Media

#WWDC14

Advances in Core Image

Session 514
David Hayward
Core Image and Raw Camera Teams

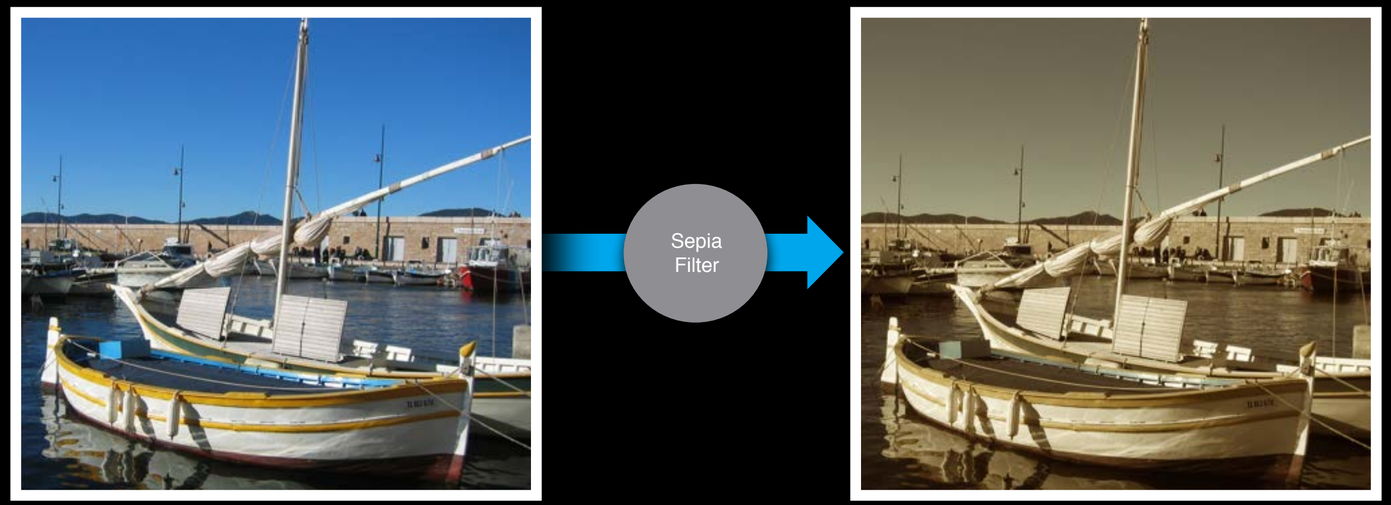
What We Will Discuss Today

What We Will Discuss Today

Key Concepts

Key Concepts

Filters perform per pixel operations on an image

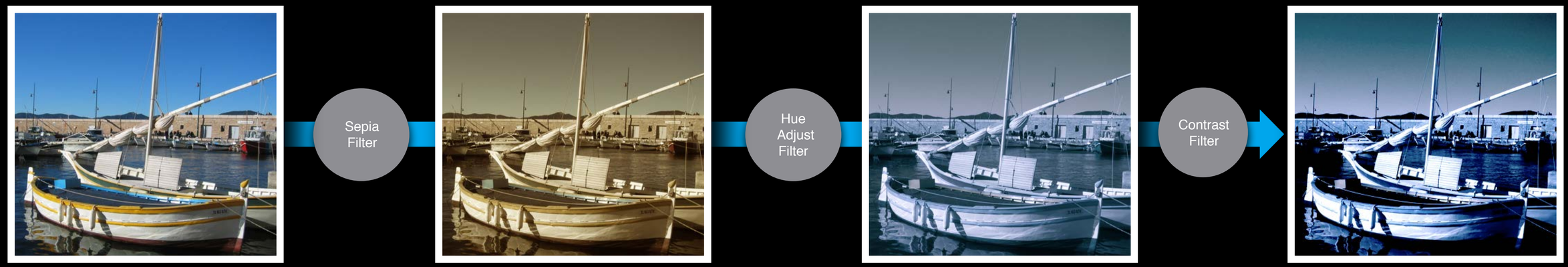


Original

Result

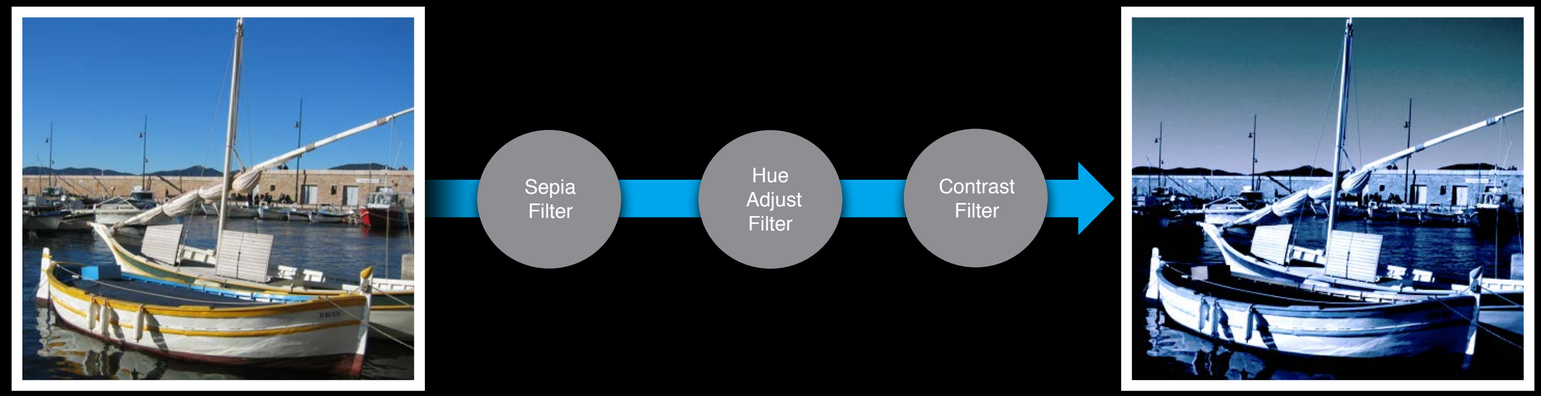
Key Concepts

Filters can be chained together for complex effects



Key Concepts

Intermediate images are lightweight objects

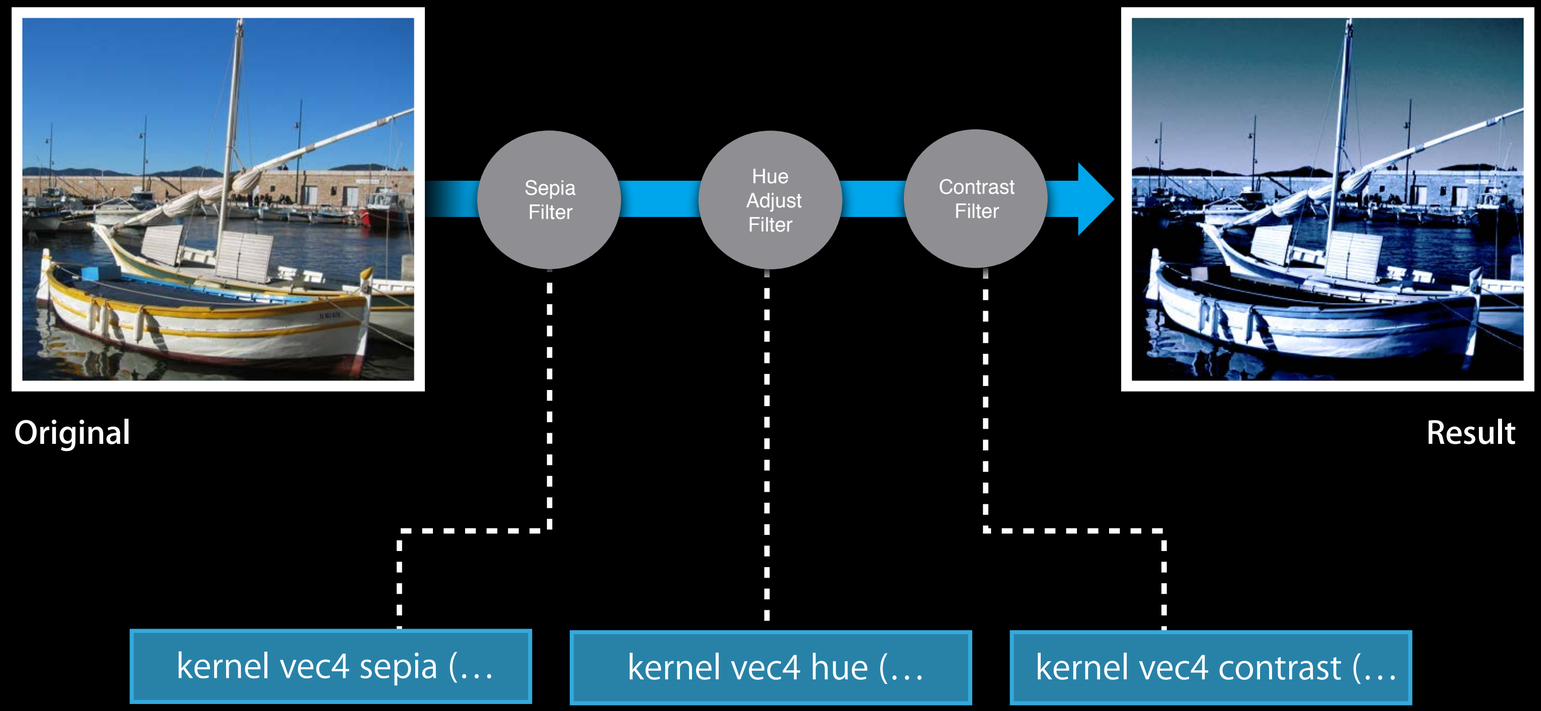


Original

Result

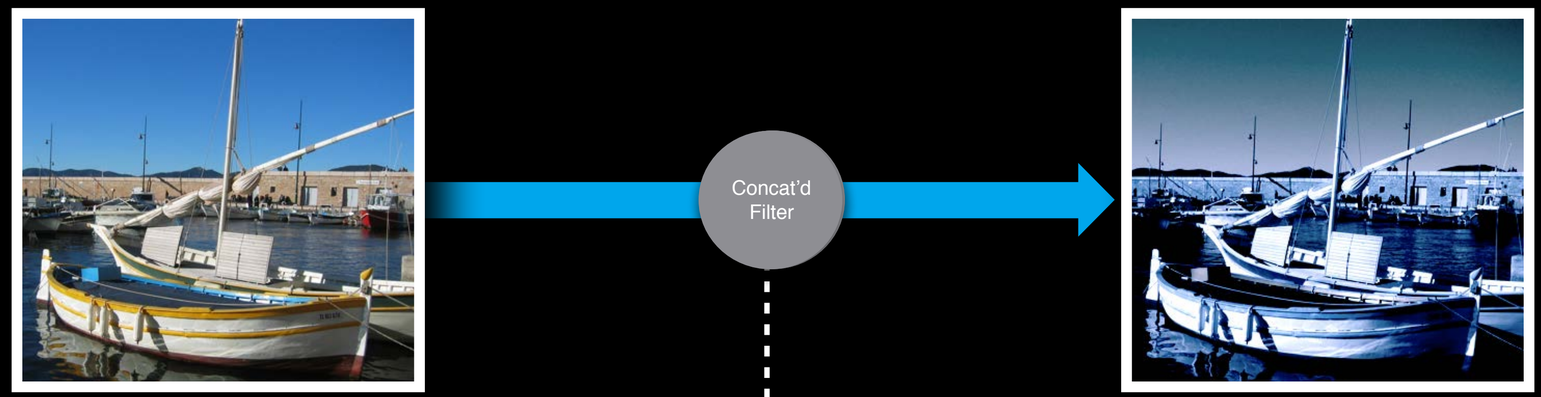
Key Concepts

Each filter has one or more kernel functions



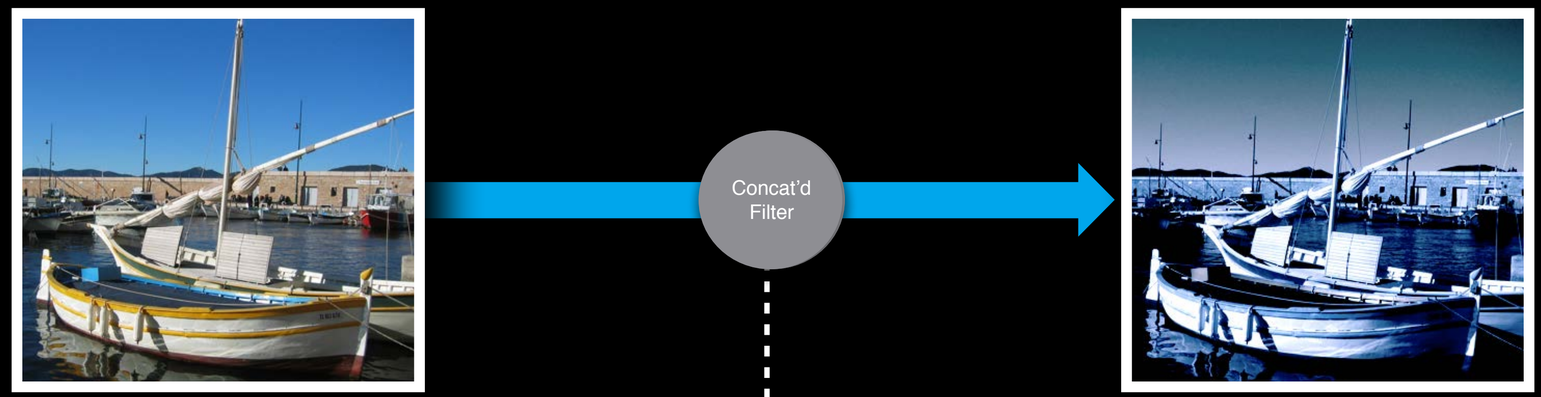
Key Concepts

Kernels concatenated into programs to minimize buffers



Key Concepts

Kernels concatenated into programs to minimize buffers




```
kernel vec4 sepia (…
kernel vec4 hue (…
kernel vec4 contrast (…
```

Core Image Classes

CIKernel

- Represents a program written in Core Image's language

CIFilter

- Has mutable input parameters
- Uses one or more CIKernels to create a new image based on inputs

CIImage

- An immutable object that represents the recipe for an image

CIContext

- A object through which Core Image draw results

What's New in Core Image this Year

What's New

Custom CIKernels on iOS

Photo Editing Extensions on iOS

Large images on iOS

Improved GPU rendering on iOS

API modernization

New built-in CIFilters

New CIDetectors

Improved RAW support on OS X

Using a second GPU on OS X

Custom CIKernels on iOS

Custom CIKernels on iOS

Core Image has over 115 great built-in filters

Now you can easily add your own

- Use the same CIKernel language as on OS X
- With new extensions to make writing typical kernels easier

Custom CIKernels

Where can they go

In your App

- The kernel code can be in a text resource or just a NSString
- The kernel is wrapped in a CIFilter subclass that applies your kernels

In your App Extension

- Photos Editing Extensions can use CIKernels
- Usable to modify photos and videos

Custom CIKernels

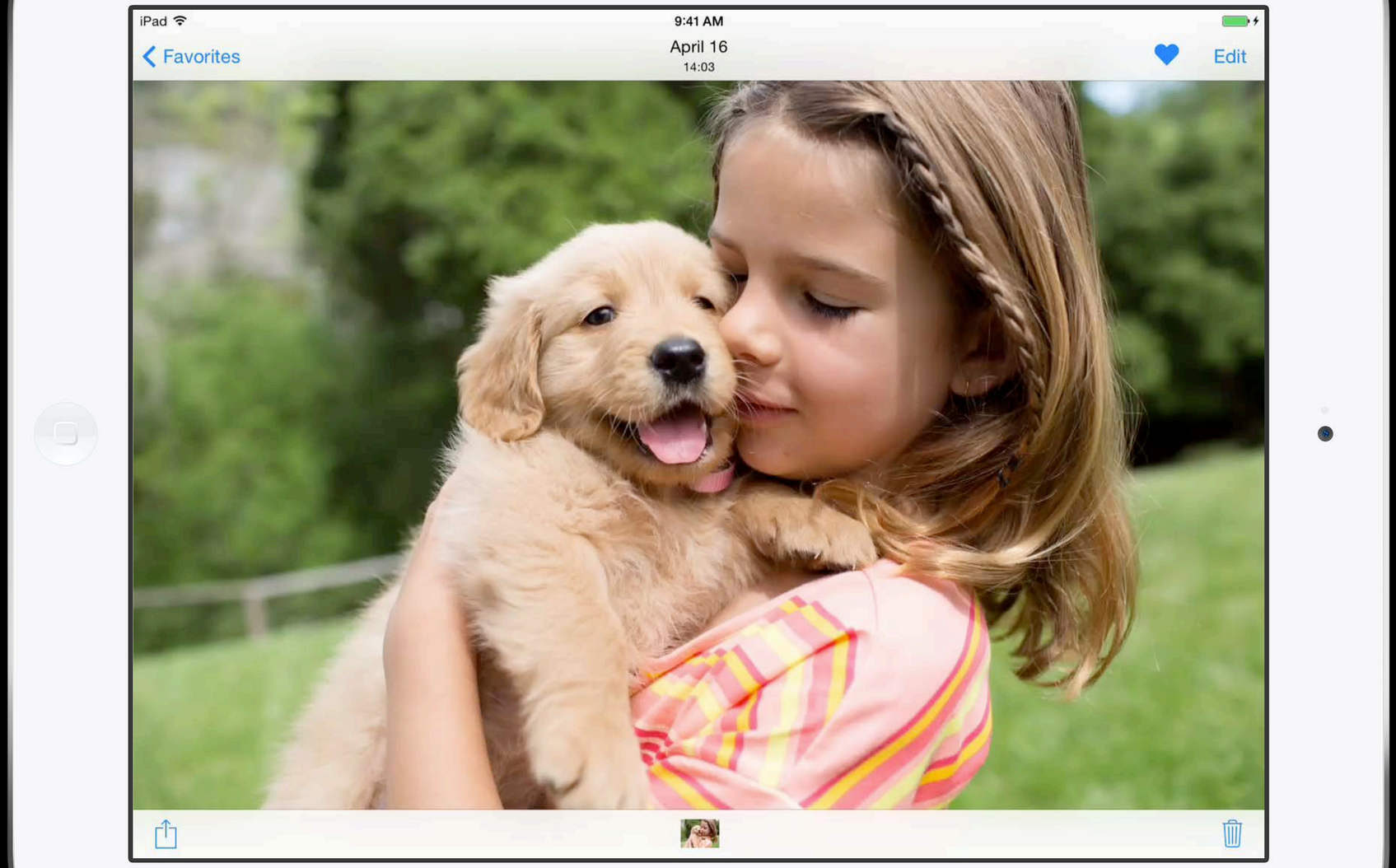
See our next presentation for all the details but heres a teaser

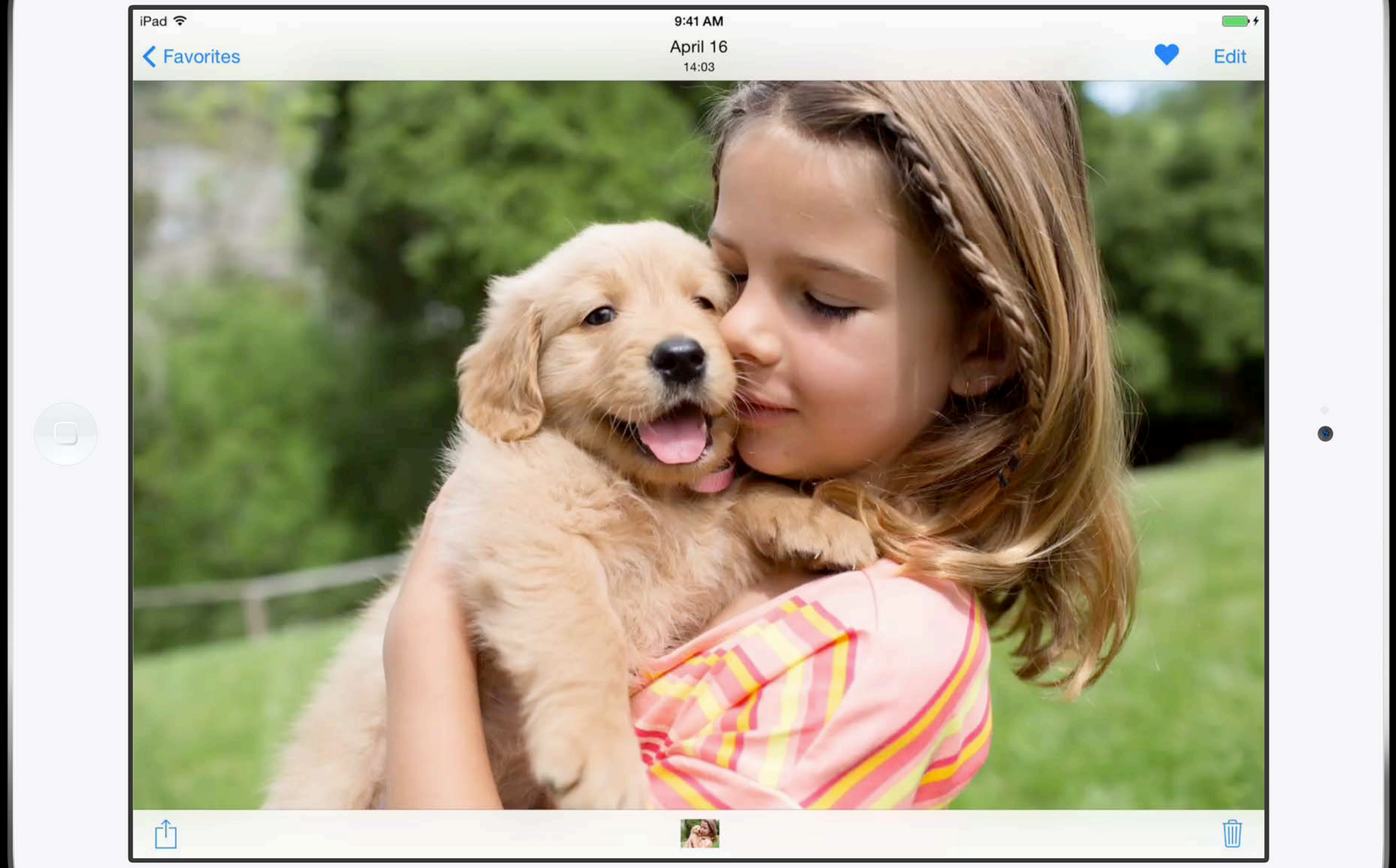
```
{
  NSString* ci_source =
        @"kernel vec4 myInvert (__sample src) \n"
         "{ \n"
         "  return vec4(s.a - src.rgb, s.a); \n"
         "}";

  return [[CIColorKernel kernelWithString:ci_source]
                        applyWithExtent:[img extent]
                              arguments:@[img] ];
}
```

Demo

Custom CIKernel example on iOS

Photo Editing Extensions on iOS


iPad
9:41 AM
Favorites
April 16
14:03
Edit


iPad
9:41 AM
Favorites
April 16
14:03
Edit

Creating a Photo Editing Extension

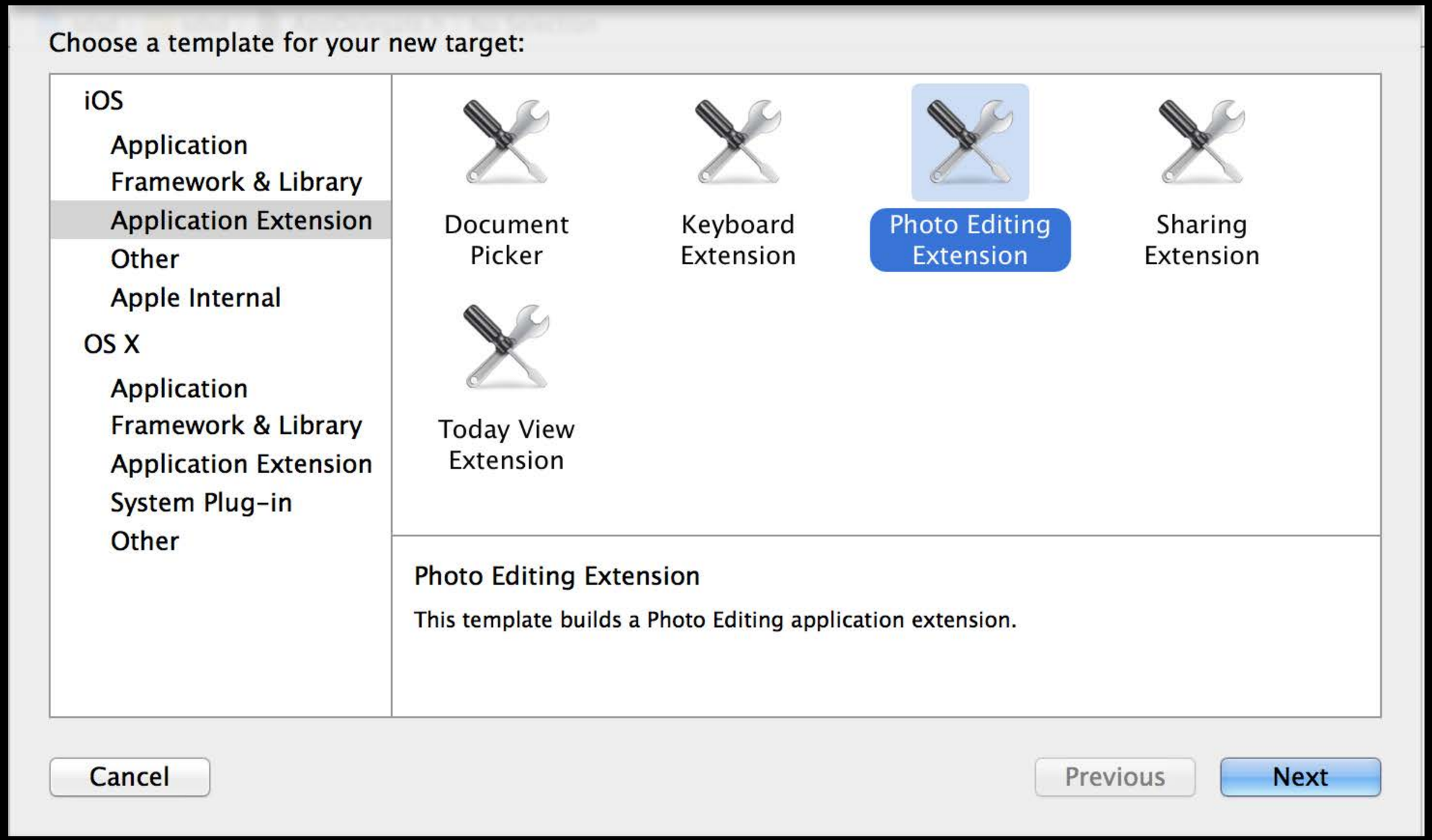


Photo Editing Extension Workflow

Using Core Image efficiently

Photo Editing Extension Workflow

Using Core Image efficiently

1 On init, create display-sized CIImage

Photo Editing Extension Workflow

Using Core Image efficiently

1 On init, create display-sized CIImage

2 On events, apply filters and draw

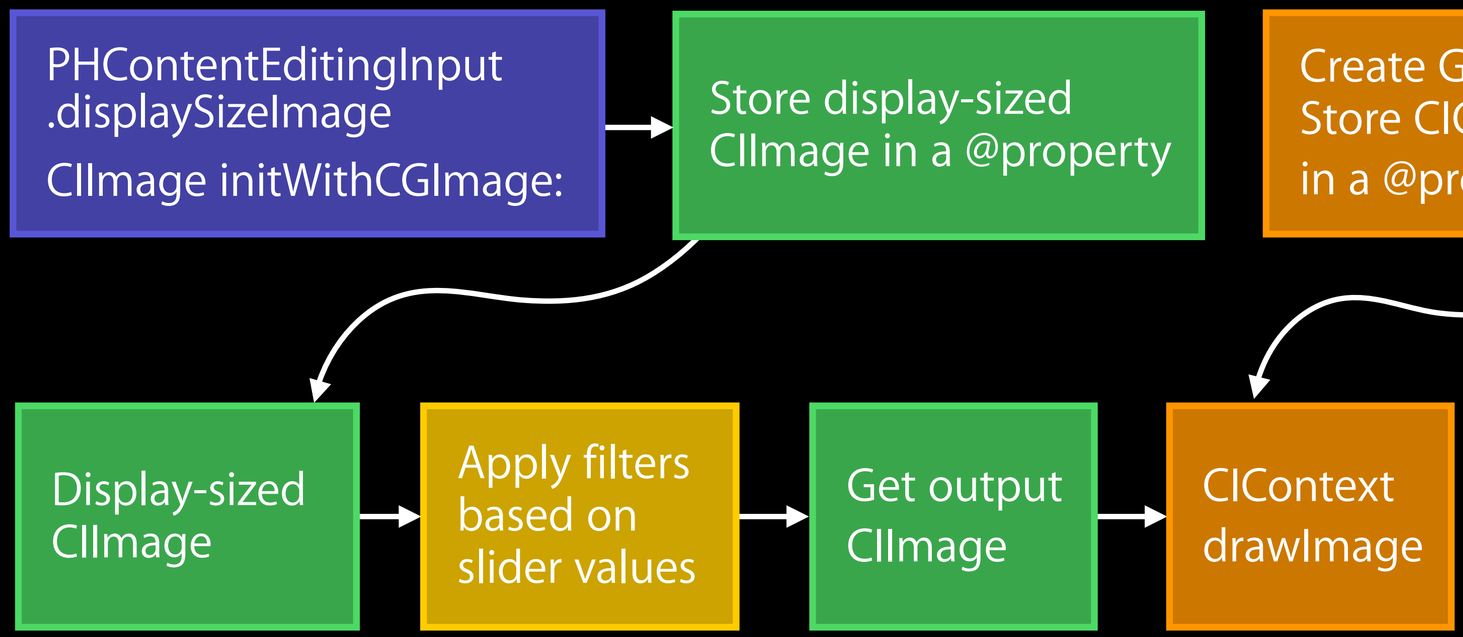
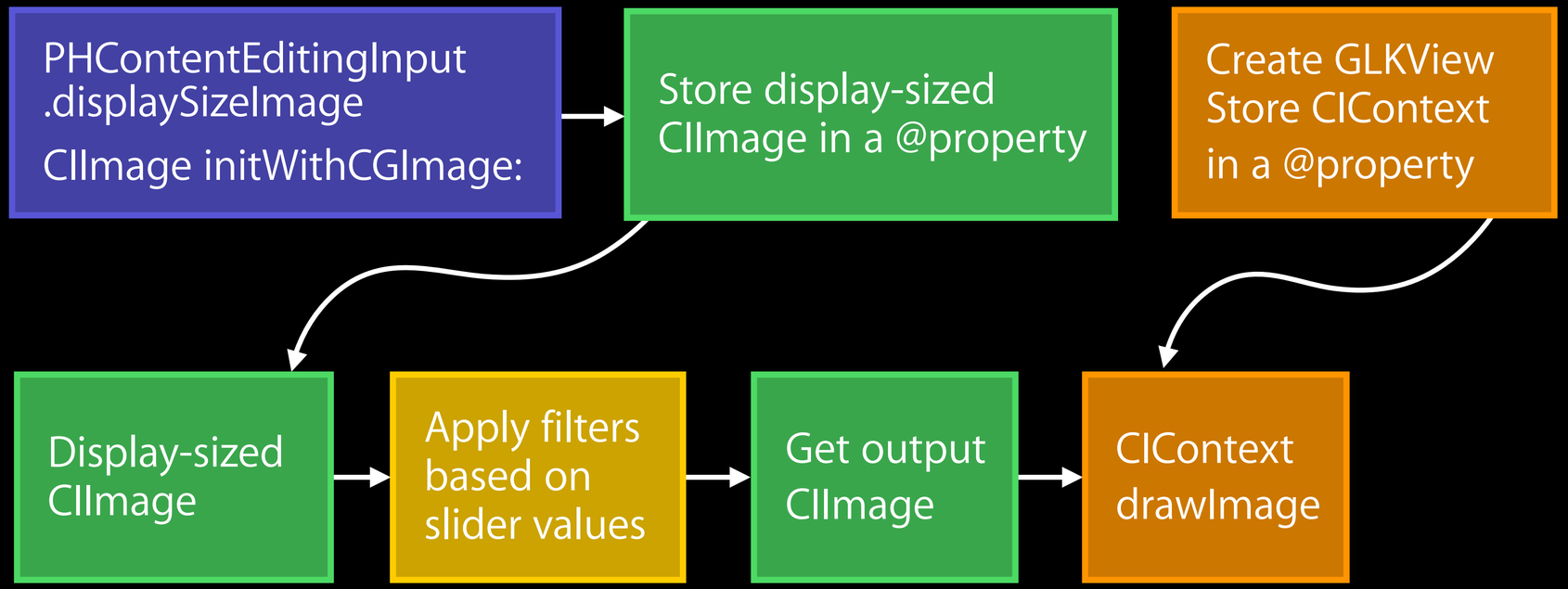


Photo Editing Extension Workflow

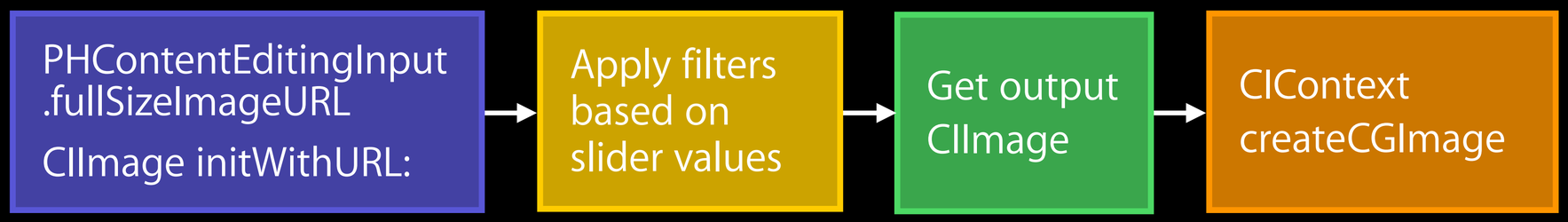
Using Core Image efficiently

1 On init, create display-sized CIImage



2 On events, apply filters and draw

3 On finish, return adjustments and full-sized filtered JPEG



Large Images on iOS

Large Images Support

Full support for images greater than the GPU limits

- Input images can be > 4K
- Output renders can be > 4K

Automatic tiling

- Leverages improvements to ImageIO's jpeg decoder/encoder
- Facilitated by features of the CIKernel language

Large Images Support

Facilitated by features of the CIKernel language

CIKernel language allows kernels to “just work” regardless of tiling

`destCoord()`

- Allows Core Image to support tiled output automatically

`samplerTransform()`

- Allows Core Image to support tiled inputs automatically

Large Images Support

Core Image, CGImageRef, and the benefits of laziness

For small input CGImages

- Image is fully decoded when [CIImage initWith] is called

For large input CGImages

- Image is decoded as needed when [CIContext render] is called

Large Images Support

Core Image, CGImageRef, and the benefits of laziness

When calling [CIContext createCGImage]

For small output CGImages

- Image is fully rendered when createCGImage is called

For large output CGImages

- Image is rendered as needed when CGImage is rendered
- Or when CGImageDestinationFinalize is called

Large Images Support

Core Image, CGImageRef, and the benefits of laziness

Very large JPEGs can be

- Decoded
- CIFiltered
- Re-encoded

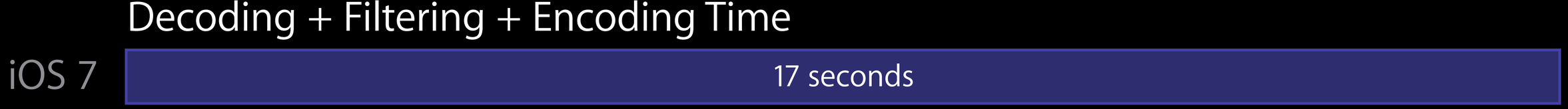
With minimal memory and great performance

Large Images Support

Apply CISepiaTone on 4200x6300 JPEG (~100 MB image)

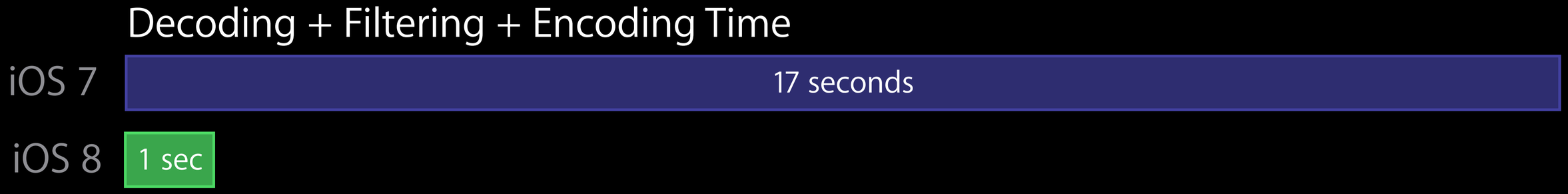
Large Images Support

Apply CISepiaTone on 4200x6300 JPEG (~100 MB image)



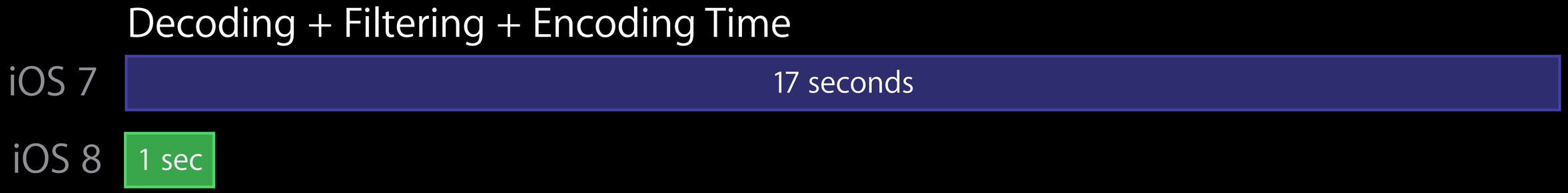
Large Images Support

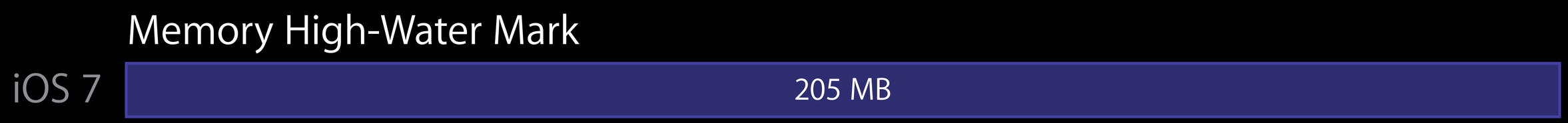
Apply CISepiaTone on 4200x6300 JPEG (~100 MB image)



Large Images Support

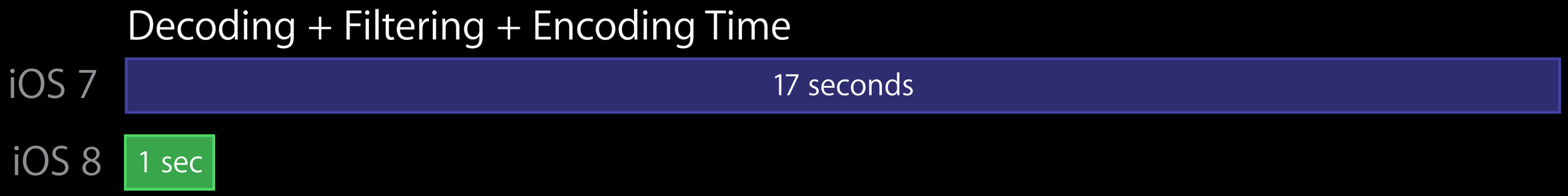
Apply CISepiaTone on 4200x6300 JPEG (~100 MB image)





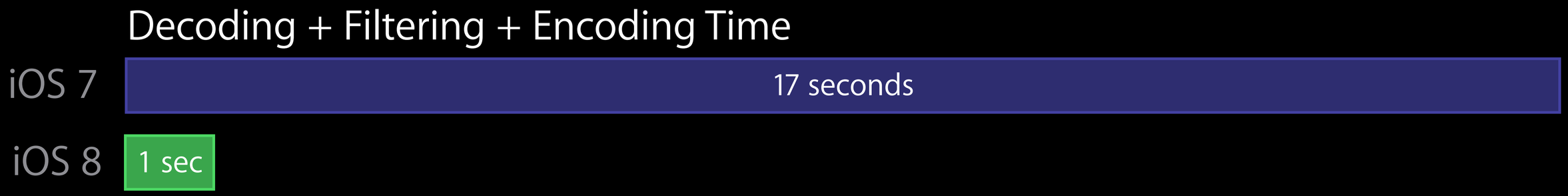
Large Images Support

Apply CISepiaTone on 4200x6300 JPEG (~100 MB image)



Large Images Support

Apply CISepiaTone on 4200x6300 JPEG (~100 MB image)



iOS 7 Full size image rendered on CPU

iOS 8 Automatically tiled image rendered on the GPU!

Improved GPU Rendering on iOS

Renders When Your App Is Background

In iOS 7

- All background renders use Core Image CPU Render

In iOS 8

- Renders within a short time of switching to background
  - Use faster GPU renderer
  - Serviced with a lower priority
  - Will not disturb foreground GPU usage

Background GPU Restrictions

Not allowed if you call

```
[CIContext drawImage:inRect:fromRect:]
```

Use other renders methods like

```
[CIContext createCGImage:fromRect:]
[CIContext render:toCVPixelBuffer:]
[CIContext render:toBitmap:rowBytes:bounds:format:colorSpace:]
```

Low Priority Foreground Rendering

When rendering from a secondary thread

In iOS 7

- Required care to avoid interrupting UI thread renders
- Or using slower CPU rendering

In iOS 8

- Secondary thread renders can use `kCIContextPriorityRequestLow`
- Will not disturb UI thread GPU usage

Thoughts on Core Image CPU Renderer

Thoughts on Core Image CPU Renderer

In iOS 7, the CPU renderer was used when

Thoughts on Core Image CPU Renderer

In iOS 7, the CPU renderer was used when

- GPU texture limits were exceeded

Thoughts on Core Image CPU Renderer

In iOS 7, the CPU renderer was used when

- ~~GPU texture limits were exceeded~~
  - No longer a limit in iOS 8 Core Image

Thoughts on Core Image CPU Renderer

In iOS 7, the CPU renderer was used when

- ~~GPU texture limits were exceeded~~
  - No longer a limit in iOS 8 Core Image
- The application needed to render briefly in the background

Thoughts on Core Image CPU Renderer

In iOS 7, the CPU renderer was used when

- ~~GPU texture limits were exceeded~~
  - No longer a limit in iOS 8 Core Image
- ~~The application needed to render briefly in the background~~
  - No longer prohibited in iOS 8 Core Image

Thoughts on Core Image CPU Renderer

In iOS 7, the CPU renderer was used when

- ~~GPU texture limits were exceeded~~
  - No longer a limit in iOS 8 Core Image
- ~~The application needed to render briefly in the background~~
  - No longer prohibited in iOS 8 Core Image
- The application wanted to render in a low priority thread

Thoughts on Core Image CPU Renderer

In iOS 7, the CPU renderer was used when

- ~~GPU texture limits were exceeded~~
  - No longer a limit in iOS 8 Core Image
- ~~The application needed to render briefly in the background~~
  - No longer prohibited in iOS 8 Core Image
- ~~The application wanted to render in a low priority thread~~
  - Can now request `kCIContextPriorityRequestLow` in iOS 8 Core Image

API Modernization

Core Image API Modernization

Properties are fully supported on OS X

CIFilter subclasses can use @property instead of ivars

- Reminder—filter subclasses don't need to release input ivars/properties

Code that looked like this

```
outImage = [filter valueForKey: kCIOutputImageKey ];
```

Can look like this now on OS X

```
outImage = filter.outputImage;
```

Core Image API Modernization

Set several parameters on a filter

```
f = [CIFilter filterWithName: @"CIColorControls"
        withInputParameters: @{
                        @"inputImage"      : inImage,
                        @"inputSaturation" : @0.5,
                        @"inputBrightness" : @1.2,
                        @"inputContrast"   : @1.3 }];
outImage = f.outputImage;
```

Core Image API Modernization

Apply a filter with several parameters on an image


```
outImage = [inImage
          imageByApplyingFilter: @"CIColorControls"
            withInputParameters: @{
                        @"inputSaturation" : @0.5,
                        @"inputBrightness" : @1.2,
                        @"inputContrast"   : @1.3 }];
```

Core Image API Modernization

Convenient methods for orienting images

Orientation values from one to eight as defined in the TIFF specification

- `(CIImage*) imageByApplyingOrientation:(int)orientation`
- `(CGAffineTransform)imageTransformForOrientation:(int)orientation`

Core Image API Modernization

Color spaces

Default RGB color space is now sRGB

- Matches default RGB color space on iOS
- Matches what most apps expect for untagged content

Core Image API Modernization

Color spaces

Default RGB color space is now sRGB

- Matches default RGB color space on iOS
- Matches what most apps expect for untagged content

Working color space is linear Rec. 709

- Matches the default working color space on iOS
- No matrix math is needed when converting
  input RGB → working space → output RGB

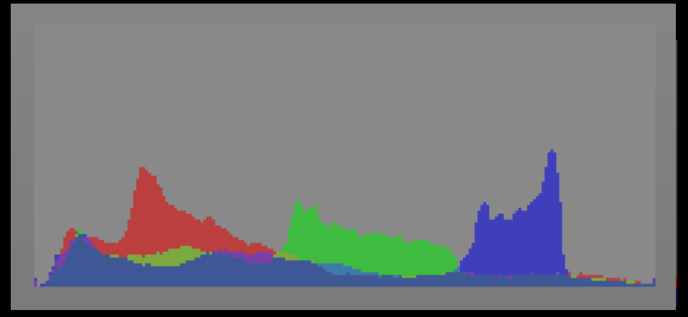
New Built-in CIFilters

New CIFilters on iOS

CIAreaHistogram + CIHistogramDisplayFilter

histogram data image
256 x 1 pixels

=

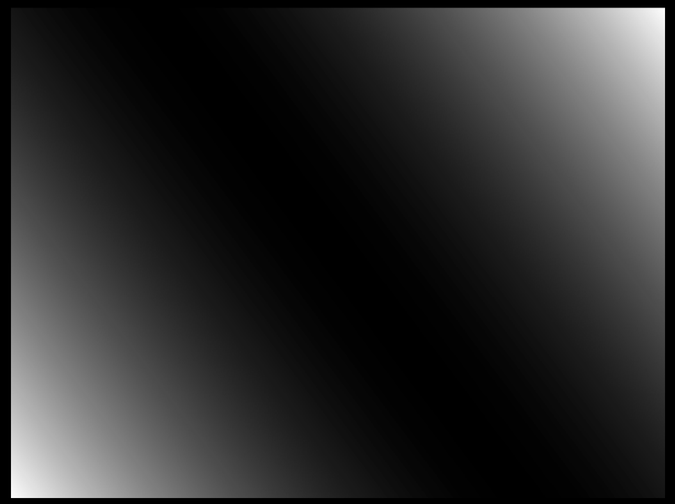


New CIFilters

CIMaskedVariableBlur

NEW

+



=

New CIFilters

CIAccordionFoldTransition


```
f.inputNumberOfFolds = @3;
f.inputBottomHeight = @50;
```

```
kernel vec4 accordianFoldTransition (
    sampler shortImage, sampler tallImage,
    vec3 foldParms, float time, vec4 dims)
{
}
```

New CIFilters

CIAccordionFoldTransition


```
kernel vec4 accordianFoldTransition (
    sampler shortImage, sampler tallImage,
    vec3 foldParms, float time, vec4 dims)
{
}
```

New CIFilters

CICode128BarcodeGenerator


```
f.inputMessage = [NSData dataWithBytes:"Robot Barf" length:10];
f.inputQuietSpace = @7;
```

New CIFilters

CIAztecCodeGenerator


```
f.inputMessage = [NSData dataWithBytes:"Robot Barf" length:10];
f.inputCorrectionLevel= @23;
```

New CIFilters

CIPerspectiveCorrection

NEW

New CIFilters

More blend modes

CILinearDodgeBlendMode

CILinearBurnBlendMode

CIPinLightBlendMode

CISubtractBlendMode

CIDivideBlendMode

CISoftLightBlendMode changed to better match its spec

New CIFilters on iOS

CIGlassDistortion

CIStretchCrop

CIDroste

And some more if time permits…

115 Built-in CIFilters on iOS

CIAccordionFoldTransition
CIAdditionCompositing
CIAffineClamp
CIAffineTile
CIAffineTransform
CIAreaHistogram
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CIBumpDistortionLinear
CICheckerboardGenerator
CICircleSplashDistortion
CICircularScreen
CICode128BarcodeGenerator
CIColorBlendMode
CIColorBurnBlendMode
CIColorControls
CIColorCube
CIColorDodgeBlendMode
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp
CIColorCrossPolynomial
CIColorPolynomial
CIColorPosterize
CIConstantColorGenerator
CIConvolution3X3
CIConvolution5X5
CIConvolution9Horizontal
CIConvolution9Vertical
CICopyMachineTransition
CICrop
CIDarkenBlendMode
CIDifferenceBlendMode
CIDisintegrateWithMask
CIDissolveTransition
CIDivideBlendMode
CIDotScreen
CIEightfoldReflectedTile
CIExclusionBlendMode
CIExposureAdjust
CIFalseColor
CIFlashTransition
CIFourfoldReflectedTile
CIFourfoldTranslatedTile
CIGammaAdjust
CIGaussianBlur
CIGaussianGradient
CIGlassDistortion
CIGlideReflectedTile
CIGloom
CIHardLightBlendMode
CIHatchedScreen
CIHighlightShadowAdjust
CIHistogramDisplayFilter
CIHoleDistortion
CIHueAdjust
CIHueBlendMode
CILanczosScaleTransform
CILightenBlendMode
CILightTunnel
CILinearBurnBlendMode
CILinearDodgeBlendMode
CILinearGradient
CILineScreen
CILuminosityBlendMode
CIMaskToAlpha
CIMaximumComponent
CIMaximumCompositing
CIMinimumComponent
CIMinimumCompositing
CIModTransition
CIMultiplyBlendMode
CIMultiplyCompositing
CIOverlayBlendMode
CIPerspectiveCorrection
CIPerspectiveTile
CIPerspectiveTransform
CIPinchDistortion
CIPinLightBlendMode
CIPixellate
CIRadialGradient
CIRandomGenerator
CISaturationBlendMode
CIScreenBlendMode
CISepiaTone
CISharpenLuminance
CISixfoldReflectedTile
CISixfoldRotatedTile
CISoftLightBlendMode
CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISubtractBlendMode
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

New CIDetectors

New CIDetectors

CIDetector is a an abstract class to find things within an image

Three types of detectors

- `CIDetectorTypeFace`
- `CIDetectorTypeRectangle`
- `CIDetectorTypeQRCode`

New CIDetectors

Creating a CIDetector object

Creating a detector

```
CIDetector* detector = [CIDetector detectorOfType:CIDetectorTypeFace
                                          context:nil
                                          options:opts];
```

Options

- Tell the detector to be fast or thorough

```
opts = @{CIDetectorAccuracy : CIDetectorAccuracyLow };
opts = @{CIDetectorAccuracy : CIDetectorAccuracyHigh };
```

- Tell the detector the smallest size to search for

```
opts = @{CIDetectorMinFeatureSize : @100 };
```

New CIDetectors

Creating a CIDetector object

Creating a detector

```
CIDetector* detector = [CIDetector detectorOfType:CIDetectorTypeRectangle
                                          context:nil
                                          options:opts];
```

Options

- Tell the detector to be fast or thorough

```
opts = @{CIDetectorAccuracy : CIDetectorAccuracyLow };
opts = @{CIDetectorAccuracy : CIDetectorAccuracyHigh };
```

- Tell the detector the smallest size to search for

```
opts = @{CIDetectorMinFeatureSize : @100 };
```

New CIDetectors

Creating a CIDetector object

Creating a detector

```
CIDetector* detector = [CIDetector detectorOfType:CIDetectorTypeQRCode
                                          context:nil
                                          options:opts];
```

Options

- Tell the detector to be fast or thorough

```
opts = @{CIDetectorAccuracy : CIDetectorAccuracyLow };
opts = @{CIDetectorAccuracy : CIDetectorAccuracyHigh };
```

- Tell the detector the smallest size to search for

```
opts = @{CIDetectorMinFeatureSize : @100 };
```

New CIDetectors

Asking a detector for CIFaceFeatures

Important—Tell the detector what what direction is up

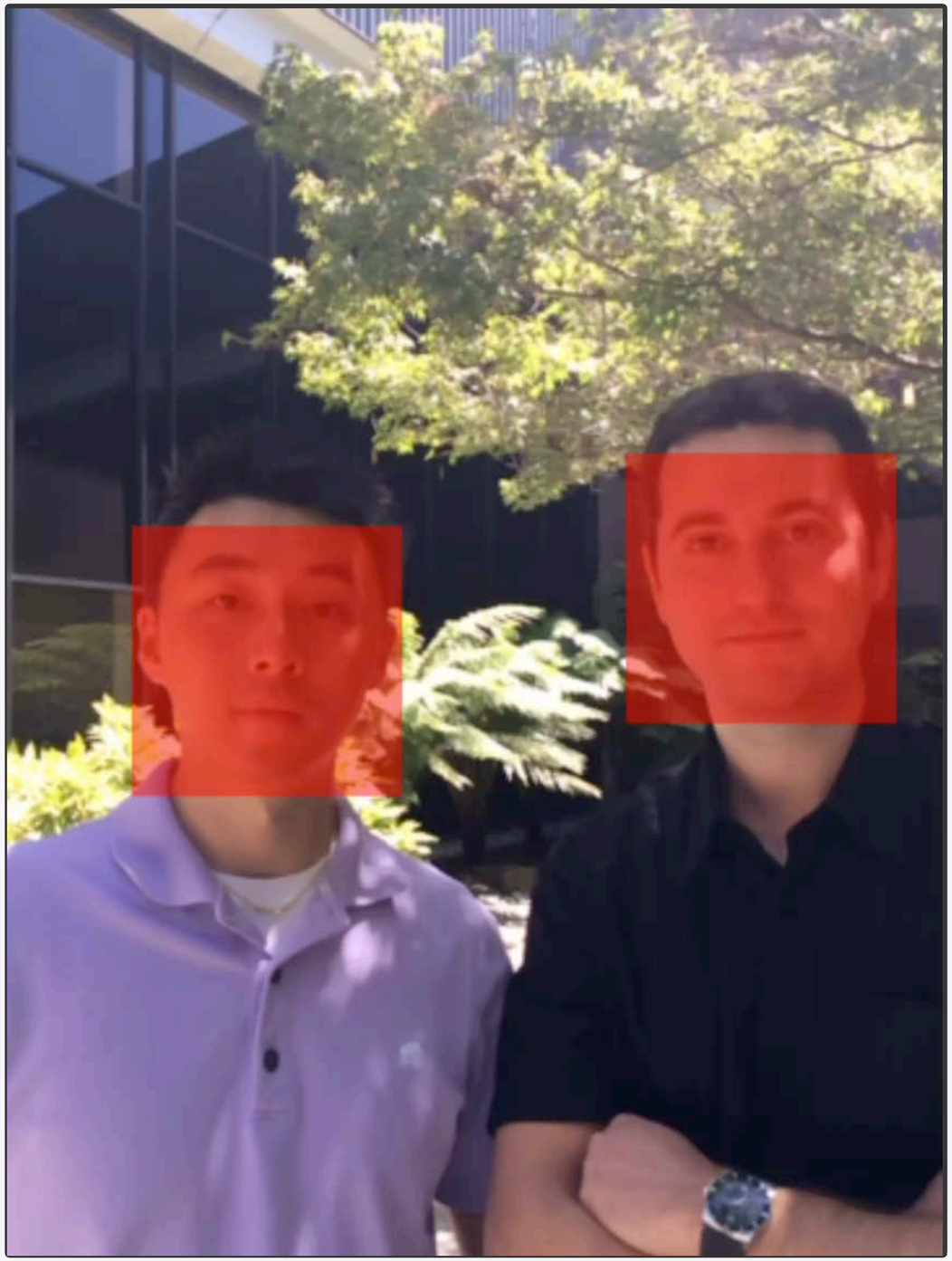
```
opts = @{ CIDetectorImageOrientation :
          [[image properties] valueForKey:kCGImagePropertyOrientation],
          CIDetectorEyeBlink : @YES,
          CIDetectorSmile : @YES};
```

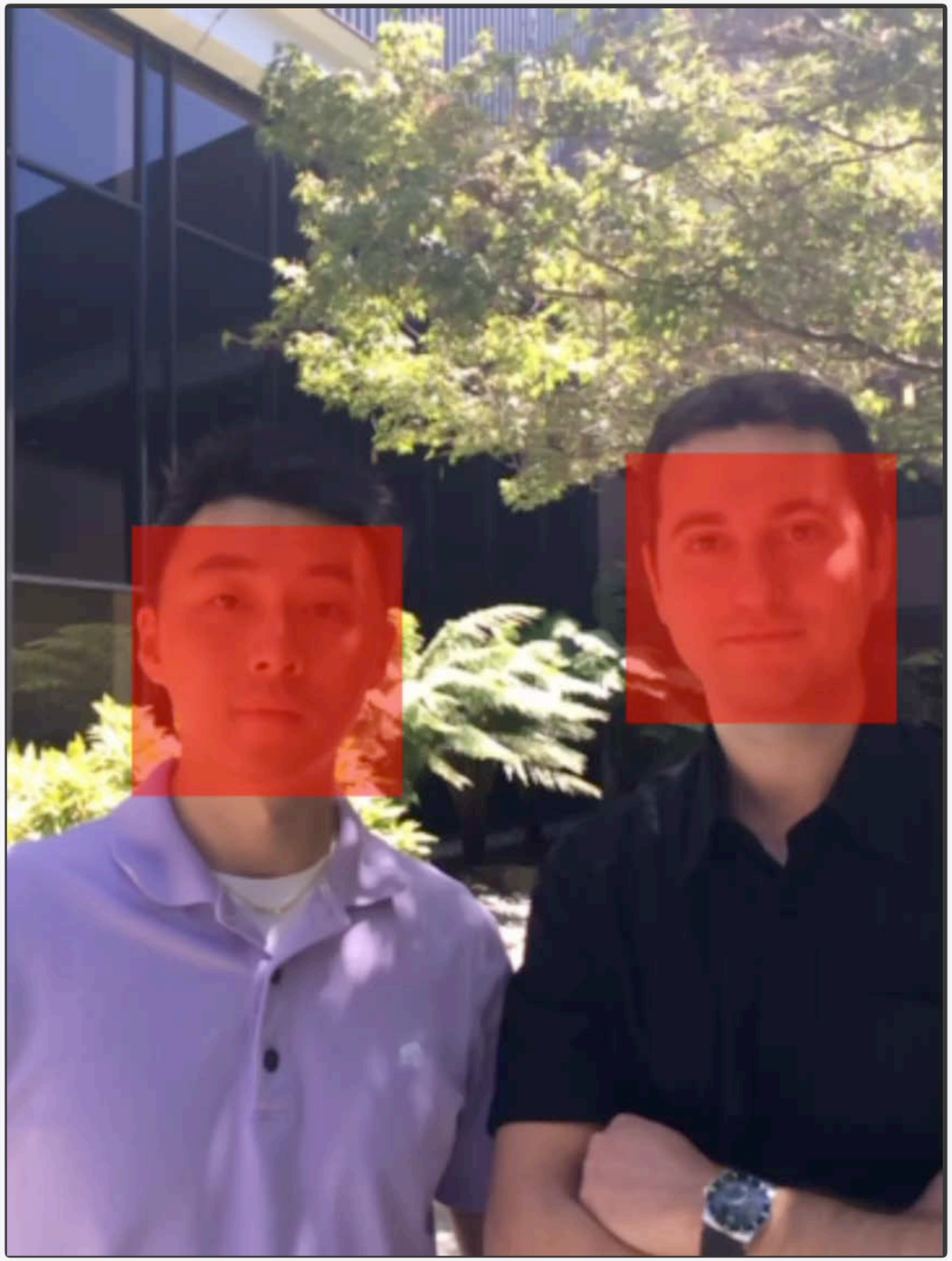
```
NSArray* features = [detector featuresInImage:image
                                      options:opts];
```

New CIDetectors

Augmenting an image with CIFaceFeatures

```
result = image;
for (CIFaceFeature *f in features)
{
  bool eyeClosed = f.leftEyeClosed || f.rightEyeClosed;
  CIImage * overlay = [CIImage imageWithColor: eyeClosed ? color1 : color2];
  overlay = [overlay imageByCroppingToRect:f.bounds];

  result = [overlay imageByCompositingOverImage:result];
}
```





New CIDetectors

Asking a detector for CIRectangleFeatures

Important—tell the detector what aspect ratio and minimum size to look for

```
opts = @{ CIDetectorAspectRatio : @2.0 };

NSArray* features = [detector featuresInImage:image
                                      options:opts];
```

New CIDetectors

Augmenting an image with CIRectangleFeatures

```
result = image;
for (CIRectangleFeature *f in features)
{
  CIImage * overlay = [CIImage imageWithColor:color];
  overlay = [overlay imageByApplyingFilter:
               @"CIPerspectiveTransformWithExtent"
             withInputParameters: @{
               @"inputExtent", [CIVector vectorWithX:0 Y:0 Z:1 W:1],
              @"inputTopLeft", [CIVector vectorWithCGPoint:f.topLeft],
               @"inputTopRight", [CIVector vectorWithCGPoint:f.topRight],
            @"inputBottomLeft", [CIVector vectorWithCGPoint:f.bottomLeft],
           @"inputBottomRight", [CIVector vectorWithCGPoint:f.bottomRight]}];
 result = [overlay imageByCompositingOverImage:result];
}
```


Record
18.6 fps
Settings
Filters
David
Hayward


Record
18.6 fps
Settings
Filters
David
Hayward

New CIDetectors

Augmenting an image with CIQRCodeFeatures

```
result = image;
for (CIQRCodeFeature *f in features)
{
  CIImage * overlay = [CIImage imageWithColor:color];
  overlay = [overlay imageByApplyingFilter:
               @"CIPerspectiveTransformWithExtent"
             withInputParameters: @{
               @"inputExtent", [CIVector vectorWithX:0 Y:0 Z:1 W:1],
              @"inputTopLeft", [CIVector vectorWithCGPoint:f.topLeft],
               @"inputTopRight", [CIVector vectorWithCGPoint:f.topRight],
            @"inputBottomLeft", [CIVector vectorWithCGPoint:f.bottomLeft],
           @"inputBottomRight", [CIVector vectorWithCGPoint:f.bottomRight]}];
 result = [overlay imageByCompositingOverImage:result];
}
```


Record
29.0 fps
Settings
Filters


Record
29.0 fps
Settings
Filters

Improved RAW Support on OS X

Improved RAW Support on OS X

History

Fundamentals of RAW image processing

Architecture overview

Using the CIRAWFilter

Adjusting RAW Images

History

Apple has been supporting RAW files since April 2005

Adjusting RAW Images

History

Apple has been supporting RAW files since April 2005

We have continuously added support for cameras and improved quality

Adjusting RAW Images

History

Apple has been supporting RAW files since April 2005

We have continuously added support for cameras and improved quality

RAW Support is provided for the entire OS X

Adjusting RAW Images

History

Apple has been supporting RAW files since April 2005

We have continuously added support for cameras and improved quality

RAW Support is provided for the entire OS X

- NSImage, CGImage

Adjusting RAW Images

History

Apple has been supporting RAW files since April 2005

We have continuously added support for cameras and improved quality

RAW Support is provided for the entire OS X

- NSImage, CGImage
- Spotlight, Quick Look

Adjusting RAW Images

History

Apple has been supporting RAW files since April 2005

We have continuously added support for cameras and improved quality

RAW Support is provided for the entire OS X

- NSImage, CGImage
- Spotlight, Quick Look
- Preview, Finder, Mail

Adjusting RAW Images

History

Apple has been supporting RAW files since April 2005

We have continuously added support for cameras and improved quality

RAW Support is provided for the entire OS X

- NSImage, CGImage
- Spotlight, Quick Look
- Preview, Finder, Mail
- Aperture, iPhoto, Photos

Adjusting RAW Images

History

Apple has been supporting RAW files since April 2005

We have continuously added support for cameras and improved quality

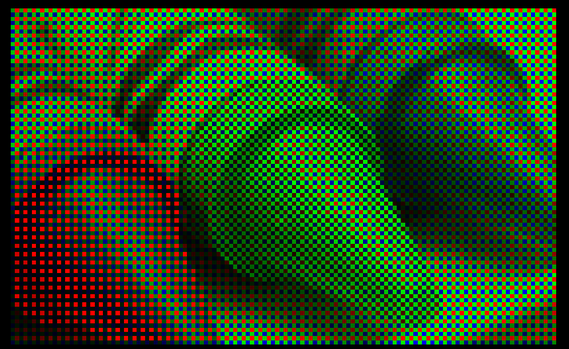
RAW Support is provided for the entire OS X

- NSImage, CGImage
- Spotlight, Quick Look
- Preview, Finder, Mail
- Aperture, iPhoto, Photos
- Third-party Apps

Adjusting RAW Images

Fundamentals of RAW image processing

RAW files contains minimally processed
data from the camera image sensor

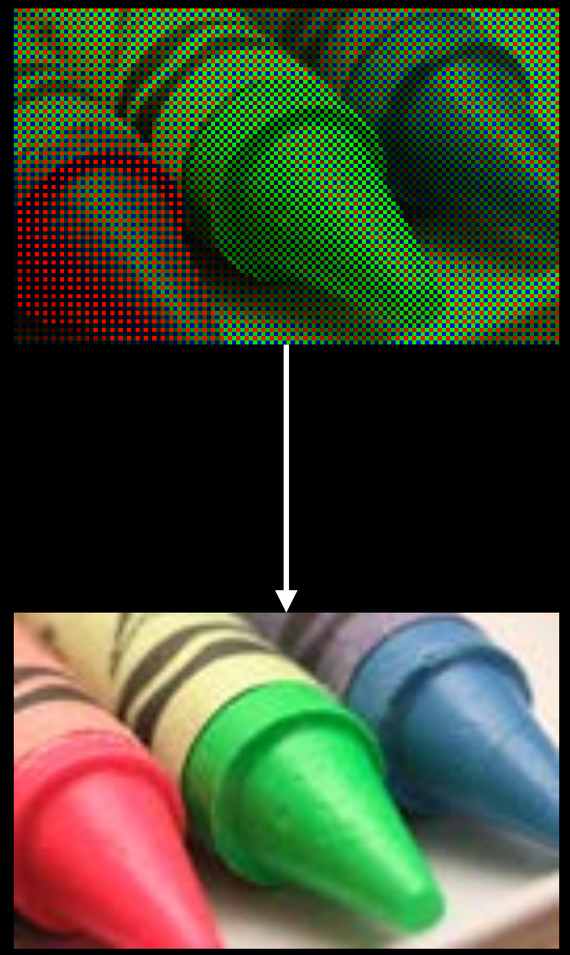


Adjusting RAW Images

Fundamentals of RAW image processing

RAW files contains minimally processed data from the camera image sensor

Requires advanced image processing to produce a great image



Adjusting RAW Images

Stages of RAW image processing

Extract critical metadata

Decode raw sensor image

De-mosaic reconstruction

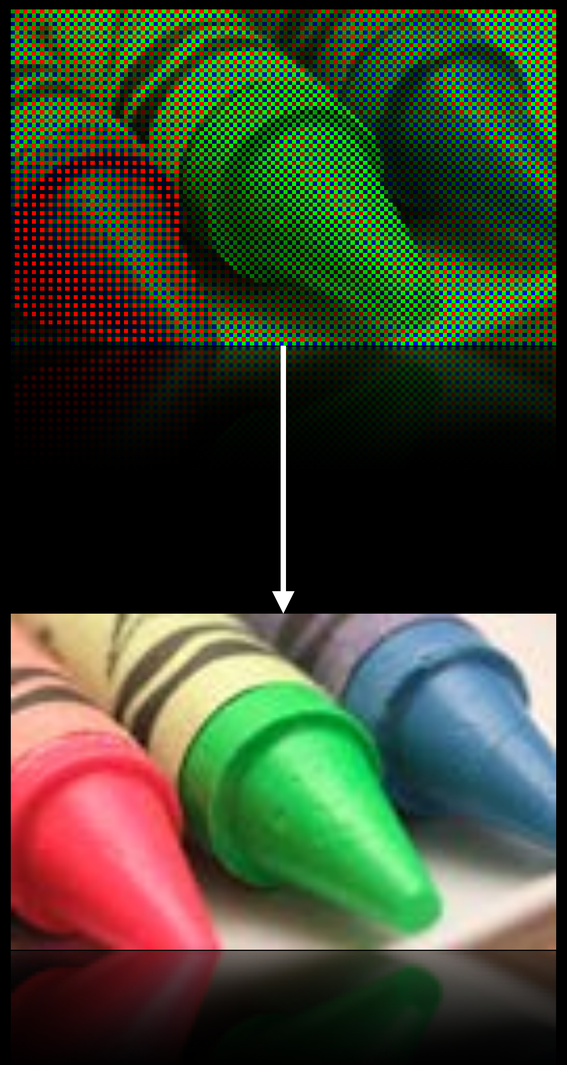
Lens correction

Noise reduction

Map scene-referred sensor values
to output-referred color space

Adjust exposure and temperature/tint

Add contrast and saturation for a pleasing look



Adjusting RAW Images

Stages of RAW image processing

NEW

Extract critical metadata

Decode raw sensor image

De-mosaic reconstruction

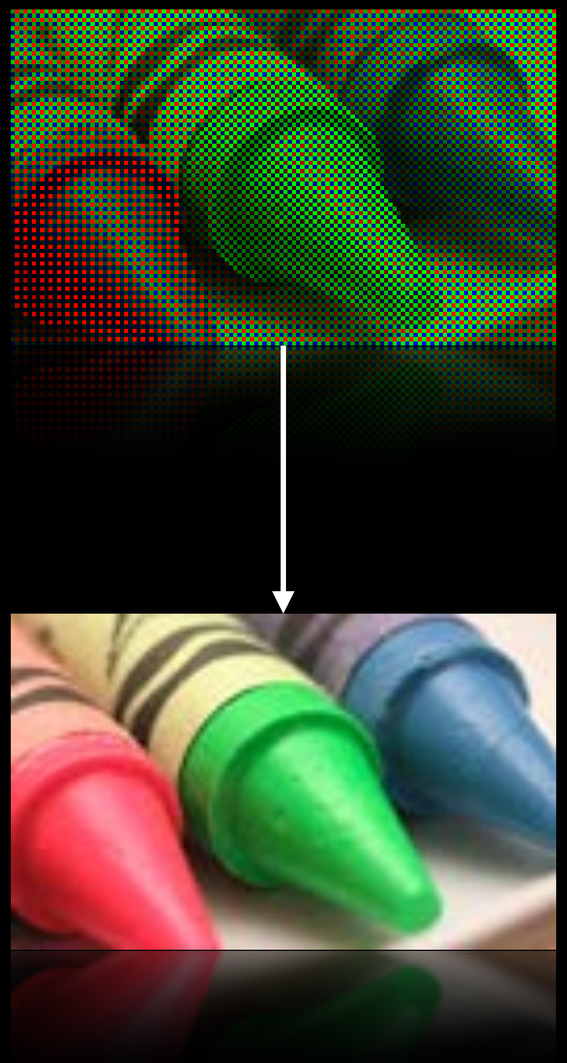
Lens correction

Noise reduction

Map scene-referred sensor values
to output-referred color space

Adjust exposure and temperature/tint

Add contrast and saturation for a pleasing look



Adjusting RAW Images

Beyond the basics

ImageIO just returns a CGImageRef

- Processed according to the default parameters and latest algorithms

Adjusting RAW Images

Beyond the basics

ImageIO just returns a CGImageRef

- Processed according to the default parameters and latest algorithms

CIRAWFilter gives your application

- CIImage with extend range, floating point precision
- Easy control over RAW processing parameters
- Fast, interactive performance using GPU

Adjusting RAW Images

Architecture overview

RAW Image File

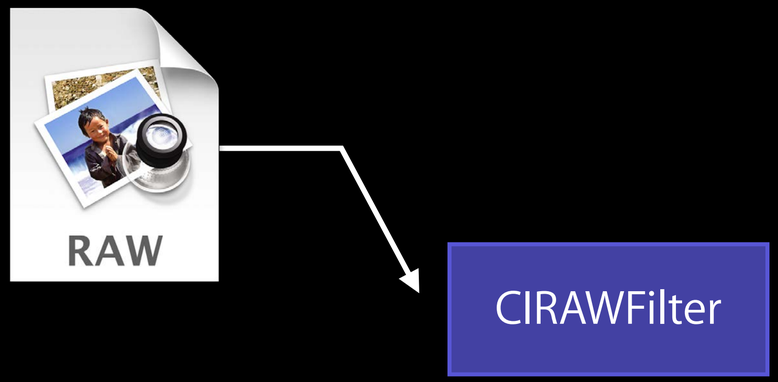
(File's URL or Data)

Adjusting RAW Images

Architecture overview

RAW Image File

(File's URL or Data)



Adjusting RAW Images

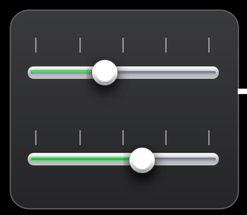
Architecture overview

RAW Image File

(File's URL or Data)

User Adjustments

(Exposure,
Temperature,
Noise Reduction, …)



CIRAWFilter

Adjusting RAW Images

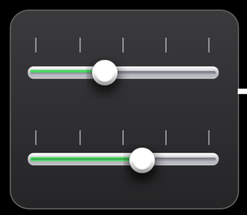
Architecture overview

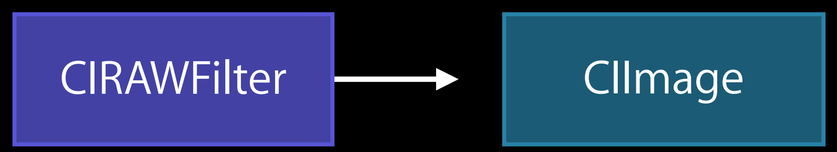
RAW Image File

(File's URL or Data)

User Adjustments

(Exposure,
Temperature,
Noise Reduction, …)





Adjusting RAW Images

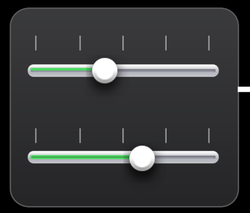
Architecture overview

RAW Image File

(File's URL or Data)

User Adjustments

(Exposure,
Temperature,
Noise Reduction, …)



CIRAWFilter

CIImage

Adjusting RAW Images

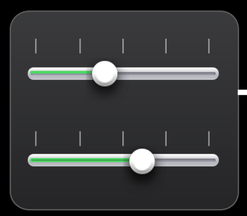
Architecture overview

RAW Image File

(File's URL or Data)

User Adjustments

(Exposure,
Temperature,
Noise Reduction, …)



CIRAWFilter

CIImage

Adjusting RAW Images

Architecture overview

RAW Image File

(File's URL or Data)

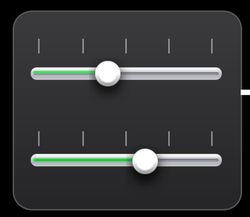
Custom CIFilter

CIRAWFilter

CIImage

User Adjustments

(Exposure,
Temperature,
Noise Reduction, …)



Adjusting RAW Images

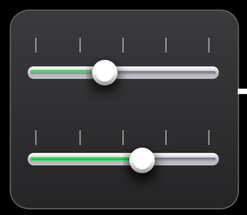
Architecture overview

RAW Image File

(File's URL or Data)

User Adjustments

(Exposure,
Temperature,
Noise Reduction, …)



Custom CIFilter

CIRAWFilter

CIImage

Adjusting RAW Images

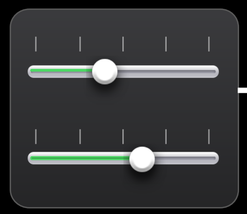
Architecture overview

RAW Image File

(File's URL or Data)

User Adjustments

(Exposure,
Temperature,
Noise Reduction, …)



Custom CIFilter

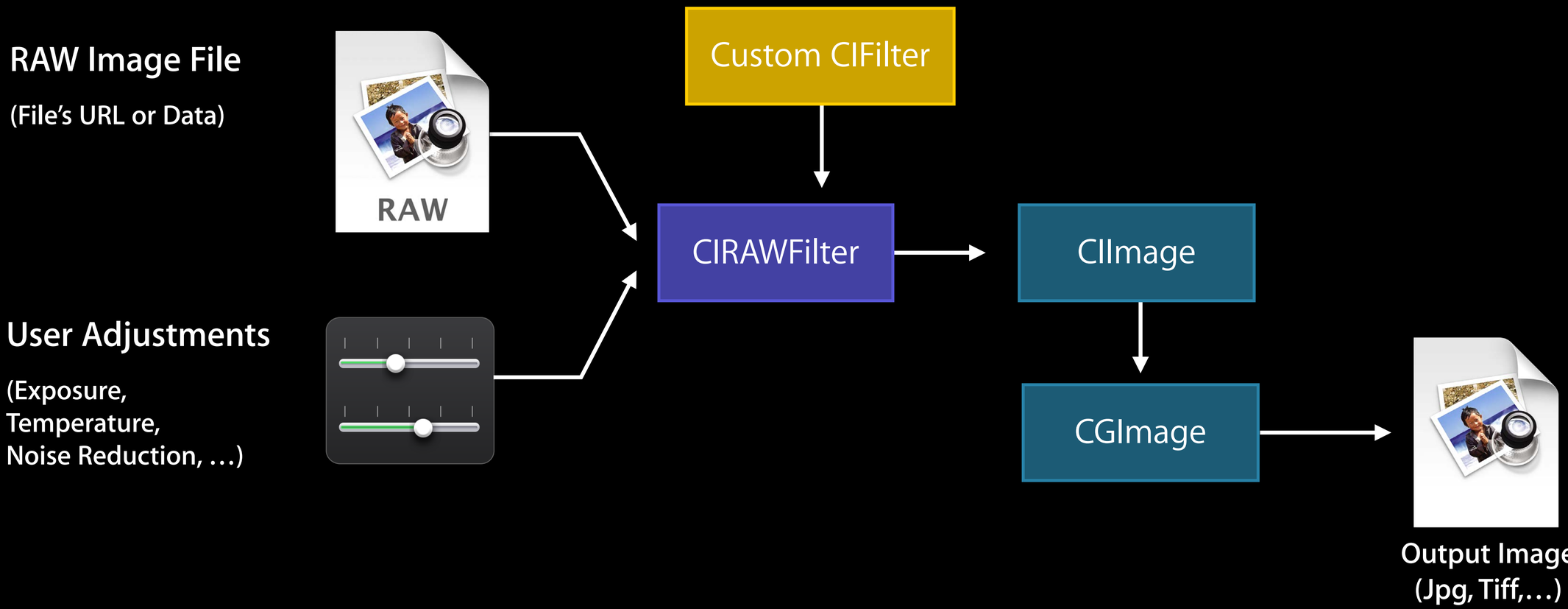
CIRAWFilter

CIImage

CGImage

Adjusting RAW Images

Architecture overview



Adjusting RAW Images

Using the CIRAWFilter

```
CIImage* GetAdjustedRaw (CFURLRef url)
{
    // Load the image
    CIFilter* f = [CIFilter filterWithImageURL:url options:nil];

    // Get the NR amount
    NSNumber* nr = [f valueForKey: kCIInputLuminanceNoiseReductionAmountKey];

    // Adjust the NR amount
    [f setValue: @(nr.doubleValue + 0.1)
         forKey: kCIInputLuminanceNoiseReductionAmountKey];

    // Get the adjusted image
    return f.outputImage
}
```

Demo

Adjusting RAW Images with CIRawFilterSample

Serhan Uslubas
RawCamera Engineer

Using the Second GPU on Mac Pro

Using the Second GPU

When does using the second GPU make sense?

- Speculative renders
- Background renders

Will not cause UI rendering on the display's GPU to stutter

Using the Second GPU

Creating the CIContext for the second GPU

In Mavericks

- It takes around 80 lines of OpenGL code

Now you just need this

- [CIContext offlineGPUAtIndex:0]

Demo

Batch Processing RAW files on a second GPU

Serhan Uslubas

Summary

Summary

Key concepts

Summary

Key concepts

What's new in Core Image in iOS 8

Summary

Key concepts

What's new in Core Image in iOS 8

What's new in Core Image in OS X Yosemite

Summary

Key concepts

What's new in Core Image in iOS 8

What's new in Core Image in OS X Yosemite

How to use CIDetectors

Summary

Key concepts

What's new in Core Image in iOS 8

What's new in Core Image in OS X Yosemite

How to use CIDetectors

How to adjust RAW image with Core Image

More Information

Allan Schaffer
Graphics and Game Technologies Evangelist
aschaffer@apple.com

Developer Technical Support
http://developer.apple.com/contact

Apple Developer Forums
http://devforums.apple.com

Related Sessions

• Camera Capture: Manual Controls	Marina	Wednesday 11:30AM
• Introducing the Photos Frameworks	Nob Hill	Thursday 10:15AM
• Developing Core Image Filters for iOS	Pacific Heights	Thursday 3:15PM

Labs

• Core Image Lab	Media Lab B	Thursday 4:30PM

WWDC14