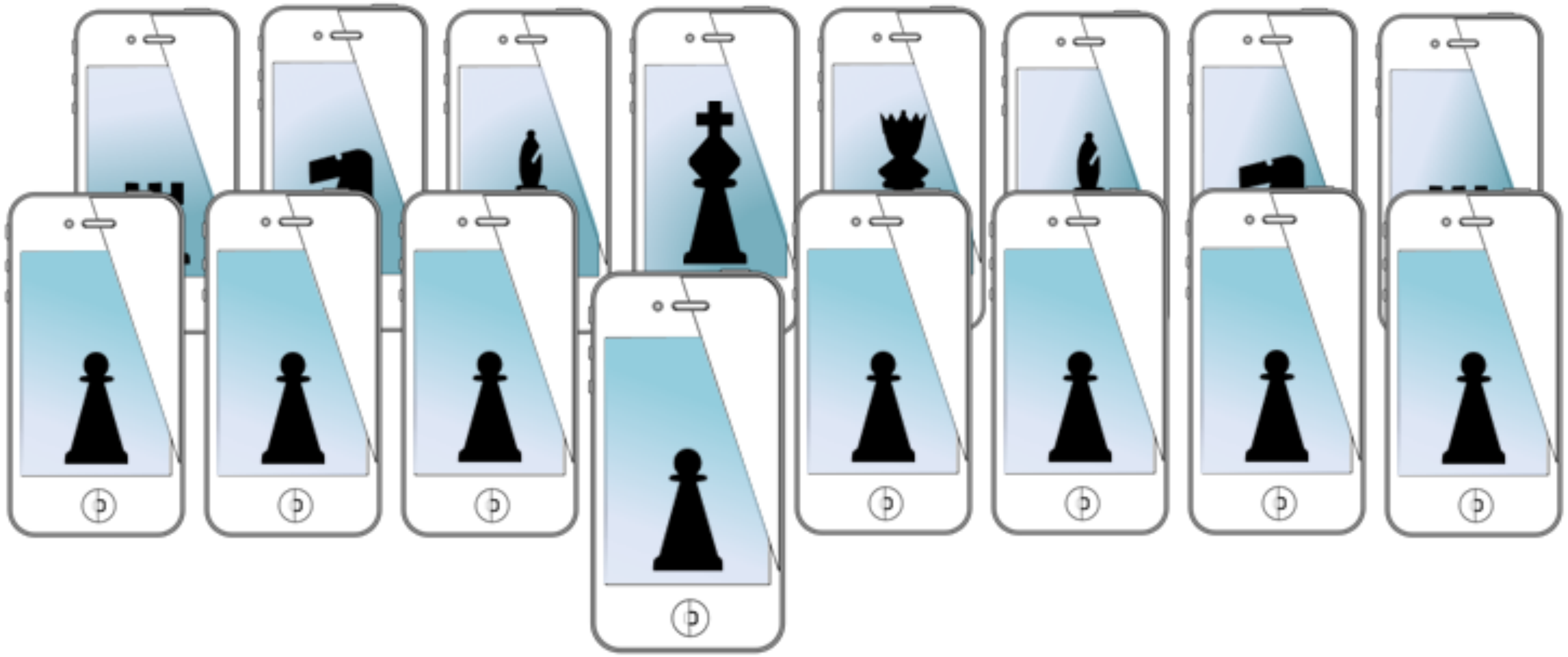
MOBILE SENSING LEARNING & CONTROL



CSE5323 & 7323

Mobile Sensing, Learning, and Control

lecture eleven: image processing and starting computer vision

Eric C. Larson, Lyle School of Engineering,
Computer Science and Engineering, Southern Methodist University

course logistics

- A2 grades will be up sometime this week
- A3 will be due Monday, March 3rd 6PM
    - see updated schedule

course logistics

- A2 grades will be up sometime this week
- A3 will be due Monday, March 3rd 6PM
  - see updated schedule

Create an iOS application using the MotionExample template that:

- Displays the number of steps a user has walked today and the previous day
- Displays a realtime count of the number of steps a users has taken today
- Displays the number of steps until the user reaches a (user settable) daily goal
- Displays the current activity of the user: {still, walking, running, in car}
- Uses {acceleration, gyro, fused motion} to distinguish when the user is climbing stairs
- Estimates the number of stairs climbed (coarse estimate)
  - this does NOT need to count stair steps in the background
- (extra credit, up to 0.5 points) add a feature that vibrates the motor and infers pressure of the grip based on the accelerometer and motor

The application should make use of the M7 co-processor whenever possible.

course logistics

- A2 grades will be up sometime this week
- A3 will be due Monday, March 3rd 6PM
    - see updated schedule

Create an iOS application using the MotionExample template that:

- Displays the number of steps a user has walked today and the previous day
- Displays a realtime count of the number of steps a users has taken today
- Displays the number of steps until the user reaches a (user settable) daily goal
- Displays the current activity of the user: {still, walking, running, in car}
- Uses {acceleration, gyro, fused motion} to distinguish when the user is climbing stairs
- Estimates the number of stairs climbed (coarse estimate)
    - this does NOT need to count stair steps in the background
- (extra credit, up to 0.5 points) add a feature that vibrates the motor and infers pressure of the grip based on the accelerometer and motor

The application should make use of the M7 co-processor whenever possible.

make the user interface engaging!

agenda

- core image
  - filtering
- video processing
- face detection

what is filtering?

- same as audio
  - convolution (linear)

.11	.11	.11
.11	.11	.11
.11	.11	.11

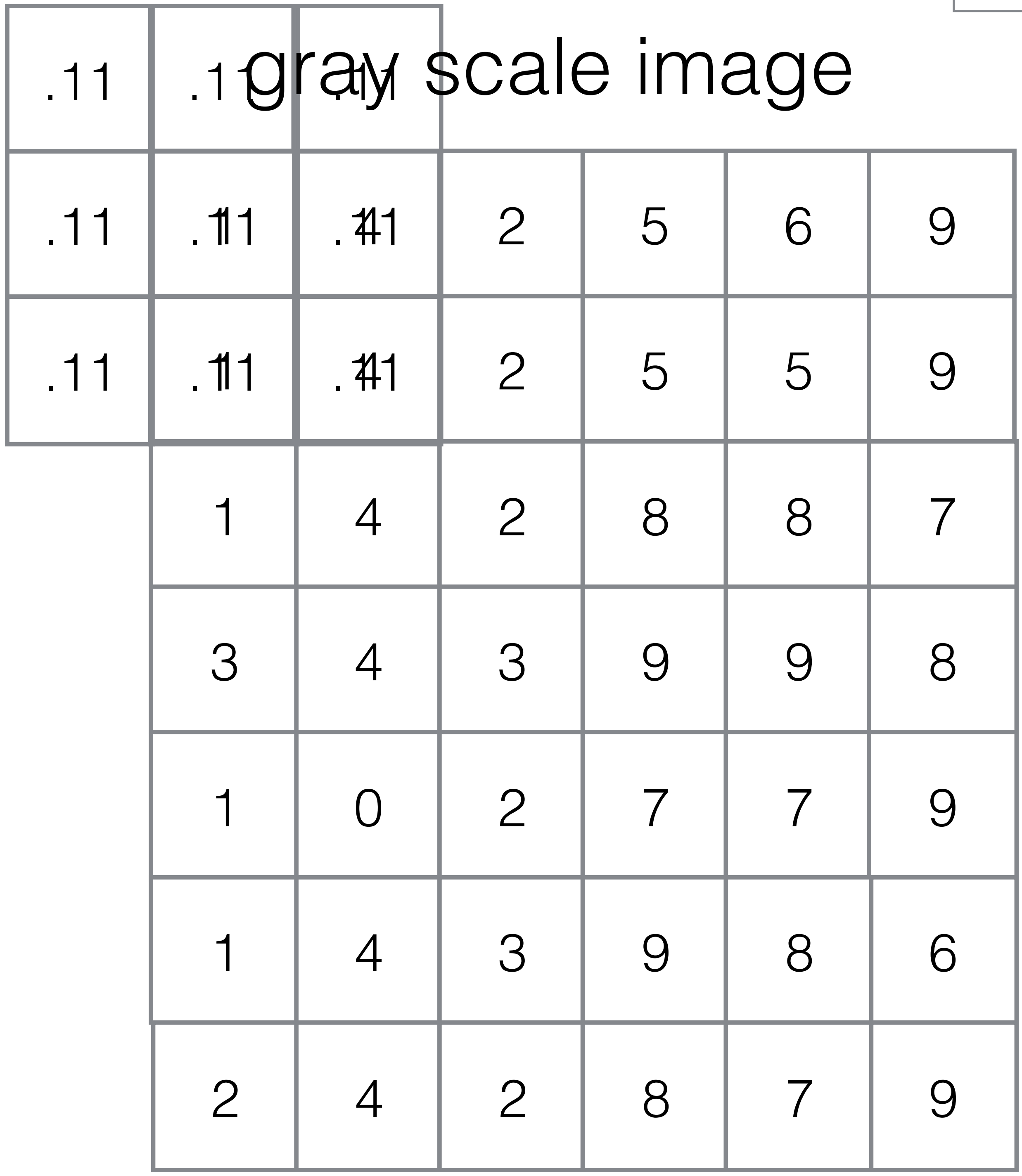
kernel

gray scale image

1	4	2	5	6	9
1	4	2	5	5	9
1	4	2	8	8	7
3	4	3	9	9	8
1	0	2	7	7	9
1	4	3	9	8	6
2	4	2	8	7	9

what is filtering?

- same as audio
  - convolution (linear)



.11	.11	.11
.11	.11	.11
.11	.11	.11

[illegible]	[illegible]	2	5	6	9
[illegible]	[illegible]	2	5	5	9
1	4	2	8	8	7
3	4	3	9	9	8
1	0	2	7	7	9
1	4	3	9	8	6
2	4	2	8	7	9

kernel

what is filtering?

- same as audio
  - convolution (linear)

gray scale image

1.1	1.5	2.4	2.7	4.3	3.2
1	4	2	5	5	9
1	4	2	8	8	7
3	4	3	9	9	8
1	0	2	7	7	9
1	4	3	9	8	6
2	4	2	8	7	9

kernel

.11	.11	.11
.11	.11	.11
.11	.11	.11

what is filtering?

- same as audio
  - convolution (linear)

gray scale image

1.1	1.5	2.4	2.7	4.3	3.2
1.6	2.3	4.0	4.7	6.8	4.8
1.8	2.6	4.5	5.6	7.5	5.1
1.4	2.2	4.3	6.1	8.0	5.2
1.4	2.3	4.5	6.3	8.0	5.2
1.3	2.1	4.3	5.8	7.7	5.1
1.2	1.7	3.3	4.1	5.2	3.3

kernel

.11	.11	.11
.11	.11	.11
.11	.11	.11

what is filtering?

-1	0	1
-1	0	1
-1	0	1

kernel

gray scale image

1	4	2	5	6	9
1	4	2	5	5	9
1	4	2	8	8	7
3	4	3	9	9	8
1	0	2	7	7	9
1	4	3	9	8	6
2	4	2	8	7	9

what is filtering?

-1	0	1
-1	0	1
-1	0	1

kernel

gray scale image

-8	-2	-2	-7	-8	4
-12	-3	-6	-13	-7	5
-8	-2	-10	-15	-2	2
-8	-2	-16	-17	0	3
-8	-3	-17	-16	2	4
-8	-3	-16	-15	0	1
-8	-2	-9	-10	2	2

what is filtering?

-	-	-
-	max	-
-	-	-

kernel

gray scale image

1	4	2	5	6	9
1	4	2	5	5	9
1	4	2	8	8	7
3	4	3	9	9	8
1	0	2	7	7	9
1	4	3	9	8	6
2	4	2	8	7	9

what is filtering?

-	-	-
-	max	-
-	-	-

kernel

gray scale image

4	4	5	6	9	9
4	4	8	8	9	9
4	4	8	8	8	8
4	4	9	9	9	9
4	4	7	9	9	9
4	4	9	9	9	9
4	4	9	9	9	9

what is filtering?

-	-	-
-	max	-
-	-	-

kernel

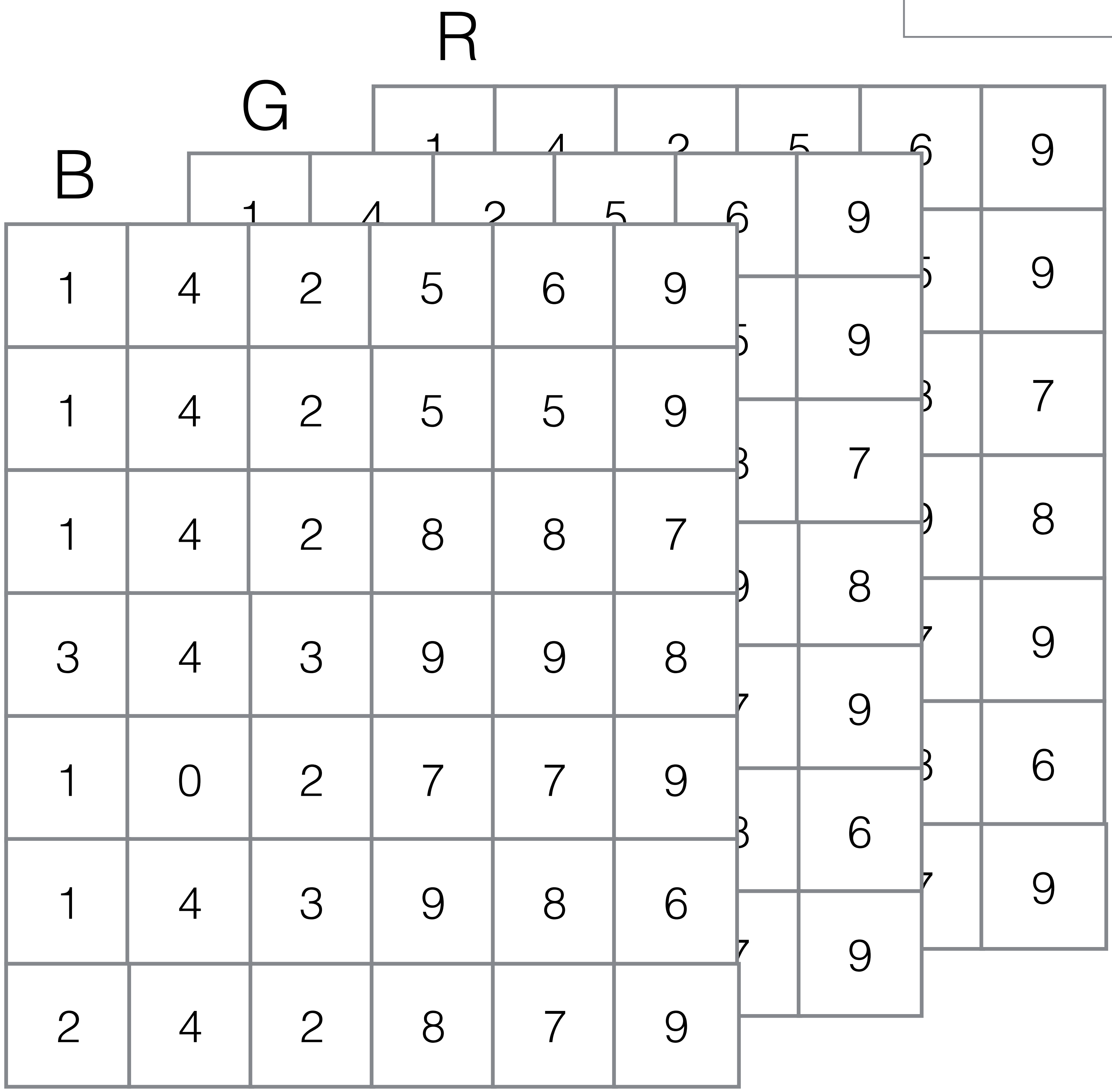
-	-	-
-	median	-
-	-	-

gray scale image

4	4	5	6	9	9
4	4	8	8	9	9
4	4	8	8	8	8
4	4	9	9	9	9
4	4	7	9	9	9
4	4	9	9	9	9
4	4	9	9	9	9

filtering with color

-	-	-
-	max	-
-	-	-



B

1	4	2	5	6	9
1	4	2	5	5	9
1	4	2	8	8	7
3	4	3	9	9	8
1	0	2	7	7	9
1	4	3	9	8	6
2	4	2	8	7	9

filtering video

filtering video

- back camera is capable of capturing
  - 8MP photos (~30 MB raw)
  - 1080p HD video at 30 fps

filtering video

- back camera is capable of capturing
  - 8MP photos (~30 MB raw)
  - 1080p HD video at 30 fps
- face camera
  - 1.2MP photos
  - 720p HD video at 30 fps

filtering video

- back camera is capable of capturing
  - 8MP photos (~30 MB raw)
  - 1080p HD video at 30 fps
- face camera
  - 1.2MP photos
  - 720p HD video at 30 fps
- video on the face camera is 1280x720 x3 channels x30fps
  - 82.9 million samples per second

so much data !!!

so much data !!!

- we need to hardware accelerate

so much data !!!

- we need to hardware accelerate
- look back to audio:
  - why is this:

```
float one = 1.0;
vDSP_vdbcon(fftMagnitudeBuffer,1,&one,fftMagnitudeBuffer,1,kBufferLength/2,0);
```

  - faster than this:

so much data !!!

- we need to hardware accelerate
- look back to audio:
    - why is this:

```
float one = 1.0;
vDSP_vdbcon(fftMagnitudeBuffer,1,&one,fftMagnitudeBuffer,1,kBufferLength/2,0);
```

- faster than this:

```
 for(int i=0;i<kBufferLength/2;i++){
            fftMagnitudeBuffer[i] = 20*logb(fftMagnitudeBuffer[i]);
     }
```

so much data !!!

- we need to hardware accelerate
- look back to audio:
  - why is this:

```
float one = 1.0;
vDSP_vdbcon(fftMagnitudeBuffer,1,&one,fftMagnitudeBuffer,1,kBufferLength/2,0);
```

  - faster than this:

```
for(int i=0;i<kBufferLength/2;i++){
        fftMagnitudeBuffer[i] = 20*logb(fftMagnitudeBuffer[i]);
    }
```

parallelized data processing

so much data !!!

- we need to hardware accelerate
- look back to audio:
    - why is this:

```
float one = 1.0;
vDSP_vdbcon(fftMagnitudeBuffer,1,&one,fftMagnitudeBuffer,1,kBufferLength/2,0);
```

- faster than this:

```
for(int i=0;i<kBufferLength/2;i++){
        fftMagnitudeBuffer[i] = 20*logb(fftMagnitudeBuffer[i]);
    }
```

parallelized data processing

images: GPU

core image framework

core image framework

- defines images as CIImage instances

core image framework

- defines images as CIImage instances
- defines a set of filters that (can be) GPU accelerated

core image framework

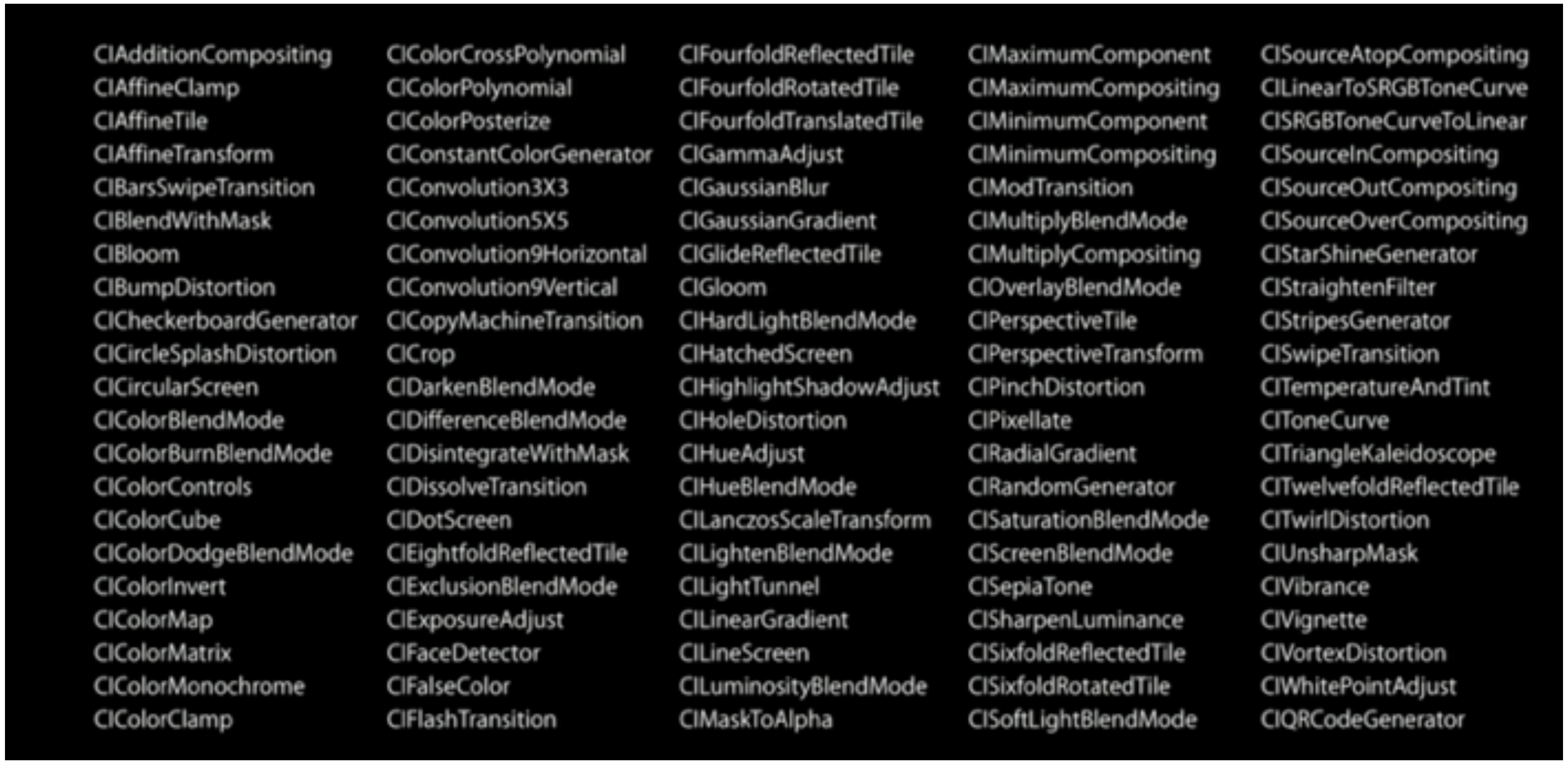
- defines images as CIImage instances
- defines a set of filters that (can be) GPU accelerated
- optimizes filters when cascaded

core image framework

- defines images as CIImage instances
- defines a set of filters that (can be) GPU accelerated
- optimizes filters when cascaded
- filters created through CIFilter class instances

core image framework

- defines images as CIImage instances
- defines a set of filters that (can be) GPU accelerated
- optimizes filters when cascaded
- filters created through CIFilter class instances



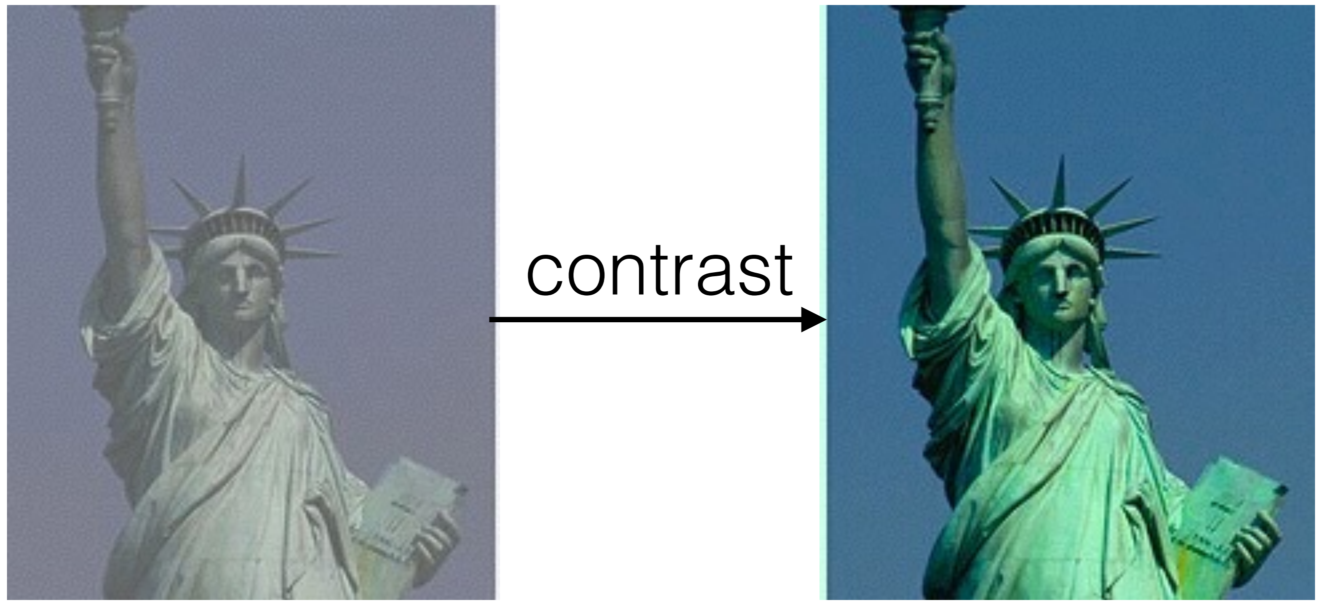
CIAdditionCompositing
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGenerator
CICircleSplashDistortion
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMode
CIColorControls
CIColorCube
CIColorDodgeBlendMode
CIColorInvert
CIColorMap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

core image framework

- nothing happens until the image is rendered!

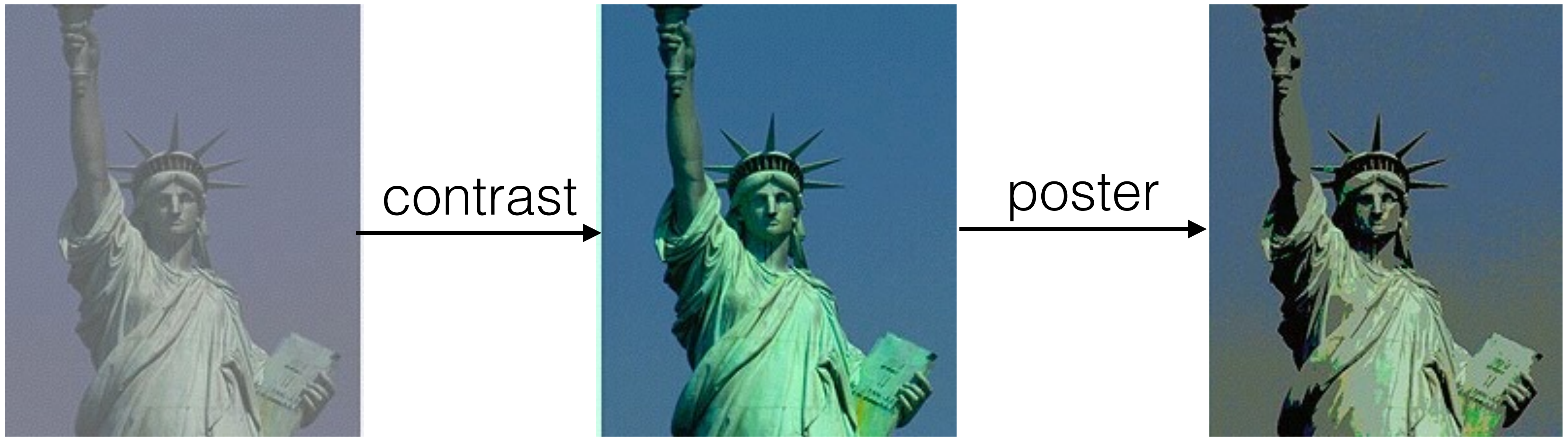
core image framework

- nothing happens until the image is rendered!



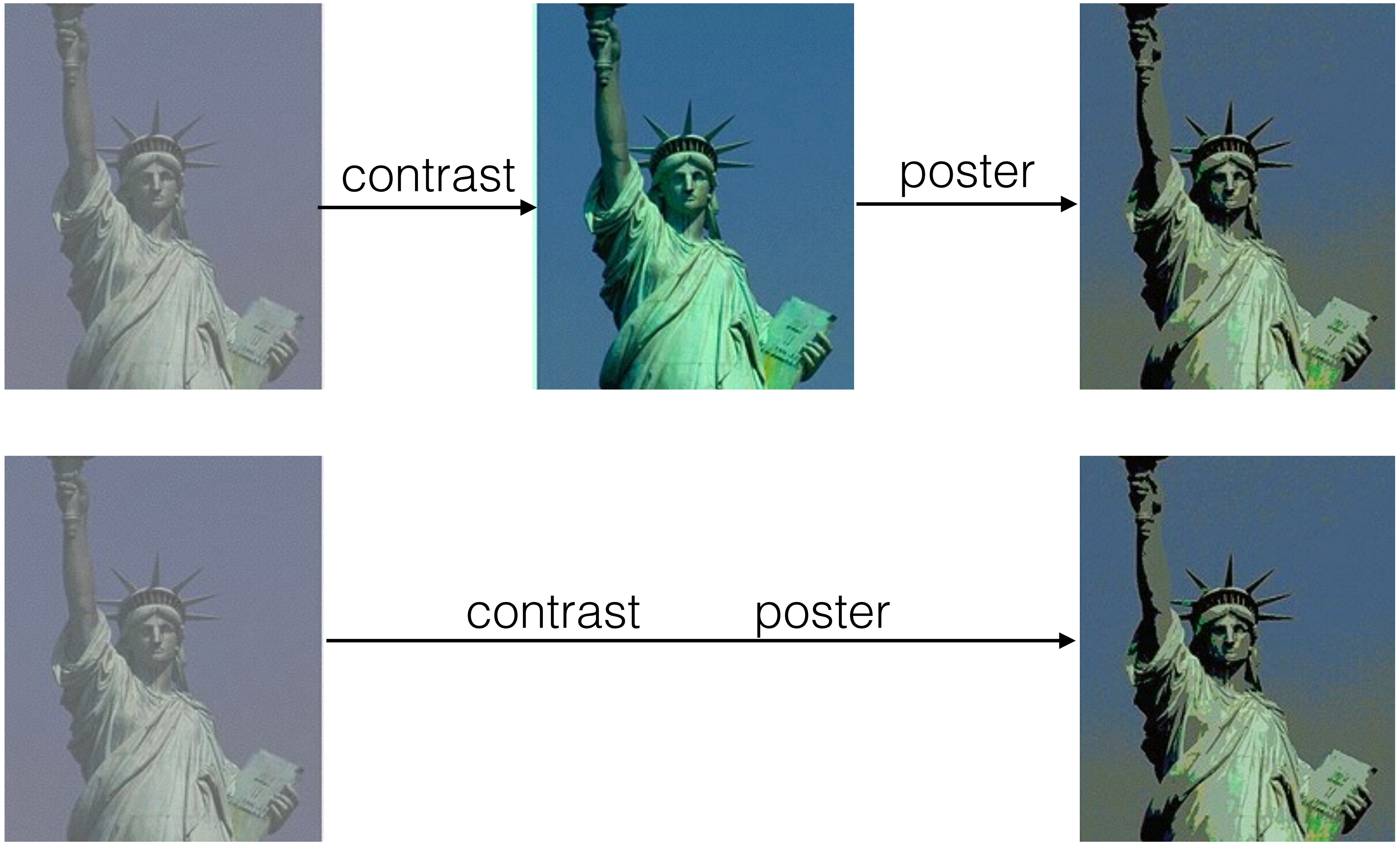
core image framework

- nothing happens until the image is rendered!



core image framework

- nothing happens until the image is rendered!



core image syntax

- Loading an image from the bundle
  - we need a CIImage instance, which stores more than just pixels

core image syntax

- Loading an image from the bundle
    - we need a CIImage instance, which stores more than just pixels

```
NSString *filePath =
   [[NSBundle mainBundle] pathForResource:@"image" ofType:@"png"];
NSURL *fileNameAndPath = [NSURL fileURLWithPath:filePath];

CIImage *myImage = [CIImage imageWithContentsOfURL:fileNameAndPath];
```

core image syntax

- Loading an image from the bundle
  - we need a CIImage instance, which stores more than just pixels

get image path from bundle

```
NSString *filePath =
   [[NSBundle mainBundle] pathForResource:@"image" ofType:@"png"];
NSURL *fileNameAndPath = [NSURL fileURLWithPath:filePath];

CIImage *myImage = [CIImage imageWithContentsOfURL:fileNameAndPath];
```

core image syntax

- Loading an image from the bundle
  - we need a CIImage instance, which stores more than just pixels

get image path from bundle

```
NSString *filePath =
   [[NSBundle mainBundle] pathForResource:@"image" ofType:@"png"];
NSURL *fileNameAndPath = [NSURL fileURLWithPath:filePath];

CIImage *myImage = [CIImage imageWithContentsOfURL:fileNameAndPath];
```

load image

core image syntax

- Loading an image from the bundle
  - we need a CIImage instance, which stores more than just pixels

get image path from bundle

```
NSString *filePath =
   [[NSBundle mainBundle] pathForResource:@"image" ofType:@"png"];
NSURL *fileNameAndPath = [NSURL fileURLWithPath:filePath];

CIImage *myImage = [CIImage imageWithContentsOfURL:fileNameAndPath];
```

load image

```
self.sourceImageView.image = [UIImage imageWithCIImage:myImage];
```

core image syntax

- Loading an image from the bundle
  - we need a CIImage instance, which stores more than just pixels

get image path from bundle

```
NSString *filePath =
   [[NSBundle mainBundle] pathForResource:@"image" ofType:@"png"];
NSURL *fileNameAndPath = [NSURL fileURLWithPath:filePath];

CIImage *myImage = [CIImage imageWithContentsOfURL:fileNameAndPath];
```

load image

```
self.sourceImageView.image = [UIImage imageWithCIImage:myImage];
```

show inside a UIImageView

core image syntax

- Loading an image from the bundle
  - we need a CIImage instance, which stores more than just pixels

get image path from bundle

```
NSString *filePath =
   [[NSBundle mainBundle] pathForResource:@"image" ofType:@"png"];
NSURL *fileNameAndPath = [NSURL fileURLWithPath:filePath];

CIImage *myImage = [CIImage imageWithContentsOfURL:fileNameAndPath];
```

load image

…processing here…

```
self.sourceImageView.image = [UIImage imageWithCIImage:myImage];
```

show inside a UIImageView

core image syntax

```
NSString *filePath =
   [[NSBundle mainBundle] pathForResource:@"image" ofType:@"png"];
NSURL *fileNameAndPath = [NSURL fileURLWithPath:filePath];

CIImage *myImage = [CIImage imageWithContentsOfURL:fileNameAndPath];
```

```
self.sourceImageView.image = [UIImage imageWithCIImage:myImage];
```

core image syntax

```
NSString *filePath =
   [[NSBundle mainBundle] pathForResource:@"image" ofType:@"png"];
NSURL *fileNameAndPath = [NSURL fileURLWithPath:filePath];

CIImage *myImage = [CIImage imageWithContentsOfURL:fileNameAndPath];
```

create filter

```
self.sourceImageView.image = [UIImage imageWithCIImage:myImage];
```

core image syntax

```
NSString *filePath =
   [[NSBundle mainBundle] pathForResource:@"image" ofType:@"png"];
NSURL *fileNameAndPath = [NSURL fileURLWithPath:filePath];

CIImage *myImage = [CIImage imageWithContentsOfURL:fileNameAndPath];
```

create filter

set parameters

```
self.sourceImageView.image = [UIImage imageWithCIImage:myImage];
```

core image syntax

```
NSString *filePath =
   [[NSBundle mainBundle] pathForResource:@"image" ofType:@"png"];
NSURL *fileNameAndPath = [NSURL fileURLWithPath:filePath];

CIImage *myImage = [CIImage imageWithContentsOfURL:fileNameAndPath];
```

create filter

set parameters

```
CIFilter *filter = [CIFilter filterWithName:@"CISepiaTone"
                keysAndValues:
                 kCIInputImageKey,  myImage,
                 @“inputIntensity", @0.8,
                 nil];
```

```
self.sourceImageView.image = [UIImage imageWithCIImage:myImage];
```

core image syntax

```
NSString *filePath =
   [[NSBundle mainBundle] pathForResource:@"image" ofType:@"png"];
NSURL *fileNameAndPath = [NSURL fileURLWithPath:filePath];

CIImage *myImage = [CIImage imageWithContentsOfURL:fileNameAndPath];
```

create filter

set parameters

```
CIFilter *filter = [CIFilter filterWithName:@"CISepiaTone"
                 keysAndValues:
                  kCIInputImageKey,  myImage,
                  @“inputIntensity", @0.8,
                  nil];
```

↑ filter type

```
self.sourceImageView.image = [UIImage imageWithCIImage:myImage];
```

core image syntax

```
NSString *filePath =
   [[NSBundle mainBundle] pathForResource:@"image" ofType:@"png"];
NSURL *fileNameAndPath = [NSURL fileURLWithPath:filePath];

CIImage *myImage = [CIImage imageWithContentsOfURL:fileNameAndPath];
```

create filter

set parameters

input image

thresholds

```
CIFilter *filter = [CIFilter filterWithName:@"CISepiaTone"
                 keysAndValues:
                  kCIInputImageKey,  myImage,
                  @“inputIntensity", @0.8,
                  nil];
```

filter type

```
self.sourceImageView.image = [UIImage imageWithCIImage:myImage];
```

core image syntax

```
NSString *filePath =
   [[NSBundle mainBundle] pathForResource:@"image" ofType:@"png"];
NSURL *fileNameAndPath = [NSURL fileURLWithPath:filePath];

CIImage *myImage = [CIImage imageWithContentsOfURL:fileNameAndPath];
```

create filter

set parameters

input image

thresholds

```
CIFilter *filter = [CIFilter filterWithName:@"CISepiaTone"
                    keysAndValues:
                     kCIInputImageKey,  myImage,
                     @“inputIntensity", @0.8,
                     nil];
```

filter type

get output

```
CIImage *outputImage = [filter outputImage];
CIImage *outputImage = filter.outputImage;
```

```
self.sourceImageView.image = [UIImage imageWithCIImage:myImage];
```

core image syntax

```
NSString *filePath =
   [[NSBundle mainBundle] pathForResource:@"image" ofType:@"png"];
NSURL *fileNameAndPath = [NSURL fileURLWithPath:filePath];

CIImage *myImage = [CIImage imageWithContentsOfURL:fileNameAndPath];
```

create filter

set parameters

input image

thresholds

```
   CIFilter *filter = [CIFilter filterWithName:@"CISepiaTone"
                            keysAndValues:
                             kCIInputImageKey,  myImage,
                             @“inputIntensity", @0.8,
                             nil];
```

filter type

get output

```
CIImage *outputImage = [filter outputImage];
CIImage *outputImage = filter.outputImage;
```

```
self.sourceImageView.image = [UIImage imageWithCIImage:outputImage];
```

core image syntax

```
NSString *filePath =
   [[NSBundle mainBundle] pathForResource:@"image" ofType:@"png"];
NSURL *fileNameAndPath = [NSURL fileURLWithPath:filePath];

CIImage *myImage = [CIImage imageWithContentsOfURL:fileNameAndPath];
```

create filter

set parameters

input image

thresholds

```
   CIFilter *filter = [CIFilter filterWithName:@"CISepiaTone"
                        keysAndValues:
                         kCIInputImageKey,  myImage,
                         @“inputIntensity", @0.8,
                         nil];
```

filter type

get output

```
 CIImage *outputImage = [filter outputImage];
 CIImage *outputImage = filter.outputImage;
```

processing

```
   self.sourceImageView.image = [UIImage imageWithCIImage:outputImage];
```

core image filters

- https://developer.apple.com/library/mac/documentation/graphicsimaging/reference/CoreImageFilterReference/Reference/reference.html
- names, parameters, examples, etc.
- best practice: create filters and reuse them
  - but **not** at the same time…

core image filters

CIBloom

Softens edges and applies a pleasant glow to an image.

Parameters

inputImage
A `CIImage` object whose display name is Image.

inputRadius
An `NSNumber` object whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 10.00

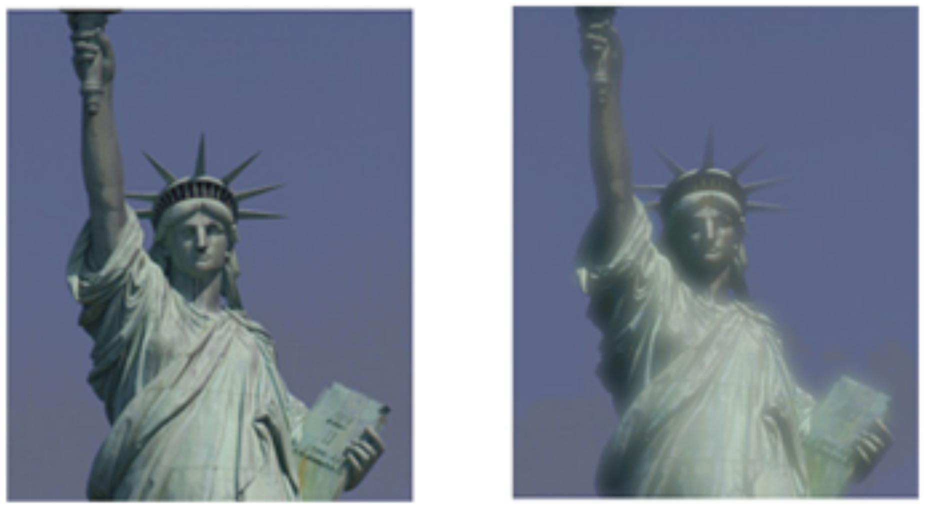
inputIntensity
An `NSNumber` object whose attribute type is `CIAttributeTypeScalar` and whose display name is Intensity.

Default value: 1.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryStylize`

Localized Display Name
Bloom

Figure 8 The result of using the CIBloom filter



Availability
Available in OS X v10.4 and later nd in iOS 6.0 and later. ← available?

core image filters

CIBloom

Softens edges and applies a pleasant glow to an image.

Parameters

inputImage

A CIImage object whose display name is Image.

inputRadius

An NSNumber object whose attribute type is CIAttributeTypeDistance and whose display name is Radius.

Default value: 10.00

inputIntensity

An NSNumber object whose attribute type is CIAttributeTypeScalar and whose display name is Intensity.

Default value: 1.00

Member of

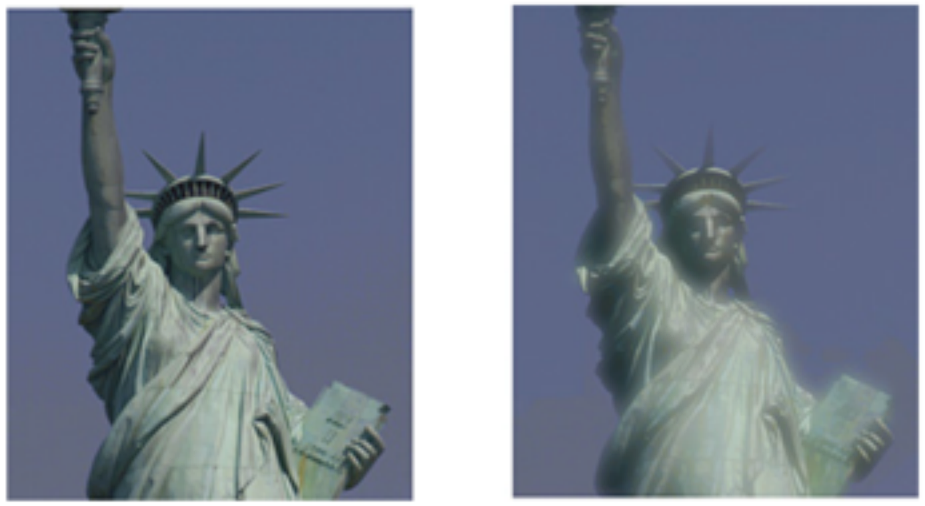
CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryStylize

Localized Display Name

Bloom

```
radius = 100;
CIFilter *filter =
[CIFilter filterWithName:@"CIBloom" keysAndValues:
             @"inputImage", myImage,
                  @“inputRadius", @(radius),
                  @"inputIntensity", @0.5,
                  nil];
```

Figure 8 The result of using the CIBloom filter



Availability

Available in OS X v10.4 and later and in iOS 6.0 and later. ← available?

core image filters

CIBloom

Softens edges and applies a pleasant glow to an image.

Parameters

inputImage
A `CIImage` object whose display name is Image.

inputRadius
An `NSNumber` object whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 10.00

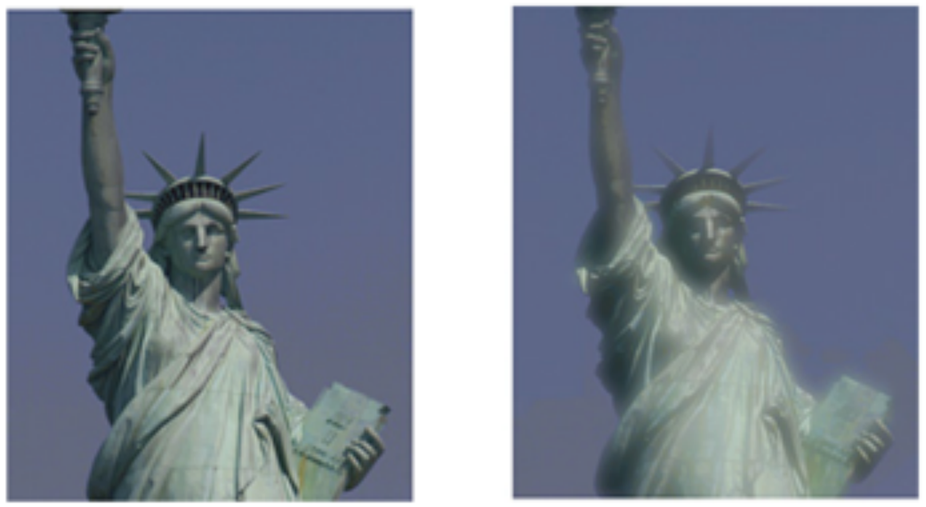
inputIntensity
An `NSNumber` object whose attribute type is `CIAttributeTypeScalar` and whose display name is Intensity.

Default value: 1.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryStylize`

Localized Display Name
Bloom

Figure 8 The result of using the CIBloom filter



Availability
Available in OS X v10.4 and later nd in iOS 6.0 and later. ← available?

```
radius = 100;
CIFilter *filter =
[CIFilter filterWithName:@"CIBloom" keysAndValues:
          @"inputImage", myImage,
              @“inputRadius", @(radius),
              @"inputIntensity", @0.5,
              nil];
```

```
  CIFilter *filter =
  [CIFilter filterWithName:@“CIBloom”];

  [filter setValue:myImage
             forKey:kCIInputImageKey];
```

core image demo

- PhotoInWonderland

custom filters

```
const CGFloat weights[] = { 1, 0, -1,
                            2, 0, -2,
                            1, 0, -1};
```

custom filters

```
const CGFloat weights[] = { 1, 0, -1,
                            2, 0, -2,
                            1, 0, -1};
```

```
result =
[CIFilter filterWithName:@"CIConvolution3X3" keysAndValues:
                         @"inputImage", inputImage,
                         @"inputWeights",
                               [CIVector vectorWithValues:weights count:9],
                         @"inputBias", @0.5,
                      nil].outputImage;
```

chaining filters

```
NSMutableArray *filters = [[NSMutableArray alloc]init];
  [filters addObject:[CIFilter filterWithName:@"CISepiaTone"]];
  [filters addObject:[CIFilter filterWithName:@"CIBloom"]];
  [filters addObject:[CIFilter filterWithName:@"CIColorInvert"]];
```

chaining filters

```
NSMutableArray *filters = [[NSMutableArray alloc]init];
  [filters addObject:[CIFilter filterWithName:@"CISepiaTone"]];
  [filters addObject:[CIFilter filterWithName:@"CIBloom"]];
  [filters addObject:[CIFilter filterWithName:@"CIColorInvert"]];

  outputImage = inputImage;
   for(CIFilter *filter in filters){
       [filter setValue:outputImage forKey:kCIInputImageKey];
       outputImage = filter.outputImage;
   }
```

camera access

camera access

- easy to get photos!

camera access

- easy to get photos!
- from library:
  - use the UIImagePickerControllerDelegate protocol

camera access

- easy to get photos!
- from library:
  - use the UIImagePickerControllerDelegate protocol
- from camera:
  - use the UIImagePickerControllerDelegate protocol
  - cameraUI.sourceType = UIImagePickerControllerSourceTypeCamera

camera access

- easy to get photos!
- from library:
  - use the UIImagePickerControllerDelegate protocol
- from camera:
  - use the UIImagePickerControllerDelegate protocol
  - cameraUI.sourceType = UIImagePickerControllerSourceTypeCamera
- https://developer.apple.com/library/ios/documentation/AudioVideo/Conceptual/CameraAndPhotoLib_TopicsForIOS/Introduction/Introduction.html#//apple_ref/doc/uid/TP40010405-SW1

access album

```
<UIImagePickerControllerDelegate, UINavigationControllerDelegate>


- (IBAction)pickImageFromAlbum:(id)sender {
    UIImagePickerController *myPicker =
            [[UIImagePickerController alloc] init];
    myPicker.delegate = self;
   [self presentViewController:myPicker animated:YES completion:nil];
}
```

access album

```
<UIImagePickerControllerDelegate, UINavigationControllerDelegate>


- (IBAction)pickImageFromAlbum:(id)sender {
    UIImagePickerController *myPicker =
            [[UIImagePickerController alloc] init];
    myPicker.delegate = self;
   [self presentViewController:myPicker animated:YES completion:nil];
}
```

this example uses a button

access album

```
<UIImagePickerControllerDelegate, UINavigationControllerDelegate>

- (IBAction)pickImageFromAlbum:(id)sender {
    UIImagePickerController *myPicker =
            [[UIImagePickerController alloc] init];
    myPicker.delegate = self;
   [self presentViewController:myPicker animated:YES completion:nil];
}
```

this example uses a button

allocate & set options

access album

```
<UIImagePickerControllerDelegate, UINavigationControllerDelegate>

- (IBAction)pickImageFromAlbum:(id)sender {
    UIImagePickerController *myPicker =
           [[UIImagePickerController alloc] init];
    myPicker.delegate = self;
   [self presentViewController:myPicker animated:YES completion:nil];
}
```

this example uses a button

allocate & set options

present modal view

access album

```
<UIImagePickerControllerDelegate, UINavigationControllerDelegate>

- (IBAction)pickImageFromAlbum:(id)sender {
    UIImagePickerController *myPicker =
            [[UIImagePickerController alloc] init];
    myPicker.delegate = self;
   [self presentViewController:myPicker animated:YES completion:nil];
}

- (void)imagePickerControllerDidCancel:
(UIImagePickerController *)picker {
    [self dismissViewControllerAnimated:YES completion:nil];
}
```

this example uses a button

allocate & set options

present modal view

interact with modal VC

access album

```
<UIImagePickerControllerDelegate, UINavigationControllerDelegate>

- (IBAction)pickImageFromAlbum:(id)sender {
    UIImagePickerController *myPicker =
            [[UIImagePickerController alloc] init];
    myPicker.delegate = self;
   [self presentViewController:myPicker animated:YES completion:nil];
}
```

this example uses a button

allocate & set options

present modal view

interact with modal VC

```
- (void)imagePickerControllerDidCancel:
(UIImagePickerController *)picker {
    [self dismissViewControllerAnimated:YES completion:nil];
}
```

get image from VC

```
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    [self dismissViewControllerAnimated:YES completion:nil];
    UIImage *pickedImage =
            [info objectForKey:UIImagePickerControllerOriginalImage];
}
```

take a photo

```
<UIImagePickerControllerDelegate, UINavigationControllerDelegate>

- (IBAction)pickImageFromAlbum:(id)sender {
    UIImagePickerController *myPicker =
            [[UIImagePickerController alloc] init];
    myPicker.delegate = self;


   [self presentViewController:myPicker animated:YES completion:nil];
}
```

```
- (void)imagePickerControllerDidCancel:
(UIImagePickerController *)picker {
    [self dismissViewControllerAnimated:YES completion:nil];
}
```

```
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    [self dismissViewControllerAnimated:YES completion:nil];
    UIImage *pickedImage =
            [info objectForKey:UIImagePickerControllerOriginalImage];
}
```

take a photo

```
<UIImagePickerControllerDelegate, UINavigationControllerDelegate>

- (IBAction)pickImageFromAlbum:(id)sender {
    UIImagePickerController *myPicker =
            [[UIImagePickerController alloc] init];
    myPicker.delegate = self;

    myPicker.sourceType = UIImagePickerControllerSourceTypeCamera;

   [self presentViewController:myPicker animated:YES completion:nil];
}
```

```
- (void)imagePickerControllerDidCancel:
(UIImagePickerController *)picker {
    [self dismissViewControllerAnimated:YES completion:nil];
}
```

```
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    [self dismissViewControllerAnimated:YES completion:nil];
    UIImage *pickedImage =
            [info objectForKey:UIImagePickerControllerOriginalImage];
}
```

raw camera access

raw camera access

- want to access incoming video in real time
  - and display to screen!

raw camera access

- want to access incoming video in real time
  - and display to screen!
- that is a lot of processing

raw camera access

- want to access incoming video in real time
    - and display to screen!
- that is a lot of processing
- setup needs to occur in conjunction with GPU
    - setup proper context (i.e., OpenGL)
    - renderers, processing pipeline

raw camera access

- want to access incoming video in real time
  - and display to screen!
- that is a lot of processing
- setup needs to occur in conjunction with GPU
  - setup proper context (i.e., OpenGL)
  - renderers, processing pipeline
- you need to understand this
  - but you won't write the code to do it

AVCaptureSession

AVCaptureSession

- mediates all access to incoming video and screen

AVCaptureSession

- mediates all access to incoming video and screen
- the screen output and camera input need to speak the same language
    - same color representation (BGRA vs ARGB)
    - and transforms (mirroring, rotation)
    - important: same rendering context (for speed)

AVCaptureSession

- mediates all access to incoming video and screen
- the screen output and camera input need to speak the same language
  - same color representation (BGRA vs ARGB)
  - and transforms (mirroring, rotation)
  - important: same rendering context (for speed)
- capture session is optimized for video chat
  - so audio can also be captured here (not unlike Novocaine)

AVCaptureSession

- setup the capture
  - device: front or back camera

```
AVCaptureDevicePositionFront
AVCaptureDevicePositionBack
```

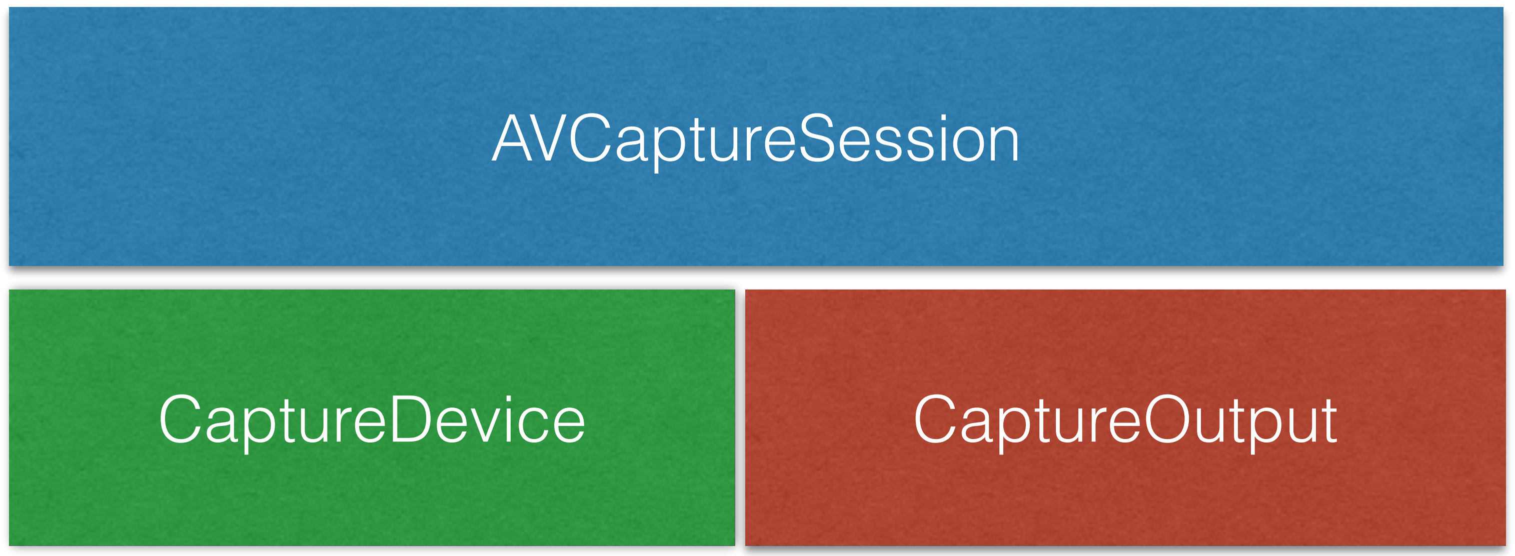
  - quality preset:

```
NSString *const AVCaptureSessionPresetPhoto;
NSString *const AVCaptureSessionPresetHigh;
NSString *const AVCaptureSessionPresetMedium;
NSString *const AVCaptureSessionPresetLow;
NSString *const AVCaptureSessionPreset352x288;
NSString *const AVCaptureSessionPreset640x480;
NSString *const AVCaptureSessionPreset1280x720;
NSString *const AVCaptureSessionPreset1920x1080;
NSString *const AVCaptureSessionPresetiFrame960x540;
NSString *const AVCaptureSessionPresetiFrame1280x720;
```

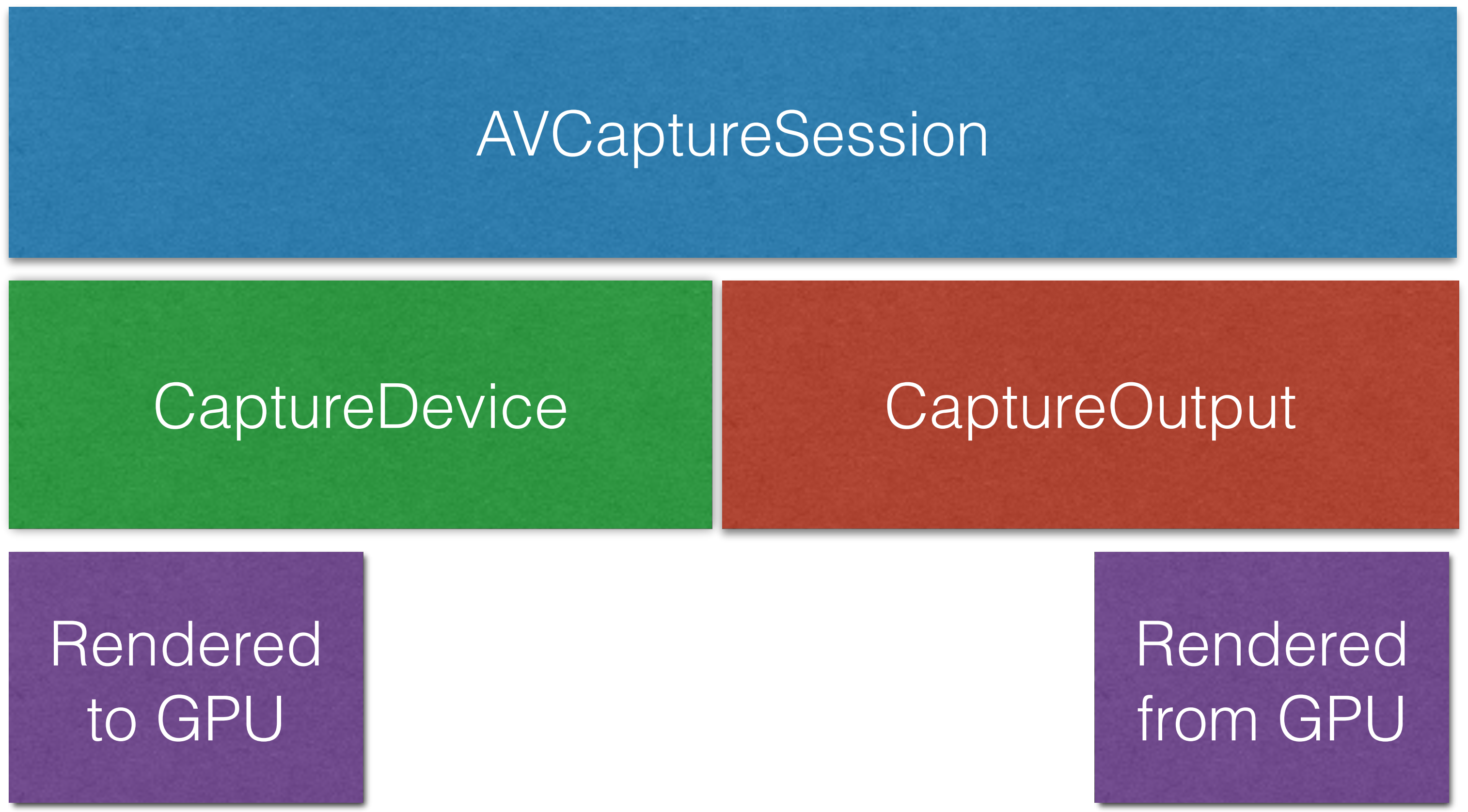
conceptual architecture

AVCaptureSession

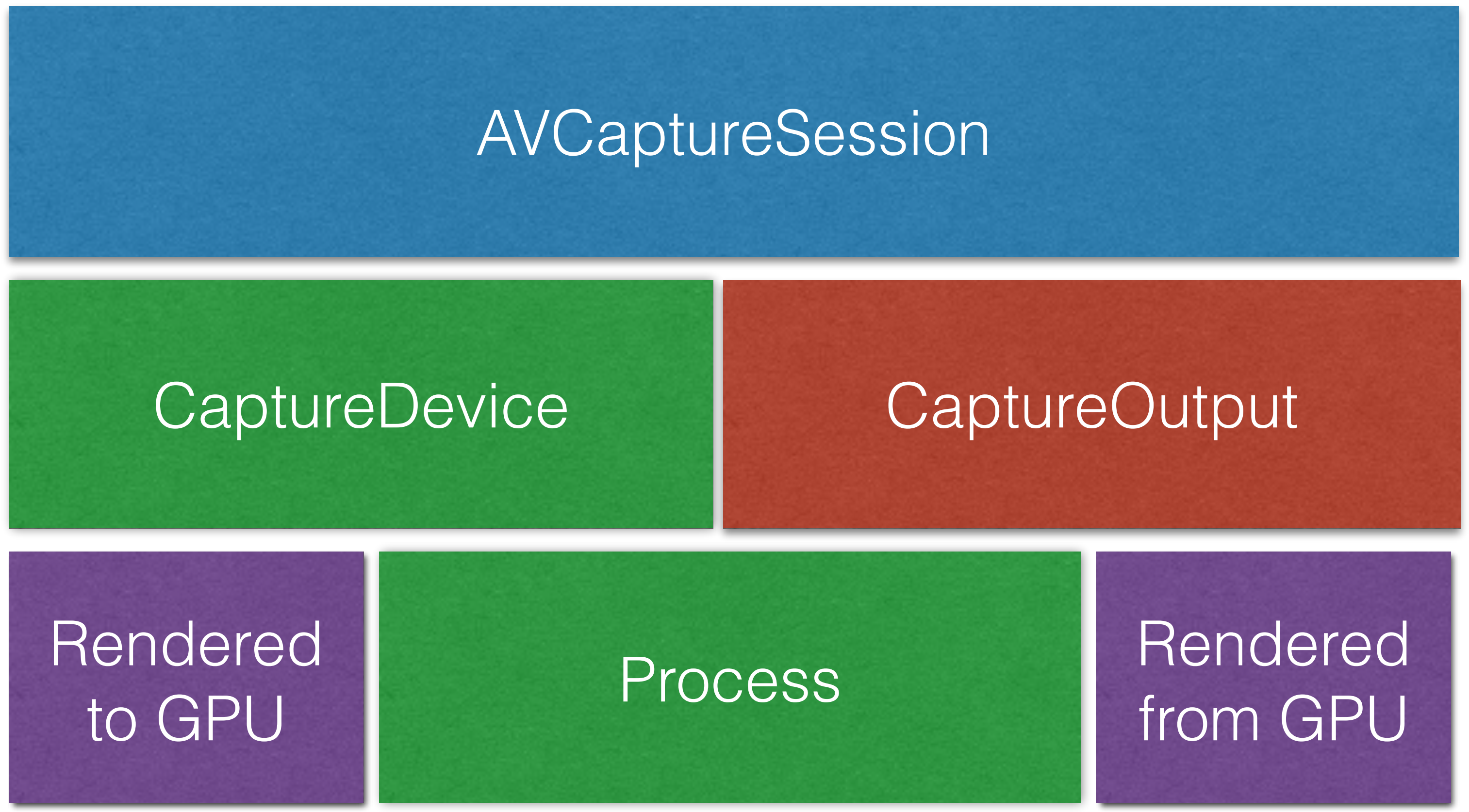
conceptual architecture



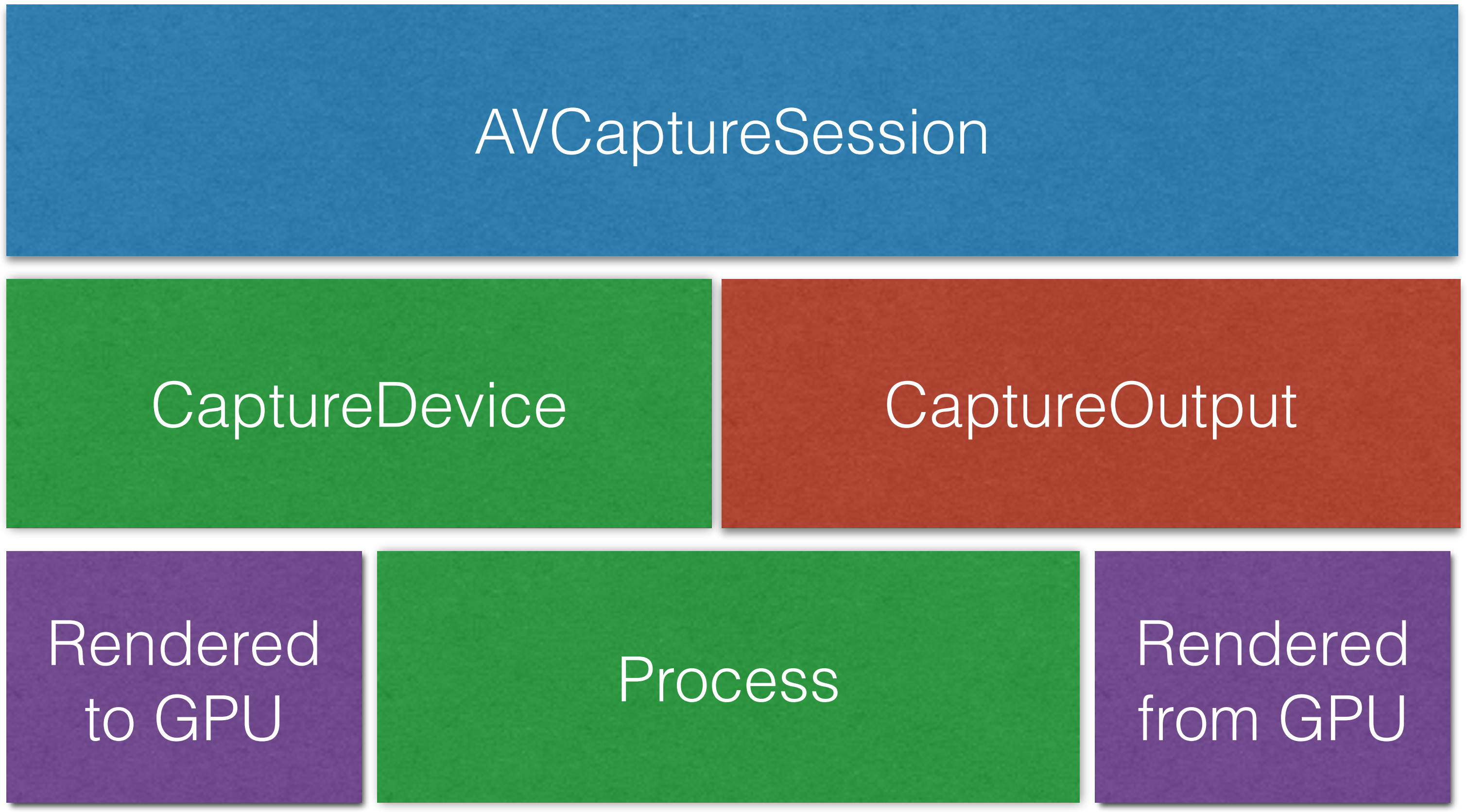
conceptual architecture



conceptual architecture



conceptual architecture



use core image, with processing setup for GPU
no data transfer from the GPU!

how to program this?

how to program this?

- what did we do with audio?
    - don’t reinvent the wheel: use Novocaine

how to program this?

- what did we do with audio?
  - don’t reinvent the wheel: use Novocaine
- now: use VideoAnalgesic
  - takes the pain out of GPU capture, render, and processing

how to program this?

- what did we do with audio?
    - don’t reinvent the wheel: use Novocaine
- now: use VideoAnalgesic
    - takes the pain out of GPU capture, render, and processing

declare

```
#import "VideoAnalgesic.h"

@property (strong,nonatomic) VideoAnalgesic *videoAnalgesic;
```

how to program this?

- what did we do with audio?
  - don’t reinvent the wheel: use Novocaine
- now: use VideoAnalgesic
  - takes the pain out of GPU capture, render, and processing

declare

```
#import "VideoAnalgesic.h"

@property (strong,nonatomic) VideoAnalgesic *videoAnalgesic;
```

init

```
self.captureManager = [VideoAnalgesic captureManager];
```

how to program this?

- what did we do with audio?
  - don’t reinvent the wheel: use Novocaine
- now: use VideoAnalgesic
  - takes the pain out of GPU capture, render, and processing

declare

```
#import "VideoAnalgesic.h"

@property (strong,nonatomic) VideoAnalgesic *videoAnalgesic;
```

init

```
self.captureManager = [VideoAnalgesic captureManager];
```

start

```
if(![self.captureManager isRunning])
     [self.captureManager start];
```

how to program this?

- what did we do with audio?
  - don’t reinvent the wheel: use Novocaine
- now: use VideoAnalgesic
  - takes the pain out of GPU capture, render, and processing

declare

```
#import "VideoAnalgesic.h"

@property (strong,nonatomic) VideoAnalgesic *videoAnalgesic;
```

init

```
self.captureManager = [VideoAnalgesic captureManager];
```

start

```
if(![self.captureManager isRunning])
     [self.captureManager start];
```

options

```
self.captureManager.preset = AVCaptureSessionPresetMedium;
[self.captureManager setCameraPosition:AVCaptureDevicePositionFront];
```

how to program this?

- what did we do with audio?
  - don't reinvent the wheel: use Novocaine
- now: use VideoAnalgesic
  - takes the pain out of GPU capture, render, and processing

declare

```
#import "VideoAnalgesic.h"

@property (strong,nonatomic) VideoAnalgesic *videoAnalgesic;
```

init

```
self.captureManager = [VideoAnalgesic captureManager];
```

start

```
if(![self.captureManager isRunning])
     [self.captureManager start];
```

options

```
self.captureManager.preset = AVCaptureSessionPresetMedium;
[self.captureManager setCameraPosition:AVCaptureDevicePositionFront];
```

stop

```
[self.captureManager stop];
```

VideoAnalgesic

- processing: similar to Novocaine
  - assumed that the output is always the screen of phone
  - use blocks and return image to draw to screen

VideoAnalgesic

- processing: similar to Novocaine
  - assumed that the output is always the screen of phone
  - use blocks and return image to draw to screen

```
[self.captureManager setProcessBlock:^(CIImage *cameraImage){
      return cameraImage;
 }];
```

VideoAnalgesic

- processing: similar to Novocaine
  - assumed that the output is always the screen of phone
  - use blocks and return image to draw to screen

```
[self.captureManager setProcessBlock:^(CIImage *cameraImage){
     return cameraImage;
 }];
```

image from camera

VideoAnalgesic

- processing: similar to Novocaine
    - assumed that the output is always the screen of phone
    - use blocks and return image to draw to screen

```
[self.captureManager setProcessBlock:^(CIImage *cameraImage){
        return cameraImage;
 }];
```

image to draw to screen

image from camera

VideoAnalgesic

- processing: similar to Novocaine
  - assumed that the output is always the screen of phone
  - use blocks and return image to draw to screen

```
[self.captureManager setProcessBlock:^(CIImage *cameraImage){
     return cameraImage;
 }];
```

image to draw to screen

image from camera

```
 __block CIFilter *filter = [CIFilter filterWithName:@"CISepiaTone"];
[self.captureManager setProcessBlock:^(CIImage *cameraImage){

    [filter setValue:cameraImage forKey:kCIInputImageKey];
    CIImage* output = filter.outputImage;
    return output;

}];
```

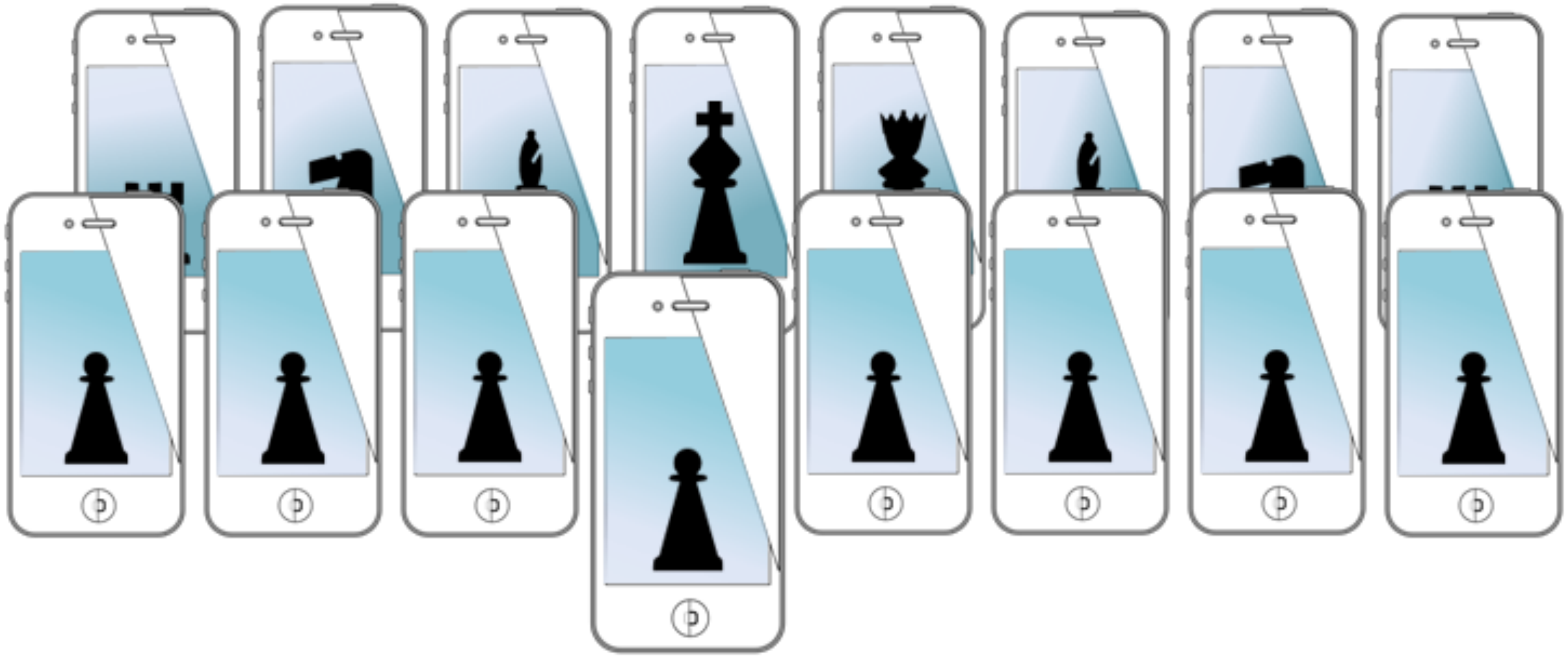
video process demo

- LookingLive!

for next time…

- computer vision with OpenCV
  - generic operations
  - tracking

MOBILE SENSING LEARNING & CONTROL



CSE5323 & 7323

Mobile Sensing, Learning, and Control

lecture eleven: image processing and starting computer vision

Eric C. Larson, Lyle School of Engineering,
Computer Science and Engineering, Southern Methodist University