# Using Core Image on iOS and Mac OS X

Session 422

**David Hayward**
Advanced Imaging Team

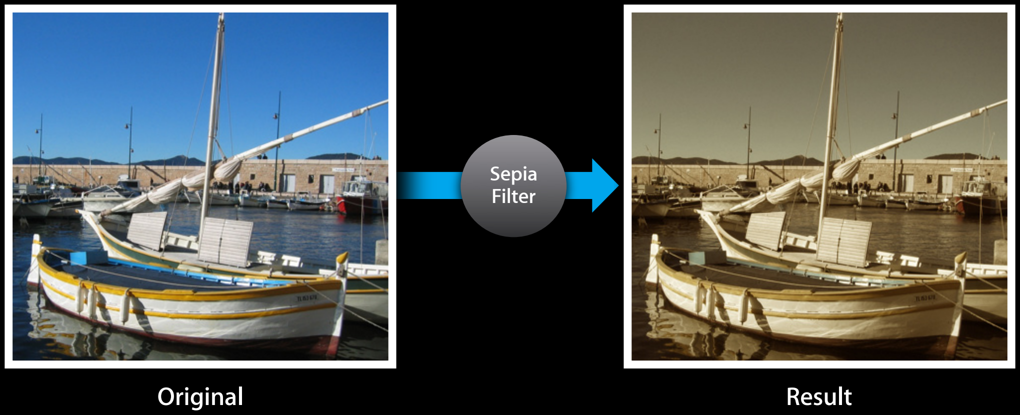
## What We Will Discuss Today

- Introducing Core Image on iOS 5
  - Key concepts
  - Basic architecture
  - Classes and API
  - Platform specifics
- Using Core Image in iOS
  - Initializing a CIImage
  - Filtering a CIImage
  - Rendering through a CIContext
- Image analysis

## Introducing Core Image in iOS 5

## Basic Concept

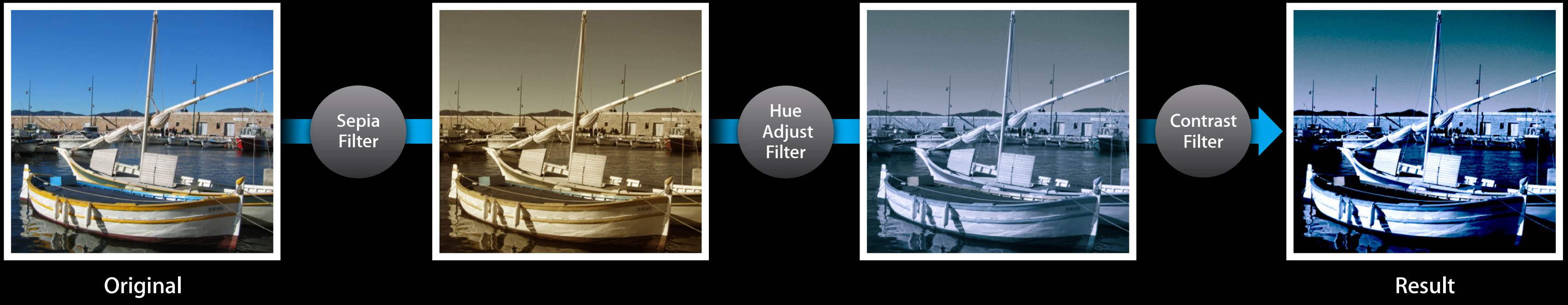
### Filters perform per pixel operations on an image



The final result is a new image

## Basic Concept

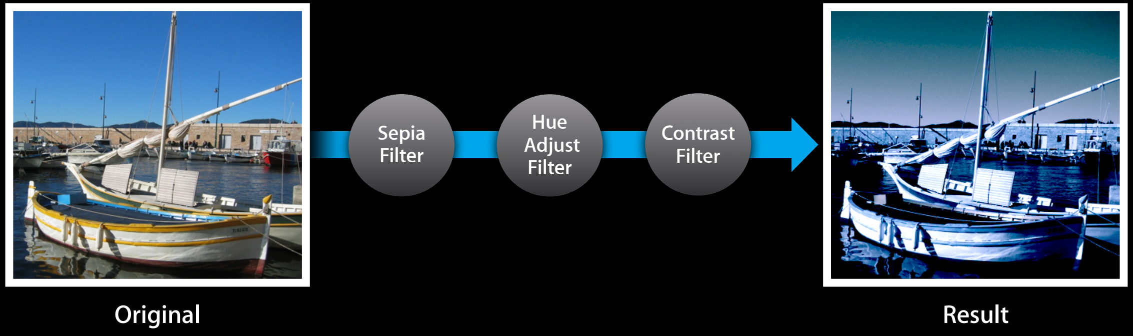
### Filters can be chained together



This allows for complex effects

## Basic Concept

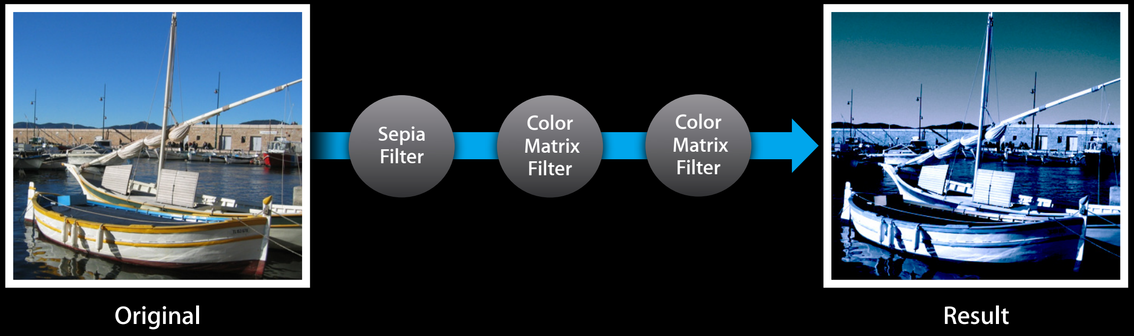
### Filters chains are concatenated



This eliminates intermediate buffers

## Basic Concept

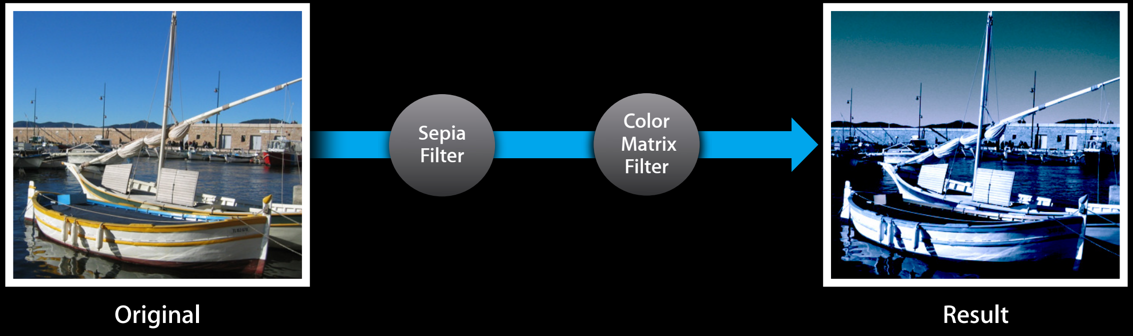
### Filters chains are optimized



This further improves performance

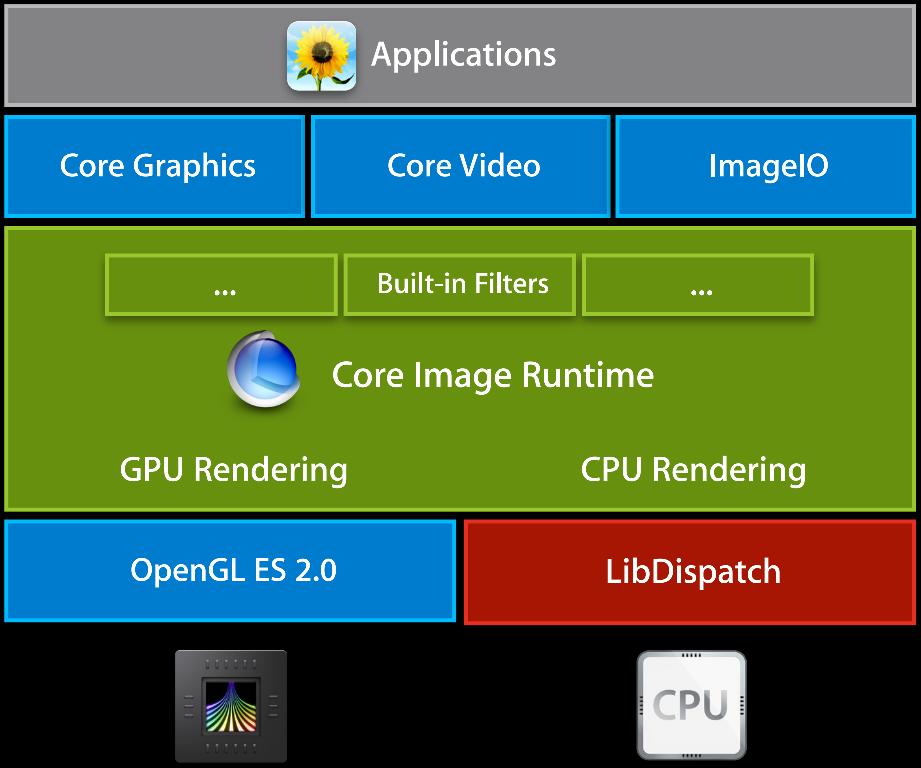
## Basic Concept

### Filters chains are optimized



This further improves performance

## Basic Architecture



## Core Image Classes

- CIFilter
  - A mutable object that represents an effect
  - Has image or numeric input parameters
  - Produces one output image based on current inputs
- CIImage
  - An immutable object that represents the recipe for an image
  - Can represent a file from disk or the output of a CIFilter
- CIContext
  - A object through which Core Image draw results
  - Can be based on on CPU or GPU

## CIContext CPU vs. GPU

- Both have their place

| CPU | GPU |
|---|---|
| **Fidelity** | Performance |
| **Background Friendly** | Offloads the CPU |

## Core Image Classes

1. Create a CIImage object

```
image = [CIImage imageWithContentsOfURL:myURL];
```

2. Create a CIFilter object

```
filter = [CIFilter filterWithName:@"CISepiaTone"];
[filter setValue:image forKey:kCIInputImageKey];
[filter setValue:[NSNumber numberWithFloat:0.8f] forKey:@”inputIntensity”];
```

3. Create a CIContext object

```
context = [CIContext contextWithOptions:nil];
```

4. Render the filter output image into a CGImage

```
result = [filter valueForKey:kCIOutputImageKey];
cgimage = [context createCGImage:result fromRect:[result extent]];
```

## Current Built-in Filters

- CIAffineTransform
- CIColorControls
- CIColorMatrix
- CIConstantColorGenerator
- CICrop
- CIExposureAdjust
- CIGammaAdjust
- CIHighlightShadowAdjust
- CIHueAdjust
- CISepiaTone
- CISourceOverCompositing
- CIStraightenFilter
- CITemperatureAndTint
- CIToneCurve
- CIVibrance
- CIWhitePointAdjust

## Platform Specifics

| | iOS | Mac OS |
|---|---|---|
| **Filters** | 16 Built-in Filters (emphasis on photo adjustment) | 130 Built-in Filters + Developer Extendable |
| **Core API** | CIFilter, CIImage, CIContext, | CIFilter, CIImage, CIContext, CIKernel, CIFilterShape |
| **Performance** | Render-time optimizations of filter graph | |
| **Rendering** | CPU or Open GL ES 2 | CPU or Open GL |

## Demo

Core Image in action

**Chendi Zhang**
Employee

## Using Core Image in iOS 5

## Using Core Image in iOS 5

### Initializing a CIImage

## Initializing a CIImage

### A CIImage can be initialized from:

- ImageIO supported formats

```
+imageWithURL:options:
+imageWithData:options:
```

- Other image types

```
+imageWithCGImage:options:

+imageWithCVPixelBuffer:options:
```

Only on iOS

```
+imageWithCVImageBuffer:options:
+imageWithIOSurface:options:
```

Only on Mac OS

- Raw pixel data

```
+imageWithBitmapData:bytesPerRow:size:format:colorSpace:
```

## Initializing a CIImage's Colorspace

- On Mac OS
  - A CIImage can be tagged with any colorspace
    - If tagged, pixels are converted to linear working space before filtering
- On iOS
  - A CIImage can be tagged with "Device RGB" colorspace
    - If tagged, pixels are gamma corrected to linear before filtering
- Use the `kCIImageColorSpace` option to override the default colorspace
  - Set value of this key to `[NSNull null]` to leave image unmanaged

## Initializing a CIImage's Metadata

Only on iOS

- New method to get metadata properties from an image

  `-(NSDictionary*) properties`

  - Contains same key/values as `CGImageSourceCopyPropertiesAtIndex`
  - One notable key is `kCGImagePropertyOrientation`
- Properties are automatic if you use `imageWithURL:` or `imageWithData:`
  - Otherwise properties can be specified using `kCIImageProperties` option

## Using Core Image in iOS 5

Filtering a CIImage

## Filter Application

- Query Core Image for the list of built-in filters

```
NSArray *list = [CIFilter filterNamesInCategory:kCICategoryBuiltIn];
```

- Filters are instantiated by name

```
CIFilter *filter = [CIFilter filterWithName:@”CISepiaTone”]
```

- Calling `[filter attributes]` will tell you about the filter’s inputs
  - The key for each input
  - The expected data type of each input
    - NSNumber, CIVector, CIImage, etc.
  - Common values of each input
    - Default, identity, minimum, and maximum

## Filter Application

- Inputs of a filter are set using key value conventions

```
[filter setValue:image forKey:kCIInputImageKey];
[filter setValue:[NSNumber numberWithFloat:0.8f] forKey:@”inputIntensity”];
```

- Output of a filter is obtained via the outputImage property

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

output = [filter outputImage];
output = filter.outputImage;
```

Only on iOS

- Shortcut

```
output = [CIFilter filterWithName:@”CISepiaTone” keysAndValues:
             kCIInputImageKey, image,
             @”inputIntensity”, [NSNumber numberWithFloat: 0.8f],
             nil].outputImage;
```

## Filter Application

### Chaining multiple filters

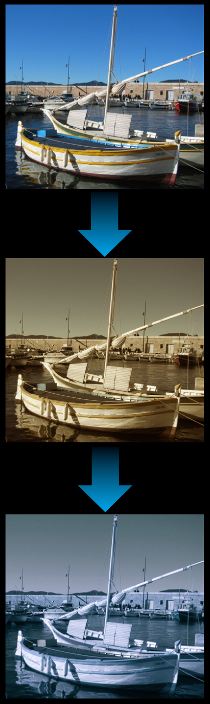
- Apply first filter to input image

```
output = [CIFilter filterWithName:@”CISepiaTone” keysAndValues:
          kCIInputImageKey, image,
          @”inputIntensity”, [NSNumber numberWithFloat: 0.8f],
          nil].outputImage;
```

- Apply next filter

```
output = [CIFilter filterWithName:@”CIHueAdjust” keysAndValues:
          kCIInputImageKey, output,
          @”inputAngle”, [NSNumber numberWithFloat: 0.8f],
          nil].outputImage;
```

- No pixel processing is performed while building the chain
  - That work is deferred until render is requested...



## Using Core Image in iOS 5

Rendering Core Image output

## Rendering Core Image Output

- Now that you have a filtered CIImage what can you do with it?
- We'll show how to render in these common use cases:
    - Displaying in a UIImageView
    - Save the result into the photo library
    - Displaying in a CAEAGLLayer-backed view
    - Passing results back to Core Video

## Rendering Core Image Output

### UIImageView

- Render the CIImage into a CGImageRef

```
CIContext *context = [CIContext context];
CIImage *ciimage = [filter outputImage];
CGImageRef cgimg = [context createCGImage:ciimage
                                 fromRect:[ciimage extent]];
view.image = [UIImage imageWithCGImage:cgimg
                      orientation:ui_orientation([ciimage properties]];
CGImageRelease(cgimg);
```

## Rendering Core Image Output

### UIImageView

- Convert the orientation property to UIImageOrientation

```
UIImageOrientation ui_orientation (NSDictionary* props)
{
    int o = [[props valueForKey:(id)kCGImagePropertyOrientation] intValue];
    UIImageOrientation map[] = {
        UIImageOrientationUp,
        UIImageOrientationUp, UIImageOrientationUpMirrored,
        UIImageOrientationDown, UIImageOrientationDownMirrored,
        UIImageOrientationLeftMirrored, UIImageOrientationRight,
        UIImageOrientationRightMirrored, UIImageOrientationLeft,
    };
    return map[o];
}
```

## Rendering Core Image Output

### Photo library

- Create a CPU context

```
CIContext *context = [CIContext contextWithOptions:
   [NSDictionary dictionaryWithObject:[NSNumber numberWithBool:YES]
                 forKey:kCIContextUseSoftwareRenderer]];
```

- Why a CPU context?
  - Will allow your app to do processing in the background
  - GPU contexts have texture size limits
  - CPU context supports larger input and output images

## Rendering Core Image Output

### Photo library

- Create a CGImage from the CIImage

```
CGImageRef cgimg = [cpu_context createCGImage:outputImage
                                     fromRect:[outputImage extent]];
```

- Add the CGImage to the photo library

```
ALAssetsLibrary *library = [ALAssetsLibrary new];
[library writeImageToSavedPhotosAlbum:cgimg
         metadata:[outputImage properties]
         completionBlock:^(NSURL *assetURL, NSError *error) {
             CGImageRelease(cgimg);
         }];
```

## Rendering Core Image Output

### CAEAGLLayer

- Render directly to the screen via a CAEAGLLayer
  - Avoids unnecessary CGImageRef intermediates buffers
  - Reduces GPU downloads/uploads
- Use the "OpenGL Game" Xcode template to setup a basic Open GL ES 2.0 project
- Create the CIContext using the same EAGLContext that contains the framebuffer and renderbuffer

```
EAGLContext *eagl_ctx = [[EAGLContext alloc]
                         initWithAPI:kEAGLRenderingAPIOpenGLES2];

CIContext *ci_ctx = [CIContext contextWithEAGLContext:eagl_ctx];
```

## Rendering Core Image Output

### CAEAGLLayer

- When you need to update the screen

```
- (void)updateScreen
{
    CIImage *image = [filter outputImage];
    [context drawImage:image atPoint:CGPointZero fromRect:[image extent]];
    glBindRenderbuffer(GL_RENDERBUFFER, render_buffer);
    [eaglContext presentRenderbuffer:GL_RENDERBUFFER];
}
```

## Demo

Rendering directly to screen

## Rendering Core Image Output

### Core Video

- Render to a CVPixelBufferRef

```
CIContext *context = [CIContext context];
CIImage *ciimage = [filter outputImage];
[context render:ciimage toCVPixelBuffer:pixelbuffer
         bounds:[ciimage extent] colorSpace:nil];
```

- Allows you to process video frames within a Core Video pipeline

## Demo

Using Core Image to filter and save live video

## Performance Best Practices

- CIImages and CIFilter are autoreleased
  - Use autorelease pools to reduce memory pressure
- When creating CVPixelBuffers use the attribute `kCVPixelBufferIOSurfacePropertiesKey`
- Don’t create a CIContext every time you render
- Core Animation and Core Image both can use the GPU
  - Avoid CA animations while rendering CIImages with a GPU context

## Performance Best Practices

- CPU and GPU CIContexts have limits on image sizes
  - Check the context limits by using:

```
– (CGSize) inputImageMaximumSize;
– (CGSize) outputImageMaximumSize;
```

Only on iOS

- Use subsampled images when possible
  - Performance generally scales linearly with the number of output pixels
  - You can use Core Graphics or ImageIO APIs to crop or down-sample

```
CGImageCreateWithImageInRect
CGImageSourceCreateThumbnailAtIndex
```

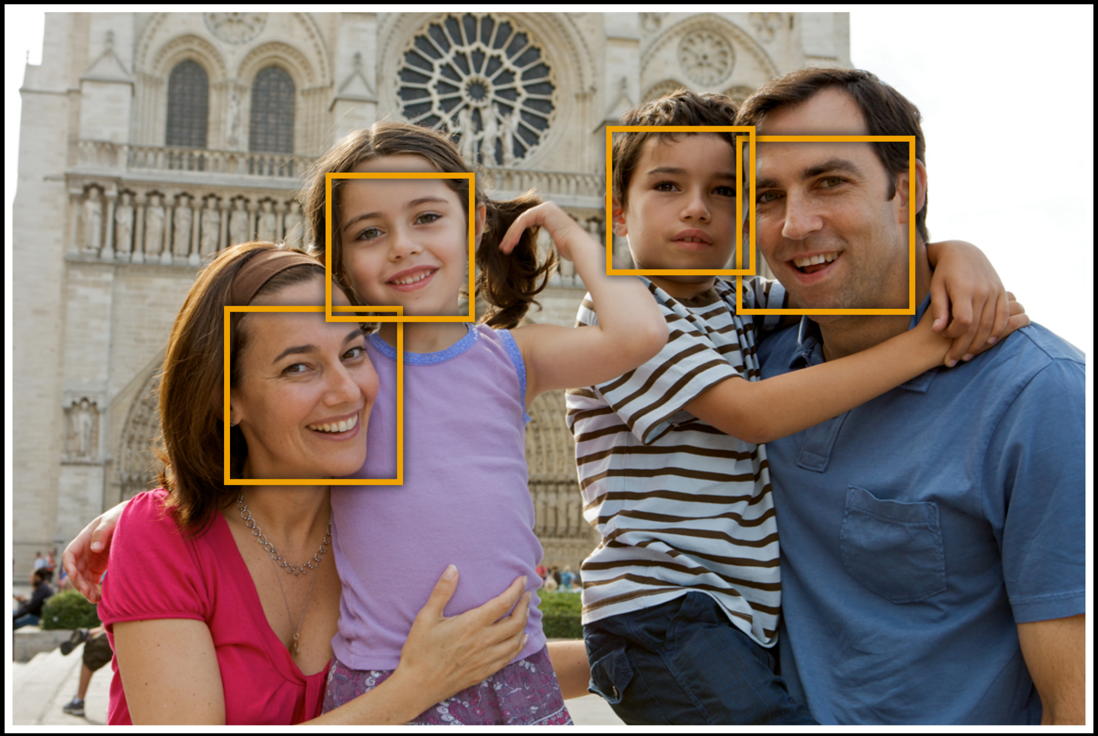
## Image Analysis

## Image Analysis

- Image Analysis goes beyond normal image filtering
  - Requires reading pixels from the source image
- Introducing two new Image Analysis tools for iOS
  - Face Detection
  - Auto Enhance

## Core Image Face Detection

## Face Detection



## Core Image Face Detection

- Same API on Lion and iOS 5
- Two classes
  - CIDetector
  - CIFeature
    - CIFaceFeature

## Core Image Face Detection

- Creating a detector

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

- Options: Tell the detector to be fast or thorough

```
opts = [NSDictionary dictionaryWithObject:CIDetectorAccuracyLow
                                   forKey:CIDetectorAccuracy];
opts = [NSDictionary dictionaryWithObject:CIDetectorAccuracyHigh
                                   forKey:CIDetectorAccuracy];
```

## Core Image Face Detection

- Finding features in an image

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

- Options: Tell the detector what direction is up

```
opts = [NSDictionary dictionaryWithObject:
      [[image properties] valueForKey:kCGImagePropertyOrientation]
      forKey:CIDetectorImageOrientation]];
```

## Core Image Face Detection

- Looking at the results

```
for (CIFaceFeature *f in features)
{
    NSLog(NSStringFromRect(f.bounds));

    if (f.hasLeftEyePosition)
        printf(“lEye %g %g\n”, f.leftEyePosition.x,
                               f.leftEyePosition.y);
    if (f.hasRightEyePosition)
        printf(“rEye %g %g\n”, f.rightEyePosition.x,
                               f.rightEyePosition.y);
    if (f.hasMouthPosition)
        printf(“rEye %g %g\n”, f.mouthPosition.x,
                               f.mouthPosition.y);
}
```

## Demo

### Face Detection

Piotr Maj
Software Engineer

## Core Image Auto Enhance

## Auto Enhance

- Analyzes an image for its
  - Histogram
  - Face region contents
  - Metadata properties
- Returns an array of CIFilters
  - Filter inputs parameters customized improve the specific image

## Auto Enhance

| Filter | Purpose |
|---|---|
| **CIRedEyeCorrection** | Repair red/amber/white eye from camera flash |
| **CIFaceBalance** | Adjust color of image to give pleasing skin tones |
| **CIVibrance** | Increase saturation without distorting skin tones |
| **CIToneCurve** | Adjust image contrast |
| **CIHighlightShadowAdjust** | Adjust shadow details |

## Auto Enhance

## Auto Enhance API

- A simple API

```
-(NSArray *)autoAdjustmentFiltersWithOptions:(NSDictionary*)options;
```

- Some important options
  - If you just want red eye correction set `kCIImageAutoAdjustEnhance` to false
  - If you just want other enhancements set `kCIImageAutoAdjustRedEye` to false
  - Use orientation to tell Core Image what's up

```
NSDictionary* opts = [NSDictionary dictionaryWithObject:
    [[image properties] valueForKey:kCGImagePropertyOrientation]
    forKey:CIDetectorImageOrientation]];
```

## Auto Enhance API

- Get the array of adjustment filters

```
NSArray *adjustments = [image autoAdjustmentFiltersWithOptions:options];
```

- Chain the filters together

```
for ( CIFilter *filter in adjustments ) {
   [filter setValue:image forKey:kCIImageInputImage];
   image = filter.outputImage;
}
```

- You can save the filter names and parameter values for later
  - Allows the enhancements to be done later without the cost of re-analyzing the image.
- Now ready to render

## Demo

Auto Enhance

## More Information

**Allan Schaffer**
Graphics and Imaging Evangelist
aschaffer@apple.com

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

## Related Sessions

| | |
|---|---|
| Capturing the Camera using AV Foundation on iOS 5 | Pacific Heights<br>Wednesday 4:30PM |

## Labs

| | |
|---|---|
| Core Image | Graphics & Media Lab A<br>Thursday 4:30PM |

WWDC2011