vImage Convolution Reference

Performance > Graphics & Imaging

2007-07-12

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

vImage Convolution Reference

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

Overview

Convolution functions implement various techniques for smoothing or sharpening an image by replacing a pixel with a weighted sum of itself and nearby pixels. Image convolution does not alter the size of an image.

Each convolution function requires that you pass it a convolution kernel, which 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 consist 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.

Typically, you use normalized values for the convolution kernel. For floating-point formats, this means the sum of the elements of the kernel is 1.0. For integer formats, the sum of the elements of the kernel, divided by the given divisor, is 1. A non-normalized kernel either lightens or darkens the image.

For integer formats, the sum of any subset of elements of the kernel must be in the range -2^{24} to $2^{24} - 1$, inclusive to prevent integer overflow. If your kernel does not meet this restriction, either use a floating-point format or scale the kernel to use smaller values.

A convolution function transforms a source image as follows:

1. Places the kernel over the image so that the center element of the kernel lies over the source pixel.
2. For floating-point formats, performs this calculation:

$$\sum kernel_{(x,y)} * pixel_{(x,y)}$$

For integer formats, performs this calculation:

$$\frac{\sum kernel_{(x,y)} * pixel_{(x,y)}}{M * N}$$

3. Assigns the result to the destination pixel.

If the image is in a planar format, the convolution operation uses the single-channel values of the pixels directly. If the image is in an interleaved format, the convolution operation processes each channel (alpha, red, green, and blue) separately. In both the planar and interleaved format, the kernel itself is always planar.

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 these cases you must pass a flag that specifies a technique for the convolution function to use: `kvImageCopyInPlace`, `kvImageBackgroundColorFill`, `kvImageEdgeExtend`, and `kvImageTruncateKernel`. For a discussion of these options, see *vImage Data Types and Constants Reference*.

Functions by Task

Deconvolving

`vImageRichardsonLucyDeConvolve_ARGBFFFF` (page 31)
: Sharpens an ARGBFFFF image by undoing a previous convolution that blurred the image, such as diffraction effects in a camera lens.

`vImageRichardsonLucyDeConvolve_ARGB8888` (page 29)
: Sharpens an ARGB8888 image by undoing a previous convolution that blurred the image, such as diffraction effects in a camera lens.

`vImageRichardsonLucyDeConvolve_PlanarF` (page 35)
: Sharpens a PlanarF image by undoing a previous convolution that blurred the image, such as diffraction effects in a camera lens.

`vImageRichardsonLucyDeConvolve_Planar8` (page 33)
: Sharpens a Planar8 image by undoing a previous convolution that blurred the image, such as diffraction effects in a camera lens.

Convolving Without Bias

`vImageConvolve_ARGBFFFF` (page 23)
: Convolves a region of interest within an ARGBFFFF source image by an M x N kernel.

`vImageConvolve_ARGB8888` (page 22)
: Convolves a region of interest within a source image by an M x N kernel, then divides the pixel values by a divisor.

vImageConvolve_PlanarF (page 27)
Convolves a region of interest within a source image by an M x N kernel.

vImageConvolve_Planar8 (page 25)
Convolves a region of interest within a source image by an M x N kernel, then divides the pixel values by a divisor.

Convolving With a Bias

vImageConvolveWithBias_ARGB8888 (page 15)
Convolves a region of interest within an ARGB8888 source image by an M x N kernel, then normalizes the pixel values.

vImageConvolveWithBias_PlanarF (page 20)
Convolves a region of interest within a PlanarF source image by an M x N kernel.

vImageConvolveWithBias_Planar8 (page 18)
Convolves a region of interest within a Planar8 source image by an M x N kernel, then normalizes the pixel values.

vImageConvolveWithBias_ARGBFFFF (page 16)
Convolves a region of interest within an ARGBFFFF source image by an M x N kernel.

Convolving With Multiple Kernels

vImageConvolveMultiKernel_ARGBFFFF (page 13)
Convolves each channel of a region of interest within an ARGBFFFF source image by one of the four M x N kernels.

vImageConvolveMultiKernel_ARGB8888 (page 11)
Convolves each channel of a region of interest within an ARGB8888 source image by one of the four M x N kernels, then divides the pixel values by one of the four divisors.

Convolving With High-Speed Box and Tent Filters

vImageBoxConvolve_Planar8 (page 9)
Convolves a region of interest within a Planar8 source image by an implicit M x N kernel that has the effect of a box filter.

vImageBoxConvolve_ARGB8888 (page 8)
Convolves a region of interest within an ARGB8888 source image by an implicit M x N kernel that has the effect of a box filter.

vImageTentConvolve_Planar8 (page 39)
Convolves a region of interest within a Planar8 source image by an implicit M x N kernel that has the effect of a tent filter.

vImageTentConvolve_ARGB8888 (page 37)
Convolves a region of interest within an ARGB8888 source image by an implicit M x N kernel that has the effect of a tent filter.

Getting the Minimum Buffer Size

`vImageGetMinimumTempBufferSizeForConvolution` (page 28)
Returns the minimum size, in bytes, for the temporary buffer that the caller supplies to any of the convolution functions. (Deprecated. Use the `kvImageGetTempBufferSize` flag with the appropriate convolution function instead of calling this function.)

Functions

vImageBoxConvolve_ARGB8888

Convolves a region of interest within an ARGB8888 source image by an implicit M x N kernel that has the effect of a box filter.

```
vImage_Error vImageBoxConvolve_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   uint32_t kernel_height,
   uint32_t kernel_width,
   Pixel_8888 backgroundColor,
   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.

backgroundColor
A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

flags
The options to use when performing the convolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

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

Discussion
This function uses an implicit divisor and an implicit kernel of specified size instead of a kernel provided by the caller.

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.

Availability
Available in Mac OS X v10.4 and later.

See Also
`vImageConvolve_ARGB8888` (page 22)
`vImageTentConvolve_ARGB8888` (page 37)

Declared In
`Convolution.h`

vImageBoxConvolve_Planar8

Convolves a region of interest within a Planar8 source image by an implicit M x N kernel that has the effect of a box filter.

```
vImage_Error vImageBoxConvolve_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   uint32_t kernel_height,
   uint32_t kernel_width,
   Pixel_8 backgroundColor,
   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.

backgroundColor

A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

flags

The options to use when performing the convolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

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

Discussion
This function uses an implicit divisor and an implicit kernel of specified size instead of a kernel provided by the caller.

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.

Availability
Available in Mac OS X v10.4 and later.

See Also
`vImageConvolve_Planar8` (page 25)
`vImageTentConvolve_Planar8` (page 39)

Declared In
`Convolution.h`

vImageConvolveMultiKernel_ARGB8888

Convolves each channel of a region of interest within an ARGB8888 source image by one of the four M x N kernels, then divides the pixel values by one of the four divisors.

```
vImage_Error vImageConvolveMultiKernel_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const int16_t *kernels[4],
   uint32_t kernel_height,
   uint32_t kernel_width,
   const int32_t divisors[4],
   const int32_t biases[4],
   Pixel_8888 backgroundColor,
   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.

kernels

An array of pointers to the data for four kernels. The first kernel is for the alpha channel, the second for red, the third for green, and the fourth for blue. The data for each kernel is a packed array of integer values.

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.

divisors

An array of values, for normalization purposes, to divide into the convolution results. Supply one value for each channel.

biases

An array of four values to be added to each element of the convolution result for one channel, before clipping. The first value is for the alpha channel, the second for red, the third for green, and the fourth for blue

backgroundColor

A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

flags

The options to use when performing the convolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

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.

Availability
Available in Mac OS X v10.4 and later.

See Also
`vImageConvolveWithBias_ARGB8888` (page 15)

Declared In
`Convolution.h`

vImageConvolveMultiKernel_ARGBFFFF

Convolves each channel of a region of interest within an ARGBFFFF source image by one of the four M x N kernels.

```
vImage_Error vImageConvolveMultiKernel_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const float *kernels[4],
   uint32_t kernel_height,
   uint32_t kernel_width,
   const float biases[4],
   Pixel_FFFF backgroundColor,
   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.

kernels

An array of pointers to the data for four kernels. The first kernel is for the alpha channel, the second for red, the third for green, and the fourth for blue. The data for each kernel is a packed array of floating-point values.

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.

biases

An array of four values to be added to each element of the convolution result for one channel, before clipping. The first value is for the alpha channel, the second for red, the third for green, and the fourth for blue

backgroundColor

A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

flags

The options to use when performing the convolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

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.

Availability
Available in Mac OS X v10.4 and later.

See Also
vImageConvolveWithBias_ARGBFFFF (page 16)

Declared In
Convolution.h

vImageConvolveWithBias_ARGB8888

Convolves a region of interest within an ARGB8888 source image by an M x N kernel, then normalizes the pixel values.

```
vImage_Error vImageConvolveWithBias_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const int16_t *kernel,
   uint32_t kernel_height,
   uint32_t kernel_width,
   int32_t divisor,
   int32_t bias,
   Pixel_8888 backgroundColor,
   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

A pointer to the convolution 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.

divisor
The value, for normalization purposes, to divide into the convolution results.

bias
The value to add to each element in the convolution result, before applying the divisor or performing any clipping.

backgroundColor
A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

flags
The options to use when performing the convolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

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.

Availability
Available in Mac OS X v10.4 and later.

See Also
`vImageConvolve_ARGB8888` (page 22)

Declared In
`Convolution.h`

vImageConvolveWithBias_ARGBFFFF

Convolves a region of interest within an ARGBFFFF source image by an M x N kernel.

```
vImage_Error vImageConvolveWithBias_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const float *kernel,
   uint32_t kernel_height,
   uint32_t kernel_width,
   float bias,
   Pixel_FFFF backgroundColor,
   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

A pointer to the convolution 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.

bias

The value to add to each element in the convolution result, before performing any clipping.

backgroundColor

A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

flags

The options to use when performing the convolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

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.

Availability
Available in Mac OS X v10.4 and later.

See Also
`vImageConvolve_ARGBFFFF` (page 23)

Declared In
`Convolution.h`

vImageConvolveWithBias_Planar8

Convolves a region of interest within a Planar8 source image by an M x N kernel, then normalizes the pixel values.

```
vImage_Error vImageConvolveWithBias_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const int16_t *kernel,
   uint32_t kernel_height,
   uint32_t kernel_width,
   int32_t divisor,
   int32_t bias,
   Pixel_8 backgroundColor,
   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

A pointer to the convolution 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.

divisor

The value, for normalization purposes, to divide into the convolution results.

bias

The value to add to each element in the convolution result, before applying the divisor or performing any clipping.

backgroundColor

A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

flags

The options to use when performing the convolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

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.

Availability
Available in Mac OS X v10.4 and later.

See Also
`vImageConvolve_Planar8` (page 25)

Declared In
`Convolution.h`

vImageConvolveWithBias_PlanarF

Convolves a region of interest within a PlanarF source image by an M x N kernel.

```
vImage_Error vImageConvolveWithBias_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const float *kernel,
   uint32_t kernel_height,
   uint32_t kernel_width,
   float bias,
   Pixel_F backgroundColor,
   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

A pointer to the convolution 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.

bias

The value to add to each element in the convolution result, before performing any clipping.

backgroundColor

A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

flags

The options to use when performing the convolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

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.

Availability
Available in Mac OS X v10.4 and later.

See Also
`vImageConvolve_PlanarF` (page 27)

Declared In
`Convolution.h`

vImageConvolve_ARGB8888

Convolves a region of interest within a source image by an M x N kernel, then divides the pixel values by a divisor.

```
vImage_Error vImageConvolve_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const int16_t *kernel,
   uint32_t kernel_height,
   uint32_t kernel_width,
   int32_t divisor,
   Pixel_8888 backgroundColor,
   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
A pointer to the convolution 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.

divisor
A value to divide the results of the convolution by. This is commonly used for normalization.

backgroundColor
A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

flags
The options to use when performing the convolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

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.

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

Declared In
`Convolution.h`

vImageConvolve_ARGBFFFF

Convolves a region of interest within an ARGBFFFF source image by an M x N kernel.

```
vImage_Error vImageConvolve_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const float *kernel,
   uint32_t kernel_height,
   uint32_t kernel_width,
   Pixel_FFFF backgroundColor,
   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. offsets to a point within the source image to define the upper left-hand point of the region of interest.

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 convolution 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.

backgroundColor

A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

flags

The options to use when performing the convolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

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.

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

Declared In
`Convolution.h`

vImageConvolve_Planar8

Convolves a region of interest within a source image by an M x N kernel, then divides the pixel values by a divisor.

```
vImage_Error vImageConvolve_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const int16_t *kernel,
   uint32_t kernel_height,
   uint32_t kernel_width,
   int32_t divisor,
   Pixel_8 backgroundColor,
   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

A pointer to the convolution 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.

divisor

A value to divide the results of the convolution with. This is commonly used for normalization.

backgroundColor

A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

flags

The options to use when performing the convolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

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.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Convolution.h`

vImageConvolve_PlanarF

Convolves a region of interest within a source image by an M x N kernel.

```
vImage_Error vImageConvolve_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const float *kernel,
   uint32_t kernel_height,
   uint32_t kernel_width,
   Pixel_F backgroundColor,
   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

A pointer to the convolution 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.

backgroundColor

A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

flags

The options to use when performing the convolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

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.

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

Declared In
`Convolution.h`

vImageGetMinimumTempBufferSizeForConvolution

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

```
size_t vImageGetMinimumTempBufferSizeForConvolution (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   uint32_t kernel_height,
   uint32_t kernel_width,
   vImage_Flags flags,
   size_t bytesPerPixel
);
```

Parameters

src

A pointer to the vImage buffer structure that you plan to pass to the convolution function.

dest

A pointer to the vImage buffer structure that you plan to pass to the convolution function.

kernel_height

The height, in pixels, of the kernel that you plan to use in the convolution function.

kernel_width

The width, in pixels, of the kernel that you plan to use in the convolution function.

flags

The flags that you plan to pass to the convolution function.

bytesPerPixel

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 does not depend on the *data* or *rowBytes* fields of the *src* or *dest* parameters; it only uses the *height* and *width* fields from those parameters. If the size of the images you are processing stay the same, then the required size of the buffer also stays the same. More specifically, if, between two calls to `vImageGetMinimumTempBufferSizeForConvolution`, 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 `vImageGetMinimumTempBufferSizeForConvolution` does not increase. This makes it easy to reuse the same temporary buffer when you are processing a number of images of the same size, as in tiling.

Availability

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.4.

Declared In

`Convolution.h`

vImageRichardsonLucyDeConvolve_ARGB8888

Sharpens an ARGB8888 image by undoing a previous convolution that blurred the image, such as diffraction effects in a camera lens.

```
vImage_Error vImageRichardsonLucyDeConvolve_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const int16_t *kernel,
   const int16_t *kernel2,
   uint32_t kernel_height,
   uint32_t kernel_width,
   uint32_t kernel_height2,
   uint32_t kernel_width2,
   int32_t divisor,
   int32_t divisor2,
   Pixel_8888 backgroundColor,
   uint32_t iterationCount,
   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

A pointer to the deconvolution kernel data, which must be a packed array without any padding. The kernel expresses a blurring convolution or point-spread function.

kernel2

A pointer to the data of a second kernel, which must be a packed array without any padding. Supply this kernel only if the first kernel is asymmetrical; otherwise pass `NULL`.

kernel_height

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

kernel_width

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

kernel_height2

The height of the second kernel in pixels (ignored if *kernel2* is `NULL`). This value must be odd.

kernel_width2

The width of the second kernel in pixels (ignored if *kernel2* is `NULL`). This value must be odd.

divisor

The divisor to be used in convolutions with the first kernel.

divisor2

The divisor to be used in convolutions with the second kernel.

backgroundColor

A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

iterationCount

The number of times to iterate the deconvolution algorithm.

flags

The options to use when performing the deconvolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

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

Discussion
This function performs a Richardson-Lucy deconvolution of a region of interest within a source image by an M x N kernel, performing a specified number of iterations and placing the result in a destination buffer.

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.

Availability
Available in Mac OS X v10.4 and later.

See Also
`vImageRichardsonLucyDeConvolve_Planar8` (page 33)

Declared In
`Convolution.h`

vImageRichardsonLucyDeConvolve_ARGBFFFF

Sharpens an ARGBFFFF image by undoing a previous convolution that blurred the image, such as diffraction effects in a camera lens.

```
vImage_Error vImageRichardsonLucyDeConvolve_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const float *kernel,
   const float *kernel2,
   uint32_t kernel_height,
   uint32_t kernel_width,
   uint32_t kernel_height2,
   uint32_t kernel_width2,
   Pixel_FFFF backgroundColor,
   uint32_t iterationCount,
   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

A pointer to the deconvolution kernel data, which must be a packed array without any padding. The kernel expresses a blurring convolution or point-spread function.

kernel2

A pointer to the data of a second kernel, which must be a packed array without any padding. Supply this kernel only if the first kernel is asymmetrical; otherwise pass `NULL`.

kernel_height

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

kernel_width

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

kernel_height2

The height of the second kernel in pixels (ignored if *kernel2* is `NULL`). This value must be odd.

kernel_width2

The width of the second kernel in pixels (ignored if *kernel2* is `NULL`). This value must be odd.

backgroundColor

A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

iterationCount

The number of times to iterate the deconvolution algorithm.

flags

The options to use when performing the deconvolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

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

Discussion
This function performs a Richardson-Lucy deconvolution of a region of interest within a source image by an M x N kernel, performing a specified number of iterations and placing the result in a destination buffer.

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.

Availability
Available in Mac OS X v10.4 and later.

See Also
`vImageRichardsonLucyDeConvolve_PlanarF` (page 35)

Declared In
`Convolution.h`

vImageRichardsonLucyDeConvolve_Planar8

Sharpens a Planar8 image by undoing a previous convolution that blurred the image, such as diffraction effects in a camera lens.

```
vImage_Error vImageRichardsonLucyDeConvolve_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const int16_t *kernel,
   const int16_t *kernel2,
   uint32_t kernel_height,
   uint32_t kernel_width,
   uint32_t kernel_height2,
   uint32_t kernel_width2,
   int32_t divisor,
   int32_t divisor2,
   Pixel_8 backgroundColor,
   uint32_t iterationCount,
   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

A pointer to the deconvolution kernel data, which must be a packed array without any padding. The kernel expresses a blurring convolution or point-spread function.

kernel2

A pointer to the data of a second kernel, which must be a packed array without any padding. Supply this kernel only if the first kernel is asymmetrical; otherwise pass `NULL`.

kernel_height

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

kernel_width

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

kernel_height2

The height of the second kernel in pixels (ignored if *kernel2* is `NULL`). This value must be odd.

kernel_width2

The width of the second kernel in pixels (ignored if *kernel2* is `NULL`). This value must be odd.

backgroundColor

A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

iterationCount

The number of times to iterate the deconvolution algorithm.

flags

The options to use when performing the deconvolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

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

Discussion
This function performs a Richardson-Lucy deconvolution of a region of interest within a source image by an M x N kernel, performing a specified number of iterations and placing the result in a destination buffer.

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.

Availability
Available in Mac OS X v10.4 and later.

See Also
`vImageRichardsonLucyDeConvolve_ARGB8888` (page 29)

Declared In
`Convolution.h`

vImageRichardsonLucyDeConvolve_PlanarF

Sharpens a PlanarF image by undoing a previous convolution that blurred the image, such as diffraction effects in a camera lens.

```
vImage_Error vImageRichardsonLucyDeConvolve_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   const float *kernel,
   const float *kernel2,
   uint32_t kernel_height,
   uint32_t kernel_width,
   uint32_t kernel_height2,
   uint32_t kernel_width2,
   Pixel_F backgroundColor,
   uint32_t iterationCount,
   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

A pointer to the deconvolution kernel data, which must be a packed array without any padding. The kernel expresses a blurring convolution or point-spread function.

kernel2

A pointer to the data of a second kernel, which must be a packed array without any padding. Supply this kernel only if the first kernel is asymmetrical; otherwise pass `NULL`.

kernel_height

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

kernel_width

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

kernel_height2

The height of the second kernel in pixels (ignored if *kernel2* is `NULL`). This value must be odd.

kernel_width2

The width of the second kernel in pixels (ignored if *kernel2* is `NULL`). This value must be odd.

backgroundColor

A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

iterationCount

The number of times to iterate the deconvolution algorithm.

flags

The options to use when performing the deconvolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

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

Discussion
The function performs a Richardson-Lucy deconvolution of a region of interest within a source image by an M x N kernel, performing a specified number of iterations and placing the result in a destination buffer.

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.

Availability
Available in Mac OS X v10.4 and later.

See Also
`vImageRichardsonLucyDeConvolve_ARGBFFFF` (page 31)

Declared In
`Convolution.h`

vImageTentConvolve_ARGB8888

Convolves a region of interest within an ARGB8888 source image by an implicit M x N kernel that has the effect of a tent filter.

```
vImage_Error vImageTentConvolve_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   uint32_t kernel_height,
   uint32_t kernel_width,
   Pixel_8888 backgroundColor,
   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.

backgroundColor

A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

flags

The options to use when performing the convolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

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

Discussion

This function uses an implicit divisor and an implicit kernel of specified size instead of a kernel provided by the caller.

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.

Availability

Available in Mac OS X v10.4 and later.

See Also

`vImageConvolve_ARGB8888` (page 22)

`vImageBoxConvolve_ARGB8888` (page 8)

Declared In
`Convolution.h`

vImageTentConvolve_Planar8

Convolves a region of interest within a Planar8 source image by an implicit M x N kernel that has the effect of a tent filter.

```
vImage_Error vImageTentConvolve_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   uint32_t kernel_height,
   uint32_t kernel_width,
   Pixel_8 backgroundColor,
   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.

backgroundColor

A background color. If you supply a color, you must also set the `kvImageBackgroundColorFill` flag, otherwise the function ignores the color.

flags

The options to use when performing the convolution operation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageCopyInPlace`, `kvImageTruncateKernel`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`.

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

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

Discussion
This function uses an implicit divisor and an implicit kernel of specified size instead of a kernel provided by the caller.

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.

Availability
Available in Mac OS X v10.4 and later.

See Also
`vImageConvolve_Planar8` (page 25)
`vImageBoxConvolve_Planar8` (page 9)

Declared In
`Convolution.h`

Document Revision History

This table describes the changes to *vImage Convolution 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*.

Document Revision History

Index

V