vImage Morphology Reference

Performance > Graphics & Imaging

2007-07-12

Contents

vImage Morphology Reference

Framework:	Accelerate/vImage
Companion guide	vImage Programming Guide
Declared in	Morphology.h

Overview

Morphological functions change the shape of an object by performing dilatation, erosion, maximum, and minimum operations. Dilation expands objects. Erosion contracts them. Maximum is a special case of dilation, while minimum is a special case of erosion. The precise nature of the expanding or shrinking is determined by a kernel (also known as a *structure element*) provided by the caller. The number of rows and number of columns of the image does not change after applying a morphological operation.

You can use morphological functions on grayscale images, where the source image is planar (single-channel) or on full-color images. The kernel itself is always planar.

vImage applies morphological operations to an **object**, which is not the same as the entire image. An object is either comprised of the brightest pixels in an image or the darkest pixels in the image, where brightness is defined relative to the particular image. When you define bright pixels as the object, dark pixels become the background. In this case dilation expands objects with erosion contracts them. When you define dark pixels as the object, bright pixels become the background. In this case, dilation contracts objects and erosion expands them.

Each morphological function requires that you pass it a convolution kernel that determines how the values of neighboring pixels are used to compute the value of a destination pixel. A kernel is a packed array, without padding at the ends of the rows. The elements of the array must be of type `uint8_t` (for the Planar8 and ARGB8888 formats) or of type `float` (for the PlanarF and ARGBFFFF formats). The height and the width of the array must both be odd numbers.

For example, a 3 x 3 convolution kernel for a Planar8 image consists of nine 8-bit (1-byte) values, arranged consecutively. The first three values represent the first row of the kernel, the next three values the second row, and the last three values the third row.

Morphology functions perform clipping to prevent overflow for the Planar8 and ARGB8888 formats. Saturated clipping maps all intensity levels above 255, to 255, all intensity levels below 0, to 0, and leaves intensity levels between 0 and 255, inclusive, unchanged.

When the pixel to be transformed is near the edge of the image—not merely the region of interest, but the entire image of which it is a part—the kernel may extend beyond the edge of the image, so that there are no existing pixels beneath some of the kernel elements. In this case the morphology functions only make use of that part of the kernel which overlaps the source buffer. The other kernel elements are ignored.

Functions by Task

Dilating an Object

`vImageDilate_ARGBFFFF` (page 8)
: Dilates a region of interest within an ARGBFFFF source image using an M x N kernel.

`vImageDilate_ARGB8888` (page 7)
: Dilates a region of interest within an ARGB8888 source image using an M x N kernel.

`vImageDilate_PlanarF` (page 10)
: Dilates a region of interest within a PlanarF source image using an M x N kernel.

`vImageDilate_Planar8` (page 9)
: Dilates a region of interest within a Planar8 source image using an M x N kernel.

Eroding an Object

`vImageErode_ARGBFFFF` (page 12)
: Erodes a region of interest within an ARGBFFFF source image using an M x N kernel.

`vImageErode_ARGB8888` (page 11)
: Erodes a region of interest within an ARGB8888 source image using an M x N kernel.

`vImageErode_PlanarF` (page 14)
: Erodes a region of interest within a PlanarF source image using an M x N kernel.

`vImageErode_Planar8` (page 13)
: Erodes a region of interest within a Planar8 source image using an M x N kernel.

Maximizing an Object

`vImageMax_ARGBFFFF` (page 17)
: Maximizes a region of interest within an ARGBFFFF source image using an M x N kernel.

`vImageMax_ARGB8888` (page 16)
: Maximizes a region of interest within an ARGB8888 source image using an M x N kernel.

`vImageMax_PlanarF` (page 20)
: Maximizes with a region of interest within a PlanarF source image using an M x N kernel.

`vImageMax_Planar8` (page 19)
: Maximizes a region of interest within a Planar8 source image using an M x N kernel.

Minimizing an Object

`vImageMin_ARGBFFFF` (page 23)
: Minimizes a region of interest within an ARGBFFFF source image using an M x N kernel.

`vImageMin_ARGB8888` (page 22)
: Minimizes a region of interest within an ARGB8888 source image using an M x N kernel.

`vImageMin_PlanarF` (page 26)
: Minimizes a region of interest within a PlanarF source image using an M x N kernel.

`vImageMin_Planar8` (page 25)
: Minimizes a region of interest within a Planar8 source image using an M x N kernel.

Getting the Buffer Size

`vImageGetMinimumTempBufferSizeForMinMax` (page 15)
: Returns the minimum size, in bytes, for the temporary buffer that the caller supplies to any of the Min or Max morphological functions. (Deprecated. Use the `kvImageGetTempBufferSize` flag with the appropriate morphological function instead of calling this function.)

Functions

vImageDilate_ARGB8888

Dilates a region of interest within an ARGB8888 source image using an M x N kernel.

```
vImage_Error vImageDilate_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const unsigned char *kernel,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags
);
```

Parameters

src
: A pointer to a vImage buffer structure that contains data for the source image.

dest
: A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

  The size (number of rows and number of columns) of the destination buffer also specifies the size of the region of interest in the source buffer.

srcOffsetToROI_X
: The horizontal offset, in pixels, to the upper-left pixel of the region of interest within the source image.

srcOffsetToROI_Y
: The vertical offset, in pixels, to the upper-left pixel of the region of interest within the source image.

kernel
: A pointer to the kernel data, which must be a packed array without any padding. The function uses the same kernel for all channels.

kernel_height

The height of the kernel in pixels. This value must be odd.

kernel_width

The width of the kernel in pixels. This value must be odd.

flags

The options to use when performing the morphological operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading. Set the `kvImageLeaveAlphaUnchanged` flag to specify that the alpha channel should be copied to the destination image unchanged.

Return Value

`kvImageNoError`, otherwise it returns one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Morphology.h`

vImageDilate_ARGBFFFF

Dilates a region of interest within an ARGBFFFF source image using an M x N kernel.

```
vImage_Error vImageDilate_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const float *kernel,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains data for the source image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

The size (number of rows and number of columns) of the destination buffer also specifies the size of the region of interest in the source buffer.

srcOffsetToROI_X

The horizontal offset, in pixels, to the upper-left pixel of the region of interest within the source image.

srcOffsetToROI_Y

The vertical offset, in pixels, to the upper-left pixel of the region of interest within the source image.

kernel

A pointer to the kernel data, which must be a packed array without any padding.

kernel_height

The height of the kernel in pixels. This value must be odd.

kernel_width

The width of the kernel in pixels. This value must be odd.

flags

The options to use when performing the morphological operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading. Set the `kvImageLeaveAlphaUnchanged` flag to specify that the alpha channel should be copied to the destination image unchanged.

Return Value

`kvImageNoError`, otherwise it returns one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

This function uses the same kernel is for all channels. In addition to supplying space for the destination image, the `dest` parameter also specifies the size of the region of interest in within the source image. The region of interest has the same height and width (in pixels) as the destination buffer pointed to by dest.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Morphology.h`

vImageDilate_Planar8

Dilates a region of interest within a Planar8 source image using an M x N kernel.

```
vImage_Error vImageDilate_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const unsigned char *kernel,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains data for the source image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

The size (number of rows and number of columns) of the destination buffer also specifies the size of the region of interest in the source buffer.

srcOffsetToROI_X

The horizontal offset, in pixels, to the upper-left pixel of the region of interest within the source image.

srcOffsetToROI_Y

The vertical offset, in pixels, to the upper-left pixel of the region of interest within the source image.

kernel

A pointer to the kernel data, which must be a packed array without any padding.

kernel_height

The height of the kernel in pixels. This value must be odd.

kernel_width

The width of the kernel in pixels. This value must be odd.

flags

The options to use when performing the morphological operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading. Set the `kvImageLeaveAlphaUnchanged` flag to specify that the alpha channel should be copied to the destination image unchanged.

Return Value

`kvImageNoError`, otherwise it returns one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Morphology.h`

vImageDilate_PlanarF

Dilates a region of interest within a PlanarF source image using an M x N kernel.

```
vImage_Error vImageDilate_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const float *kernel,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains data for the source image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

The size (number of rows and number of columns) of the destination buffer also specifies the size of the region of interest in the source buffer.

srcOffsetToROI_X

The horizontal offset, in pixels, to the upper-left pixel of the region of interest within the source image.

srcOffsetToROI_Y

The vertical offset, in pixels, to the upper-left pixel of the region of interest within the source image.

kernel

A pointer to the kernel data, which must be a packed array without any padding.

kernel_height

The height of the kernel in pixels. This value must be odd.

kernel_width

The width of the kernel in pixels. This value must be odd.

flags

The options to use when performing the morphological operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading. Set the `kvImageLeaveAlphaUnchanged` flag to specify that the alpha channel should be copied to the destination image unchanged.

Return Value

`kvImageNoError`, otherwise it returns one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Morphology.h`

vImageErode_ARGB8888

Erodes a region of interest within an ARGB8888 source image using an M x N kernel.

```
vImage_Error vImageErode_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const unsigned char *kernel,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains data for the source image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

The size (number of rows and number of columns) of the destination buffer also specifies the size of the region of interest in the source buffer.

srcOffsetToROI_X

The horizontal offset, in pixels, to the upper-left pixel of the region of interest within the source image.

srcOffsetToROI_Y

The vertical offset, in pixels, to the upper-left pixel of the region of interest within the source image.

kernel

A pointer to the kernel data, which must be a packed array without any padding. The function uses the same kernel for all channels.

kernel_height

The height of the kernel in pixels. This value must be odd.

kernel_width

The width of the kernel in pixels. This value must be odd.

flags

The options to use when performing the morphological operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading. Set the `kvImageLeaveAlphaUnchanged` flag to specify that the alpha channel should be copied to the destination image unchanged.

Return Value

`kvImageNoError`, otherwise it returns one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Morphology.h`

vImageErode_ARGBFFFF

Erodes a region of interest within an ARGBFFFF source image using an M x N kernel.

```
vImage_Error vImageErode_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const float *kernel,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains data for the source image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

The size (number of rows and number of columns) of the destination buffer also specifies the size of the region of interest in the source buffer.

srcOffsetToROI_X

The horizontal offset, in pixels, to the upper-left pixel of the region of interest within the source image.

srcOffsetToROI_Y

The vertical offset, in pixels, to the upper-left pixel of the region of interest within the source image.

kernel

A pointer to the kernel data, which must be a packed array without any padding. The function uses the same kernel for all channels.

kernel_height

The height of the kernel in pixels. This value must be odd.

kernel_width

The width of the kernel in pixels. This value must be odd.

flags

The options to use when performing the morphological operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading. Set the `kvImageLeaveAlphaUnchanged` flag to specify that the alpha channel should be copied to the destination image unchanged.

Return Value

`kvImageNoError`, otherwise it returns one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Morphology.h`

vImageErode_Planar8

Erodes a region of interest within a Planar8 source image using an M x N kernel.

```
vImage_Error vImageErode_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const unsigned char *kernel,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains data for the source image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

The size (number of rows and number of columns) of the destination buffer also specifies the size of the region of interest in the source buffer.

srcOffsetToROI_X

The horizontal offset, in pixels, to the upper-left pixel of the region of interest within the source image.

srcOffsetToROI_Y

The vertical offset, in pixels, to the upper-left pixel of the region of interest within the source image.

kernel

A pointer to the kernel data, which must be a packed array without any padding.

kernel_height

The height of the kernel in pixels. This value must be odd.

kernel_width

The width of the kernel in pixels. This value must be odd.

flags

The options to use when performing the morphological operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading. Set the `kvImageLeaveAlphaUnchanged` flag to specify that the alpha channel should be copied to the destination image unchanged.

Return Value

`kvImageNoError`, otherwise it returns one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Morphology.h`

vImageErode_PlanarF

Erodes a region of interest within a PlanarF source image using an M x N kernel.

```
vImage_Error vImageErode_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const float *kernel,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains data for the source image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

The size (number of rows and number of columns) of the destination buffer also specifies the size of the region of interest in the source buffer.

srcOffsetToROI_X

The horizontal offset, in pixels, to the upper-left pixel of the region of interest within the source image.

srcOffsetToROI_Y
: The vertical offset, in pixels, to the upper-left pixel of the region of interest within the source image.

kernel
: A pointer to the kernel data, which must be a packed array without any padding.

kernel_height
: The height of the kernel in pixels. This value must be odd.

kernel_width
: The width of the kernel in pixels. This value must be odd.

flags
: The options to use when performing the morphological operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading. Set the `kvImageLeaveAlphaUnchanged` flag to specify that the alpha channel should be copied to the destination image unchanged.

Return Value
`kvImageNoError`, otherwise it returns one of the error codes described in *vImage Data Types and Constants Reference*.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`Morphology.h`

vImageGetMinimumTempBufferSizeForMinMax

Returns the minimum size, in bytes, for the temporary buffer that the caller supplies to any of the Min or Max morphological functions. (**Deprecated.** Use the `kvImageGetTempBufferSize` flag with the appropriate morphological function instead of calling this function.)

```
size_t vImageGetMinimumTempBufferSizeForMinMax (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags,
   size_t bytesPerPixel
);
```

Parameters

src
: A pointer to the vImage buffer structure that you plan to pass to the Min or Max function.

dest
: A pointer to the vImage buffer structure that you plan to pass to the Min or Max function.

kernel_height
: The height, in pixels, of the kernel that you plan to pass to the Min or Max function.

kernel_width
: The width, in pixels, of the kernel that you plan to pass to the Min or Max function.

flags
: The flags that you plan to pass to the Min or Max function.

pixelBytes

The number of bytes in a pixel. Make sure to pass the value appropriate for the format of the pixel.

Return Value

The minimum size, in bytes, of the temporary buffer.

Discussion

This function uses the *height* and *width* fields from the *src* or *dest* parameters; it does not use the *data* or *rowBytes* fields. If the size of the images you are processing remain the same, then the required size of the buffer also remains the same. More specifically, if, between two calls to `vImageGetMinimumTempBufferSizeForMinMax`, the `height` and `width` of the *src* and *dest* parameters do not increase, and the other parameters remain the same, then the result of the `vImageGetMinimumTempBufferSizeForMinMax` does not increase. This makes it easy to reuse the same temporary buffer when you process a number of images of the same size, as you would when tiling.

Availability

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.4.

Declared In

`Morphology.h`

vImageMax_ARGB8888

Maximizes a region of interest within an ARGB8888 source image using an M x N kernel.

```
vImage_Error vImageMax_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains data for the source image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

The size (number of rows and number of columns) of the destination buffer also specifies the size of the region of interest in the source buffer.

tempBuffer

A pointer to a temporary buffer. If you pass `NULL`, the function allocates the buffer, then deallocates it before returning. Alternatively, you can allocate the buffer yourself, in which case you are responsible for deallocating it when you is no longer need it.

If you want to allocate the buffer yourself, see the Discussion for information on how to determine the minimum size that you must allocate.

srcOffsetToROI_X
: The horizontal offset, in pixels, to the upper-left pixel of the region of interest within the source image.

srcOffsetToROI_Y
: The vertical offset, in pixels, to the upper-left pixel of the region of interest within the source image.

kernel_height
: The height of the kernel in pixels. This value must be odd.

kernel_width
: The width of the kernel in pixels. This value must be odd.

flags
: The options to use when performing the morphological operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading. Set the `kvImageLeaveAlphaUnchanged` flag to specify that the alpha channel should be copied to the destination image unchanged. Set the `kvImageGetTempBufferSize` flag if you want to determine the minimum size to allocate for the `tempBuffer` parameter.

Return Value
`kvImageNoError`, otherwise it returns one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
If you want to allocate the memory for the `tempBuffer` parameter yourself, call this function twice, as follows:

1. To determine the minimum size for the temporary buffer, the first time you call this function pass the `kvImageGetTempBufferSize` flag. Pass the same values for all other parameters that you intend to use in for the second call. The function returns the required minimum size, which should be a positive value. (A negative returned value indicates an error.) The `kvImageGetTempBufferSize` flag prevents the function from performing any processing other than to determine the minimum buffer size.

2. After you allocate enough space for a buffer of the returned size, call the function a second time, passing a valid pointer in the `tempBuffer` parameter. This time, do not pass the `kvImageGetTempBufferSize` flag.

The morphological operation Max is a special case of the dilation operation. In the Max operation, all the elements of the kernel have the same value. The actual value does not matter; only the size of the kernel is significant. vImage optimizes this special case, so the Max function is considerably faster than the Dilate function called with a uniform kernel.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`Morphology.h`

vImageMax_ARGBFFFF

Maximizes a region of interest within an ARGBFFFF source image using an M x N kernel.

```
vImage_Error vImageMax_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains data for the source image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

The size (number of rows and number of columns) of the destination buffer also specifies the size of the region of interest in the source buffer.

tempBuffer

A pointer to a temporary buffer. If you pass `NULL`, the function allocates the buffer, then deallocates it before returning. Alternatively, you can allocate the buffer yourself, in which case you are responsible for deallocating it when you is no longer need it.

If you want to allocate the buffer yourself, see the Discussion for information on how to determine the minimum size that you must allocate.

srcOffsetToROI_X

The horizontal offset, in pixels, to the upper-left pixel of the region of interest within the source image.

srcOffsetToROI_Y

The vertical offset, in pixels, to the upper-left pixel of the region of interest within the source image.

kernel_height

The height of the kernel in pixels. This value must be odd.

kernel_width

The width of the kernel in pixels. This value must be odd.

flags

The options to use when performing the morphological operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading. Set the `kvImageLeaveAlphaUnchanged` flag to specify that the alpha channel should be copied to the destination image unchanged.

Return Value

`kvImageNoError`, otherwise it returns one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

If you want to allocate the memory for the `tempBuffer` parameter yourself, call this function twice, as follows:

1. To determine the minimum size for the temporary buffer, the first time you call this function pass the `kvImageGetTempBufferSize` flag. Pass the same values for all other parameters that you intend to use in for the second call. The function returns the required minimum size, which should be a positive value. (A negative returned value indicates an error.) The `kvImageGetTempBufferSize` flag prevents the function from performing any processing other than to determine the minimum buffer size.

2. After you allocate enough space for a buffer of the returned size, call the function a second time, passing a valid pointer in the `tempBuffer` parameter. This time, do not pass the `kvImageGetTempBufferSize` flag.

The morphological operation Max is a special case of the dilation operation. In the Max operation, all the elements of the kernel have the same value. The actual value does not matter; only the size of the kernel is significant. vImage optimizes this special case, so the Max function is considerably faster than the Dilate function called with a uniform kernel.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`Morphology.h`

vImageMax_Planar8

Maximizes a region of interest within a Planar8 source image using an M x N kernel.

```
vImage_Error vImageMax_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags
);
```

Parameters

`src`

A pointer to a vImage buffer structure that contains data for the source image.

`dest`

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

The size (number of rows and number of columns) of the destination buffer also specifies the size of the region of interest in the source buffer.

tempBuffer
: A pointer to a temporary buffer. If you pass `NULL`, the function allocates the buffer, then deallocates it before returning. Alternatively, you can allocate the buffer yourself, in which case you are responsible for deallocating it when you is no longer need it.

  If you want to allocate the buffer yourself, see the Discussion for information on how to determine the minimum size that you must allocate.

srcOffsetToROI_X
: The horizontal offset, in pixels, to the upper-left pixel of the region of interest within the source image.

srcOffsetToROI_Y
: The vertical offset, in pixels, to the upper-left pixel of the region of interest within the source image.

kernel_height
: The height of the kernel in pixels. This value must be odd.

kernel_width
: The width of the kernel in pixels. This value must be odd.

flags
: The options to use when performing the morphological operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading. Set the `kvImageLeaveAlphaUnchanged` flag to specify that the alpha channel should be copied to the destination image unchanged.

Return Value
`kvImageNoError`, otherwise it returns one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
If you want to allocate the memory for the `tempBuffer` parameter yourself, call this function twice, as follows:

1. To determine the minimum size for the temporary buffer, the first time you call this function pass the `kvImageGetTempBufferSize` flag. Pass the same values for all other parameters that you intend to use in for the second call. The function returns the required minimum size, which should be a positive value. (A negative returned value indicates an error.) The `kvImageGetTempBufferSize` flag prevents the function from performing any processing other than to determine the minimum buffer size.

2. After you allocate enough space for a buffer of the returned size, call the function a second time, passing a valid pointer in the `tempBuffer` parameter. This time, do not pass the `kvImageGetTempBufferSize` flag.

The morphological operation Max is a special case of the dilation operation. In the Max operation, all the elements of the kernel have the same value. The actual value does not matter; only the size of the kernel is significant. vImage optimizes this special case, so the Max function is considerably faster than the Dilate function called with a uniform kernel.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`Morphology.h`

vImageMax_PlanarF

Maximizes with a region of interest within a PlanarF source image using an M x N kernel.

```
vImage_Error vImageMax_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains data for the source image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

The size (number of rows and number of columns) of the destination buffer also specifies the size of the region of interest in the source buffer.

tempBuffer

A pointer to a temporary buffer. If you pass `NULL`, the function allocates the buffer, then deallocates it before returning. Alternatively, you can allocate the buffer yourself, in which case you are responsible for deallocating it when you is no longer need it.

If you want to allocate the buffer yourself, see the Discussion for information on how to determine the minimum size that you must allocate.

srcOffsetToROI_X

The horizontal offset, in pixels, to the upper-left pixel of the region of interest within the source image.

srcOffsetToROI_Y

The vertical offset, in pixels, to the upper-left pixel of the region of interest within the source image.

kernel_height

The height of the kernel in pixels. This value must be odd.

kernel_width

The width of the kernel in pixels. This value must be odd.

flags

The options to use when performing the morphological operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading. Set the `kvImageLeaveAlphaUnchanged` flag to specify that the alpha channel should be copied to the destination image unchanged.

Return Value

`kvImageNoError`, otherwise it returns one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

If you want to allocate the memory for the `tempBuffer` parameter yourself, call this function twice, as follows:

1. To determine the minimum size for the temporary buffer, the first time you call this function pass the `kvImageGetTempBufferSize` flag. Pass the same values for all other parameters that you intend to use in for the second call. The function returns the required minimum size, which should be a positive value. (A negative returned value indicates an error.) The `kvImageGetTempBufferSize` flag prevents the function from performing any processing other than to determine the minimum buffer size.

2. After you allocate enough space for a buffer of the returned size, call the function a second time, passing a valid pointer in the `tempBuffer` parameter. This time, do not pass the `kvImageGetTempBufferSize` flag.

The morphological operation Max is a special case of the dilation operation. In the Max operation, all the elements of the kernel have the same value. The actual value does not matter; only the size of the kernel is significant. vImage optimizes this special case, so the Max function is considerably faster than the Dilate function called with a uniform kernel.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`Morphology.h`

vImageMin_ARGB8888

Minimizes a region of interest within an ARGB8888 source image using an M x N kernel.

```
vImage_Error vImageMin_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags
);
```

Parameters

`src`

A pointer to a vImage buffer structure that contains data for the source image.

`dest`

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

The size (number of rows and number of columns) of the destination buffer also specifies the size of the region of interest in the source buffer.

tempBuffer

A pointer to a temporary buffer. If you pass `NULL`, the function allocates the buffer, then deallocates it before returning. Alternatively, you can allocate the buffer yourself, in which case you are responsible for deallocating it when you is no longer need it.

If you want to allocate the buffer yourself, see the Discussion for information on how to determine the minimum size that you must allocate.

srcOffsetToROI_X

The horizontal offset, in pixels, to the upper-left pixel of the region of interest within the source image.

srcOffsetToROI_Y

The vertical offset, in pixels, to the upper-left pixel of the region of interest within the source image.

kernel_height

The height of the kernel in pixels. This value must be odd.

kernel_width

The width of the kernel in pixels. This value must be odd.

flags

The options to use when performing the morphological operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading. Set the `kvImageLeaveAlphaUnchanged` flag to specify that the alpha channel should be copied to the destination image unchanged.

Return Value

`kvImageNoError`, otherwise it returns one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

If you want to allocate the memory for the `tempBuffer` parameter yourself, call this function twice, as follows:

1. To determine the minimum size for the temporary buffer, the first time you call this function pass the `kvImageGetTempBufferSize` flag. Pass the same values for all other parameters that you intend to use in for the second call. The function returns the required minimum size, which should be a positive value. (A negative returned value indicates an error.) The `kvImageGetTempBufferSize` flag prevents the function from performing any processing other than to determine the minimum buffer size.

2. After you allocate enough space for a buffer of the returned size, call the function a second time, passing a valid pointer in the `tempBuffer` parameter. This time, do not pass the `kvImageGetTempBufferSize` flag.

The morphological operation Min is a special case of the erosion operation. In the Min operation, all the elements of the kernel have the same value. The actual value does not matter; only the size of the kernel is significant. vImage optimizes this special case, so the Min function is considerably faster than the Erode function called with a uniform kernel.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Morphology.h`

vImageMin_ARGBFFFF

Minimizes a region of interest within an ARGBFFFF source image using an M x N kernel.

```
vImage_Error vImageMin_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains data for the source image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

The size (number of rows and number of columns) of the destination buffer also specifies the size of the region of interest in the source buffer.

tempBuffer

A pointer to a temporary buffer. If you pass `NULL`, the function allocates the buffer, then deallocates it before returning. Alternatively, you can allocate the buffer yourself, in which case you are responsible for deallocating it when you is no longer need it.

If you want to allocate the buffer yourself, see the Discussion for information on how to determine the minimum size that you must allocate.

srcOffsetToROI_X

The horizontal offset, in pixels, to the upper-left pixel of the region of interest within the source image.

srcOffsetToROI_Y

The vertical offset, in pixels, to the upper-left pixel of the region of interest within the source image.

kernel_height

The height of the kernel in pixels. This value must be odd.

kernel_width

The width of the kernel in pixels. This value must be odd.

flags

The options to use when performing the morphological operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading. Set the `kvImageLeaveAlphaUnchanged` flag to specify that the alpha channel should be copied to the destination image unchanged.

Return Value

`kvImageNoError`, otherwise it returns one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

If you want to allocate the memory for the `tempBuffer` parameter yourself, call this function twice, as follows:

1. To determine the minimum size for the temporary buffer, the first time you call this function pass the `kvImageGetTempBufferSize` flag. Pass the same values for all other parameters that you intend to use in for the second call. The function returns the required minimum size, which should be a positive value. (A negative returned value indicates an error.) The `kvImageGetTempBufferSize` flag prevents the function from performing any processing other than to determine the minimum buffer size.

2. After you allocate enough space for a buffer of the returned size, call the function a second time, passing a valid pointer in the `tempBuffer` parameter. This time, do not pass the `kvImageGetTempBufferSize` flag.

The morphological operation Min is a special case of the erosion operation. In the Min operation, all the elements of the kernel have the same value. The actual value does not matter; only the size of the kernel is significant. vImage optimizes this special case, so the Min function is considerably faster than the Erode function called with a uniform kernel.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`Morphology.h`

vImageMin_Planar8

Minimizes a region of interest within a Planar8 source image using an M x N kernel.

```
vImage_Error vImageMin_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains data for the source image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

The size (number of rows and number of columns) of the destination buffer also specifies the size of the region of interest in the source buffer.

tempBuffer
: A pointer to a temporary buffer. If you pass `NULL`, the function allocates the buffer, then deallocates it before returning. Alternatively, you can allocate the buffer yourself, in which case you are responsible for deallocating it when you is no longer need it.

  If you want to allocate the buffer yourself, see the Discussion for information on how to determine the minimum size that you must allocate.

srcOffsetToROI_X
: The horizontal offset, in pixels, to the upper-left pixel of the region of interest within the source image.

srcOffsetToROI_Y
: The vertical offset, in pixels, to the upper-left pixel of the region of interest within the source image.

kernel_height
: The height of the kernel in pixels. This value must be odd.

kernel_width
: The width of the kernel in pixels. This value must be odd.

flags
: The options to use when performing the morphological operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading. Set the `kvImageLeaveAlphaUnchanged` flag to specify that the alpha channel should be copied to the destination image unchanged.

Return Value
`kvImageNoError`, otherwise it returns one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
If you want to allocate the memory for the `tempBuffer` parameter yourself, call this function twice, as follows:

1. To determine the minimum size for the temporary buffer, the first time you call this function pass the `kvImageGetTempBufferSize` flag. Pass the same values for all other parameters that you intend to use in for the second call. The function returns the required minimum size, which should be a positive value. (A negative returned value indicates an error.) The `kvImageGetTempBufferSize` flag prevents the function from performing any processing other than to determine the minimum buffer size.

2. After you allocate enough space for a buffer of the returned size, call the function a second time, passing a valid pointer in the `tempBuffer` parameter. This time, do not pass the `kvImageGetTempBufferSize` flag.

The morphological operation Min is a special case of the erosion operation. In the Min operation, all the elements of the kernel have the same value. The actual value does not matter; only the size of the kernel is significant. vImage optimizes this special case, so the Min function is considerably faster than the Erode function called with a uniform kernel.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`Morphology.h`

vImageMin_PlanarF

Minimizes a region of interest within a PlanarF source image using an M x N kernel.

```
vImage_Error vImageMin_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   vImagePixelCount kernel_height,
   vImagePixelCount kernel_width,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains data for the source image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

The size (number of rows and number of columns) of the destination buffer also specifies the size of the region of interest in the source buffer.

tempBuffer

A pointer to a temporary buffer. If you pass `NULL`, the function allocates the buffer, then deallocates it before returning. Alternatively, you can allocate the buffer yourself, in which case you are responsible for deallocating it when you is no longer need it.

If you want to allocate the buffer yourself, see the Discussion for information on how to determine the minimum size that you must allocate.

srcOffsetToROI_X

The horizontal offset, in pixels, to the upper-left pixel of the region of interest within the source image.

srcOffsetToROI_Y

The vertical offset, in pixels, to the upper-left pixel of the region of interest within the source image.

kernel_height

The height of the kernel in pixels. This value must be odd.

kernel_width

The width of the kernel in pixels. This value must be odd.

flags

The options to use when performing the morphological operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading. Set the `kvImageLeaveAlphaUnchanged` flag to specify that the alpha channel should be copied to the destination image unchanged.

Return Value

`kvImageNoError`, otherwise it returns one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

If you want to allocate the memory for the `tempBuffer` parameter yourself, call this function twice, as follows:

1. To determine the minimum size for the temporary buffer, the first time you call this function pass the `kvImageGetTempBufferSize` flag. Pass the same values for all other parameters that you intend to use in for the second call. The function returns the required minimum size, which should be a positive value. (A negative returned value indicates an error.) The `kvImageGetTempBufferSize` flag prevents the function from performing any processing other than to determine the minimum buffer size.

2. After you allocate enough space for a buffer of the returned size, call the function a second time, passing a valid pointer in the `tempBuffer` parameter. This time, do not pass the `kvImageGetTempBufferSize` flag.

The morphological operation Min is a special case of the erosion operation. In the Min operation, all the elements of the kernel have the same value. The actual value does not matter; only the size of the kernel is significant. vImage optimizes this special case, so the Min function is considerably faster than the Erode function called with a uniform kernel.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`Morphology.h`

Document Revision History

This table describes the changes to *vImage Morphology Reference*.

Date	Notes
2007-07-12	Updated for Mac OS X v10.4.
	The content in this document was formerly part of *Optimizing Image Processing With vImage*.
	In Mac OS X 10.4, some of the parameter types for some functions changed. For example, `unsigned int` is now `uint32_t` or (for byte counts) `size_t` or (for pixel counts)`vImagePixelCount` (for pixel counts). Although the changes are fully backward-compatible with existing binaries, it is advisable to update and recompile existing code.

Index