

[Accelerate](#) / Specifying histograms with vImage

Sample Code

# Specifying histograms with vImage

Calculate the histogram of one image, and apply it to a second image.

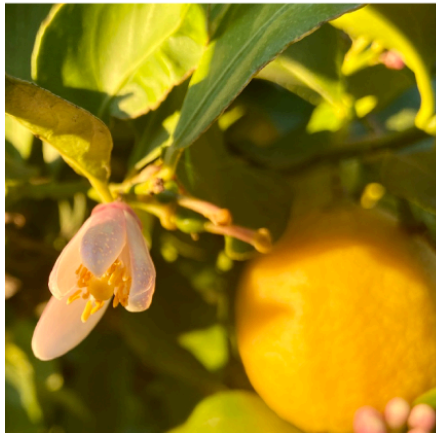
Download

macOS 13.3+ | Xcode 14.3+

## Overview

*Histogram specification* is an image-processing operation that calculates the histogram of a reference image and applies it to an input image. The operation changes the colors and tones of the input image to match those of the reference image.

The example below shows a source image (bottom left) and a histogram reference image (top left), with the histogram specification output on the right.



Before exploring the code, build and run the app to familiarize yourself with the different visual results the app generates when you select different source and reference images.

## Perform histogram specification using pixel buffers

The `vImage.PixelBuffer` structure provides a simple API to calculate and specify a histogram. The `histogram()` function returns the histogram of a pixel buffer, and the `specifyHistogram(destination:)` function performs the histogram specification operation.

The following code creates the pixel buffers that the operation requires, performs the specification, and returns a Core Graphics image that contains the result:

```
/// Performs a histogram specification operation using `vImage.PixelBuffer` structure
static func applyHistogramSpecification_PixelBuffer(
    histogramSourceImage: CGImage,
    imageSourceImage: CGImage) -> CGImage {

    let histogramSource = try! vImage.PixelBuffer<vImage.Interleaved8x4>(
        cgImage: histogramSourceImage,
        cgImageFormat: &imageFormat)

    let imageSource = try! vImage.PixelBuffer<vImage.Interleaved8x4>(
```

```

        cgImage: imageSourceImage,
        cgImageFormat: &imageFormat)

    let destinationBuffer = vImage.PixelBuffer<vImage.Interleaved8x4>(
        size: imageSource.size)

    let histogram = histogramSource.histogram()

    imageSource.specifyHistogram(histogram, destination: destinationBuffer)

    return destinationBuffer.makeCGImage(cgImageFormat: imageFormat)!
}

```

## Calculate the reference histogram using vImage buffers

If you're creating apps for older operating systems that don't support the [vImage.PixelBuffer](#) API, the sample code project also includes source code for performing histogram specification using [vImage\\_Buffer](#) structures.

The [vImageHistogramCalculation\\_ARGB8888\( : : :\)](#) calculates and stores histogram data in four arrays — one for each channel — where the value of each element is the number of pixels in the reference image with that color value. In an 8-bit-per-channel image, each color channel can hold 256 different values, and the sample code defines each array with a count of 256.

```

var histogramBinZero = [vImagePixelCount](repeating: 0, count: 256)
var histogramBinOne = [vImagePixelCount](repeating: 0, count: 256)
var histogramBinTwo = [vImagePixelCount](repeating: 0, count: 256)
var histogramBinThree = [vImagePixelCount](repeating: 0, count: 256)

```

The following code calculates the histogram of the histogramSource [vImage\\_Buffer](#) structure:

```

histogramBinZero.withUnsafeMutableBufferPointer { zeroPtr in
    histogramBinOne.withUnsafeMutableBufferPointer { onePtr in
        histogramBinTwo.withUnsafeMutableBufferPointer { twoPtr in
            histogramBinThree.withUnsafeMutableBufferPointer { threePtr in

                var histogramBins = [zeroPtr.baseAddress, onePtr.baseAddress,
                                     twoPtr.baseAddress, threePtr.baseAddress]

                histogramBins.withUnsafeMutableBufferPointer { histogramBinsPtr in

```

```
let error = vImageHistogramCalculation_ARGB8888(&histogramSource,
                                                histogramBinsPtr,
                                                vImage_Flags(kvImage_...))

guard error == kvImageNoError else {
    fatalError("Error calculating histogram: \(error)")
}

}
```

On return, the four arrays contain the histogram data from the `histogramSource`.

## Specify the image histogram using vImage buffers

The `vImageHistogramSpecification ARGB8888( : : : :)` performs the histogram specification operation. The following code matches the histogram of the reference image to the input image:

```

    histogramBinZero.withUnsafeBufferPointer { zeroPtr in
        histogramBinOne.withUnsafeBufferPointer { onePtr in
            histogramBinTwo.withUnsafeBufferPointer { twoPtr in
                histogramBinThree.withUnsafeBufferPointer { threePtr in

                    var histogramBins = [zeroPtr.baseAddress, onePtr.baseAddress,
                                          twoPtr.baseAddress, threePtr.baseAddress]

                    histogramBins.withUnsafeMutableBufferPointer { histogramBinsPtr in
                        let error = vImageHistogramSpecification_ARGB8888(&imageSource,
                                                                           &destinationBuffer,
                                                                           histogramBinsPtr,
                                                                           vImage_Flags(0))

                        guard error == kvImageNoError else {
                            fatalError("Error specifying histogram: \(error)")
                        }
                    }
                }
            }
        }
    }
}

```

}

On return, `destinationBuffer` contains the original input image with the histogram that the reference image specified.

---

## See Also

### Color and Tone Adjustment

{ } Adjusting the brightness and contrast of an image

Use a gamma function to apply a linear or exponential curve.

{ } Adjusting saturation and applying tone mapping

Convert an RGB image to discrete luminance and chrominance channels, and apply color and contrast treatments.

{ } Applying tone curve adjustments to images

Use the `vlImage` library's polynomial transform to apply tone curve adjustments to images.

{ } Adjusting the hue of an image

Convert an image to L\*a\*b\* color space and apply hue adjustment.

📄 Enhancing image contrast with histogram manipulation

Enhance and adjust the contrast of an image with histogram equalization and contrast stretching.

☰ Histogram

Calculate or manipulate an image's histogram.