vImage Geometry 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 Geometry Reference

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

Overview

Geometric functions rotate, resize, and distort the geometry of images. vImage provides both high-level (rotation, scaling, and warping) and low-level geometric functions (reflection, shearing, and low-level rotation).

Most vImage geometric functions resample image data to avoid creating artifacts, such as interference patterns, in the destination image. vImage uses resampling kernels, which combine data from a target pixel and other nearby pixels to calculate a value for the destination pixel, a procedure somewhat similar to that used for convolution. However, for geometric operations, the resampling kernel itself is resampled during the process of pairing kernel values against the sampled pixel data. The kernel is evaluated at both fractional and integral pixel locations. This has implications for the nature of the kernel—which must be supplied as a function rather than as an M by N matrix. A resampling kernel function is also called a resampling filter, or simply a filter.

For almost all geometric operations, vImage supplies a default resampling filter unless you set the flag `kvImageHighQualityResampling`, in which case vImage uses a higher-quality filter, but that filter may be slower to use.

The reflection and high-level rotation functions don't resample. The shear functions can either use a default resampling filter or, if you require more control, a custom filter that you provide.

Functions by Task

Applying Affine Transforms

`vImageAffineWarp_ARGBFFFF` (page 9)
Applies an affine transform to an ARGBFFFF source image.

`vImageAffineWarp_ARGB8888` (page 8)
Applies an affine transform to an ARGB8888 source image.

`vImageAffineWarp_PlanarF` (page 12)
Applies an affine transform to a PlanarF source image.

`vImageAffineWarp_Planar8` (page 11)
Applies an affine transform to a Planar8 source image.

Reflecting

`vImageHorizontalReflect_ARGBFFFF` (page 17)
Reflects an ARGBFFFF source image left to right across the center vertical line of the image.

`vImageHorizontalReflect_PlanarF` (page 18)
Reflects a PlanarF source image left to right across the center vertical line of the image, placing the result in a destination buffer.

`vImageHorizontalReflect_Planar8` (page 18)
Reflects a Planar9 source image left to right across the center vertical line of the image.

`vImageHorizontalReflect_ARGB8888` (page 16)
Reflects an ARGB8888 source image left to right across the center vertical line of the image.

`vImageVerticalReflect_ARGBFFFF` (page 42)
Reflects an ARGBFFFF source image top to bottom across the center vertical line of the image.

`vImageVerticalReflect_ARGB8888` (page 42)
Reflects an ARGBFFFF source image top to bottom across the center vertical line of the image.

`vImageVerticalReflect_PlanarF` (page 44)
Reflects a PlanarF source image top to bottom across the center vertical line of the image.

`vImageVerticalReflect_Planar8` (page 43)
Reflects a Planar 8 source image top to bottom across the center vertical line of the image.

Shearing

`vImageHorizontalShear_ARGBFFFF` (page 20)
Performs a horizontal shear operation on a region of interest of an ARGBFFFF source image.

`vImageHorizontalShear_ARGB8888` (page 19)
Performs a horizontal shear operation on a region of interest of an ARGB8888 source image.

`vImageHorizontalShear_PlanarF` (page 23)
Performs a horizontal shear operation on a region of interest of a PlanarF source image.

`vImageHorizontalShear_Planar8` (page 22)
Performs a horizontal shear operation on a region of interest of a Planar8 source image.

`vImageVerticalShear_ARGBFFFF` (page 46)
Performs a vertical shear operation on a region of interest of an ARGBFFFF source image.

`vImageVerticalShear_ARGB8888` (page 45)
Performs a vertical shear operation on a region of interest of an ARGB8888 source image.

`vImageVerticalShear_PlanarF` (page 49)
Performs a vertical shear operation on a region of interest of a PlanarF source image.

`vImageVerticalShear_Planar8` (page 47)
Performs a vertical shear operation on a region of interest of a Planar8 source image.

Rotating

`vImageRotate90_ARGBFFFF` (page 28)
Rotates an ARGBFFFF source image by the provided factor of 90.

`vImageRotate90_ARGB8888` (page 27)
Rotates an ARGB8888 source image by the provided factor of 90.

`vImageRotate90_PlanarF` (page 30)
Rotates a PlanarF source image by the provided factor of 90.

`vImageRotate90_Planar8` (page 29)
Rotates a Planar8 source image by the provided factor of 90.

`vImageRotate_ARGBFFFF` (page 33)
Rotates an ARGBFFFF source image by the provided angle.

`vImageRotate_ARGB8888` (page 31)
Rotates an ARGB8888 source image by the provided angle.

`vImageRotate_PlanarF` (page 36)
Rotates a PlanarF source image by the provided angle.

`vImageRotate_Planar8` (page 34)
Rotates a Planar8 source image by the provided angle.

Scaling

`vImageScale_ARGBFFFF` (page 38)
Scales an ARGBFFFF source image to fit a destination buffer.

`vImageScale_ARGB8888` (page 37)
Scales an ARGB8888 source image to fit a destination buffer.

`vImageScale_PlanarF` (page 40)
Scales a PlanarF source image to fit a destination buffer.

`vImageScale_Planar8` (page 39)
Scales a Planar8 source image to fit a destination buffer.

Resampling

`vImageDestroyResamplingFilter` (page 14)
Disposes of a resampling filter object.

`vImageGetResamplingFilterSize` (page 15)
Returns the minimum size, in bytes, for the buffer needed by the function `vImageNewResamplingFilterForFunctionUsingBuffer`.

`vImageNewResamplingFilter` (page 25)
Creates a resampling filter object that corresponds to the default kernel supplied by the vImage framework.

`vImageNewResamplingFilterForFunctionUsingBuffer` (page 25)
Creates a resampling filter object that encapsulates a resampling kernel function that you provide.

Getting the Buffer Size

`vImageGetMinimumGeometryTempBufferSize` (page 14)
Returns the minimum size, in bytes, for the temporary buffer needed by a high-level geometry function. (Deprecated. Use the `kvImageGetTempBufferSize` flag with the appropriate geometry function instead of calling this function.)

Functions

vImageAffineWarp_ARGB8888

Applies an affine transform to an ARGB8888 source image.

```
vImage_Error vImageAffineWarp_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   const vImage_AffineTransform *transform,
   Pixel_8888 backColor,
   vImage_Flags flags
);
```

Parameters

src
A pointer to a vImage buffer structure that contains the source image whose data you want to transform.

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.

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.

transform
The affine transformation matrix to apply to the source image.

backgroundColor
A background color. Pass a pixel value only if you also set the `kvImageBackgroundColorFill` flag.

flags

The options to use when applying the transform. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageBackgroundColorFill` or `kvImageEdgeExtend`.

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

This function ignores the `kvImageLeaveAlphaUnchanged` flag.

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

Discussion
This function maps each pixel in the source image `[x, y]` to a new position `[x', y']` in the destination image by the formula:

```
(x', y') = (x, y) * transform
```

where `transform` is the 3x3 affine transformation matrix.

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
`Geometry.h`

vImageAffineWarp_ARGBFFFF

Applies an affine transform to an ARGBFFFF source image.

```
vImage_Error vImageAffineWarp_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   const vImage_AffineTransform *transform,
   Pixel_FFFF backColor,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to transform.

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.

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.

transform

The affine transformation matrix to apply to the source image.

backgroundColor

A background color. Pass a pixel value only if you also set the `kvImageBackgroundColorFill` flag.

flags

The options to use when applying the transform. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageBackgroundColorFill` or `kvImageEdgeExtend`.

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

This function ignores the `kvImageLeaveAlphaUnchanged` flag.

Return Value

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

Discussion

This function maps each pixel in the source image `[x, y]` to a new position `[x', y']` in the destination image by the formula:

```
(x', y') = (x, y) * transform
```

where `transform` is the 3x3 affine transformation matrix.

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
`Geometry.h`

vImageAffineWarp_Planar8

Applies an affine transform to a Planar8 source image.

```
vImage_Error vImageAffineWarp_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   const vImage_AffineTransform *transform,
   Pixel_8 backColor,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to transform.

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.

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.

transform

The affine transformation matrix to apply to the source image.

backgroundColor

A background color. Pass a pixel value only if you also set the `kvImageBackgroundColorFill` flag.

flags

The options to use when applying the transform. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageBackgroundColorFill` or `kvImageEdgeExtend`.

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

This function ignores the `kvImageLeaveAlphaUnchanged` flag.

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

Discussion
This function maps each pixel in the source image `[x, y]` to a new position `[x', y']` in the destination image by the formula:

```
(x', y') = (x, y) * transform
```

where `transform` is the 3x3 affine transformation matrix.

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
`Geometry.h`

vImageAffineWarp_PlanarF

Applies an affine transform to a PlanarF source image.

```
vImage_Error vImageAffineWarp_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   const vImage_AffineTransform *transform,
   Pixel_F backColor,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to transform.

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.

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.

transform

The affine transformation matrix to apply to the source image.

backgroundColor

A background color. Pass a pixel value only if you also set the `kvImageBackgroundColorFill` flag.

flags

The options to use when applying the transform. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageBackgroundColorFill` or `kvImageEdgeExtend`.

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

This function ignores the `kvImageLeaveAlphaUnchanged` flag.

Return Value

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

Discussion

This function maps each pixel in the source image `[x, y]` to a new position `[x', y']` in the destination image by the formula:

```
(x', y') = (x, y) * transform
```

where `transform` is the 3x3 affine transformation matrix.

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
`Geometry.h`

vImageDestroyResamplingFilter

Disposes of a resampling filter object.

```
void vImageDestroyResamplingFilter (
   ResamplingFilter filter
);
```

Parameters

filter

The resampling filter object to dispose of.

Discussion
This function deallocates the memory associated with a resampling filter object that was created by calling the function `vImageNewResamplingFilter` (page 25). Do not directly deallocate this memory yourself.

Do not pass this function a resampling filter object created by the function `vImageNewResamplingFilterForFunctionUsingBuffer` (page 25). You are responsible for deallocating the memory associated with resampling filter objects created by that call yourself.

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

See Also
`vImageNewResamplingFilterForFunctionUsingBuffer` (page 25)

Declared In
`Geometry.h`

vImageGetMinimumGeometryTempBufferSize

Returns the minimum size, in bytes, for the temporary buffer needed by a high-level geometry function. (**Deprecated.** Use the `kvImageGetTempBufferSize` flag with the appropriate geometry function instead of calling this function.)

```
size_t vImageGetMinimumGeometryTempBufferSize (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImage_Flags flags,
   size_t bytesPerPixel
);
```

Parameters

src

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

dest

A pointer to a vImage buffer structure that you plan to pass to the geometry function. You must set the fields of this structure yourself, and allocate memory for its data. When you are done with the buffer structure, you must deallocate the memory.

flags

The flags that you plan to pass to the geometry 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 will also stay the same. More specifically, if, between two calls to `vImageGetMinimumGeometryTempBufferSize`, the `height` and `width` of the *src* and *dest* parameters do not increase, and the and the other parameters remain the same, then the result of the `vImageGetMinimumGeometryTempBufferSize` will 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

`Geometry.h`

vImageGetResamplingFilterSize

Returns the minimum size, in bytes, for the buffer needed by the function `vImageNewResamplingFilterForFunctionUsingBuffer`.

```
size_t vImageGetResamplingFilterSize (
   const float *xArray,
   float *yArray,
   unsigned long count,
   void *userData
);
```

Parameters

scale

The scale factor that you plan to pass to the function `vImageNewResamplingFilterForFunctionUsingBuffer`.

kernelFunc

The function pointer that you plan to pass to the function `vImageNewResamplingFilterForFunctionUsingBuffer`.

userData

The user data pointer that you plan to pass to the function `vImageNewResamplingFilterForFunctionUsingBuffer`.

kernelWidth

The kernel width that you plan to pass to the function `vImageNewResamplingFilterForFunctionUsingBuffer`.

flags

The flags that you plan to pass to the function `vImageNewResamplingFilterForFunctionUsingBuffer`.

Return Value
The minimum size, in bytes, of the buffer.

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

Declared In
`Geometry.h`

vImageHorizontalReflect_ARGB8888

Reflects an ARGB8888 source image left to right across the center vertical line of the image.

```
vImage_Error vImageHorizontalReflect_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to reflect.

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.

flags

The options to use when performing the reflection. 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 does not scale or resample. The source and destination buffers must have the same height and the same width.

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

Declared In
`Geometry.h`

vImageHorizontalReflect_ARGBFFFF

Reflects an ARGBFFFF source image left to right across the center vertical line of the image.

```
vImage_Error vImageHorizontalReflect_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to reflect.

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.

flags

The options to use when performing the reflection. 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 does not scale or resample. The source and destination buffers must have the same height and the same width.

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

Declared In
`Geometry.h`

vImageHorizontalReflect_Planar8

Reflects a Planar9 source image left to right across the center vertical line of the image.

```
vImage_Error vImageHorizontalReflect_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to reflect.

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.

flags

The options to use when performing the reflection. 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 does not scale or resample. The source and destination buffers must have the same height and the same width.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Geometry.h`

vImageHorizontalReflect_PlanarF

Reflects a PlanarF source image left to right across the center vertical line of the image, placing the result in a destination buffer.

```
vImage_Error vImageHorizontalReflect_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to reflect.

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.

flags

The options to use when performing the reflection. 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 does not scale or resample. The source and destination buffers must have the same height and the same width.

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

Declared In
`Geometry.h`

vImageHorizontalShear_ARGB8888

Performs a horizontal shear operation on a region of interest of an ARGB8888 source image.

```
vImage_Error vImageHorizontalShear_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   float xTranslate,
   float shearSlope,
   ResamplingFilter filter,
   Pixel_8888 backColor,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to shear.

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.

This 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 as the destination image 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.

xTranslate
A translation value for the horizontal direction.

shearSlope
The slope of the front edge of the sheared image, measured in a clockwise direction.

filter
The resampling filter to be used with this function. You create this object by calling `vImageNewResamplingFilter` (to use a default resampling filter supplied by vImage) or `vImageNewResamplingFilterForFunctionUsingBuffer` (to use a custom resampling filter that you supply). When the resampling filter is created, you can also set a scale factor that will be used in the horizontal shear operation.

backgroundColor
A background color. Pass a pixel value only if you also set the `kvImageBackgroundColorFill` flag.

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

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

The `kvImageBackgroundColorFill` and `kvImageEdgeExtend` flags are mutually exclusive. You must set exactly one of these flags.

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

Discussion
This function also translates and scales the image, both in the horizontal direction. The function transforms as much of the source image as it needs in order to attempt to fill the destination buffer, which means it can transform pixels outside the region of interest.

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

Declared In
`Geometry.h`

vImageHorizontalShear_ARGBFFFF

Performs a horizontal shear operation on a region of interest of an ARGBFFFF source image.

```
vImage_Error vImageHorizontalShear_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   float xTranslate,
   float shearSlope,
   ResamplingFilter filter,
   Pixel_FFFF backColor,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to shear.

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.

This 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 as the destination image 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.

xTranslate

A translation value for the horizontal direction.

shearSlope

The slope of the front edge of the sheared image, measured in a clockwise direction.

filter

The resampling filter to be used with this function. You create this object by calling `vImageNewResamplingFilter` (to use a default resampling filter supplied by vImage) or `vImageNewResamplingFilterForFunctionUsingBuffer` (to use a custom resampling filter that you supply). When the resampling filter is created, you can also set a scale factor that will be used in the horizontal shear operation.

backgroundColor

A background color. Pass a pixel value only if you also set the `kvImageBackgroundColorFill` flag.

flags

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

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

The `kvImageBackgroundColorFill` and `kvImageEdgeExtend` flags are mutually exclusive. You must set exactly one of these flags.

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

Discussion
This function also translates and scales the image, both in the horizontal direction. The function transforms as much of the source image as it needs in order to attempt to fill the destination buffer, which means it can transform pixels outside the region of interest.

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

Declared In
`Geometry.h`

vImageHorizontalShear_Planar8

Performs a horizontal shear operation on a region of interest of a Planar8 source image.

```
vImage_Error vImageHorizontalShear_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   float xTranslate,
   float shearSlope,
   ResamplingFilter filter,
   Pixel_8 backColor,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to shear.

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.

This 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 as the destination image 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.

xTranslate

A translation value for the horizontal direction.

shearSlope

The slope of the front edge of the sheared image, measured in a clockwise direction.

filter

The resampling filter to be used with this function. You create this object by calling `vImageNewResamplingFilter` (to use a default resampling filter supplied by vImage) or `vImageNewResamplingFilterForFunctionUsingBuffer` (to use a custom resampling filter that you supply). When the resampling filter is created, you can also set a scale factor that will be used in the horizontal shear operation.

backgroundColor

A background color. Pass a pixel value only if you also set the `kvImageBackgroundColorFill` flag.

flags

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

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

The `kvImageBackgroundColorFill` and `kvImageEdgeExtend` flags are mutually exclusive. You must set exactly one of these flags.

Return Value

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

Discussion

This function also translates and scales the image, both in the horizontal direction. The function transforms as much of the source image as it needs in order to attempt to fill the destination buffer, which means it can transform pixels outside the region of interest.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Geometry.h`

vImageHorizontalShear_PlanarF

Performs a horizontal shear operation on a region of interest of a PlanarF source image.

```
vImage_Error vImageHorizontalShear_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   float xTranslate,
   float shearSlope,
   ResamplingFilter filter,
   Pixel_F backColor,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to shear.

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.

This 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 as the destination image 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.

xTranslate

A translation value for the horizontal direction.

shearSlope

The slope of the front edge of the sheared image, measured in a clockwise direction.

filter

The resampling filter to be used with this function. You create this object by calling `vImageNewResamplingFilter` (to use a default resampling filter supplied by vImage) or `vImageNewResamplingFilterForFunctionUsingBuffer` (to use a custom resampling filter that you supply). When the resampling filter is created, you can also set a scale factor that will be used in the horizontal shear operation.

backgroundColor

A background color. Pass a pixel value only if you also set the `kvImageBackgroundColorFill` flag.

flags

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

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

The `kvImageBackgroundColorFill` and `kvImageEdgeExtend` flags are mutually exclusive. You must set exactly one of these flags.

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

Discussion
This function also translates and scales the image, both in the horizontal direction. The function transforms as much of the source image as it needs in order to attempt to fill the destination buffer, which means it can transform pixels outside the region of interest.

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

Declared In
`Geometry.h`

vImageNewResamplingFilter

Creates a resampling filter object that corresponds to the default kernel supplied by the vImage framework.

```
ResamplingFilter vImageNewResamplingFilter (
   float scale,
   vImage_Flags flags
);
```

Parameters

scale

A scale factor to associated with the resampling filter object. Shear functions to which you pass the resampling filter object use this factor when performing a shear operation. The shear function applies the scale factor to the entire image, in a direction appropriate to the shear function, either horizontal or vertical.

flags

The options to use when creating the resampling filter object. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageBackgroundColorFill` or `kvImageEdgeExtend`.

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

This function ignores the `kvImageLeaveAlphaUnchanged` flag.

Return Value

A pointer to a newly created resampling filter object; otherwise `NULL`.

Discussion

This function creates a reusable resampling filter object that you can pass to a shear function. The resampling filter encapsulated by the object is the default kernel for vImage This function allocates the memory needed for the resampling filter object. To deallocate this memory, call the function `vImageDestroyResamplingFilter` (page 14). Do not attempt to deallocate the memory yourself.

Availability

Available in Mac OS X v10.3 and later.

See Also

`vImageNewResamplingFilterForFunctionUsingBuffer` (page 25)

Declared In

`Geometry.h`

vImageNewResamplingFilterForFunctionUsingBuffer

Creates a resampling filter object that encapsulates a resampling kernel function that you provide.

```
vImage_Error vImageNewResamplingFilterForFunctionUsingBuffer (
   const float *xArray,
   float *yArray,
   unsigned long count,
   void *userData
);
```

Parameters

filter

A pointer to a buffer. On return, the buffer contains a resampling filter object. You must allocate the memory for this buffer yourself. Call the function `vImageGetResamplingFilterSize` (page 15) to obtain the size of this buffer.

scale

A scale factor to associated with the resampling filter object. Shear functions to which you pass the resampling filter object use this factor when performing a shear operation. The shear function applies the scale factor to the entire image, in a direction appropriate to the shear function, either horizontal or vertical.

kernelFunc

A pointer to your custom resampling kernel function. If this pointer is `NULL`, vImage creates a resampling filter object that corresponds to its default kernel. The kernel function must remain valid for the life of the resampling filter object.

If you need precise control over the memory used for a resampling filter object but want to use the default, pass `NULL` as the `kernelFunc` parameter. This causes vImage to create a resampling filter object that corresponds to its default kernel. You can then determine where in memory the resampling filter object is located. You can reuse the buffer to reduce memory allocation, and so on. Note that a resampling filter object created in this way is not necessarily the same as one created by the function `vImageNewResamplingFilter` (page 25). You must still deallocate the object yourself; you can not pass it to the function `vImageDestroyResamplingFilter` (page 14).

kernelWidth

A bounding value for the domain of your resampling kernel function. When your function is called, the x-values it will be passed will lie between –*kernelWidth* and +*kernelWidth*, inclusive.

userData

A pointer to custom data that you want to use when calculating your resampling kernel function. When vImage invokes your resampling kernel function, this pointer is passed to the function. The data can be anything you want to use in calculating your resampling kernel function—a table, a list of pointers to related functions, or so on. The data must remain valid for the life of the resampling kernel object.

If your resampling kernel function does nor require user data, pass `NULL`.

flags

The options to use when creating the resampling filter object. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageBackgroundColorFill` or `kvImageEdgeExtend`.

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

This function ignores the `kvImageLeaveAlphaUnchanged` flag.

Return Value

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

Discussion
This function creates a reusable resampling filter object that you can pass to a shear function. Before calling this function, you must allocate a buffer to contain the resampling filter object returned by this function. You can get the necessary size by calling the function `vImageGetResamplingFilterSize` (page 15). When you no longer need this object, you are responsible for deallocating its memory. (Do not use the function `vImageDestroyResamplingFilter` (page 14) to deallocate custom resampling filter objects; it is not designed to deallocate them.)

If you need precise control over the memory used for a resampling filter object but want to use the default, pass `NULL` as the `kernelFunc` parameter. This causes vImage to create a resampling filter object that corresponds to its default kernel. You can then determine where in memory the resampling filter object is located. You can reuse the buffer to reduce memory allocation, and so on. Note that a resampling filter object created in this way is not necessarily the same as one created by the function `vImageNewResamplingFilter` (page 25). You must still deallocate the object yourself; you can not pass it to the function `vImageDestroyResamplingFilter` (page 14).

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

Declared In
`Geometry.h`

vImageRotate90_ARGB8888

Rotates an ARGB8888 source image by the provided factor of 90.

```
vImage_Error vImageRotate90_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   uint8_t rotationConstant,
   Pixel_8888 backColor,
   vImage_Flags flags
);
```

Parameters

src
A pointer to a vImage buffer structure that contains the source image whose data you want to rotate.

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.

rotationConstant
A value specifying the angle of rotation.

backgroundColor
A background color.

flags
The options to use when performing the rotation. 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 maps the center point of the source image to the center point of the destination image. It does not scale or resample, instead, the function copies individual pixels unchanged to new locations.

This function places certain restrictions on the pixel height and widths of the source and destination buffers, so that it can map the center of the source to the center of the destination precisely. The restrictions are:

- If you are rotating the image 90 or 270 degrees, the height of the source image and the width of the destination image must both be even or both be odd; and the width of the source image and the height of the destination image must both be even or both be odd.

If your images do not meet these restrictions, you can use the general (high-level) Rotate function instead, with an angle of 90 or 270 degrees.

Depending on the relative sizes of the source image and the destination buffer, parts of the source image may be clipped. Areas outside the source image may appear in the destination image the background color passed to the function.

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

Declared In
`Geometry.h`

vImageRotate90_ARGBFFFF

Rotates an ARGBFFFF source image by the provided factor of 90.

```
vImage_Error vImageRotate90_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   uint8_t rotationConstant,
   Pixel_FFFF backColor,
   vImage_Flags flags
);
```

Parameters

src
: A pointer to a vImage buffer structure that contains the source image whose data you want to rotate.

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.

rotationConstant
: A value specifying the angle of rotation.

backgroundColor
: A background color.

flags

The options to use when performing the rotation. 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 maps the center point of the source image to the center point of the destination image. It does not scale or resample, instead, the function copies individual pixels unchanged to new locations.

This function places certain restrictions on the pixel height and widths of the source and destination buffers, so that it can map the center of the source to the center of the destination precisely. The restrictions are:

- If you are rotating the image 90 or 270 degrees, the height of the source image and the width of the destination image must both be even or both be odd; and the width of the source image and the height of the destination image must both be even or both be odd.

If your images do not meet these restrictions, you can use the general (high-level) Rotate function instead, with an angle of 90 or 270 degrees.

Depending on the relative sizes of the source image and the destination buffer, parts of the source image may be clipped. Areas outside the source image may appear in the destination image the background color passed to the function.

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

Declared In
`Geometry.h`

vImageRotate90_Planar8

Rotates a Planar8 source image by the provided factor of 90.

```
vImage_Error vImageRotate90_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   uint8_t rotationConstant,
   Pixel_8 backColor,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to rotate.

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.

rotationConstant
A value specifying the angle of rotation.

backgroundColor
A background color.

flags
The options to use when performing the rotation. 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 maps the center point of the source image to the center point of the destination image. It does not scale or resample, instead, the function copies individual pixels unchanged to new locations.

This function places certain restrictions on the pixel height and widths of the source and destination buffers, so that it can map the center of the source to the center of the destination precisely. The restrictions are:

- If you are rotating the image 90 or 270 degrees, the height of the source image and the width of the destination image must both be even or both be odd; and the width of the source image and the height of the destination image must both be even or both be odd.

If your images do not meet these restrictions, you can use the general (high-level) Rotate function instead, with an angle of 90 or 270 degrees.

Depending on the relative sizes of the source image and the destination buffer, parts of the source image may be clipped. Areas outside the source image may appear in the destination image the background color passed to the function.

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

Declared In
`Geometry.h`

vImageRotate90_PlanarF

Rotates a PlanarF source image by the provided factor of 90.

```
vImage_Error vImageRotate90_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   uint8_t rotationConstant,
   Pixel_F backColor,
   vImage_Flags flags
);
```

Parameters

src
A pointer to a vImage buffer structure that contains the source image whose data you want to rotate.

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.

rotationConstant

A value specifying the angle of rotation.

backgroundColor

A background color.

flags

The options to use when performing the rotation. 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 maps the center point of the source image to the center point of the destination image. It does not scale or resample, instead, the function copies individual pixels unchanged to new locations.

This function places certain restrictions on the pixel height and widths of the source and destination buffers, so that it can map the center of the source to the center of the destination precisely. The restrictions are:

- If you are rotating the image 90 or 270 degrees, the height of the source image and the width of the destination image must both be even or both be odd; and the width of the source image and the height of the destination image must both be even or both be odd.

If your images do not meet these restrictions, you can use the general (high-level) Rotate function instead, with an angle of 90 or 270 degrees.

Depending on the relative sizes of the source image and the destination buffer, parts of the source image may be clipped. Areas outside the source image may appear in the destination image the background color passed to the function.

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

Declared In
`Geometry.h`

vImageRotate_ARGB8888

Rotates an ARGB8888 source image by the provided angle.

```
vImage_Error vImageRotate_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   float angleInRadians,
   Pixel_8888 backColor,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to rotate.

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.

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.

angleInRadians

The angle of rotation, in radians.

backgroundColor

A background color. Pass a pixel value only if you also set the `kvImageBackgroundColorFill` flag.

flags

The options to use when applying the rotation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageBackgroundColorFill` or `kvImageEdgeExtend`.

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

This function ignores the `kvImageLeaveAlphaUnchanged` flag.

Return Value

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

Discussion

This function maps the center point of the source image to the center point of the destination image. It does not scale, but it resamples. Depending on the relative sizes of the source image and the destination buffer, parts of the source image may be clipped. Areas outside the source image may appear in the destination image the background color passed to the function.

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
`Geometry.h`

vImageRotate_ARGBFFFF

Rotates an ARGBFFFF source image by the provided angle.

```
vImage_Error vImageRotate_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   float angleInRadians,
   Pixel_FFFF backColor,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to rotate.

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.

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.

angleInRadians

The angle of rotation, in radians.

backgroundColor

A background color. Pass a pixel value only if you also set the `kvImageBackgroundColorFill` flag.

flags

The options to use when applying the rotation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageBackgroundColorFill` or `kvImageEdgeExtend`.

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

This function ignores the `kvImageLeaveAlphaUnchanged` flag.

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

Discussion
This function maps the center point of the source image to the center point of the destination image. It does not scale, but it resamples. Depending on the relative sizes of the source image and the destination buffer, parts of the source image may be clipped. Areas outside the source image may appear in the destination image the background color passed to the function.

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
`Geometry.h`

vImageRotate_Planar8

Rotates a Planar8 source image by the provided angle.

```
vImage_Error vImageRotate_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   float angleInRadians,
   Pixel_8 backColor,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to rotate.

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.

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.

angleInRadians

The angle of rotation, in radians.

backgroundColor

A background color. Pass a pixel value only if you also set the `kvImageBackgroundColorFill` flag.

flags

The options to use when applying the rotation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageBackgroundColorFill` or `kvImageEdgeExtend`.

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

This function ignores the `kvImageLeaveAlphaUnchanged` flag.

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

Discussion
This function maps the center point of the source image to the center point of the destination image. It does not scale, but it resamples. Depending on the relative sizes of the source image and the destination buffer, parts of the source image may be clipped. Areas outside the source image may appear in the destination image the background color passed to the function.

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
`Geometry.h`

vImageRotate_PlanarF

Rotates a PlanarF source image by the provided angle.

```
vImage_Error vImageRotate_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   float angleInRadians,
   Pixel_F backColor,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to rotate.

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.

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.

angleInRadians

The angle of rotation, in radians.

backgroundColor

A background color. Pass a pixel value only if you also set the `kvImageBackgroundColorFill` flag.

flags

The options to use when applying the rotation. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageBackgroundColorFill` or `kvImageEdgeExtend`.

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

This function ignores the `kvImageLeaveAlphaUnchanged` flag.

Return Value

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

Discussion

This function maps the center point of the source image to the center point of the destination image. It does not scale, but it resamples. Depending on the relative sizes of the source image and the destination buffer, parts of the source image may be clipped. Areas outside the source image may appear in the destination image the background color passed to the function.

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
`Geometry.h`

vImageScale_ARGB8888

Scales an ARGB8888 source image to fit a destination buffer.

```
vImage_Error vImageScale_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to scale.

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.

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.

flags

The options to use when applying the scaling. Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

This function ignores the `kvImageLeaveAlphaUnchanged` flag. The function uses edge extend (see `kvImageEdgeExtend`) to assign values to pixels that are outside the source buffer. Edge extend prevents a background color from impinging on the edges of the scaled image.

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

Discussion
The relative size of the source image and the destination buffer determine the scaling factors, which may be different in the X and Y directions.

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
`Geometry.h`

vImageScale_ARGBFFFF

Scales an ARGBFFFF source image to fit a destination buffer.

```
vImage_Error vImageScale_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to scale.

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.

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.

flags

The options to use when applying the scaling. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageBackgroundColorFill` or `kvImageEdgeExtend`.

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

This function ignores the `kvImageLeaveAlphaUnchanged` flag.

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

Discussion
The relative size of the source image and the destination buffer determine the scaling factors, which may be different in the X and Y directions.

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
`Geometry.h`

vImageScale_Planar8

Scales a Planar8 source image to fit a destination buffer.

```
vImage_Error vImageScale_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to scale.

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.

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.

flags

The options to use when applying the scaling. Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

This function ignores the `kvImageLeaveAlphaUnchanged` flag. The function uses edge extend (see `kvImageEdgeExtend`) to assign values to pixels that are outside the source buffer. Edge extend prevents a background color from impinging on the edges of the scaled image.

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

Discussion
The relative size of the source image and the destination buffer determine the scaling factors, which may be different in the X and Y directions.

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
`Geometry.h`

vImageScale_PlanarF

Scales a PlanarF source image to fit a destination buffer.

```
vImage_Error vImageScale_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to scale.

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.

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.

flags

The options to use when applying the scaling. Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

This function ignores the `kvImageLeaveAlphaUnchanged` flag. The function uses edge extend (see `kvImageEdgeExtend`) to assign values to pixels that are outside the source buffer. Edge extend prevents a background color from impinging on the edges of the scaled image.

Return Value

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

Discussion

The relative size of the source image and the destination buffer determine the scaling factors, which may be different in the X and Y directions.

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
Geometry.h

vImageVerticalReflect_ARGB8888

Reflects an ARGBFFFF source image top to bottom across the center vertical line of the image.

```
vImage_Error vImageVerticalReflect_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to reflect.

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.

flags

The options to use when performing the reflection. 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 resulting image appears upside down, as if seen from the back of the image.

This function does not scale or resample. The source and destination buffers must have the same height and the same width.

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

Related Sample Code
QTCoreVideo103
QTCoreVideo202

Declared In
Geometry.h

vImageVerticalReflect_ARGBFFFF

Reflects an ARGBFFFF source image top to bottom across the center vertical line of the image.

```
vImage_Error vImageVerticalReflect_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to reflect.

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.

flags

The options to use when performing the reflection. 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 resulting image appears upside down, as if seen from the back of the image.

This function does not scale or resample. The source and destination buffers must have the same height and the same width.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Geometry.h`

vImageVerticalReflect_Planar8

Reflects a Planar 8 source image top to bottom across the center vertical line of the image.

```
vImage_Error vImageVerticalReflect_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to reflect.

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.

flags

The options to use when performing the reflection. 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 resulting image appears upside down, as if seen from the back of the image.

This function does not scale or resample. The source and destination buffers must have the same height and the same width.

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

Declared In
`Geometry.h`

vImageVerticalReflect_PlanarF

Reflects a PlanarF source image top to bottom across the center vertical line of the image.

```
vImage_Error vImageVerticalReflect_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a structure of type `vImage_Buffer` containing 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.

flags

The options to use when performing the reflection. 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 resulting image appears upside down, as if seen from the back of the image.

This function does not scale or resample. The source and destination buffers must have the same height and the same width.

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

Declared In
`Geometry.h`

vImageVerticalShear_ARGB8888

Performs a vertical shear operation on a region of interest of an ARGB8888 source image.

```
vImage_Error vImageVerticalShear_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   float yTranslate,
   float shearSlope,
   ResamplingFilter filter,
   Pixel_8888 backColor,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to shear.

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.

This 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 as the destination image 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.

yTranslate

A translation value for the vertical direction.

shearSlope

The slope of the top edge of the sheared image, measured in a clockwise direction.

filter

The resampling filter to be used with this function. You create this object by calling `vImageNewResamplingFilter` (to use a default resampling filter supplied by vImage) or `vImageNewResamplingFilterForFunctionUsingBuffer` (to use a custom resampling filter that you supply). When the resampling filter is created, you can also set a scale factor that will be used in the horizontal shear operation.

backgroundColor

A background color. Pass a pixel value only if you also set the `kvImageBackgroundColorFill` flag.

flags

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

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

The `kvImageBackgroundColorFill` and `kvImageEdgeExtend` flags are mutually exclusive. You must set exactly one of these flags.

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

Discussion
This function also translates and scales the image, both in the vertical direction. The function transforms as much of the source image as it needs in order to attempt to fill the destination buffer, which means it can transform pixels outside the region of interest.

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

Declared In
`Geometry.h`

vImageVerticalShear_ARGBFFFF

Performs a vertical shear operation on a region of interest of an ARGBFFFF source image.

```
vImage_Error vImageVerticalShear_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   float yTranslate,
   float shearSlope,
   ResamplingFilter filter,
   Pixel_FFFF backColor,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to shear.

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.

This 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 as the destination image 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.

yTranslate
A translation value for the vertical direction.

shearSlope
The slope of the top edge of the sheared image, measured in a clockwise direction.

filter
The resampling filter to be used with this function. You create this object by calling `vImageNewResamplingFilter` (to use a default resampling filter supplied by vImage) or `vImageNewResamplingFilterForFunctionUsingBuffer` (to use a custom resampling filter that you supply). When the resampling filter is created, you can also set a scale factor that will be used in the horizontal shear operation.

backgroundColor
A background color. Pass a pixel value only if you also set the `kvImageBackgroundColorFill` flag.

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

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

The `kvImageBackgroundColorFill` and `kvImageEdgeExtend` flags are mutually exclusive. You must set exactly one of these flags.

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

Discussion
This function also translates and scales the image, both in the vertical direction. The function transforms as much of the source image as it needs in order to attempt to fill the destination buffer, which means it can transform pixels outside the region of interest.

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

Declared In
`Geometry.h`

vImageVerticalShear_Planar8

Performs a vertical shear operation on a region of interest of a Planar8 source image.

```
vImage_Error vImageVerticalShear_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   float yTranslate,
   float shearSlope,
   ResamplingFilter filter,
   Pixel_8 backColor,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to shear.

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.

This 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 as the destination image 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.

yTranslate

A translation value for the vertical direction.

shearSlope

The slope of the top edge of the sheared image, measured in a clockwise direction.

filter

The resampling filter to be used with this function. You create this object by calling `vImageNewResamplingFilter` (to use a default resampling filter supplied by vImage) or `vImageNewResamplingFilterForFunctionUsingBuffer` (to use a custom resampling filter that you supply). When the resampling filter is created, you can also set a scale factor that will be used in the horizontal shear operation.

backgroundColor

A background color. Pass a pixel value only if you also set the `kvImageBackgroundColorFill` flag.

flags

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

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

The `kvImageBackgroundColorFill` and `kvImageEdgeExtend` flags are mutually exclusive. You must set exactly one of these flags.

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

Discussion
This function also translates and scales the image, both in the vertical direction. The function transforms as much of the source image as it needs in order to attempt to fill the destination buffer, which means it can transform pixels outside the region of interest.

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

Declared In
`Geometry.h`

vImageVerticalShear_PlanarF

Performs a vertical shear operation on a region of interest of a PlanarF source image.

```
vImage_Error vImageVerticalShear_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImagePixelCount srcOffsetToROI_X,
   vImagePixelCount srcOffsetToROI_Y,
   float yTranslate,
   float shearSlope,
   ResamplingFilter filter,
   Pixel_F backColor,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to shear.

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.

This 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 as the destination image 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.

yTranslate

A translation value for the vertical direction.

shearSlope

The slope of the top edge of the sheared image, measured in a clockwise direction.

filter

The resampling filter to be used with this function. You create this object by calling `vImageNewResamplingFilter` (to use a default resampling filter supplied by vImage) or `vImageNewResamplingFilterForFunctionUsingBuffer` (to use a custom resampling filter that you supply). When the resampling filter is created, you can also set a scale factor that will be used in the horizontal shear operation.

backgroundColor

A background color. Pass a pixel value only if you also set the `kvImageBackgroundColorFill` flag.

flags

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

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

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

The `kvImageBackgroundColorFill` and `kvImageEdgeExtend` flags are mutually exclusive. You must set exactly one of these flags.

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

Discussion
This function also translates and scales the image, both in the vertical direction. The function transforms as much of the source image as it needs in order to attempt to fill the destination buffer, which means it can transform pixels outside the region of interest.

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

Declared In
`Geometry.h`

Constants

Rotation Constants

The number of degrees in the clockwise direction.

```
enum
{
    kRotate0DegreesClockwise        = 0,
    kRotate90DegreesClockwise       = 3,
    kRotate180DegreesClockwise      = 2,
    kRotate270DegreesClockwise      = 1
    kRotate0DegreesCounterClockwise = 0,
    kRotate90DegreesCounterClockwise= 1,
    kRotate180DegreesCounterClockwise= 2,
    kRotate270DegreesCounterClockwise= 3
};
```

Constants

`kRotate0DegreesClockwise`
Rotate 0 degrees (that is, copy without rotating).
Available in Mac OS X v10.3 and later.
Declared in `Geometry.h`.

`kRotate90DegreesClockwise`
Rotate 90 degrees clockwise.
Available in Mac OS X v10.3 and later.
Declared in `Geometry.h`.

`kRotate180DegreesClockwise`
Rotate 180 degrees clockwise.
Available in Mac OS X v10.3 and later.
Declared in `Geometry.h`.

`kRotate270DegreesClockwise`
Rotate 270 degrees clockwise.
Available in Mac OS X v10.3 and later.
Declared in `Geometry.h`.

`kRotate0DegreesCounterClockwise`
Rotate 0 degrees (that is, copy without rotating).
Available in Mac OS X v10.3 and later.
Declared in `Geometry.h`.

`kRotate90DegreesCounterClockwise`
Rotate 90 degrees counter-clockwise.
Available in Mac OS X v10.3 and later.
Declared in `Geometry.h`.

`kRotate180DegreesCounterClockwise`
Rotate 180 degrees counter-clockwise.
Available in Mac OS X v10.3 and later.
Declared in `Geometry.h`.

`kRotate270DegreesCounterClockwise`
Rotate 270 degrees counter-clockwise.
Available in Mac OS X v10.3 and later.
Declared in `Geometry.h`.

Declared In
`Geometry.h`

Document Revision History

This table describes the changes to *vImage Geometry 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

K

R

V