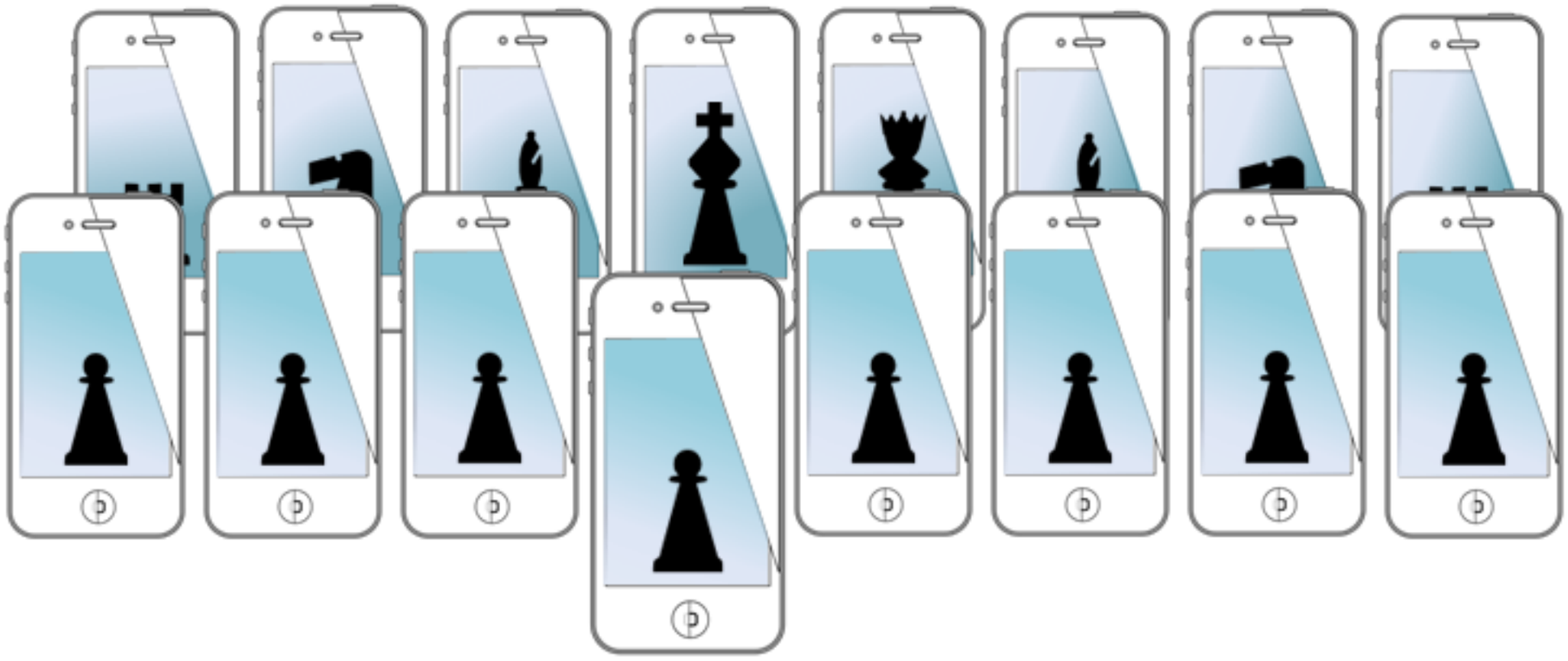
MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

core image and image processing

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

course logistics/agenda

- last time:
  - motion data (CoreMotion) and games (SceneKit, SpriteKit)
- logistics:
  - A3 due soon!
- today:
  - A3 lab explanation
  - image processing basics
  - core image filtering

storing persistent defaults

review

- iOS supports UserDefaults for primitives and encapsulated data (or lists of)

```
// standardUserDefaults variable
let defaults = UserDefaults.standardUserDefaults()

// saving
defaults.setInteger(252, forKey:@“primitiveInteger")
defaults.setDouble(3.14, forKey:@“primitiveDouble")
defaults.setFloat
defaults.setBool
defaults.setURL

// saving an object
defaults.setObject("Coding Explorer", forKey: “userNameKey")

if let name = defaults.stringForKey("userNameKey") {
    print(name)
}
```

import defaults

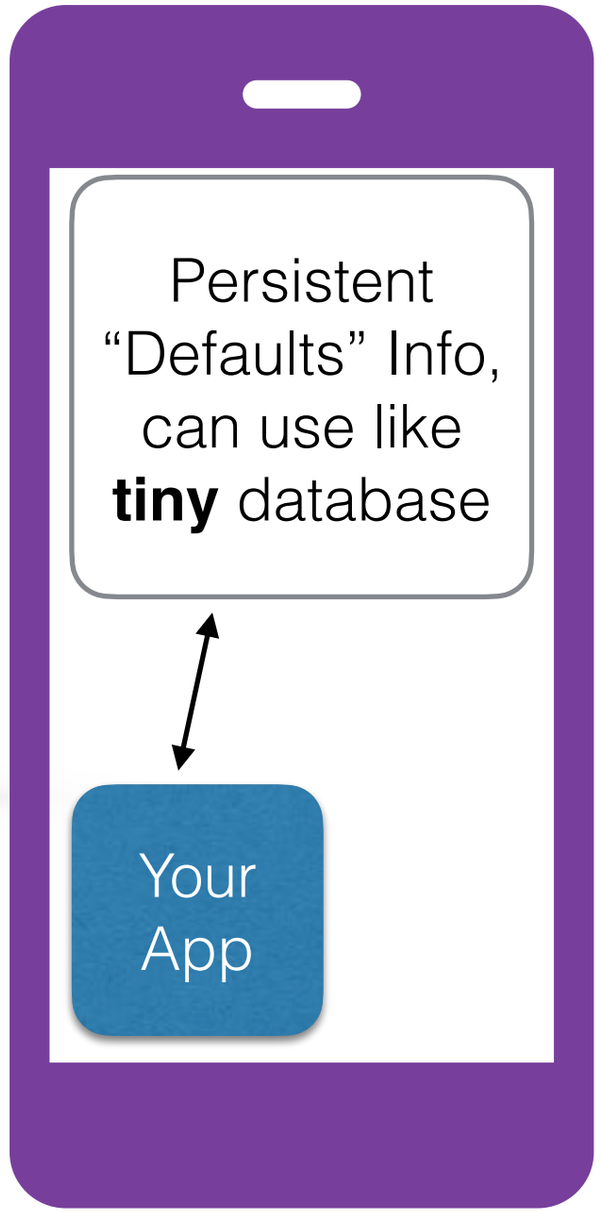
primitives

objects

access saved objects

```
boolForKey          ->  Bool
integerForKey       ->  Int
dataForKey          ->  Data?
objectForKey        ->  AnyObject?
arrayForKey         ->  [AnyObject]?
stringArrayForKey   ->  [String]?
dictionaryForKey    ->  {String:AnyObject}?
```

supported datatypes



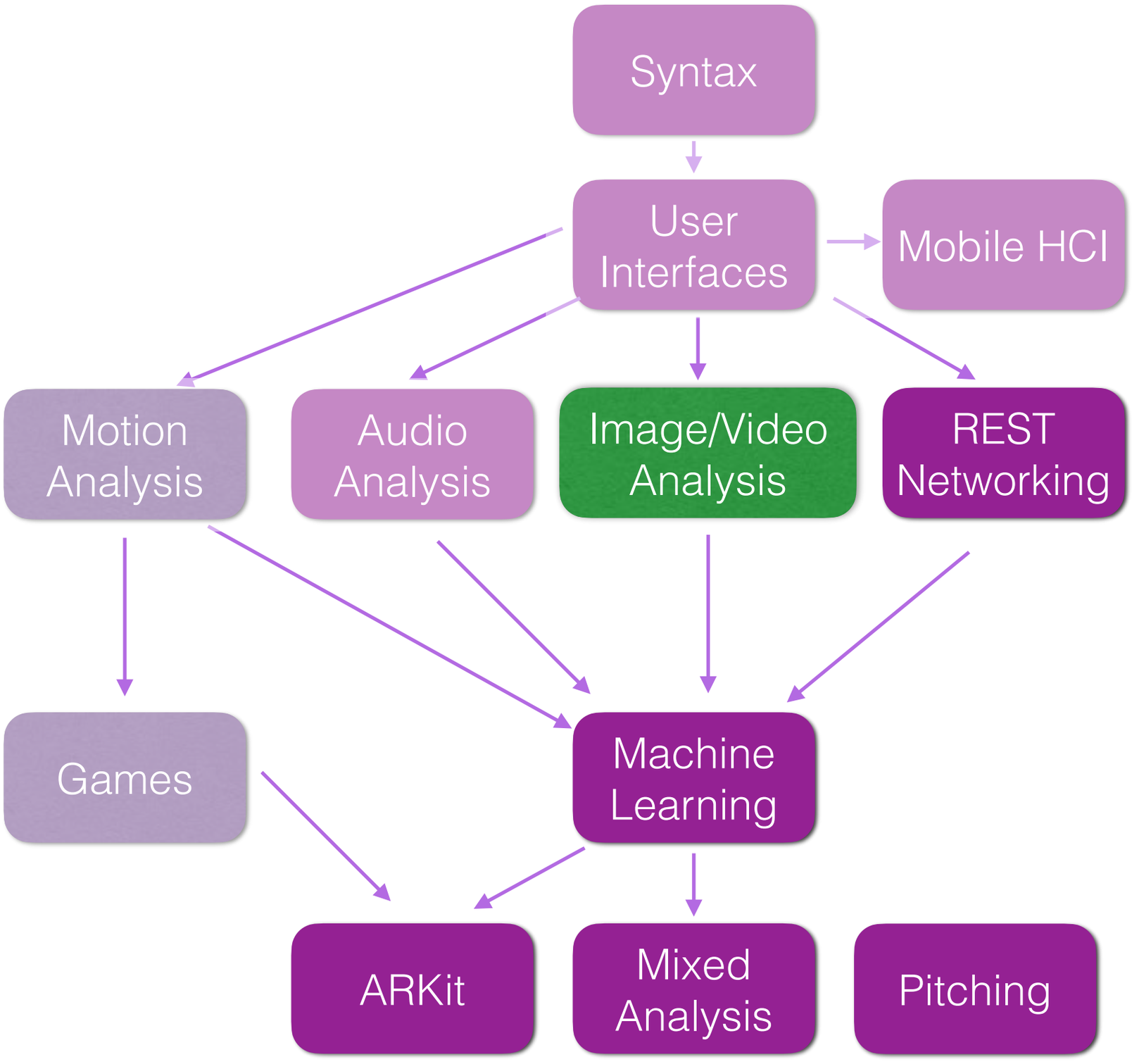
lab three town hall

step (health) reward

fun user interface…

basic game with motion that is substantially different than games from instructor

class overview

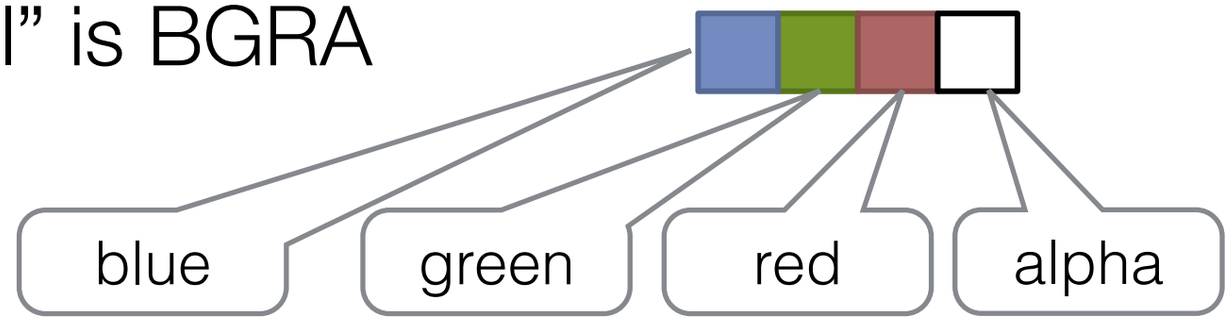


what is image processing?

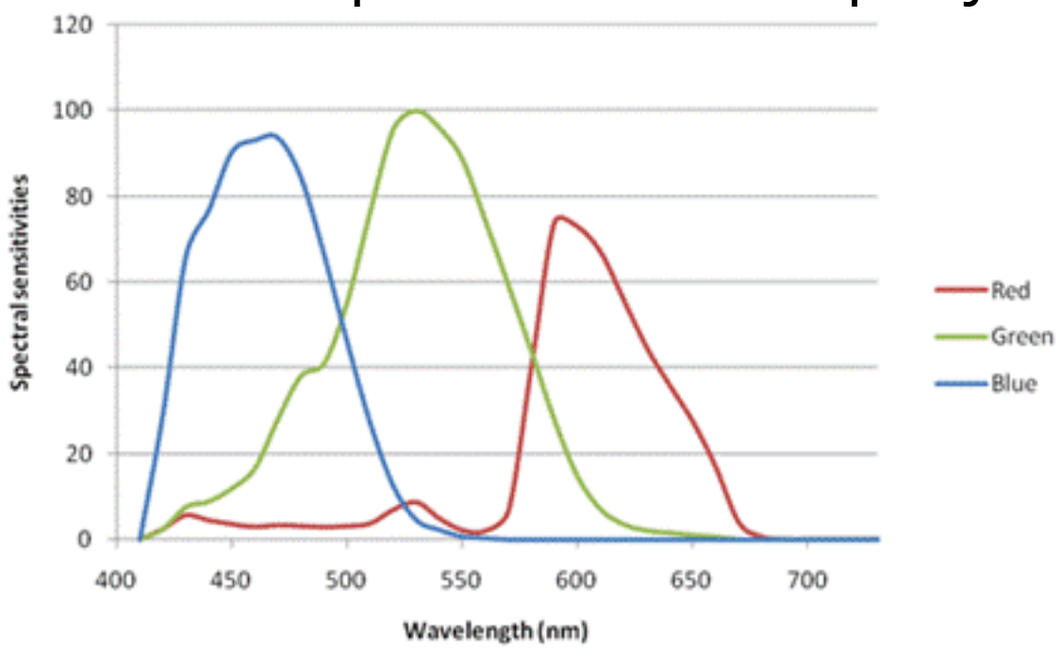
- the **art** and **science** of manipulating pixels
  - combining images (blending or compositing)
  - enhancing edges and lines
  - adjusting contrast, color
  - warping, transformation
  - filtering
  - …anything you can do in photoshop
  - also used in computer vision

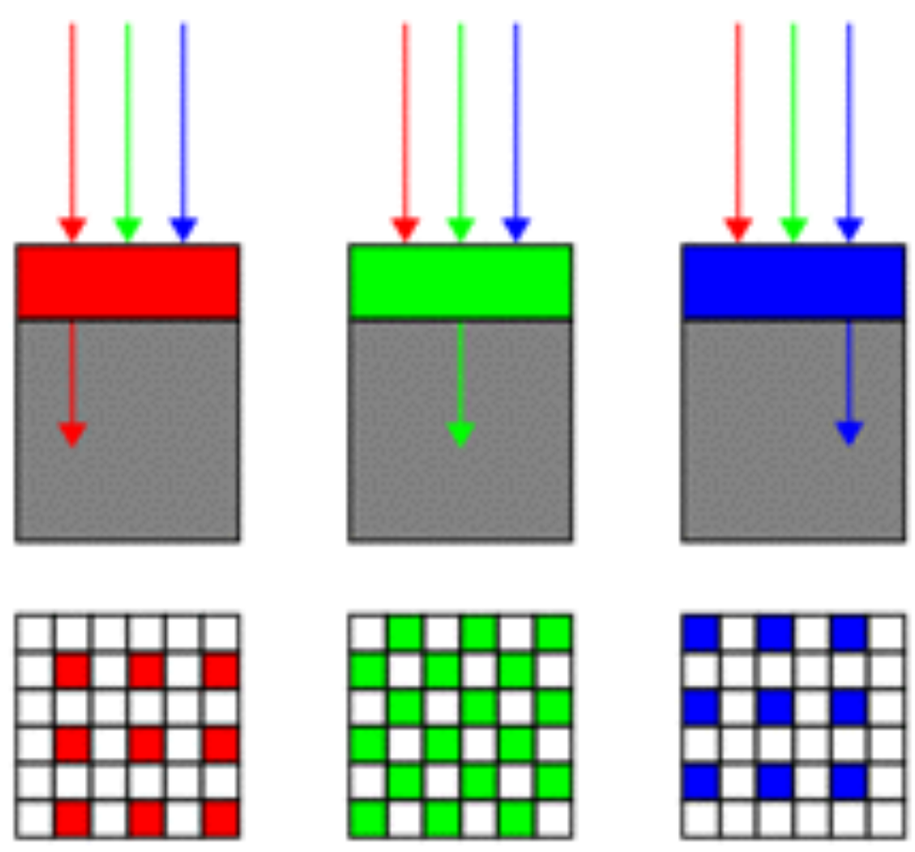
images as data

- an image can be represented in many ways
- most common format is a matrix of pixels
  - each “pixel” is BGRA



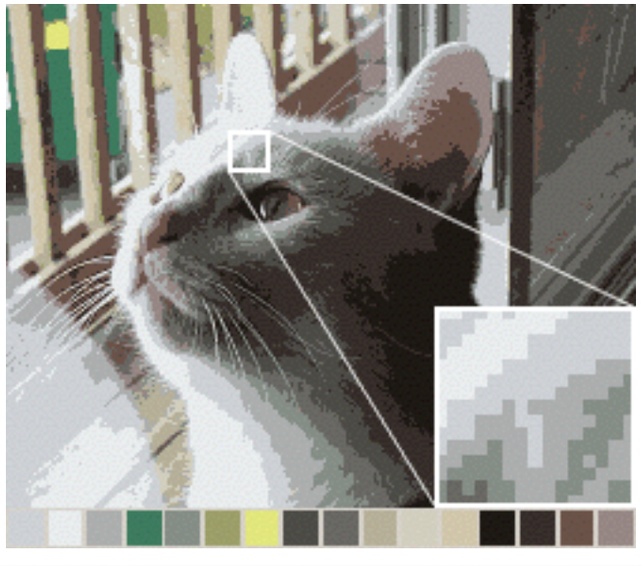
- used for capture and display





images as signals

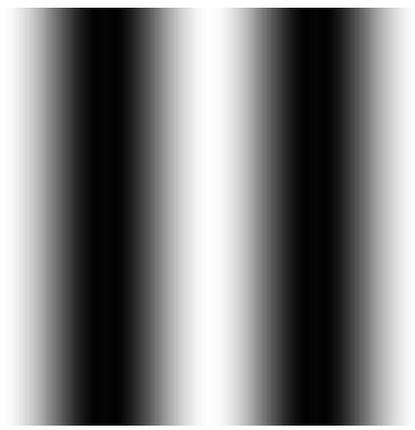
- everything from audio still applies
- quantization
  - each pixel can only take on 0-255 values
  - i.e., “stretching” in low light conditions

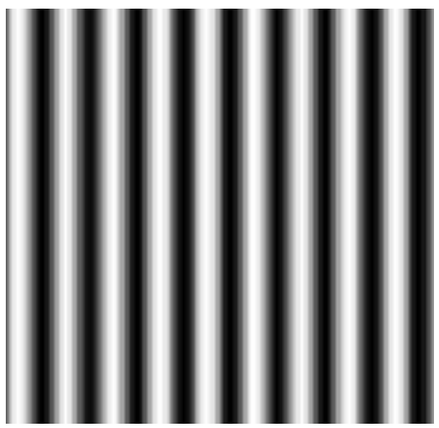


sampling errors

- in time (video)

- in space (resolution)
  - “frequency” is in terms of spatial sine waves





images as signals

image with lots of low frequency

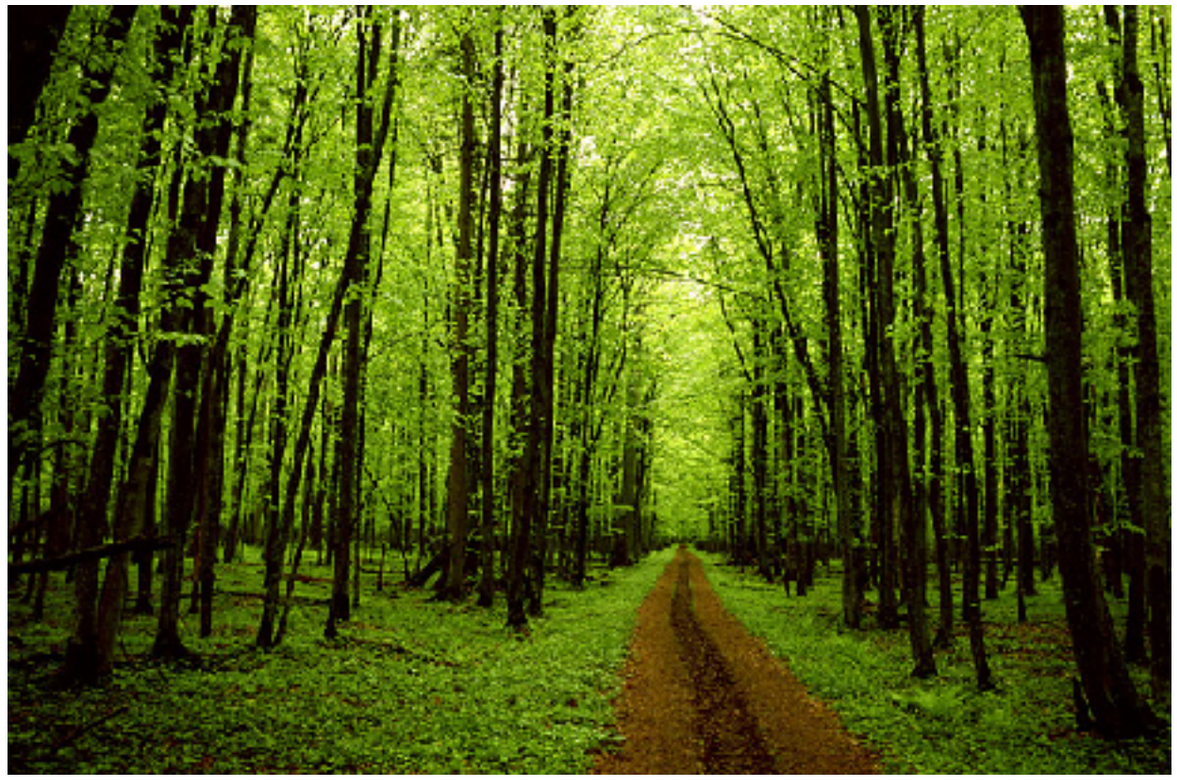


image with lots of high frequency

what is filtering?

- same as audio
  - convolution (linear)

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

kernel

averaging

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

what is filtering?

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

vertical difference

-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

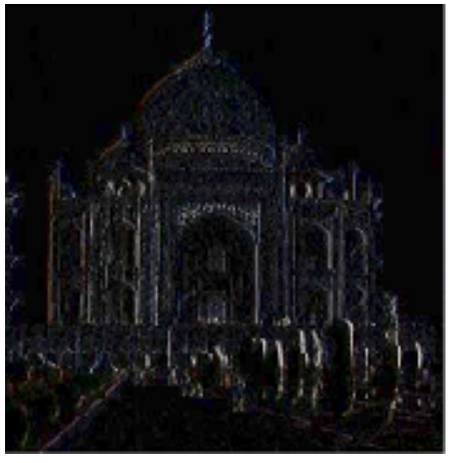
examples of filtering

Blur

1	1	1
1	1	1
1	1	1

Vertical Edges

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

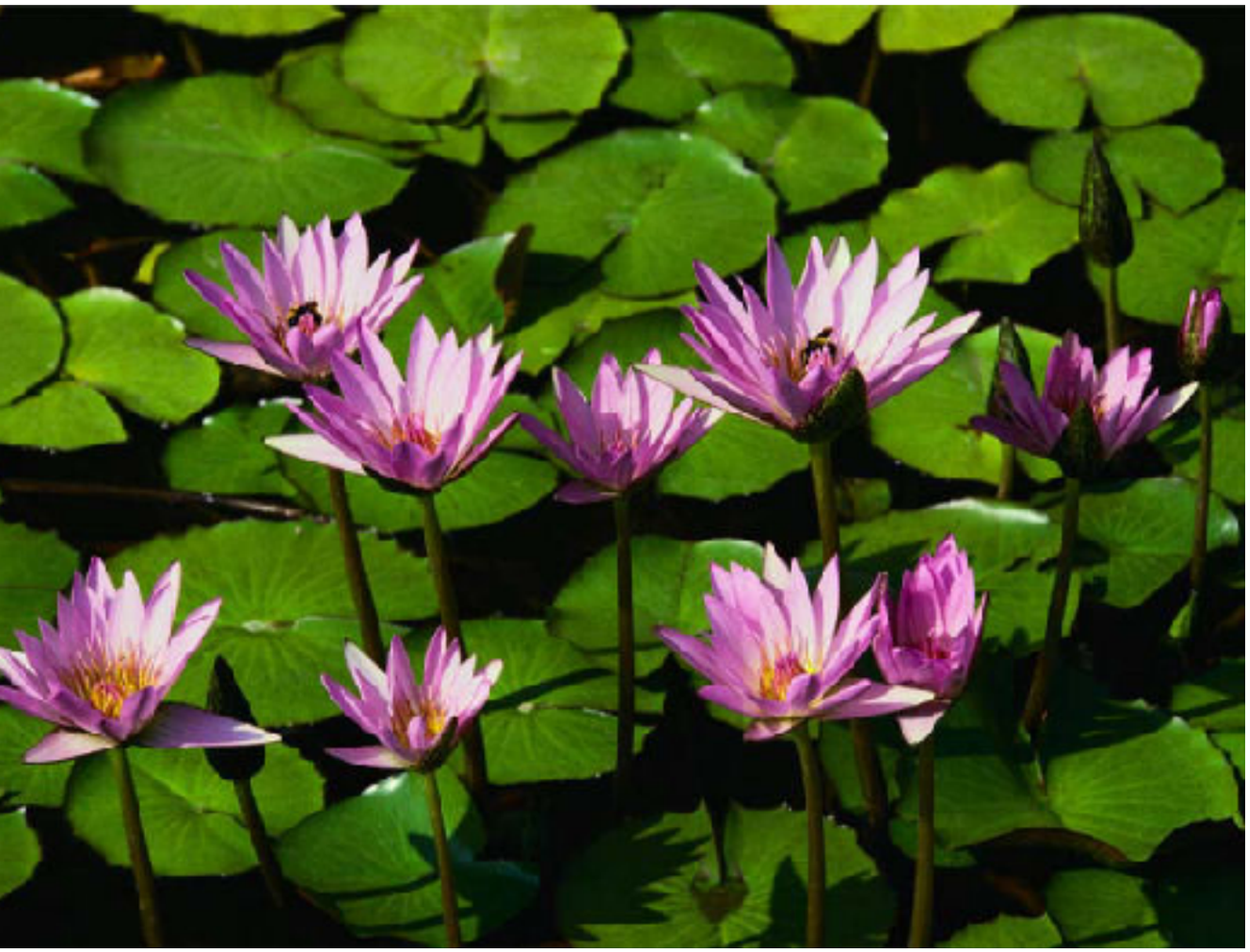


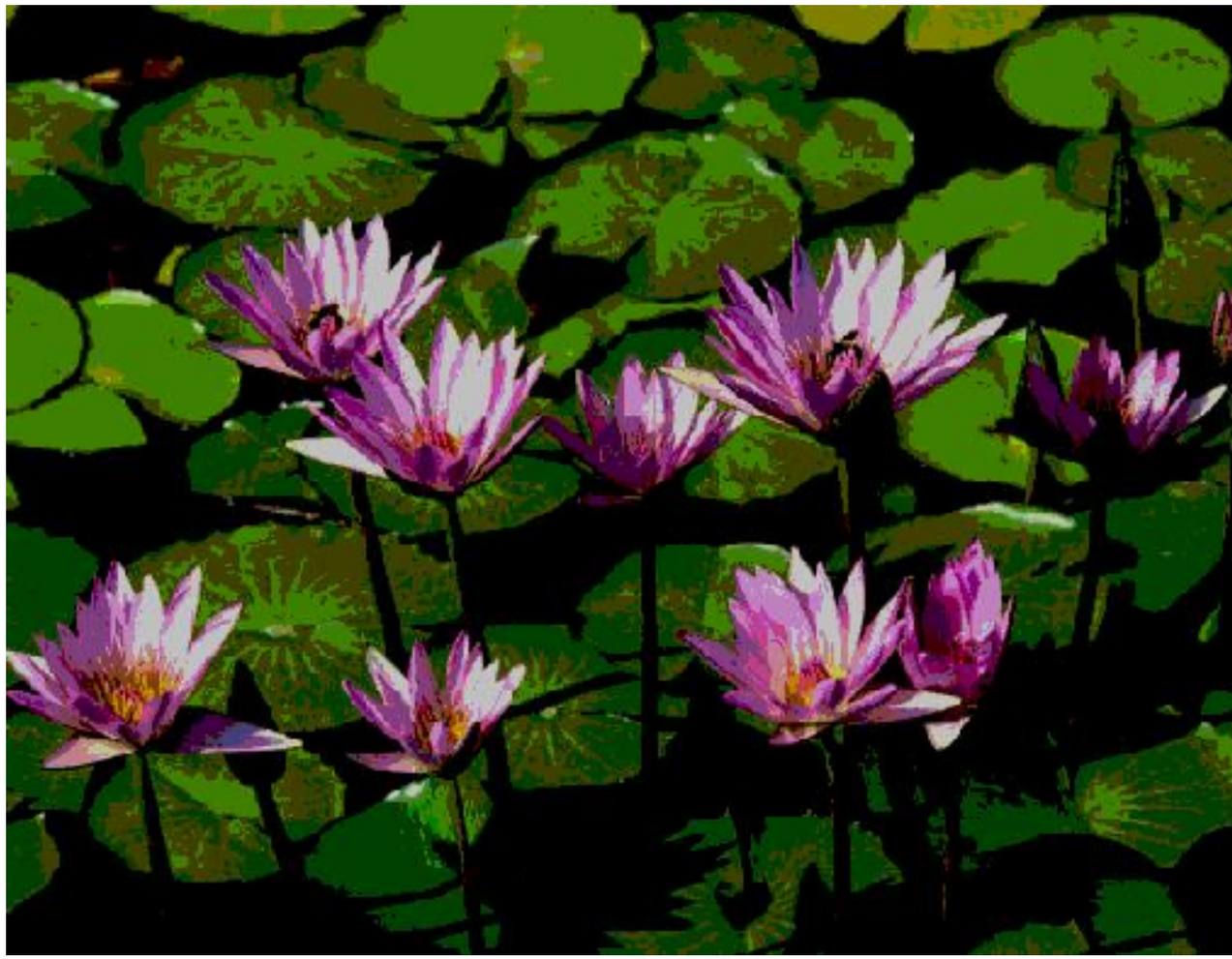
Sharpen

0	-1	0
-1	5	-1
0	-1	0

interesting operations

but there is no need to just perform linear convolution!





non-linear 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	-
-	-	-

R

G

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

- iPhone 16 Pro:
  - back camera is capable of capturing
    - 8MP photos (~48 MB raw)
    - 1080p HD video at 60 fps
    - fast capture up to 240 fps
  - face camera
    - 1.2MP photos
    - 720p HD video at 30 fps
  - video on the face camera is 1280x720pixels x3 channels x30fps
    - 82.9 million samples per second

so much data !!!

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

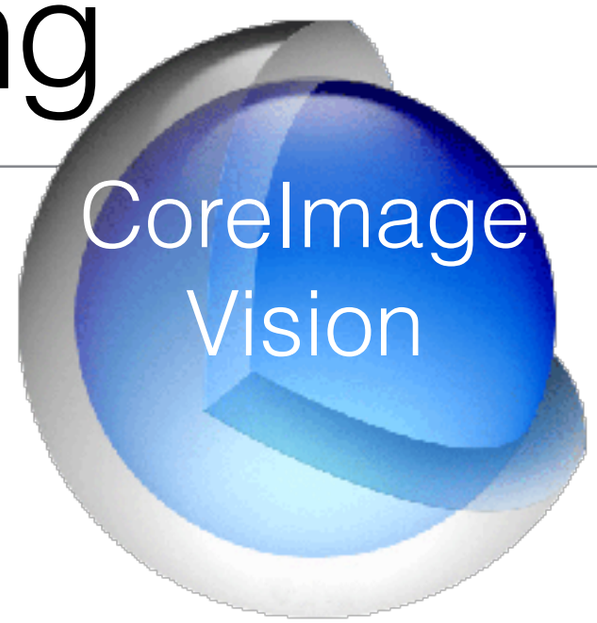
```
float one = 1.0;
vDSP_vdbcon(fftMagnitudeBuffer,1,&one,fftMagnitudeBuffer,1,kBufferLength/2,0);
```

  - faster than this:

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

parallelized data processing images: GPU

options for image processing



- **CoreImage and Vision** (written by Apple)
  - somewhat extensible, very fast, easy to use
  - if not implemented, hard to update, actively being improved, can fall back on CPU
- **GPUImage** (independent developer)
  - open source, updated for swift, lots of users, very fast (comparable to CI), very easy to use
  - developed and maintained by one guy, Brad Larson
- **OpenCV** (started by Intel)
  - slow, requires c++, huge
  - most comprehensive functionality, biggest user base (gigantic),

core image framework

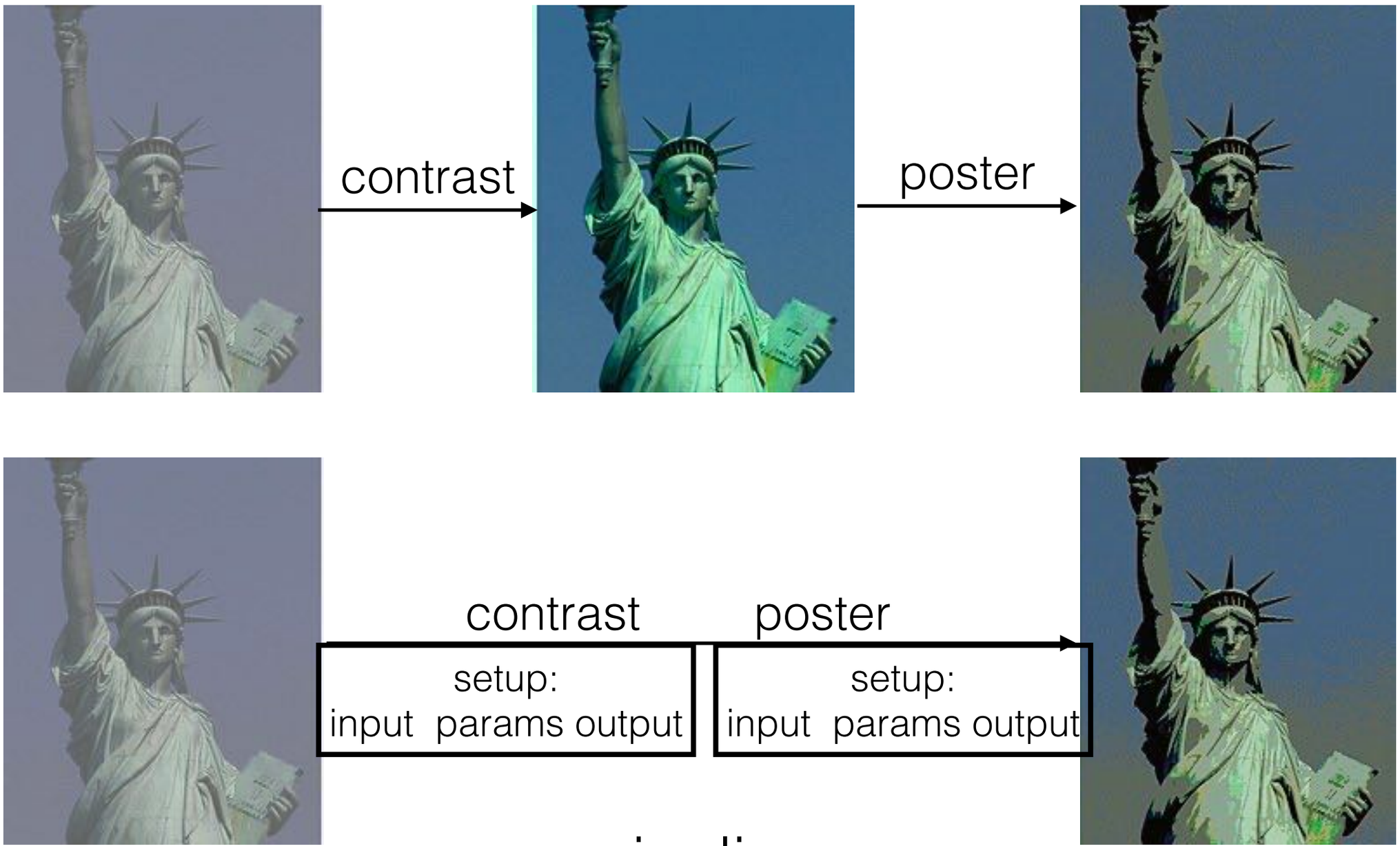
- defines images as CIImage instances
- defines a set of filters that (can be) GPU accelerated
  - optimizes filters when cascaded
- filters created through CIFilter class instances

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

could be convolution, non-linear windows, or other

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 (**note**: not a UIImage!)

```
let urlPath = Bundle.mainBundle().pathForResource("smu-campus", ofType: "jpg")
let fileURL = URL.fileURLWithPath(urlPath!)

let beginImage = CIImage(contentsOfURL: fileURL)

    …setup processing here…

self.imageView.image = UIImage(CIImage: beginImage)
```

get image path from bundle

load image

show inside a UIImageView

core image syntax

```
let urlPath = Bundle.mainBundle().pathForResource("smu-campus", ofType: "jpg")
let fileURL = URL.fileURLWithPath(urlPath!)

let beginImage = CIImage(contentsOfURL: fileURL)
```

create filter

```
let filter = CIFilter(name: “CIBloom")!
```

← filter type

set parameters

input image →
thresholds →

```
filter.setValue(beginImage, forKey: kCIInputImageKey)
filter.setValue(0.5, forKey: kCIInputIntensityKey)
```

get output

```
outputImage = filter.outputImage!
```

processing

```
self.sourceImageView.image = UIImage(CIImage: outputImage)
```

```
self.imageView.image = UIImage(CIImage: beginImage)
```

kCIInputImageKey = “inputImage”

kCIInputIntensityKey = “inputIntensity”

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 and in iOS 6.0 and later.

```
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];
```

available?

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;
 }
```

array of filters

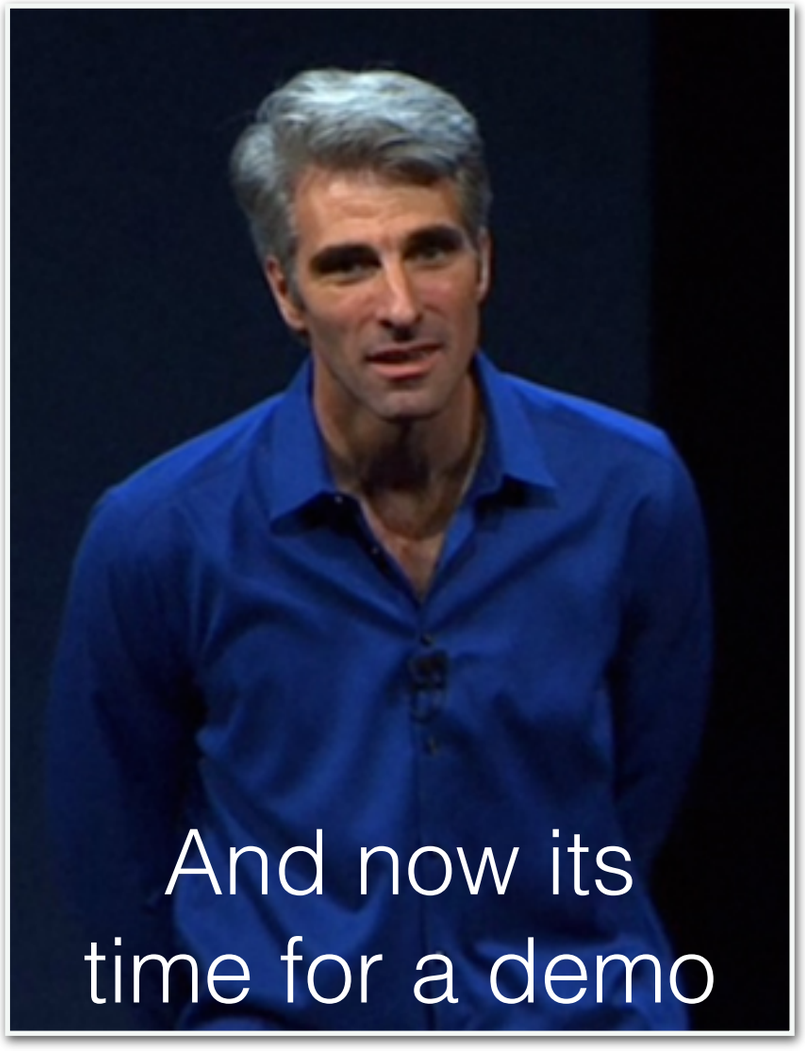
init image

apply each filter

core image demo

- ImageLab, filter image from bundle

http://www.raywenderlich.com/76285/beginning-core-image-swift



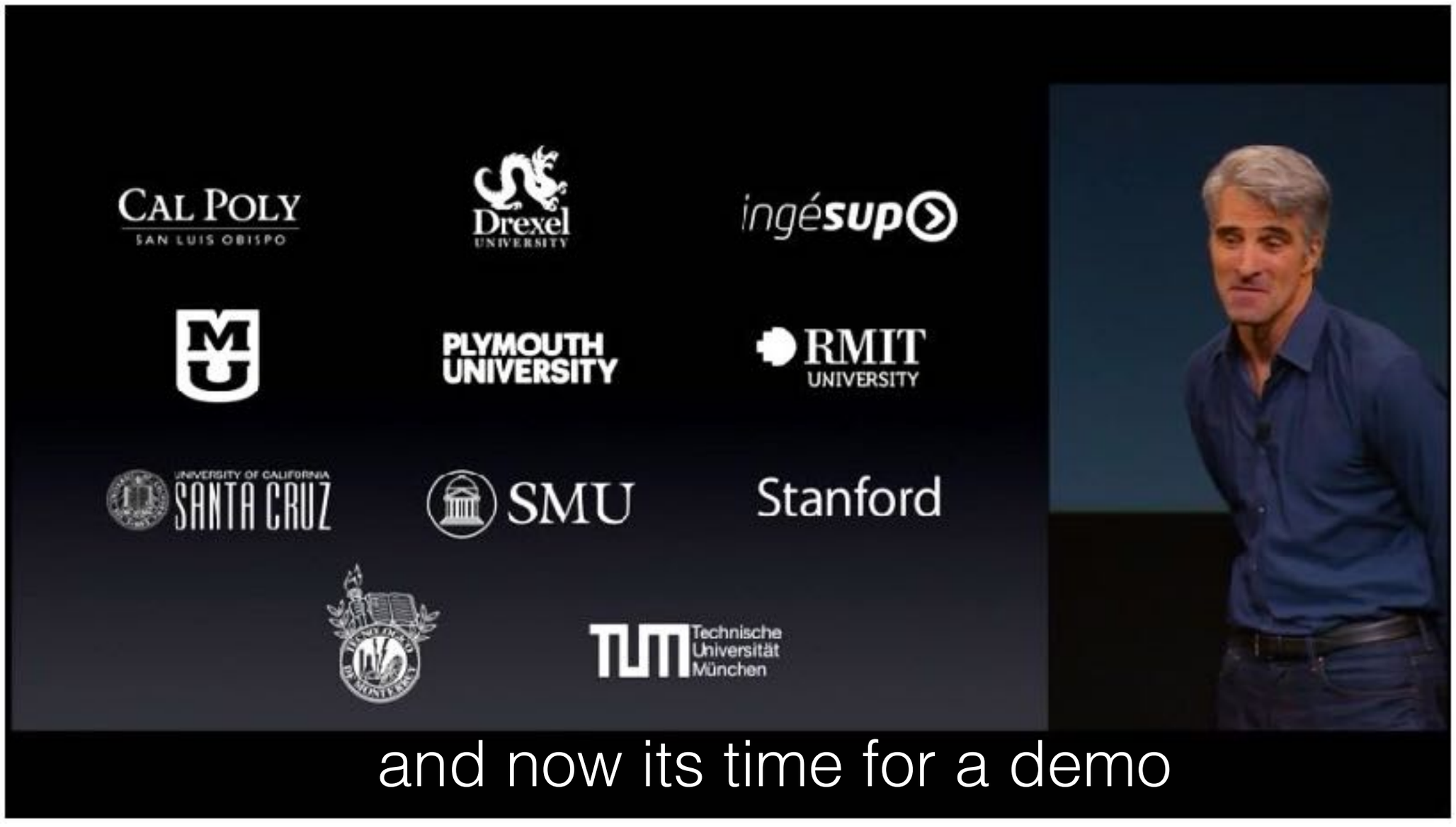
beyond the main bundle "if time"

- easy to get photos!
- demos in the branch of **ImageLab**, if you are interested!
- 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

core image demo

"if time"

- ImageLab, using the camera

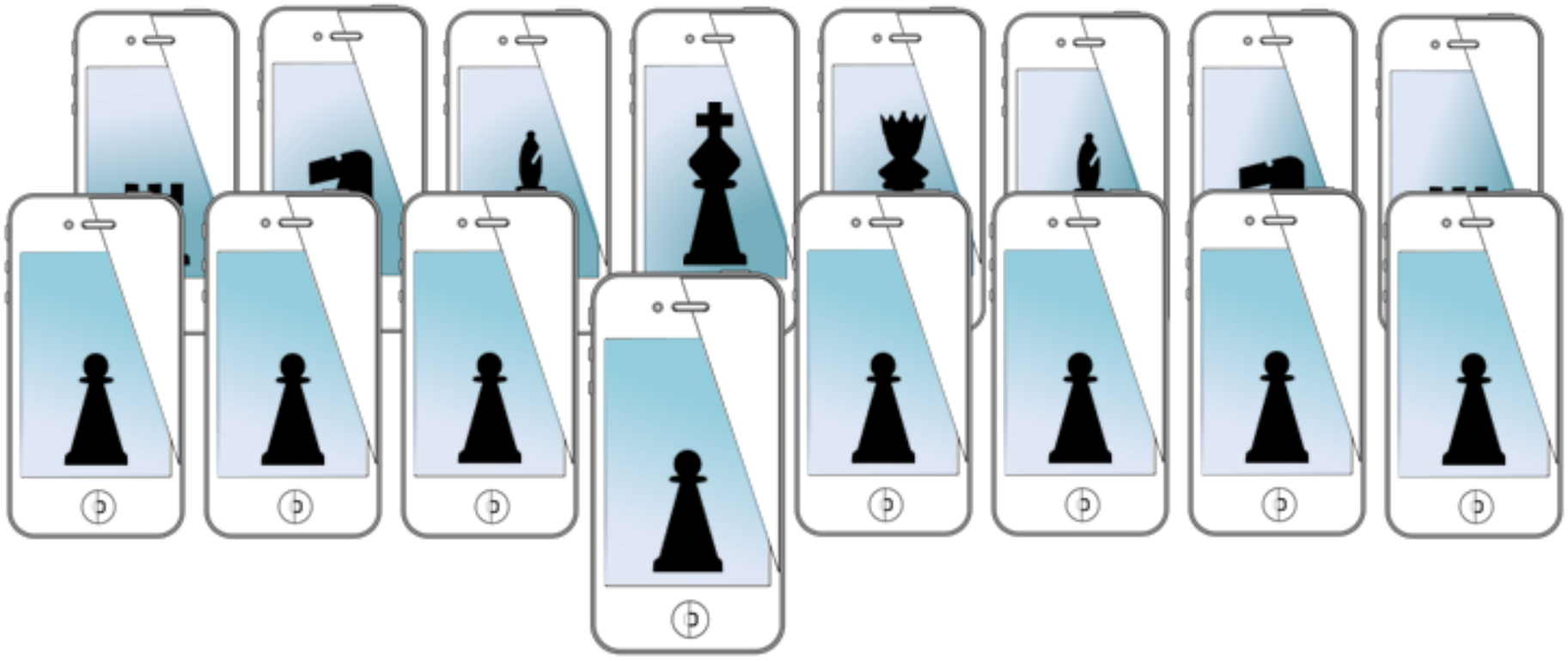


for next time…

- remainder of computer vision:
  - filters
  - faces
  - physiology

- next week in-class assignment!!
  - computer vision with Apple Vision
    - generic detection
    - tracking

MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

core image and image processing

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