames, camera positions, color channels, and vector components that the image's various bands represent. Together they should account for all bands of the image — i.e., their product should equal **nbands**. The image's bands are ordered according to these fields, with component index incrementing most rapidly and frame index incrementing least rapidly.

Image Representation in a File

*attribute-name***: image {**
  nbands: *nbands*
  nrows: *nrows*
  ncolumns: *ncolumns*
  repn: *pixel-representation*
  nframes: *nframes*
  nviewpoints: *nviewpoints*
  ncolors: *ncolors*
  ncomponents: *ncomponents*
  data: *data-offset*
  length: *data-length*
  other attributes
}

pixel-representation ::=
bit | ubyte | sbyte | short | long | float | double

An image is represented in a Vista data file as an attribute. The attribute's value is an object of type **image** containing the following attributes in any order:

nbands an integer-valued attribute specifying the number of bands in the image.

nrows an integer-valued attribute specifying the number of rows in the image.

ncolumns an integer-valued attribute specifying the number of columns in the image.

repn a string-valued attribute specifying the image's pixel value representation with one of the following keywords: **bit**, **ubyte**, **sbyte**, **short**, **long**, **float**, or **double**.

nframes an integer-valued attribute specifying the number of motion frames represented by the image's bands.

nviewpoints
an integer-valued attribute specifying the number of camera viewpoints represented by the image's bands.

ncolors an integer-valued attribute specifying the number of color channels represented by the image's bands.

ncomponents
an integer-valued attribute specifying the number of vector components represented by the image's bands.

data an integer-valued attribute specifying the offset of the image's pixel data within the binary data section of the file

length an integer-valued attribute specifying the size, in bytes, of the image's pixel data

An **nbands**, **nframes**, **nviewpoints**, **ncolors**, or **ncomponents** attribute may be omitted if its value is 1. Moreover, the **image** object's attribute list may contain any other attributes associated with the image. See, for example, the data file listed under ‘‘EXAMPLES’’, below.

Pixel Representation in a File

Image pixel values are represented in binary form in the binary data section of a file. They occupy a contiguous block in which pixel values are ordered by band, row, and column, as in memory.

Single-bit pixel values are packed eight to a byte, beginning with the most-significant bit of each byte. Pixel values of more than 8 bits are packed into multiple bytes, beginning with the most-significant byte of the pixel value. Consecutive pixel values are packed together, without intervening bits for padding or alignment. If necessary, however, the entire block of pixel values is padded so that its length is a multiple of 8 bits.

Each kind of pixel is represented as follows in the Vista data file:

VBit as a 1-bit unsigned integer

VUByte as an 8-bit unsigned integer

VSByte as an 8-bit two's-complement integer

VShort as a 16-bit two's-complement integer

VLong as a 32-bit two's-complement integer

VFloat as a 32-bit IEEE floating-point number

VDouble as a 64-bit IEEE floating point number

Note that a pixel's representation in a Vista data file is independent of the form it takes when stored in memory on any particular machine. Thus a **VBit** pixel may occupy 8 bits in the main memory of a Sun SPARCstation 32 bits on a Cray machine, but it always occupies a single bit in a Vista data file.

EXAMPLES

This code fragment sets all pixels with 1 in a one-band image of single-bit pixels:

```
for (i = 0; i < VImageNRows (image); i++)
        for (j = 0; j < VImageNColumns (image); j++)
                VPixel (image, 0, i, j, VBit) = 1;
```

The previous example may be made more efficient by avoiding the repeated computation of pixel addresses:

```
VBit *p = & VPixel (image, 0, 0, 0, VBit);

for (i = 0; i < VImageNPixels (image); i++)
        *p++ = 1;
```

This code fragment creates an image of complex pixel values:

```
image = VCreateImage (2, 256, 256, VFloatRepn);

VImageNFrames (image) = VImageNViewpoints (image) = VImageNColors (image) = 1;

VImageNComponents (image) = 2;

VSetAttr (VImageAttrList (image), VComponentInterpAttr, NULL, VLongRepn,
           (VLong) VBandInterpComplex);
```

Here is an example of a Vista data file containing two images:

```
V-data 2 {
        one: image {
                nbands: 2
                nrows: 256
                ncolumns: 256
                ncomponents: 2
                component_interp: complex
                repn: float
                data: offset of first image's pixel values
                length: 524288
                name: "UFO sighted over Vancouver"
                pixel_aspect_ratio: 1.25
        }
        two: image {
                nrows: 32
                ncolumns: 32
                repn: ubyte
                data: offset of second image's pixel values
                length: 128
                name: "UFO icon"
        }
}
^L
first image's pixel values
second image's pixel values
```

SEE ALSO

Vattribute(3Vi), **Vfile**(5Vi), **Vlib**(7Vi)

AUTHOR

Art Pope <pope@cs.ubc.ca>

LIST OF ROUTINES

The following table summarizes the Vista library routines that operate on images. Many of these routine are documented elsewhere, by a section 3Vi man page named for the routine.

For creating and destroying images in memory:

VCreateImage Create an image.

VCreateImageLike Create one image like another.

VDestroyImage Release memory occupied by an image.

For fetching and storing pixel values:

VGetPixel Fetch a pixel value with any pixel representation.

VSetPixel Store a pixel value with any pixel representation.

VSelectBand Select bands of image pixels for processing.

VSelectDestIamge Select a destination for an image processing operation.

For getting band interpretation information:

VImageFrameInterp Report meaning of frame dimension.

VImageColorInterp Report meaning of color dimension.

VImageComponentInterp
Report meaning of component dimension.

VImageViewpointInterp
Report meaning of viewpoint dimension.

For reading and writing images:

VReadImages Read a set of images from a data file.

VReadPlain Read an image in Vista plain file format.

VReadPnm Read an image in Portable Anymap (PNM) format.

VReadUbcIff Read an image in UBC image file format.

VWriteImages Write a set of images to a data file.

VWriteUbcIff Write an image in UBC image file format.

For comparing images:

VSameImageRange Test whether two images have the same size and pixel representation.

VSameImageSize Test whether two images have the same size.

For copying images:

VCombineBands Copy selected bands of pixel values from various source images to a single destination image.

VCopyBand Copy one or all bands of pixel values from one image to another.

VCopyImage Copy one image to another.

VCopyImageAttrs Copy one image’s attributes to another.

VCopyImagePixels Copy one image’s pixel values to another.

For converting an image’s pixel representation:

VConvertImageCopy Convert an image’s pixel representation while simply copying pixel values.

VConvertImageLinear
Convert an image’s pixel representation using some linear mapping of pixel

values.

VConvertImageOpt Convert an image’s pixel representation while mapping the actual range of source pixel values to the full range of possible destination pixel values.

VConvertImageRange Convert an image’s pixel representation while mapping the full range of possible source pixel values to the full range of possible destination pixel values.

For rotating or transposing an image:

VFlipImage Flip an image horizontally or vertically.

VRotateImage Rotate an image by any angle.

VTransposeImage Transpose the rows and columns of an image.

For changing the size of an image:

VCropImage Extract a rectangular region from an image.

VExpandImage Increase the size of an image by an integer factor.

VReduceImage Decrease the size of an image by an integer factor.

VScaleImage Scale the size of an image, up or down, by any factor.

For filtering and transforming images:

VAdjustImage Adjust image brightness and/or contrast.

VCanny Detect edges in an image using a Canny operator.

VConvolveImage Convolve an image with a mask.

VGaussianConvolveImage
Convolve an image with a Gaussian filter or its derivative.

VImageFFT Compute a forward or inverse Fourier transform.

VImageGradient Compute the horizontal and vertical components of image gradient.

VImageOp Perform an arithmetic or logical operation on an image’s pixel values.

VInvertImage Invert an image’s pixel values, swapping black and white.

VLinkImage Create an edge set by linking connected, non-zero image pixels.

VNegateImage Negate an image’s pixel values.

VZeroCrossings Mark the zero crossings in an image.

For computing image statistics:

VImageStats Compute the minimum, maximum, mean, and variance of an image’s pixel values.

For working with complex images:

VBuildComplexImage Build a complex image from separate images of real and imaginary components.

VImageMagnitude Compute the magnitude of a complex image.

VImagePhase Compute the phase of a complex image, or the gradient direction of a two-component gradient image.

For displaying or rendering images:

VImageBandToPS Render an image band using PostScript.

VRGBImageToPS Render an RGB color image using PostScript.

VImageView An X Toolkit widget for displaying an image.

VImageWindowSize	Compute appropriate window dimensions for displaying an image.

For filling images with patterns:

VFillImage	Fill an image with a constant pixel value.
VRampImage	Fill an image with an intensity ramp.
VSineGratingImage	Fill an image with a sine grating.
VZonePlateImage	Fill an image with a zone plate pattern.
VBilinearNoiseImage	Fill an image with a random mixture of two pixel values.
VNormalNoiseImage	Fill an image with pixel values drawn from a normal distribution.
VUniformNoiseImage	Fill an image with pixel values drawn from a uniform distribution.

Miscellaneous:

VOptFlowWLS	compute optical flow by weighted least squares.