# vImage Histogram Reference

Performance > Graphics & Imaging

2007-07-12

# Contents

# vImage Histogram Reference

|  |  |
|---|---|
| **Framework:** | Accelerate/vImage |
| **Companion guide** | vImage Programming Guide |
| **Declared in** | Histogram.h |

## Overview

Histogram functions calculate image histograms or manipulate a histogram to modify an image. There are a number of reasons to apply histogram operations to an image. An image may not make full use of the possible range of intensity values—for example, most of its pixels may be fairly dark, making details difficult to see. Changing the image so that it has a more uniform histogram can improve contrast. Also, it may be easier to compare two images (with respect to texture or other aspects) if you change each histogram to match some standard histogram.

Histogram operations are **point operations**: that is, the intensity of a destination pixel depends only on the intensity of the source pixel, modified by values that are the same over the entire image. Two pixels of the same intensity always map to two pixels of the same (but presumably altered) intensity. If the original image has N different intensity values, the transformed image will have at most N different intensity levels represented.

The vImage histogram functions either calculate histograms or perform one of these point operations:

- Contrast stretch transforms an image so that its intensity values stretch out along the full range of intensity values. It is best used on images in which all the pixels are concentrated in one area of the intensity spectrum, and intensity values outside that area are not represented.
- Ends-in contrast stretch is a more complex version of the contrast stretch operation. These types of functions are best used on images that have some pixels at or near the lowest and highest values of the intensity spectrum, but whose histogram is still mainly concentrated in one area. The ends-in contrast stretch functions map all intensities less than or equal to a certain level to 0; all intensities greater than or equal to a certain level to 255; and perform a contrast stretch on all the values in between. The low and high levels are not defined directly by two given intensity values, but by percentages: the ends-in contrast stretch operation must find intensity levels such that a certain percent of pixels are below one of the intensity values, and a certain percent are above the other intensity value
- Equalization transforms an image so that it has a more uniform histogram. A truly uniform histogram is one in which each intensity level occurs with equal frequency. These functions approximate that histogram.
- Histogram specification transforms an image so that its histogram more closely resembles a given histogram.

# Functions by Task

## Stretching the Contrast

`vImageContrastStretch_ARGBFFFF` (page 8)
: Stretches the contrast of an ARGBFFFF source image.

`vImageContrastStretch_ARGB8888` (page 7)
: Stretches the contrast of an ARGB8888 source image.

`vImageContrastStretch_PlanarF` (page 10)
: Stretches the contrast of a PlanarF source image.

`vImageContrastStretch_Planar8` (page 9)
: Stretches the contrast of a Planar8 source image.

`vImageEndsInContrastStretch_ARGBFFFF` (page 13)
: Performs an ends-in contrast stretch operation on an ARGBFFFF source image.

`vImageEndsInContrastStretch_ARGB8888` (page 12)
: Performs an ends-in contrast stretch operation on an ARGB8888 source image.

`vImageEndsInContrastStretch_PlanarF` (page 15)
: Performs an ends-in contrast stretch operation on a PlanarF source image.

`vImageEndsInContrastStretch_Planar8` (page 14)
: Performs an ends-in contrast stretch operation on a Planar8 source image.

## Equalizing a Histogram

`vImageEqualization_ARGBFFFF` (page 18)
: Equalizes the histogram of an ARGBFFFF source image.

`vImageEqualization_ARGB8888` (page 17)
: Equalizes the histogram of an ARGB8888 source image.

`vImageEqualization_PlanarF` (page 20)
: Equalizes the histogram of a PlanarF source image.

`vImageEqualization_Planar8` (page 19)
: Equalizes the histogram of an ARGB8888 source image.

## Specifying a Histogram

`vImageHistogramSpecification_ARGBFFFF` (page 26)
: Performs a histogram specification operation on an ARGBFFFF source image.

`vImageHistogramSpecification_ARGB8888` (page 25)
: Performs a histogram specification operation on an ARGB8888 source image.

`vImageHistogramSpecification_PlanarF` (page 28)
: Performs a histogram specification operation on a PlanarF source image.

`vImageHistogramSpecification_Planar8` (page 28)
: Performs a histogram specification operation on a Planar8 source image.

### Calculating a Histogram

`vImageHistogramCalculation_ARGBFFFF` (page 23)
: Calculates histograms for each channel of an ARGBFFFF image.

`vImageHistogramCalculation_ARGB8888` (page 22)
: Calculates histograms for each channel of an ARGB8888 image.

`vImageHistogramCalculation_PlanarF` (page 24)
: Calculates the histogram a PlanarF image.

`vImageHistogramCalculation_Planar8` (page 24)
: Calculates a histogram for a Planar8 image.

### Getting the Minimum Buffer Size.

`vImageGetMinimumTempBufferSizeForHistogram` (page 21)
: Returns the minimum size, in bytes, for the temporary buffer needed by a histogram function. (Deprecated. Use the `kvImageGetTempBufferSize` flag with the appropriate histogram function instead of calling this function.)

## Functions

### vImageContrastStretch_ARGB8888

Stretches the contrast of an ARGB8888 source image.

```
vImage_Error vImageContrastStretch_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.

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

  Set the `kvImageLeaveAlphaUnchanged` flag to copy the alpha channel to the destination image unchanged.

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

**Discussion**

The contrast stretch operation alters the image histogram so that pixel values can be found at both the lowest and highest end of the histogram, with values in between “stretched” in a linear fashion. The contrast stretch operation is done separately for each of the four channels—alpha, red, green, and blue. However, the size and range values are the same for each of the four histograms.

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**

`Histogram.h`

## vImageContrastStretch_ARGBFFFF

Stretches the contrast of an ARGBFFFF source image.

```
vImage_Error vImageContrastStretch_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   unsigned int histogram_entries,
   Pixel_F minVal,
   Pixel_F maxVal,
   vImage_Flags flags
);
```

**Parameters**

*src*

A pointer to a vImage buffer structure that contains 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.

*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.

*histogram_entries*

The number of histogram entries, or bins, to use in histograms for this operation.

*minVal*

A minimum pixel value, the low end of the histogram. Any pixel value less than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the first histogram entry. This minimum value is applied to each of the four channels separately.

*maxVal*

A maximum pixel value, the high end of the histogram. Any pixel value greater than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the last histogram entry. This maximum value is applied to each of the four channels separately.

*flags*

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

Set the `kvImageLeaveAlphaUnchanged` flag to copy the alpha channel to the destination image unchanged.

**Return Value**

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

**Discussion**

The contrast stretch operation alters the image histogram so that pixel values can be found at both the lowest and highest end of the histogram, with values in between "stretched" in a linear fashion. The contrast stretch operation is done separately for each of the four channels—alpha, red, green, and blue. However, the size and range values are the same for each of the four histograms.

The source and destination buffers must have the same height and the same width.

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**

`Histogram.h`

## vImageContrastStretch_Planar8

Stretches the contrast of a Planar8 source image.

```
vImage_Error vImageContrastStretch_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.

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

Set the `kvImageLeaveAlphaUnchanged` flag to copy the alpha channel to the destination image unchanged.

**Return Value**

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

**Discussion**

The contrast stretch operation alters the image histogram so that pixel values can be found at both the lowest and highest end of the histogram, with values in between "stretched" in a linear fashion. The contrast stretch operation is done separately for each of the four channels—alpha, red, green, and blue. However, the size and range values are the same for each of the four histograms.

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**

`Histogram.h`

## vImageContrastStretch_PlanarF

Stretches the contrast of a PlanarF source image.

```
vImage_Error vImageContrastStretch_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   unsigned int histogram_entries,
   Pixel_F minVal,
   Pixel_F maxVal,
   vImage_Flags flags
);
```

**Parameters**

*src*

A pointer to a vImage buffer structure that contains 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.

*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.

*histogram_entries*
: The number of histogram entries, or "bins," to be used in histograms for this operation.

*minVal*
: A minimum pixel value, the low end of the histogram. Any pixel value less than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the first histogram entry.

*maxVal*
: A maximum pixel value, the high end of the histogram. Any pixel value greater than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the last histogram entry.

*flags*
: The options to use. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

  Set the `kvImageLeaveAlphaUnchanged` flag to copy the alpha channel to the destination image unchanged.

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

**Discussion**
The contrast stretch operation alters the image histogram so that pixel values can be found at both the lowest and highest end of the histogram, with values in between "stretched" in a linear fashion. The contrast stretch operation is done separately for each of the four channels—alpha, red, green, and blue. However, the size and range values are the same for each of the four histograms.

The source and destination buffers must have the same height and the same width.

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**
`Histogram.h`

## vImageEndsInContrastStretch_ARGB8888

Performs an ends-in contrast stretch operation on an ARGB8888 source image.

```
vImage_Error vImageEndsInContrastStretch_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   const unsigned int percent_low[4],
   const unsigned int percent_high[4],
   vImage_Flags flags
);
```

**Parameters**

*src*

A pointer to a vImage buffer structure that contains 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.

*percent_low*

A percentage value. The number of pixels that map to the lowest end of the histogram of the transformed image should represent this percentage of the total pixels.

*percent_high*

A percentage value. The number of pixels that map to the highest end of the histogram of the transformed image should represent this percentage of the total pixels.

*flags*

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

Set the `kvImageLeaveAlphaUnchanged` flag to copy the alpha channel to the destination image unchanged.

**Return Value**

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

**Discussion**

The ends-in contrast stretch operation alters the image histogram so that a certain percentage of pixels are mapped to the lowest end of the histogram, a certain percentage are mapped to the highest end, and the values in between “stretched” between the lowest and the highest. The ends-in contrast stretch operation is done separately for each of the four channels—alpha, red, green, and blue. However the size and range values are the same for each of the four histograms. In general, it is not possible to get exactly the desired percentage of pixels at the low end and the high end of the histogram of the transformed image. This operation only approximates the given values.

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**

`Histogram.h`

## vImageEndsInContrastStretch_ARGBFFFF

Performs an ends-in contrast stretch operation on an ARGBFFFF source image.

```
vImage_Error vImageEndsInContrastStretch_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   const unsigned int percent_low[4],
   const unsigned int percent_high[4],
   unsigned int histogram_entries,
   Pixel_F minVal,
   Pixel_F maxVal,
   vImage_Flags flags
);
```

### Parameters

*src*

A pointer to a vImage buffer structure that contains 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.

*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.

*percent_low*

A percentage value. The number of pixels that map to the lowest end of the histogram of the transformed image should represent this percentage of the total pixels.

*percent_high*

A percentage value. The number of pixels that map to the highest end of the histogram of the transformed image should represent this percentage of the total pixels.

*histogram_entries*

The number of histogram entries, or “bins,” to be used in histograms for this operation.

*minVal*

A minimum pixel value, the low end of the histogram. Any pixel value less than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the first histogram entry. This minimum value is applied to each of the four channels separately.

*maxVal*

A maximum pixel value, the high end of the histogram. Any pixel value greater than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the last histogram entry. This maximum value is applied to each of the four channels separately.

*flags*

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

Set the `kvImageLeaveAlphaUnchanged` flag to copy the alpha channel to the destination image unchanged.

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

**Discussion**
The ends-in contrast stretch operation alters the image histogram so that a certain percentage of pixels are mapped to the lowest end of the histogram, a certain percentage are mapped to the highest end, and the values in between “stretched” between the lowest and the highest. The ends-in contrast stretch operation is done separately for each of the four channels—alpha, red, green, and blue. However the size and range values are the same for each of the four histograms. In general, it is not possible to get exactly the desired percentage of pixels at the low end and the high end of the histogram of the transformed image. This operation only approximates the given values.

The source and destination buffers must have the same height and the same width.

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**
`Histogram.h`

## vImageEndsInContrastStretch_Planar8

Performs an ends-in contrast stretch operation on a Planar8 source image.

```
vImage_Error vImageEndsInContrastStretch_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   unsigned int percent_low,
   unsigned int percent_high,
   vImage_Flags flags
);
```

**Parameters**

*`src`*

A pointer to a vImage buffer structure that contains 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.

*percent_low*

A percentage value. The number of pixels that map to the lowest end of the histogram of the transformed image should represent this percentage of the total pixels.

*percent_high*

A percentage value. The number of pixels that map to the highest end of the histogram of the transformed image should represent this percentage of the total pixels.

*flags*

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

Set the `kvImageLeaveAlphaUnchanged` flag to copy the alpha channel to the destination image unchanged.

**Return Value**

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

**Discussion**

The ends-in contrast stretch operation alters the image histogram so that a certain percentage of pixels are mapped to the lowest end of the histogram, a certain percentage are mapped to the highest end, and the values in between “stretched” between the lowest and the highest. The ends-in contrast stretch operation is done separately for each of the four channels—alpha, red, green, and blue. However the size and range values are the same for each of the four histograms. In general, it is not possible to get exactly the desired percentage of pixels at the low end and the high end of the histogram of the transformed image. This operation only approximates the given values.

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**

`Histogram.h`

## vImageEndsInContrastStretch_PlanarF

Performs an ends-in contrast stretch operation on a PlanarF source image.

```
vImage_Error vImageEndsInContrastStretch_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   unsigned int percent_low,
   unsigned int percent_high,
   unsigned int histogram_entries,
   Pixel_F minVal,
   Pixel_F maxVal,
   vImage_Flags flags
);
```

**Parameters**

*src*

A pointer to a vImage buffer structure that contains 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.

*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.

*percent_low*

A percentage value. The number of pixels that map to the lowest end of the histogram of the transformed image should represent this percentage of the total pixels.

*percent_high*

A percentage value. The number of pixels that map to the highest end of the histogram of the transformed image should represent this percentage of the total pixels.

*histogram_entries*

The number of histogram entries, or "bins," to be used in histograms for this operation.

*minVal*

A minimum pixel value, the low end of the histogram. Any pixel value less than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the first histogram entry.

*maxVal*

A maximum pixel value, the high end of the histogram. Any pixel value greater than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the last histogram entry.

*flags*

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

Set the `kvImageLeaveAlphaUnchanged` flag to copy the alpha channel to the destination image unchanged.

**Return Value**

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

**Discussion**

The ends-in contrast stretch operation alters the image histogram so that a certain percentage of pixels are mapped to the lowest end of the histogram, a certain percentage are mapped to the highest end, and the values in between "stretched" between the lowest and the highest. The ends-in contrast stretch operation is done separately for each of the four channels—alpha, red, green, and blue. However the size and range values are the same for each of the four histograms. In general, it is not possible to get exactly the desired percentage of pixels at the low end and the high end of the histogram of the transformed image. This operation only approximates the given values.

The source and destination buffers must have the same height and the same width.

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**
`Histogram.h`

## vImageEqualization_ARGB8888

Equalizes the histogram of an ARGB8888 source image.

```
vImage_Error vImageEqualization_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.

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

Set the `kvImageLeaveAlphaUnchanged` flag to copy the alpha channel to the destination image unchanged.

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

**Discussion**
The equalization operation alters the image histogram so that it is closer to a uniform intensity distribution. The histogram equalization operation is done separately for each of the four channels—alpha, red, green, and blue. However, the size and range values are the same for each of the four histograms.

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**
`Histogram.h`

## vImageEqualization_ARGBFFFF

Equalizes the histogram of an ARGBFFFF source image.

```
vImage_Error vImageEqualization_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   unsigned int histogram_entries,
   Pixel_F minVal,
   Pixel_F maxVal,
   vImage_Flags flags
);
```

**Parameters**

*src*

A pointer to a vImage buffer structure that contains 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.

*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.

*histogram_entries*

The number of histogram entries, or "bins," to be used in histograms for this operation.

*minVal*

A minimum pixel value. Any pixel value less than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the first histogram entry. This minimum value is applied to each of the four channels separately.

*maxVal*

A maximum pixel value. Any pixel value greater than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the last histogram entry. This maximum value is applied to each of the four channels separately.

*flags*

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

Set the `kvImageLeaveAlphaUnchanged` flag to copy the alpha channel to the destination image unchanged.

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

**Discussion**

The equalization operation alters the image histogram so that it is closer to a uniform intensity distribution. The histogram equalization operation is done separately for each of the four channels—alpha, red, green, and blue. However, the size and range values are the same for each of the four histograms.

The source and destination buffers must have the same height and the same width.

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**

`Histogram.h`

## vImageEqualization_Planar8

Equalizes the histogram of an ARGB8888 source image.

```
vImage_Error vImageEqualization_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.

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

Set the `kvImageLeaveAlphaUnchanged` flag to copy the alpha channel to the destination image unchanged.

**Return Value**

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

**Discussion**

The equalization operation alters the image histogram so that it is closer to a uniform intensity distribution. 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**

`Histogram.h`

## vImageEqualization_PlanarF

Equalizes the histogram of a PlanarF source image.

```
vImage_Error vImageEqualization_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   unsigned int histogram_entries,
   Pixel_F minVal,
   Pixel_F maxVal,
   vImage_Flags flags
);
```

**Parameters**

*src*

A pointer to a vImage buffer structure that contains 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.

*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.

*histogram_entries*

The number of histogram entries, or "bins," to be used in histograms for this operation.

*minVal*

A minimum pixel value. Any pixel value less than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the first histogram entry. This minimum value is applied to each of the four channels separately.

*maxVal*

A maximum pixel value. Any pixel value greater than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the last histogram entry. This maximum value is applied to each of the four channels separately.

*flags*

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

Set the `kvImageLeaveAlphaUnchanged` flag to copy the alpha channel to the destination image unchanged.

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

**Discussion**
The equalization operation alters the image histogram so that it is closer to a uniform intensity distribution. The histogram equalization operation is done separately for each of the four channels—alpha, red, green, and blue. However, the size and range values are the same for each of the four histograms.

The source and destination buffers must have the same height and the same width.

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**
`Histogram.h`

## vImageGetMinimumTempBufferSizeForHistogram

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

```
size_t vImageGetMinimumTempBufferSizeForHistogram (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   unsigned int histogram_entries,
   vImage_Flags flags,
   size_t bytesPerPixel
);
```

**Parameters**

*src*

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

*dest*
: A pointer to a vImage buffer structure that you plan to pass to the histogram function.

*histogram_entries*
: The number of histogram that you plan to pass to the histogram function.

*flags*
: The flags that you plan to pass to the histogram function.

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

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

**Discussion**
This function 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 `vImageGetMinimumTempBufferSizeForHistogram`, the `height` and `width` of the *src* and *dest* parameters do not increase, and the other parameters remain the same, then the result of the `vImageGetMinimumTempBufferSizeForHistogram` 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**
`Histogram.h`

## vImageHistogramCalculation_ARGB8888

Calculates histograms for each channel of an ARGB8888 image.

```
vImage_Error vImageHistogramCalculation_ARGB8888 (
   const vImage_Buffer *src,
   vImagePixelCount *histogram[4],
   vImage_Flags flags
);
```

**Parameters**

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

*histogram*
: A pointer to a histograms, one each for alpha, red, green, and blue (in that order). On return, this array will contain the four histograms for the corresponding channels. Each of the four histograms will be an array with 256 elements.

*flags*
: The options to use. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

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

**Discussion**
The function calculates the histogram for each channel completely separately from the others. However, size and range values are the same for each of the four histograms.

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

**Declared In**
`Histogram.h`

## vImageHistogramCalculation_ARGBFFFF

Calculates histograms for each channel of an ARGBFFFF image.

```
vImage_Error vImageHistogramCalculation_ARGBFFFF (
   const vImage_Buffer *src,
   vImagePixelCount *histogram[4],
   unsigned int histogram_entries,
   Pixel_F minVal,
   Pixel_F maxVal,
   vImage_Flags flags
);
```

**Parameters**

*`src`*
: A pointer to a vImage buffer structure that contains the source image.

*`histogram`*
: A pointer to an array of four histograms, one each for alpha, red, green, and blue (in that order). On return, this array will contain the four histograms for the corresponding channels. Each of the four histograms will be an array with histogram_entries elements.

*`histogram_entries`*
: The number of histogram entries, or “bins.” Each of the four calculated histograms will be an array with histogram_entries elements.

*`minVal`*
: A minimum pixel value. Any pixel value less than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the first histogram entry. This minimum value is applied to each of the four channels separately.

*`maxVal`*
: A maximum pixel value. Any pixel value greater than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the last histogram entry. This maximum value is applied to each of the four channels separately.

*`flags`*
: The options to use. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

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

**Discussion**
The function calculates the histogram for each channel completely separately from the others. However, size and range values are the same for each of the four histograms.

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

**Declared In**
`Histogram.h`

## vImageHistogramCalculation_Planar8

Calculates a histogram for a Planar8 image.

```
vImage_Error vImageHistogramCalculation_Planar8 (
   const vImage_Buffer *src,
   vImagePixelCount *histogram,
   vImage_Flags flags
);
```

**Parameters**

*`src`*
A pointer to a vImage buffer structure that contains the source image.

*`histogram`*
A pointer to a histogram. On return, this array will contain the histogram for the source image. The histogram will be an array with 256 elements.

*`flags`*
The options to use. 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*.

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

**Declared In**
`Histogram.h`

## vImageHistogramCalculation_PlanarF

Calculates the histogram a PlanarF image.

```
vImage_Error vImageHistogramCalculation_PlanarF (
   const vImage_Buffer *src,
   vImagePixelCount *histogram,
   unsigned int histogram_entries,
   Pixel_F minVal,
   Pixel_F maxVal,
   vImage_Flags flags
);
```

**Parameters**

*`src`*
A pointer to a vImage buffer structure that contains the source image.

*histogram*
: A pointer to a histogram. On return, this array will contain the histogram for the source image. The histogram will have histogram_entries elements.

*histogram_entries*
: The number of histogram entries, or "bins." The histogram will be an array with histogram_entries elements.

*minVal*
: A minimum pixel value. Any pixel value less than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the first histogram entry.

*maxVal*
: A maximum pixel value. Any pixel value greater than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the last histogram entry.

*flags*
: The options to use. 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*.

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

**Declared In**
`Histogram.h`

## vImageHistogramSpecification_ARGB8888

Performs a histogram specification operation on an ARGB8888 source image.

```
vImage_Error vImageHistogramSpecification_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   const vImagePixelCount *desired_histogram[4],
   vImage_Flags flags
);
```

**Parameters**

*src*
: A pointer to a vImage buffer structure that contains 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.

*desiredHistogram*
: A pointer to an array of four histograms, one each for alpha, red, green, and blue (in that order). These are the desired histograms for the transformed image. Each histogram should be an array with 256 elements.

*flags*

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

Set the `kvImageLeaveAlphaUnchanged` flag to copy the alpha channel to the destination image unchanged.

**Return Value**

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

**Discussion**

The specification operation alters the image histogram so that it more closely resembles a given histogram. The histogram specification operation is done separately for each of the four channels—alpha, red, green, and blue. However, the size and range values are the same for each of the four histograms, and for each of the four desired histograms.

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**

`Histogram.h`

## vImageHistogramSpecification_ARGBFFFF

Performs a histogram specification operation on an ARGBFFFF source image.

```
vImage_Error vImageHistogramSpecification_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   const vImagePixelCount *desired_histogram[4],
   unsigned int histogram_entries,
   Pixel_F minVal,
   Pixel_F maxVal,
   vImage_Flags flags
);
```

**Parameters**

*src*

A pointer to a vImage buffer structure that contains 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.

*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.

*desiredHistogram*

A pointer to an array of four histograms, one each for alpha, red, green, and blue (in that order). These are the desired histograms for the transformed image. Each histogram should be an array with histogram_entries elements.

*histogram_entries*

The number of histogram entries, or "bins," to be used in histograms for this operation.

*minVal*

A minimum pixel value. Any pixel value less than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the first histogram entry. This minimum value is applied to each of the four channels separately.

*maxVal*

A maximum pixel value. Any pixel value greater than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the last histogram entry. This maximum value is applied to each of the four channels separately.

*flags*

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

Set the `kvImageLeaveAlphaUnchanged` flag to copy the alpha channel to the destination image unchanged.

**Return Value**

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

**Discussion**

The specification operation alters the image histogram so that it more closely resembles a given histogram. The histogram specification operation is done separately for each of the four channels—alpha, red, green, and blue. However, the size and range values are the same for each of the four histograms, and for each of the four desired histograms.

The source and destination buffers must have the same height and the same width.

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**

`Histogram.h`

## vImageHistogramSpecification_Planar8

Performs a histogram specification operation on a Planar8 source image.

```
vImage_Error vImageHistogramSpecification_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   const vImagePixelCount *desired_histogram,
   vImage_Flags flags
);
```

**Parameters**

*src*

A pointer to a vImage buffer structure that contains 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.

*desiredHistogram*

A pointer the desired histogram for the transformed image. The histogram should be an array with 256 elements.

*flags*

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

Set the `kvImageLeaveAlphaUnchanged` flag to copy the alpha channel to the destination image unchanged.

**Return Value**

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

**Discussion**

The specification operation alters the image histogram so that it more closely resembles a given histogram. The histogram specification operation is done separately for each of the four channels—alpha, red, green, and blue. However, the size and range values are the same for each of the four histograms, and for each of the four desired histograms.

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**

`Histogram.h`

## vImageHistogramSpecification_PlanarF

Performs a histogram specification operation on a PlanarF source image.

```
vImage_Error vImageHistogramSpecification_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   void *tempBuffer,
   const vImagePixelCount *desired_histogram,
   unsigned int histogram_entries,
   Pixel_F minVal,
   Pixel_F maxVal,
   vImage_Flags flags
);
```

**Parameters**

*src*

A pointer to a vImage buffer structure that contains 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.

*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.

*desiredHistogram*

A pointer the desired histogram for the transformed image. The histogram should be an array with *histogram_entries* elements.

*histogram_entries*

The number of histogram entries, or "bins," to be used in histograms for this operation.

*minVal*

A minimum pixel value. Any pixel value less than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the first histogram entry. This minimum value is applied to each of the four channels separately.

*maxVal*

A maximum pixel value. Any pixel value greater than this will be clipped to this value (for the purposes of histogram calculation), and assigned to the last histogram entry. This maximum value is applied to each of the four channels separately.

*flags*

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

Set the `kvImageLeaveAlphaUnchanged` flag to copy the alpha channel to the destination image unchanged.

**Return Value**

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

**Discussion**
The specification operation alters the image histogram so that it more closely resembles a given histogram. The histogram specification operation is done separately for each of the four channels—alpha, red, green, and blue. However, the size and range values are the same for each of the four histograms, and for each of the four desired histograms.

The source and destination buffers must have the same height and the same width.

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**
`Histogram.h`

# Document Revision History

This table describes the changes to *vImage Histogram 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*. |

# Index