

[Accelerate](#) / `vImageMultiDimensionalInterpolatedLookupTable_PlanarF(_:_:_:_:_:_)`

Function

`vImageMultiDimensionalInterpolatedLookupTable_PlanarF(_:_:_:_:_:_)`

Uses a multidimensional lookup table to transform a 32-bit planar image.

iOS 7.0+ | iPadOS 7.0+ | Mac Catalyst 13.1+ | macOS 10.9+ | tvOS 7.0+ | visionOS 1.0+ | watchOS 1.0+

```
func vImageMultiDimensionalInterpolatedLookupTable_PlanarF(
    _ srcs: UnsafePointer<vImage_Buffer>,
    _ dests: UnsafePointer<vImage_Buffer>,
    _ tempBuffer: UnsafeMutableRawPointer!,
    _ table: vImage_MultidimensionalTable,
    _ method: vImage_InterpolationMethod,
    _ flags: vImage_Flags
) -> vImage_Error
```

Parameters

`srcs`

An array of `vImage` buffers that reference the source image planes. The number of source buffers is the `numSrcChannels` parameter you pass to `vImageMultidimensionalTable Create\(_:_:_:_:_:_\)`.

`dests`

An array of `vImage` buffers that reference the destination image planes. The number of destination buffers is the `numDestChannels` parameter you pass to `vImageMultidimensionalTable Create\(_:_:_:_:_:_\)`.

`tempBuffer`

A pointer to a temporary buffer. If you pass `nil`, the function allocates the buffer and then deallocates it before returning. Alternatively, you can allocate the buffer yourself, in which case, you're responsible for deallocating it when you no longer need it.

table

The multidimensional lookup table.

method

The interpolation method, either [`kvImageFullInterpolation`](#) or [`kvImageHalfInterpolation`](#).

flags

The options to use when performing the operation. If your code implements its own tiling or its own multithreading, pass [`kvImageDoNotTile`](#); otherwise, pass [`kvImageNoFlags`](#).

Pass [`kvImageGetTempBufferSize`](#) to specify that the function returns the minimum temporary buffer size for the operation with the specified parameters.

Return Value

[`kvImageNoError`](#); otherwise, one of the error codes in [Data Types and Constants](#).

Mentioned in

 Applying color transforms to images with a multidimensional lookup table

Discussion

The following code shows how to convert an RGB image that's represented as three planar buffers to grayscale. The call to [`vImageMultiDimensionalInterpolatedLookupTable_PlanarF\(: : : : : \)`](#) uses the lookup table from the example code in the Discussion section for [`vImageMultiDimensionalTable_Create\(: : : : : : \)`](#). The source image contains three pixels, one is pure red, one is pure green, and one is pure blue. On return, the destination image contains three pixels with the values `[0.2125887, 0.71519035, 0.072190434]`.

```
let size = vImage.Size(width: 1, height: 3)

let red = vImage.PixelBuffer<vImage.PlanarF>(
    pixelValues: [1, 0, 0],
    size: size)

let green = vImage.PixelBuffer<vImage.PlanarF>(
```

```

        pixelValues: [0, 1, 0],
        size: size)

let blue = vImage.PixelBuffer<vImage.PlanarF>(
    pixelValues: [0, 0, 1],
    size: size)

let source = vImage.PixelBuffer<vImage.PlanarFx3>(
    planarBuffers: [red, green, blue])

let destination = vImage.PixelBuffer<vImage.PlanarF>(
    size: size)

source.withUnsafeVImageBuffers { src in
    destination.withUnsafeVImageBuffer { dest in




        _ = vImageMultiDimensionalInterpolatedLookupTable_PlanarF(
            src,
            [dest],
            nil,
            lookupTable,
            kvImageFullInterpolation,
            vImage_Flags(kvImageNoFlags))
    }
}

// Prints "[0.2125887, 0.71519035, 0.072190434]".
print(destination.array)

```

See Also

Transforming with a multidimensional lookup table

-  Applying color transforms to images with a multidimensional lookup table
 Precompute translation values to optimize color space conversion and other pointwise operations.
-  Cropping to the subject in a chroma-keyed image
 Convert a chroma-key color to alpha values and trim transparent pixels using Accelerate.
-  Applying transformations to selected colors in an image
 Desaturate a range of colors in an image with a multidimensional lookup table.

```
func vImageMultidimensionalTable_Create(UnsafePointer<UInt16>, UInt32,
UInt32, UnsafePointer<UInt8>, vImageMDTableUsageHint, vImage_Flags,
UnsafeMutablePointer<vImage_Error>!) -> vImage_MultidimensionalTable!
```

Creates a multidimensional lookup table.

```
func vImageMultiDimensionalInterpolatedLookupTable_Planar16Q12(Unsafe
Pointer<vImage_Buffer>, UnsafePointer<vImage_Buffer>, UnsafeMutableRaw
Pointer!, vImage_MultidimensionalTable, vImage_InterpolationMethod, v
Image_Flags) -> vImage_Error
```

Uses a multidimensional lookup table to transform a 16Q12 planar image.

```
func vImageMultidimensionalTable_Retain(vImage_MultidimensionalTable!)
-> vImage_Error
```

Retains a multidimensional table.

```
func vImageMultidimensionalTable_Release(vImage_MultidimensionalTable!)
-> vImage_Error
```

Releases a multidimensional table.

```
 typealias vImage_MultidimensionalTable
```

An opaque pointer that represents a multidimensional lookup table.

```
struct vImageMDTableUsageHint
```

Constants that indicate the use for a multidimensional lookup table.

```
struct vImage_InterpolationMethod
```

Constants that represent different interpolation methods.