Article

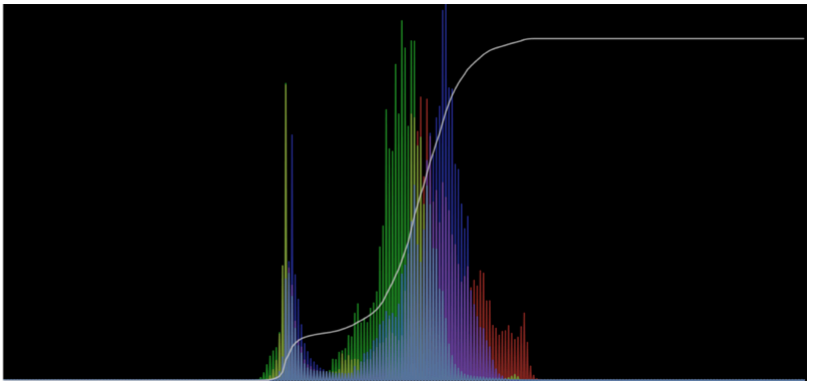
Enhancing image contrast with histogram manipulation

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

Overview

An image histogram is a representation of an image that describes its color tones distribution. The histogram contains a series of bins that represent the possible values for each color channel. For example, each channel of an 8-bit image contains 256 histogram bins. Each bin contains the image's pixel count of the corresponding value.

The histogram of the following low-contrast image shows that almost all of its pixel values are clustered around the mid values. There are no pixels on the left side — corresponding to low values — indicating the image doesn't contain any very dark colors. Similarly, there are no pixels on the right side — corresponding to high values — indicating the images doesn't contain any very bright colors. The thin, gray line shows the cumulative histogram, that is, a running sum of pixel counts at each intensity.



vImage provides functions that can either equalize or stretch an image's histogram to enhance the contrast of an image.

Apply histogram equalization to an image

Histogram equalization transforms an image so that its histogram is more uniformly distributed across the entire range of values. The operation stretches dense parts of the histogram, where contrast is low, and condenses sparse parts of the histogram, where contrast is high. A truly uniform histogram is one in which each histogram bin contains the same value, that is, its cumulative histogram is a diagonal line. The vImage histogram equalization functions approximate that truly uniform histogram.

The following code shows how to perform histogram equalization for `vImage_Buffer` and `vImage.PixelBuffer` structures:

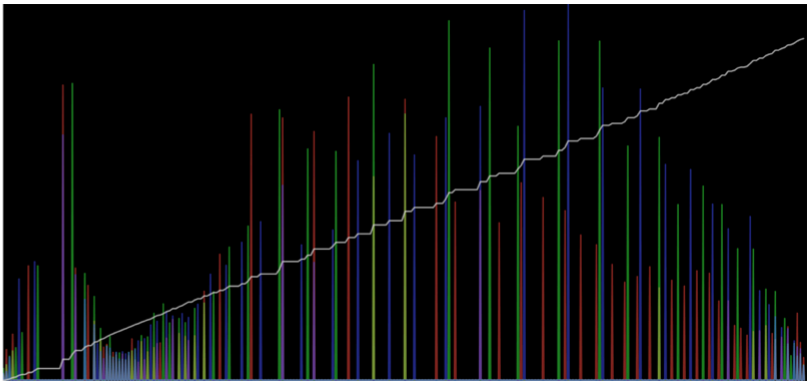
```
// vImage buffer histogram equalization.
do {
    var sourceBuffer: vImage_Buffer = ...
    var destinationBuffer: vImage_Buffer = ...

    vImageEqualization_ARGB8888(&sourceBuffer,
                                &destinationBuffer,
                                vImage_Flags(kvImageNoFlags))
}

// vImage pixel buffer histogram equalization.
do {
    let sourceBuffer: vImage.PixelBuffer<vImage.Interleaved8x4> = ...
    let destinationBuffer: vImage.PixelBuffer<vImage.Interleaved8x4> = ...

    sourceBuffer.equalizeHistogram(destination: destinationBuffer)
}
```

On return, `destinationBuffer` contains the transformed image. The picture below shows the low-contrast image after histogram equalization. The operation distributed the nonzero histogram bins across the entire range of values, and the result has a lot more contrast. The cumulative histogram is nearly a straight diagonal line, indicating an almost uniform distribution of values.



Note that the histogram bins aren't evenly distributed throughout the resulting histogram. The amount of stretching correlates to the number of pixels in each bin.

Apply contrast stretching to an image

Contrast stretching evenly distributes a histogram's pixel values across the full range of available pixel values. This technique is ideal for enhancing the contrast of an image with pixel values concentrated in one area of the intensity spectrum, such as the original low-contrast image above.

The following code shows how to perform contrast stretching for `vImage_Buffer` and `vImage.PixelBuffer` structures:

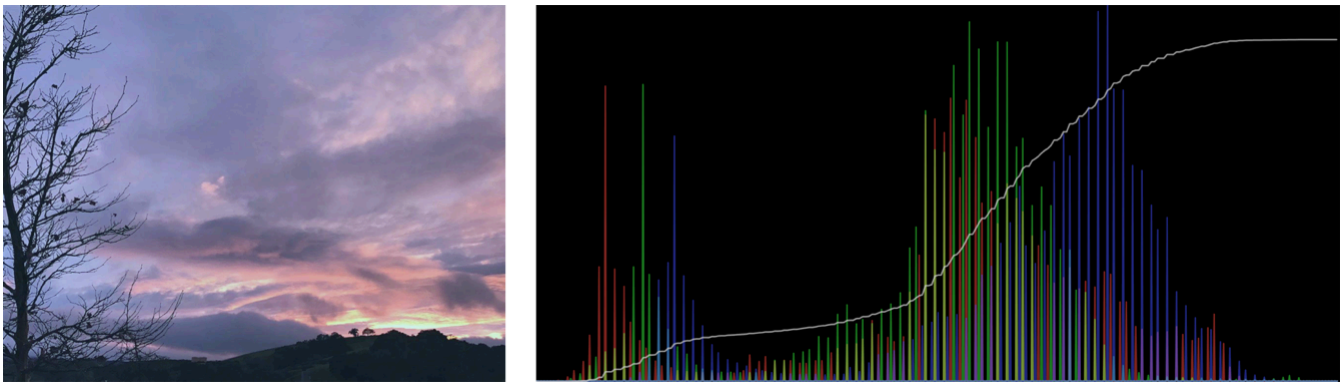
```
// vImage buffer contrast stretching.
do {
    var sourceBuffer: vImage_Buffer = ...
    var destinationBuffer: vImage_Buffer = ...

    vImageContrastStretch_ARGB8888(&sourceBuffer,
                                   &destinationBuffer,
                                   vImage_Flags(kvImageNoFlags))
}

// vImage pixel buffer contrast stretching.
do {
    let sourceBuffer: vImage.PixelBuffer<vImage.Interleaved8x4> = ...
    let destinationBuffer: vImage.PixelBuffer<vImage.Interleaved8x4> = ...

    sourceBuffer.contrastStretch(destination: destinationBuffer)
}
```

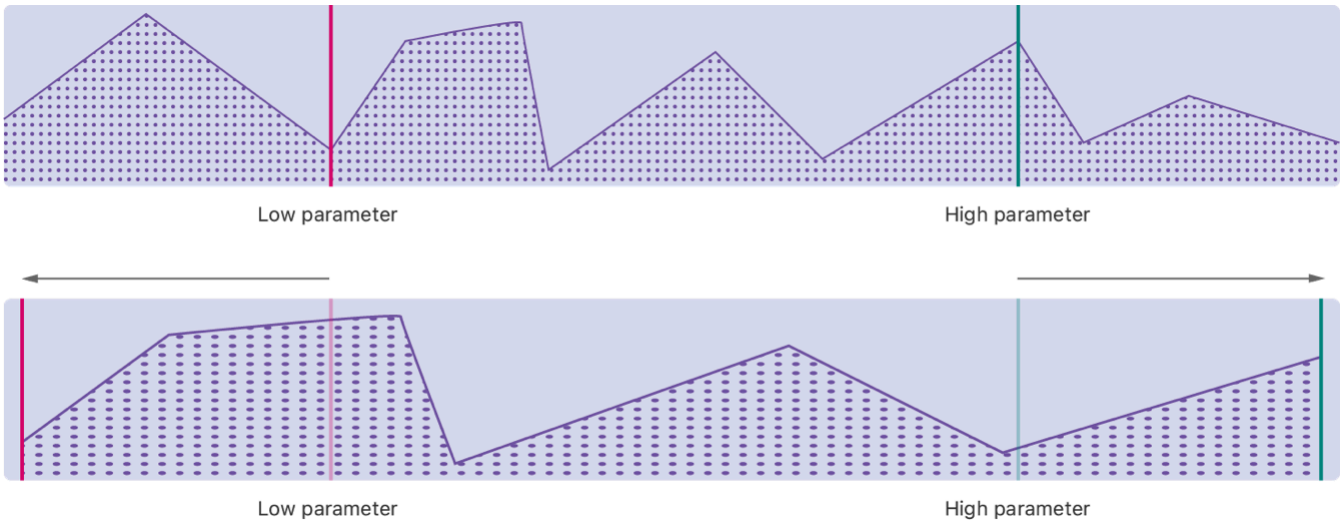
On return, `destinationBuffer` contains the transformed image. The picture below shows the low-contrast image after histogram stretching. The result has a lot more contrast, and its histogram shows that values are evenly distributed throughout the entire range. The shape of the contrast stretched image's cumulative histogram is very similar to the original image's cumulative histogram.



Apply ends-in contrast stretching to an image

The vImage ends-in contrast stretching functions accept parameters that allow you to control which part of the histogram the operation stretches. Use ends-in contrast stretching for images with the majority of their pixels clustered in a single area and a small number of pixels at either end of their histogram. In these situations, standard contrast stretching may not yield your desired result.

The following figure illustrates how ends-in contrast stretching discards elements from a histogram and stretches the remaining values. The operation maps the 25% low values and 25% high values to 0 and 255 (for an 8-bit image), respectively. The result contains the central 50% of the source histogram stretched to fill the remaining 254 bins.



Note that this illustration isn't to scale; the operation uses percentages based on the number of pixels for each intensity.

Set the `percent_low` parameter of the `vImageEndsInContrastStretch_ARGB8888(_:_:_:_:_:)` function to define the percentage of pixels that the operation maps to the lowest end of the transformed image's histogram. The following code shows how to perform ends-in contrast stretching for `vImage_Buffer` and `vImage.PixelBuffer` structures with `percent_low` set to 25% for all channels:

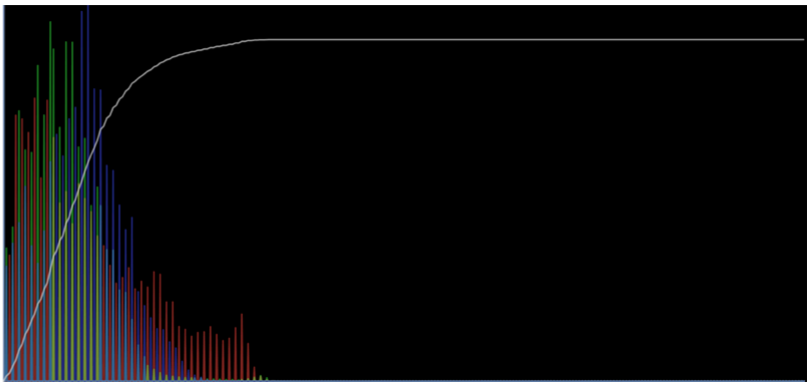
```
// vImage buffer ends-in contrast stretching.
do {
    var sourceBuffer: vImage_Buffer = ...
    var destinationBuffer: vImage_Buffer = ...

    vImageEndsInContrastStretch_ARGB8888(&sourceBuffer,
                                         &destinationBuffer,
                                         [25, 25, 25, 25],
                                         [0, 0, 0, 0],
                                         vImage_Flags(kvImageNoFlags))
}

// vImage pixel buffer ends-in contrast stretching.
do {
    let sourceBuffer: vImage.PixelBuffer<vImage.Interleaved8x4> = ...
    let destinationBuffer: vImage.PixelBuffer<vImage.Interleaved8x4> = ...

    _ = sourceBuffer.withUnsafePointerToVImageBuffer { src in
        destinationBuffer.withUnsafePointerToVImageBuffer { dst in
            vImageEndsInContrastStretch_ARGB8888(src,
                                                 dst,
                                                 [25, 25, 25, 25],
                                                 [0, 0, 0, 0],
                                                 vImage_Flags(kvImageNoFlags))
        }
    }
}
```

On return, `destinationBuffer` contains the transformed image. The picture below shows the low-contrast image after ends-in contrast stretching. This result is much darker overall with the histogram shifted to the left.



Set the `percent_high` parameter of the `vImageEndsInContrastStretch_ARGB8888(_:_:_:_:)` function to define the percentage of pixels that the operation maps to the high end of the transformed image's histogram. The following code shows how to perform ends-in contrast stretching for `vImage_Buffer` and `vImage.PixelBuffer` structures with `percent_high` set to 25% for all channels:

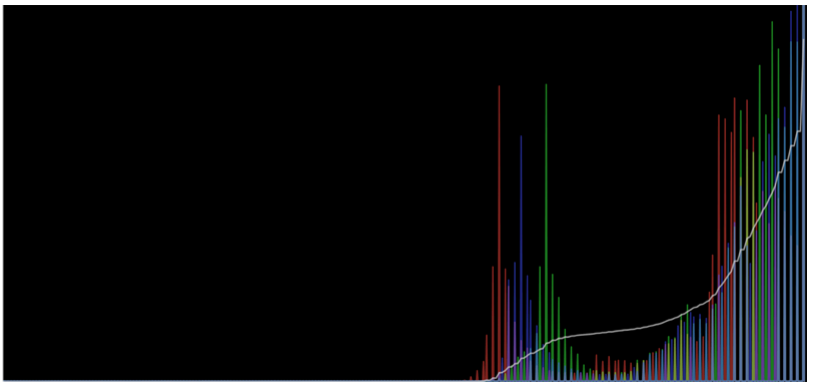
```
// vImage buffer ends-in contrast stretching.
do {
    var sourceBuffer: vImage_Buffer = ...
    var destinationBuffer: vImage_Buffer = ...

    vImageEndsInContrastStretch_ARGB8888(&sourceBuffer,
                                         &destinationBuffer,
                                         [0, 0, 0, 0],
                                         [25, 25, 25, 25],
                                         vImage_Flags(kvImageNoFlags))
}

// vImage pixel buffer ends-in contrast stretching.
do {
    let sourceBuffer: vImage.PixelBuffer<vImage.Interleaved8x4> = ...
    let destinationBuffer: vImage.PixelBuffer<vImage.Interleaved8x4> = ...

    _ = sourceBuffer.withUnsafePointerToVImageBuffer { src in
        destinationBuffer.withUnsafePointerToVImageBuffer { dst in
            vImageEndsInContrastStretch_ARGB8888(src,
                                                 dst,
                                                 [0, 0, 0, 0],
                                                 [25, 25, 25, 25],
                                                 vImage_Flags(kvImageNoFlags))
        }
    }
}
```

On return, `destinationBuffer` contains the transformed image. The picture below shows the low-contrast image after ends-in contrast stretching. This result is much brighter overall with the histogram shifted to the right.



See Also

Related Documentation

`func equalizeHistogram(destination: vImage.PixelBuffer<Format>)`

Equalizes the histogram of a multiple-plane 8-bit pixel buffer.

`func contrastStretch(destination: vImage.PixelBuffer<vImage.Interleaved8x4>)`

Stretches the histogram of an 8-bit-per-channel, 4-channel interleaved pixel buffer.

`func vImageEqualization_ARGB8888(UnsafePointer<vImage_Buffer>, UnsafePointer<vImage_Buffer>, vImage_Flags) -> vImage_Error`

Performs histogram equalization on an 8-bit-per-channel, 4-channel interleaved buffer.

`func vImageContrastStretch_ARGB8888(UnsafePointer<vImage_Buffer>, UnsafePointer<vImage_Buffer>, vImage_Flags) -> vImage_Error`

Performs contrast stretching on an 8-bit-per-channel, 4-channel interleaved buffer.

`func vImageEndsInContrastStretch_ARGB8888(UnsafePointer<vImage_Buffer>, UnsafePointer<vImage_Buffer>, UnsafePointer<UInt32>, UnsafePointer<UInt32>, vImage_Flags) -> vImage_Error`

Performs ends-in contrast stretching on an 8-bit-per-channel, 4-channel interleaved buffer.

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 vImage 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.

{} Specifying histograms with vImage

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

Histogram

Calculate or manipulate an image's histogram.