Documentation

Accelerate / Converting luminance and chrominance planes to an ARGB image

Sample Code

Converting luminance and chrominance planes to an ARGB image

Create a displayable ARGB image using the luminance and chrominance information from your device’s camera.

Download

macOS 13.3+ | Xcode 14.3+

Overview

As an alternative to the any-to-any conversion technique that Using vImage pixel buffers to generate video effects describes, vImage provides low-level functions for creating RGB images from the separate luminance and chrominance planes that an `AVCaptureSession` instance provides. These functions offer better performance and more granular configuration than using a `vImageConverter` instance.

Configure the YpCbCr-to-ARGB information

The `vImageConvert_YpCbCrToARGB_GenerateConversion(_:_:_:_:_:_:)` function generates the information that vImage requires to convert the luminance and chrominance planes to a single ARGB image.

Video-range YpCbCr formats often don’t use very low and very high values. For example, an 8-bit video range format typically uses the range `16...235` for luminance and `16...240` for chrominance. The generate conversion function accepts a `vImage_YpCbCrPixelRange` structure that defines the pixel range.

The following code example populates a `vImage_YpCbCrToARGB` structure with the required conversion information for video-range 8-bit pixels:

```
var infoYpCbCrToARGB = vImage_YpCbCrToARGB()

func configureYpCbCrToARGBInfo() {
    var pixelRange = vImage_YpCbCrPixelRange(Yp_bias: 16,
                                             CbCr_bias: 128,
                                             YpRangeMax: 235,
                                             CbCrRangeMax: 240,
                                             YpMax: 235,
                                             YpMin: 16,
                                             CbCrMax: 240,
                                             CbCrMin: 16)

    var ypCbCrToARGBMatrix = vImage_YpCbCrToARGBMatrix(Yp: 1.0,
                                                       Cr_R: 1.402, Cr_G: -0.7141363
                                                       Cb_G: -0.3441363, Cb_B: 1.772

    _ = vImageConvert_YpCbCrToARGB_GenerateConversion(
        &ypCbCrToARGBMatrix,
        &pixelRange,
        &infoYpCbCrToARGB,
        kvImage422CbYpCrYp8,
        kvImageARGB8888,
        vImage_Flags(kvImageNoFlags))
}
```

Lock the Core Video pixel buffer

Before the sample app accesses the pixel data that AVFoundation supplies as a `CVPixelBuffer`, it calls `CVPixelBufferLockBaseAddress(_:_:)` to lock the pixel buffer and make the underlying memory available.

After the YpCbCr-to-RGB conversion is complete, the code calls `CVPixelBufferUnlockBaseAddress(_:_:)` to unlock the pixel buffer.

The `convertYpCbCrToRGB(cvPixelBuffer:)` function performs the YpCbCr-to-RGB conversion.

```
CVPixelBufferLockBaseAddress(
    pixelBuffer,
    CVPixelBufferLockFlags.readOnly)

convertYpCbCrToRGB(cvPixelBuffer: pixelBuffer)
```

```
CVPixelBufferUnlockBaseAddress(
    pixelBuffer,
    CVPixelBufferLockFlags.readOnly)
```

Create the source luminance and chrominance pixel buffers

The `convertYpCbCrToRGB(cvPixelBuffer:)` function creates two pixel buffers that share memory with the `CVPixelBuffer`. The Core Video pixel buffer contains two planes: the plane at index `0` contains one channel that represents the luminance component, the plane at index 1 contains two interleaved channels that represent the two chrominance components.

The `init(referencing:planeIndex:overrideSize:pixelFormat:)` function initializes a `vImage.PixelBuffer` that references a single plane of a multiple-plane Core Video pixel buffer.

```
let lumaPixelBuffer = vImage.PixelBuffer(referencing: cvPixelBuffer,
                                         planeIndex: 0,
                                         pixelFormat: vImage.Planar8.self)

let chromaPixelBuffer = vImage.PixelBuffer(referencing: cvPixelBuffer,
                                           planeIndex: 1,
                                           pixelFormat: vImage.Interleaved8x2.self)
```

Adjust the contrast of the image

The sample app provides a `Slider` for changing the contrast of the final image. The following code example uses the tone-mapping technique that Adjusting saturation and applying tone mapping describes:

```
if contrast != 1 {
    lumaPixelBuffer.applyGamma(.halfPrecision(contrast),
                               destination: lumaPixelBuffer)
}
```

Convert the YpCbCr image to an ARGB image

The `convert(lumaSource:chromaSource:conversionInfo:)` converts the luminance and chrominance information in `lumaPixelBuffer` and `chromaPixelBuffer` to an ARGB image.

This pixel buffer method calls the underlying vImage `vImageConvert_420Yp8_CbCr8ToARGB8888(_:_:_:_:_:_:_:)` function.

```
argbPixelBuffer.convert(lumaSource: lumaPixelBuffer,
                        chromaSource: chromaPixelBuffer,
                        conversionInfo: infoYpCbCrToARGB)
```

See Also

Conversion Between Image Formats

Building a basic image conversion workflow

Learn the fundamentals of the convert-any-to-any function by converting a CMYK image to an RGB image.

{} Converting color images to grayscale

Convert an RGB image to grayscale using matrix multiplication.

Applying color transforms to images with a multidimensional lookup table

Precompute translation values to optimize color space conversion and other pointwise operations.

Building a basic image conversion workflow

Learn the fundamentals of the convert-any-to-any function by converting a CMYK image to an RGB image.

Conversion

Convert an image to a different format.