Structure

vImage_Buffer

An image buffer that stores an image's pixel data, dimensions, and row stride.

iOS | iPadOS | Mac Catalyst | macOS | tvOS | visionOS | watchOS

```
struct vImage_Buffer
```

Mentioned in

- Enhancing image contrast with histogram manipulation
- Applying vImage operations to regions of interest
- Creating and Populating Buffers from Core Graphics Images
- Building a basic image conversion workflow
- Building a Basic Image-Processing Workflow

Overview

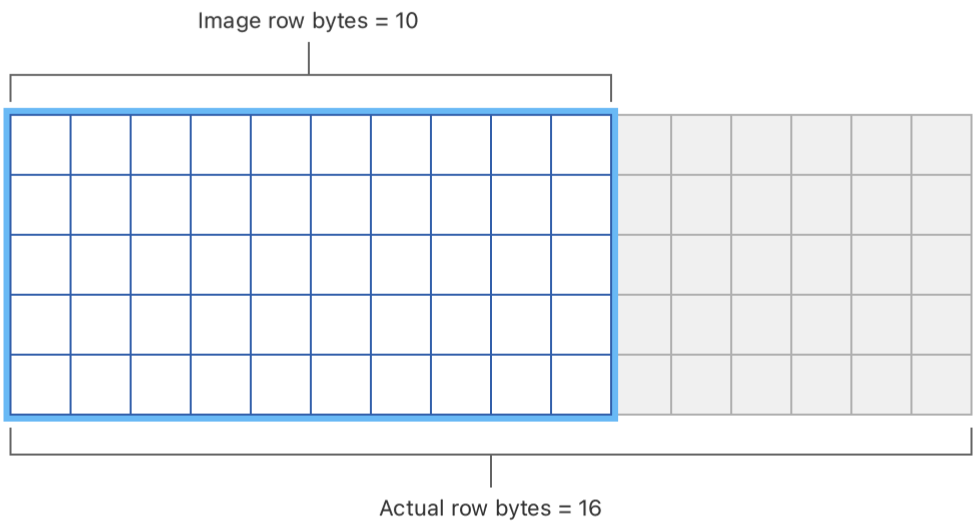
The vImage buffer is the fundamental data structure that the vImage library uses to represent image data. To ensure the best performance, the vImage buffer initialization functions may add extra padding to each row. For example, the following code declares an 8-bit per pixel buffer that's 10 pixels wide:

```
var buffer = vImage_Buffer()

vImageBuffer_Init(&buffer,
                  5,     // height
                  10,    // width
                  8,     // bits per pixel
```

```
                    vImage_Flags(kvImageNoFlags))
```

Although the code defines a buffer with 10 bytes per row, to maximize performance, vImageBuffer_Init(_:_:_:_:_:) initializes a buffer with 16 bytes per row:



If you provide your own buffer storage, call preferredAlignmentAndRowBytes(width:height:bitsPerPixel:) to get the row stride that ensures your buffer achieves the best performance.

```
let width = 10
let height = 5

let alignmentAndRowBytes = try vImage_Buffer.preferredAlignmentAndRowBytes(
    width: width,
    height: height,
    bitsPerPixel: 8)

// Prints "16".
print(alignmentAndRowBytes.rowBytes)

let data = UnsafeMutableRawPointer.allocate(
    byteCount: alignmentAndRowBytes.rowBytes * height,
    alignment: alignmentAndRowBytes.alignment)

let buffer = vImage_Buffer(data: data,
                           height: vImagePixelCount(height),
                           width: vImagePixelCount(width),
                           rowBytes: alignmentAndRowBytes.rowBytes)
```

Topics

Creating an empty vImage buffer

`init(width: Int, height: Int, bitsPerPixel: UInt32) throws`

Creates a new buffer with the specified width, height, and bits per pixel.

`init(size: CGSize, bitsPerPixel: UInt32) throws`

Creates a new buffer with the specified size and bits per pixel.

`init()`

Creates an empty vImage buffer.

Creating a buffer that references existing data

`init(data: UnsafeMutableRawPointer!, height: vImagePixelCount, width: vImagePixelCount, rowBytes: Int)`

Creates a new buffer with the specified size that references existing data.

`static func preferredAlignmentAndRowBytes(width: Int, height: Int, bitsPerPixel: UInt32) throws -> (alignment: Int, rowBytes: Int)`

Returns the preferred alignment and row bytes for a specified size and bits per pixel.

Consuming and producing Core Graphics images

`init(cgImage: CGImage, flags: vImage.Options) throws`

Creates a new buffer with the contents of a Core Graphics image.

`init(cgImage: CGImage, format: vImage_CGImageFormat, flags: vImage.Options) throws`

Creates a new buffer with the contents of a Core Graphics image using the supplied image format.

`func createCGImage(format: vImage_CGImageFormat, flags: vImage.Options) throws -> CGImage`

Creates a Core Graphics image from the vImage buffer.

Inspecting a buffer’s properties

```
var data: UnsafeMutableRawPointer!
```

A pointer to the top-left pixel of the image.

```
var height: vImagePixelCount
```

The height of the image, in pixels.

```
var width: vImagePixelCount
```

The width of the image, in pixels.

```
var size: CGSize
```

The size of the image, in pixels.

```
var rowBytes: Int
```

The distance, in bytes, between the start of one pixel row and the next in an image, including any unused space between them.

Copying a buffer’s contents

```
func copy(destinationBuffer: inout vImage_Buffer, pixelSize: Int, flags
: vImage.Options) throws
```

Copies the contents of a vImage buffer to the specified destination buffer.

Deallocating a buffer

```
func free()
```

Frees the resources associated with the vImage buffer.

Relationships

Conforms To

`BitwiseCopyable`

See Also

Data Types

`typealias vImagePixelCount`

A type for the number of pixels.

`struct vImage_AffineTransform`

A structure for values that represent an affine transformation.

`struct vImage_AffineTransform_Double`

A structure for values that represent a double-precision affine transformation.

`typealias vImage_CGAffineTransform`

A structure for values that represent a Core Graphics–compatible affine transformation.

`typealias vImage_Error`

A type for image errors.

`typealias vImage_Flags`

A type for processing options.

`typealias GammaFunction`

A type for a gamma function.

`typealias ResamplingFilter`

A pointer to a resampling filter callback function.