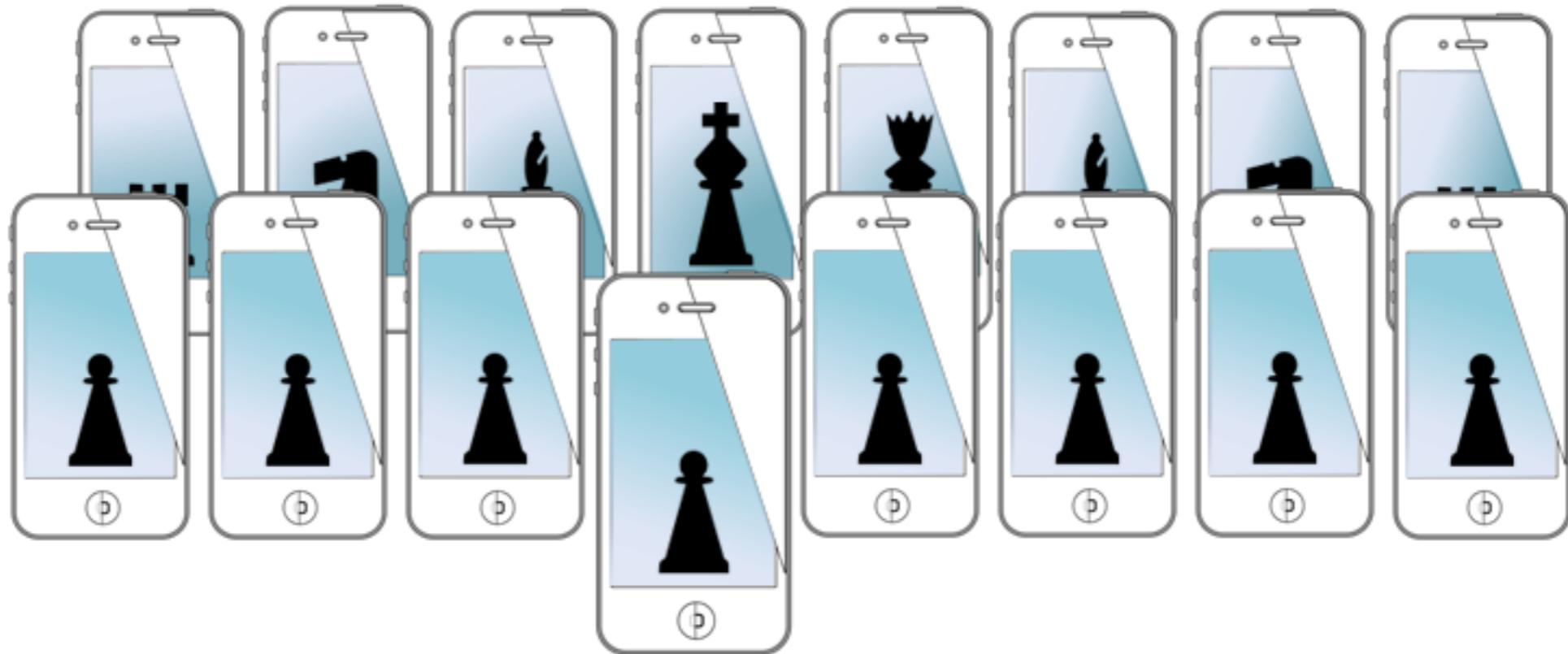


MOBILE SENSING LEARNING



CSE5323 & 7323
Mobile Sensing and Learning

week 6: core image and image processing

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

course logistics

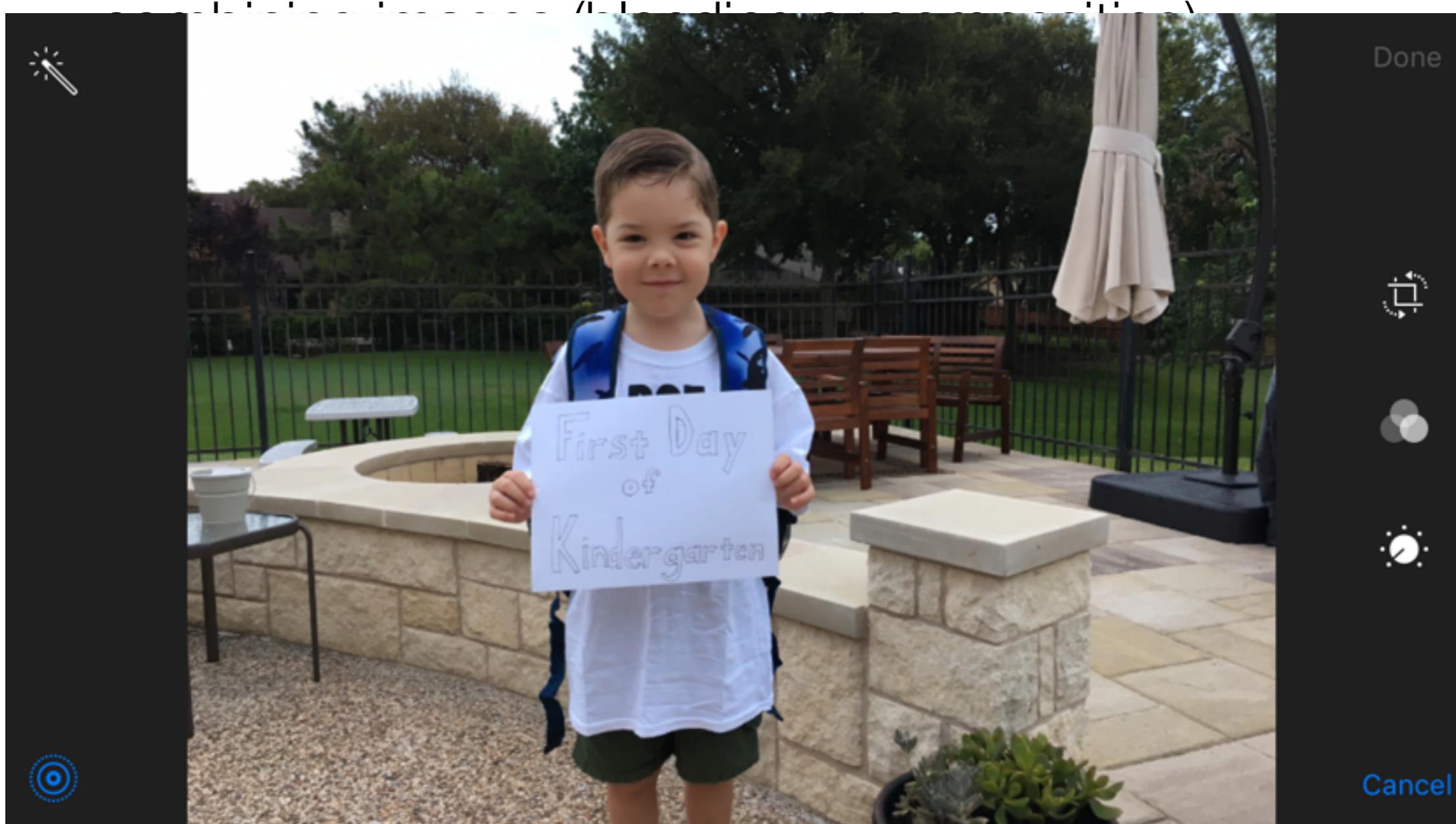
- grades coming soon
- A3 will be due Friday

agenda

- image processing basics
- core image filtering

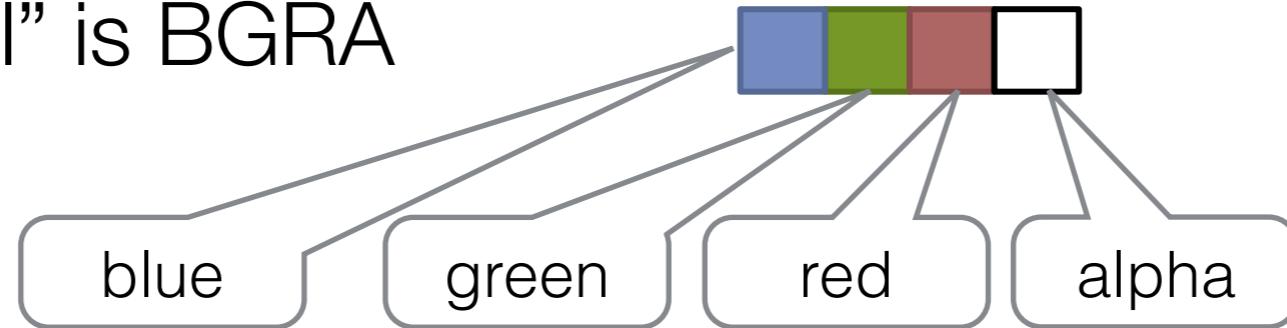
what is image processing

- the **art** and **science** of manipulating pixels

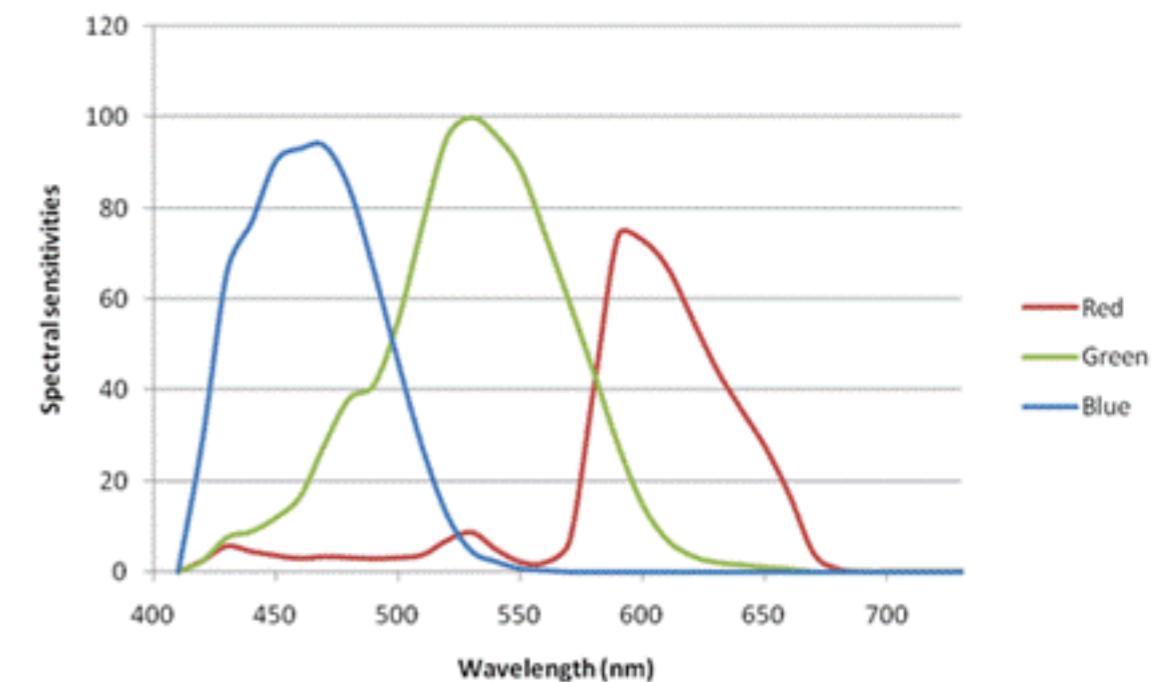
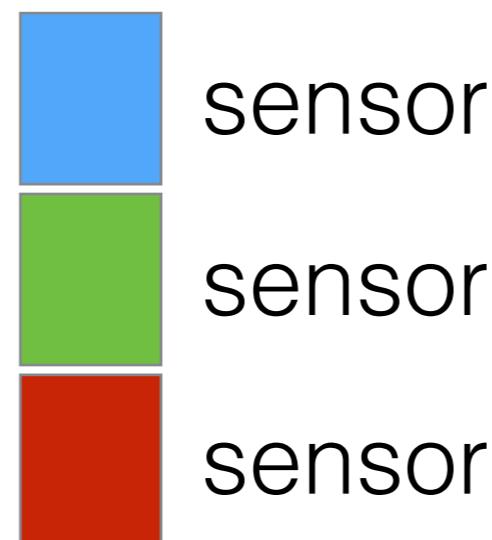
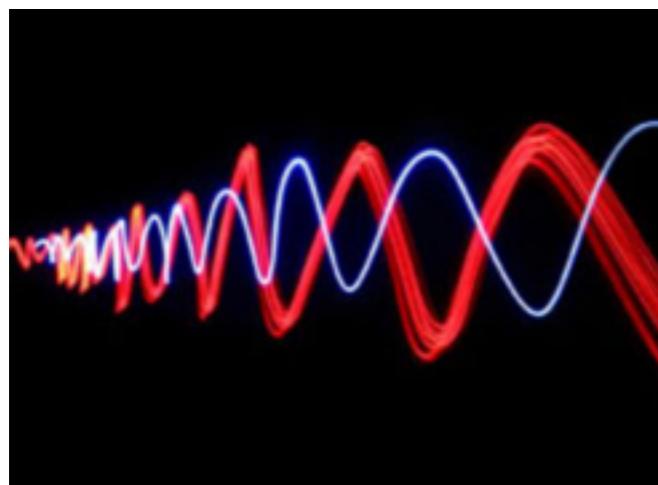


images as data

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

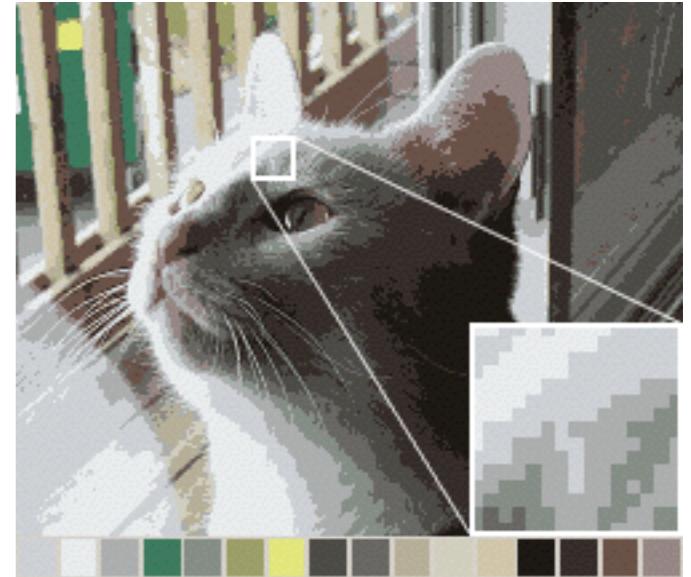


- used for capture and display



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



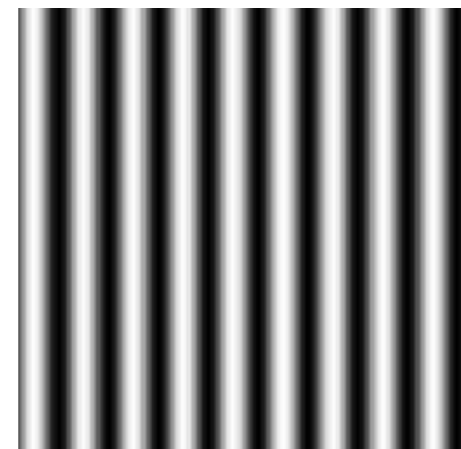
sampling errors

- in time (video)



- in space (resolution)

- “frequency” is in terms of spatial sine waves



images as signals



image with lots of high frequency

images as signals



image with lots of low 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

interesting operations

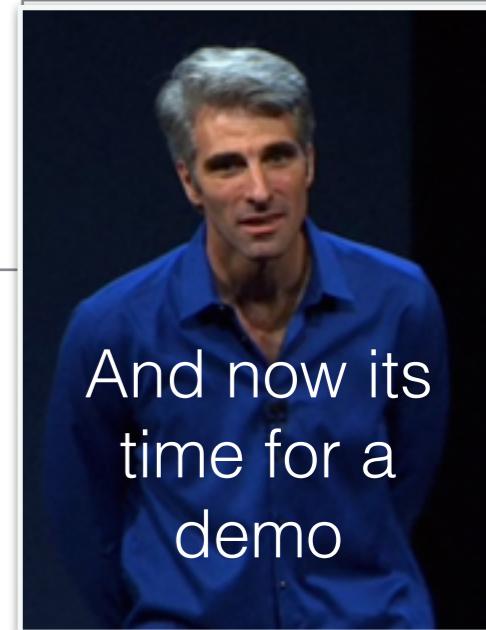
- the power of simple filtering
 - **a demo, via python**



And now its
time for a
demo

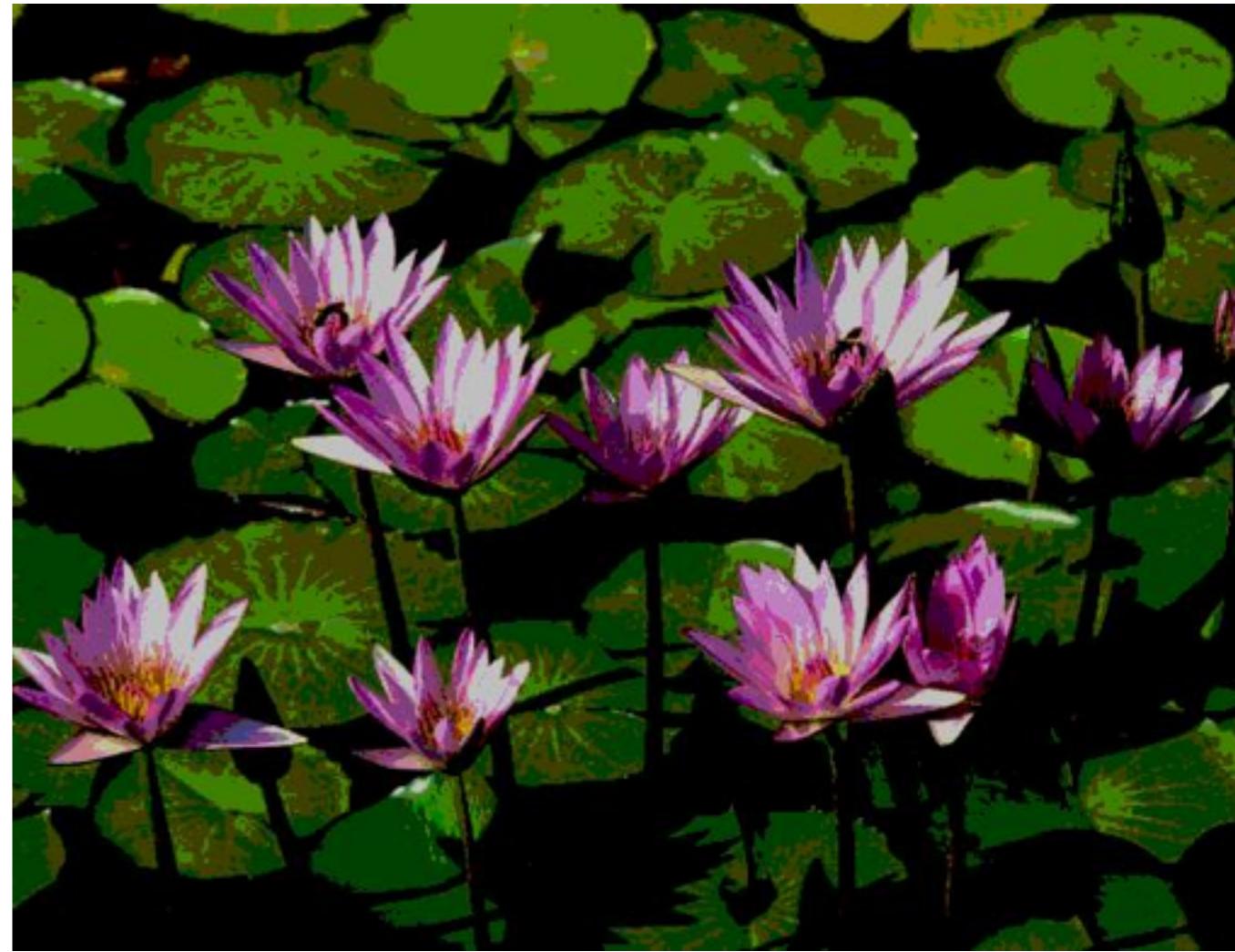
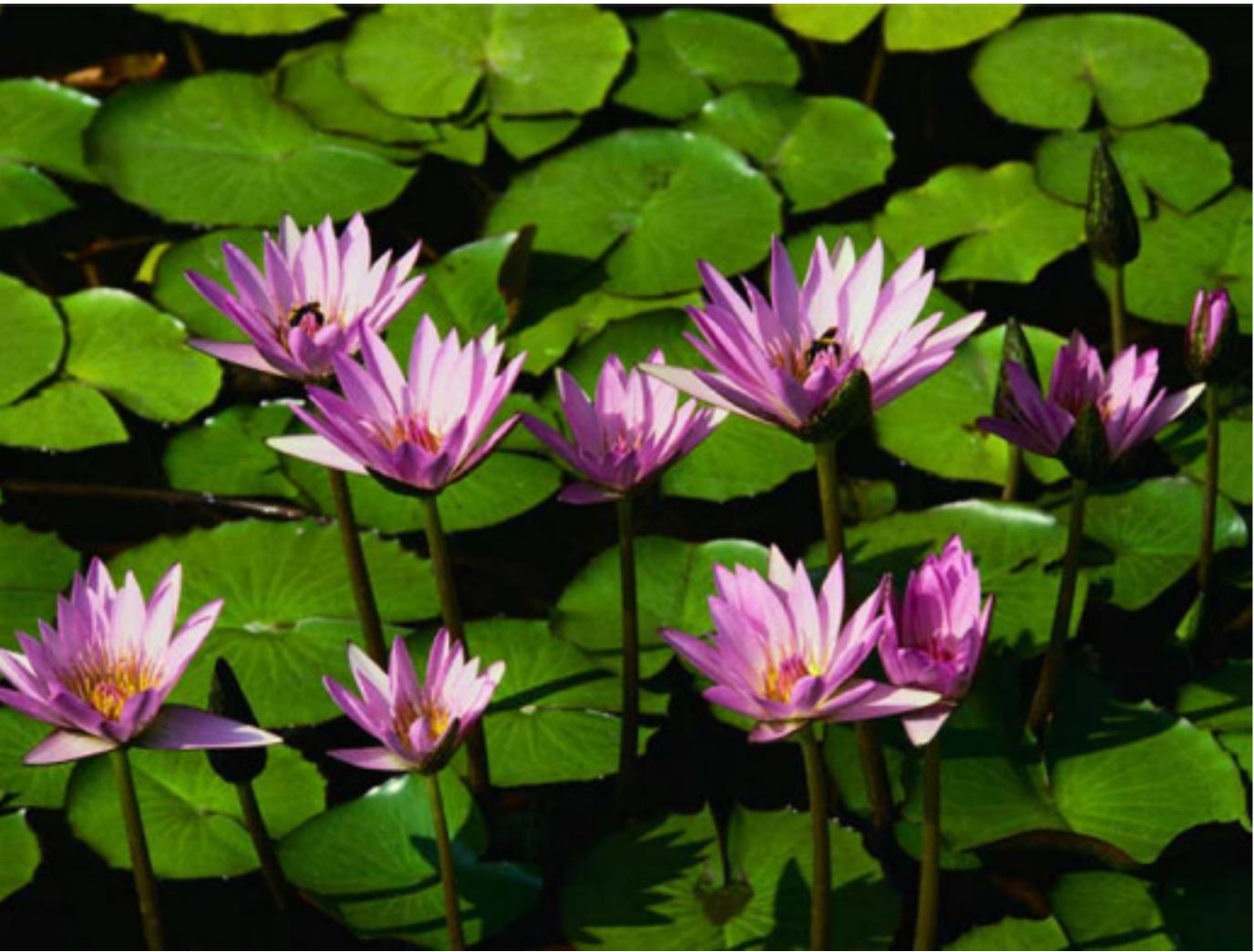
interesting operations

- the power of simple filtering
 - **a demo, via python**



And now its
time for a
demo

but there is no need to just perform linear convolution!



non-linear filtering

gray scale image

-	-	-
-	max	-
-	-	-

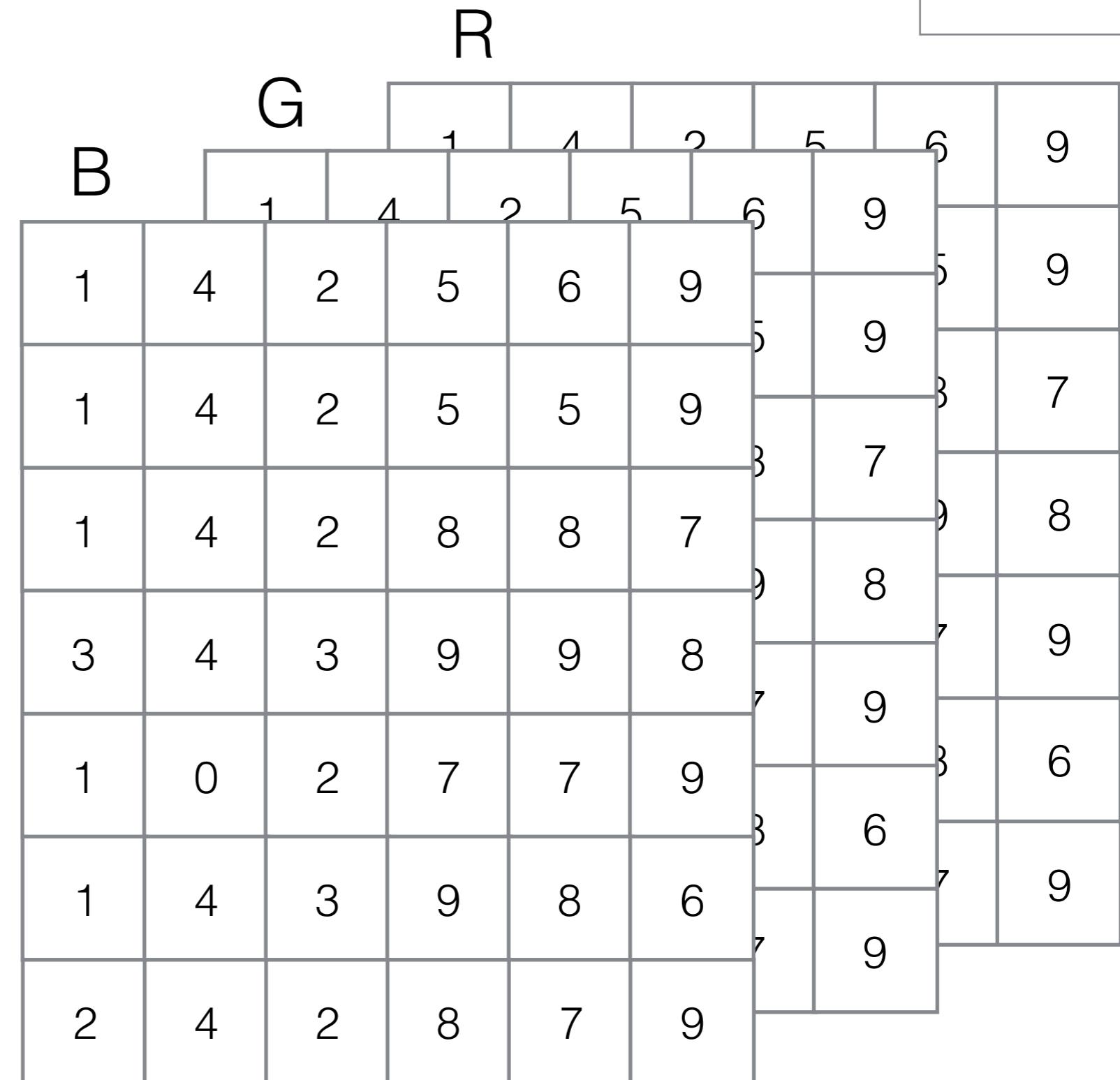
kernel

-	-	-
-	median	-
-	-	-

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	-
-	-	-



filtering video

- for default 5S available in class (it gets better with newer phones):
- 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 !!!

- 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** (written by Apple)
 - somewhat extensible, very fast, easy to use
 - if not implemented, you can't do it (like computer vision), always getting better, can fall back on CPU
- **GPUImage**
 - 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**
 - slow, requires c++, bloated
 - most comprehensive functionality, biggest user base (gigantic),



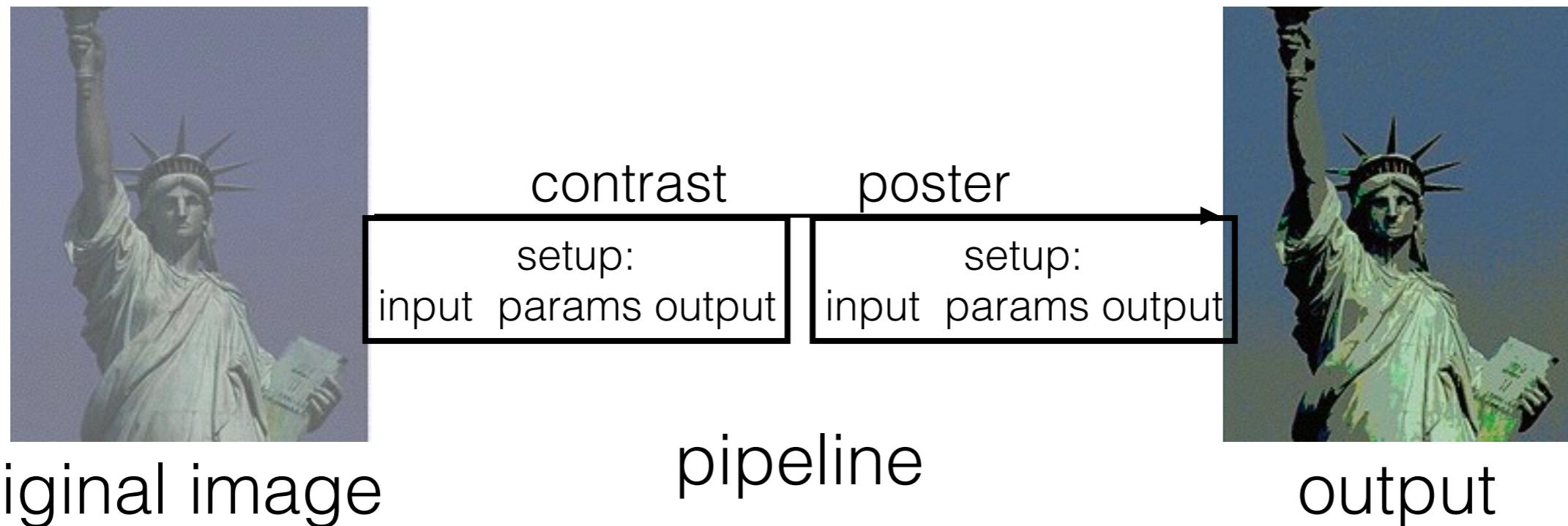
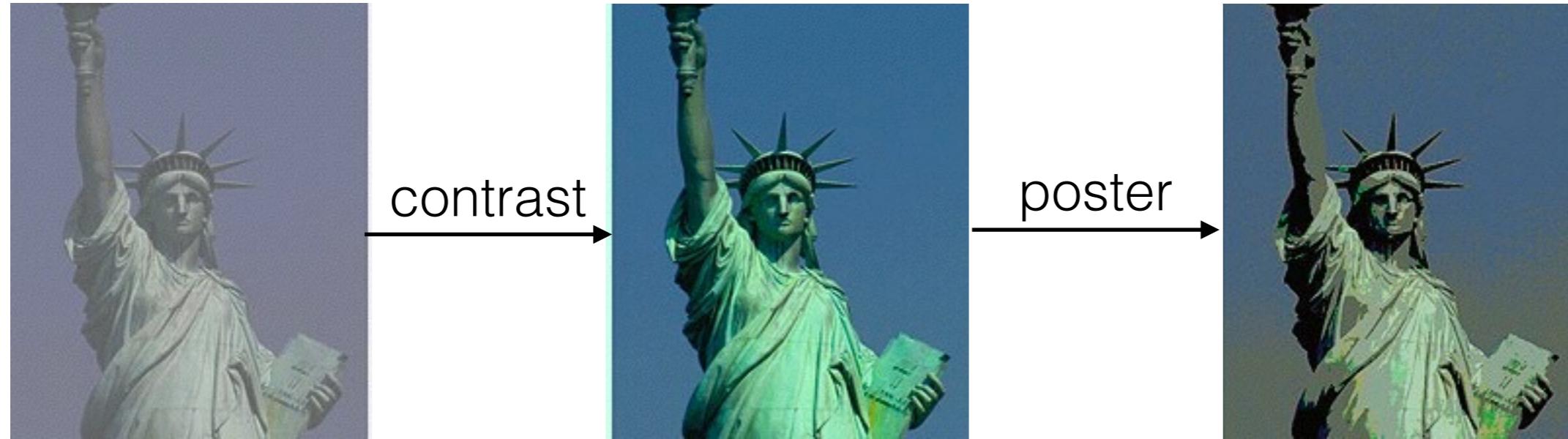
core image framework

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

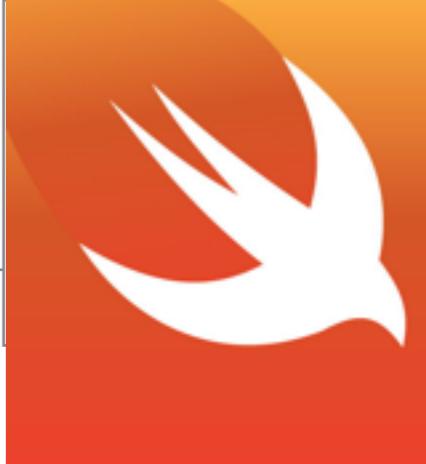
<code>CIAdditionCompositing</code>	<code>CIColorCrossPolynomial</code>	<code>CIFourfoldReflectedTile</code>	<code>CIMaximumComponent</code>	<code>CISourceAtopCompositing</code>
<code>CAffineClamp</code>	<code>CIColorPolynomial</code>	<code>CIFourfoldRotatedTile</code>	<code>CIMaximumCompositing</code>	<code>CILinearToSRGBToneCurve</code>
<code>CAffineTile</code>	<code>CIColorPosterize</code>	<code>CIFourfoldTranslatedTile</code>	<code>CIMinimumComponent</code>	<code>CISRGBToneCurveToLinear</code>
<code>CAffineTransform</code>	<code>CIConstantColorGenerator</code>	<code>CGammaAdjust</code>	<code>CIMinimumCompositing</code>	<code>CISourceInCompositing</code>
<code>CBarsSwipeTransition</code>	<code>CIConvolution3X3</code>	<code>CGaussianBlur</code>	<code>CIModTransition</code>	<code>CISourceOutCompositing</code>
<code>BlendWithMask</code>	<code>CIConvolution5X5</code>	<code>CGaussianGradient</code>	<code>CMultiplyBlendMode</code>	<code>CISourceOverCompositing</code>
<code>CBloom</code>	<code>CIConvolution9Horizontal</code>	<code>CGlideReflectedTile</code>	<code>CMultiplyCompositing</code>	<code>CIStarShineGenerator</code>
<code>CBumpDistortion</code>	<code>CIConvolution9Vertical</code>	<code>CGloom</code>	<code>COVERLAYBlendMode</code>	<code>CIStraightenFilter</code>
<code>CheckerboardGenerator</code>	<code>CIcopyMachineTransition</code>	<code>CHardLightBlendMode</code>	<code>CPerspectiveTile</code>	<code>CIStripesGenerator</code>
<code>CircleSplashDistortion</code>	<code>CICrop</code>	<code>CHatchedScreen</code>	<code>CPerspectiveTransform</code>	<code>CSwipeTransition</code>
<code>CCircularScreen</code>	<code>CDarkenBlendMode</code>	<code>CHighlightShadowAdjust</code>	<code>CPinchDistortion</code>	<code>CTemperatureAndTint</code>
<code>ColorBlendMode</code>	<code>CDifferenceBlendMode</code>	<code>CHoleDistortion</code>	<code>CPixelate</code>	<code>CToneCurve</code>
<code>ColorBurnBlendMode</code>	<code>CDisintegrateWithMask</code>	<code>CHueAdjust</code>	<code>CRadialGradient</code>	<code>CTriangleKaleidoscope</code>
<code>ColorControls</code>	<code>CDissolveTransition</code>	<code>CHueBlendMode</code>	<code>CRandomGenerator</code>	<code>CTwelvefoldReflectedTile</code>
<code>ColorCube</code>	<code>CDotScreen</code>	<code>CLanczosScaleTransform</code>	<code>CSaturationBlendMode</code>	<code>CTwirlDistortion</code>
<code>ColorDodgeBlendMode</code>	<code>CEightfoldReflectedTile</code>	<code>CLightenBlendMode</code>	<code>CScreenBlendMode</code>	<code>CUUnsharpMask</code>
<code>ColorInvert</code>	<code>CExclusionBlendMode</code>	<code>CLightTunnel</code>	<code>CSepiaTone</code>	<code>CVibrance</code>
<code>ColorMap</code>	<code>CExposureAdjust</code>	<code>CLinearGradient</code>	<code>CSHARPENLuminance</code>	<code>CVignette</code>
<code>ColorMatrix</code>	<code>CFaceDetector</code>	<code>CLineScreen</code>	<code>CSixfoldReflectedTile</code>	<code>CVortexDistortion</code>
<code>ColorMonochrome</code>	<code>CFalseColor</code>	<code>CLuminosityBlendMode</code>	<code>CSixfoldRotatedTile</code>	<code>CIWhitePointAdjust</code>
<code>ColorClamp</code>	<code>CFlashTransition</code>	<code>CMaskToAlpha</code>	<code>CSOFTLightBlendMode</code>	<code>CIQRCodeGenerator</code>

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!)

get image path from bundle

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

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

load image

...processing here...

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

show inside a UIImageView

core image syntax



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

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

create filter

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

filter type

set parameters

input image

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

thresholds

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

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

get output

processing

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

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

core image filters

- <https://developer.apple.com/library/mac/documentation/graphicsimaging/reference/CoreImageFilterReference/Reference/reference.html>
- names, parameters, examples, etc.

Filters (linear and non-linear)

CICategoryBlur
CICategorySharpen
CICategoryStylize

Distortion

CICategoryCompositeOperation
CICategoryDistortionEffect
CICategoryGeometryAdjustment
CICategoryTileEffect

Other Images

CICategoryGenerator
CICategoryHalftoneEffect
CICategoryGradient

Color

CICategoryColorAdjustment
CICategoryColorEffect

Features (histogram, averages)

CICategoryReduction

core image filters



CIBloom

Softens edges and applies a pleasant glow to an image.

Parameters

inputImage

A `CIIImage` 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"];

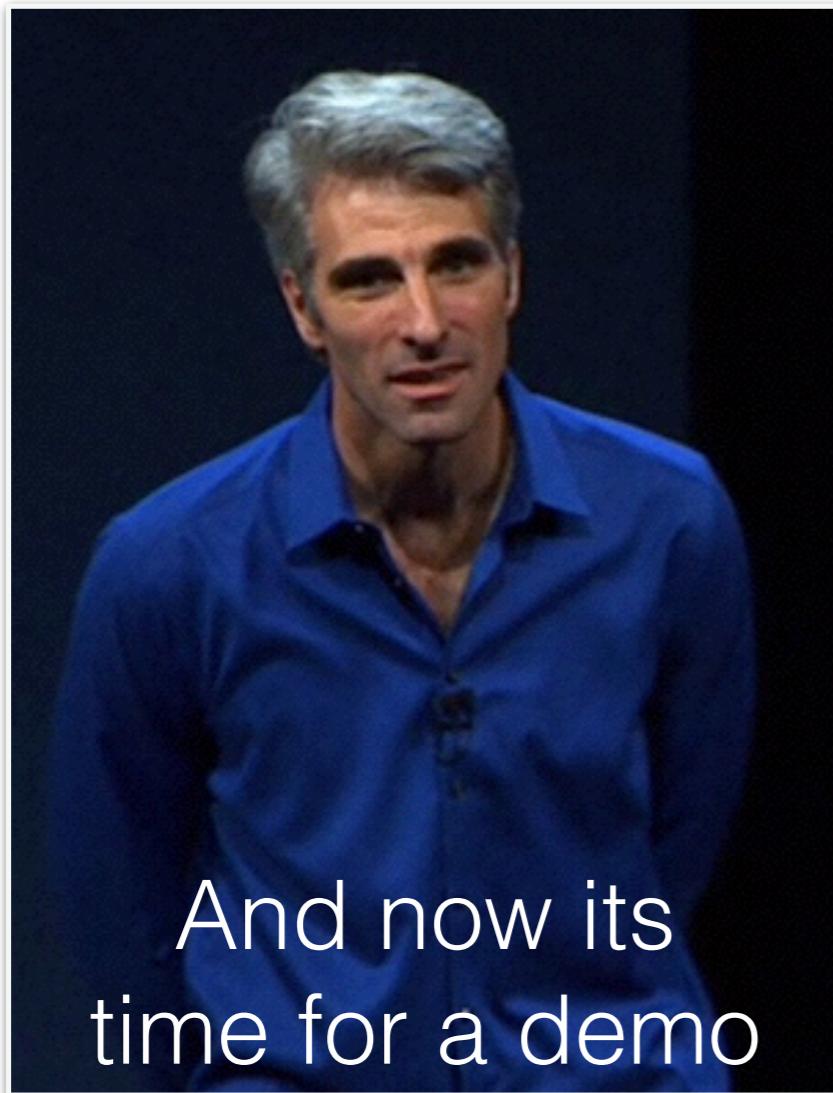
[filter setValue:myImage
 forKey:kCIInputImageKey];
```

available?

core image demo

- ImageLab, filter image from bundle

[http://www.raywenderlich.com/76285/
beginning-core-image-swift](http://www.raywenderlich.com/76285/beginning-core-image-swift)



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"]];

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

beyond the main bundle

- 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/AVFoundation/Conceptual/CameraAndPhotoLib_TopicsForIOS/Introduction/Introduction.html#/apple_ref/doc/uid/TP40010405-SW1

access still images in album

```
<UIImagePickerControllerDelegate, UINavigationControllerDelegate>
```

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

interact with modal VC

this example uses a button
allocate & set options

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

present modal view

get image from VC

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

take a still image 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];
}
```

but we want video...

- 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 intuitively
 - but you won't write the code to do it

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

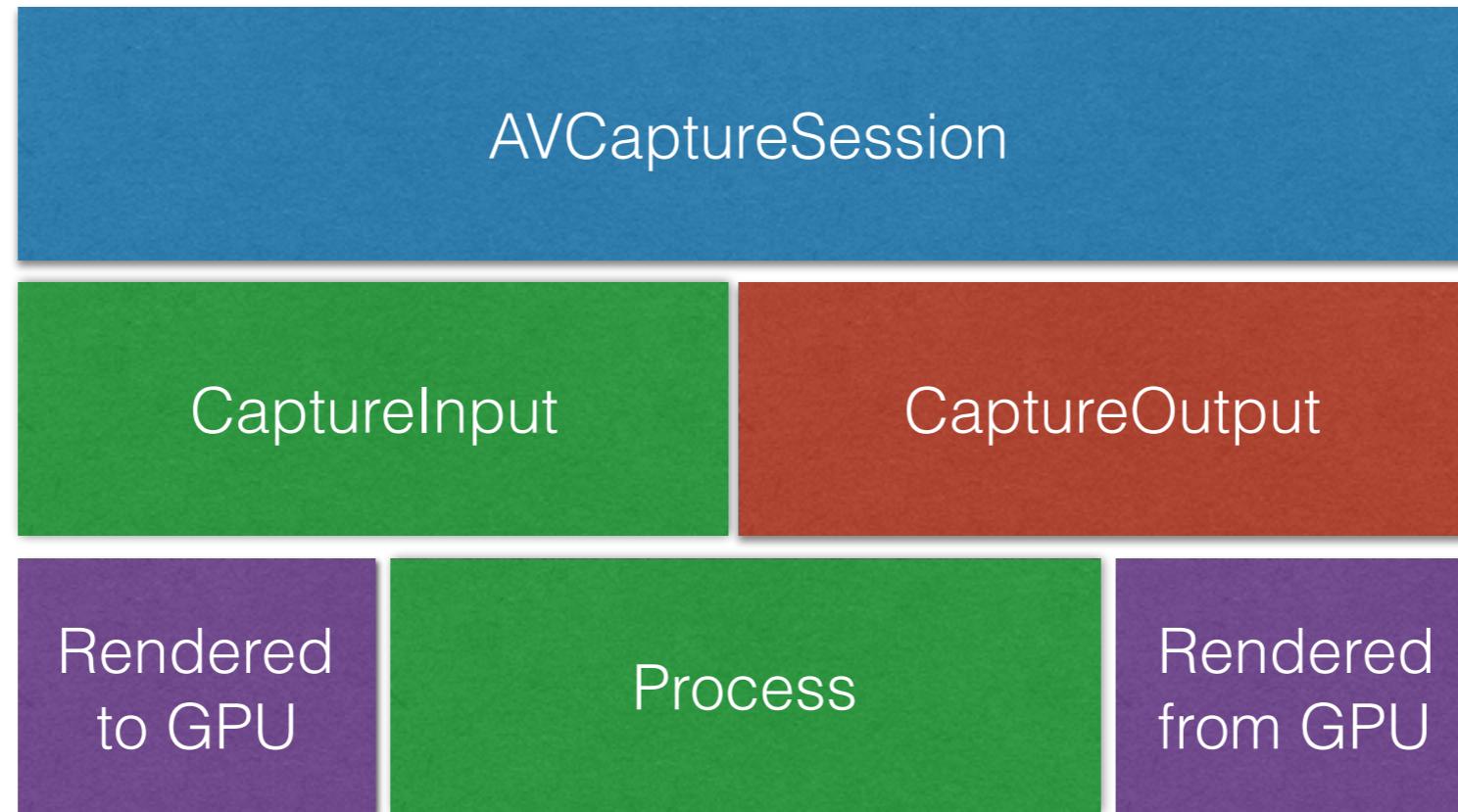
`AVCaptureDevicePositionFront`

`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



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

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
init
start
options
stop

```
#import "VideoAnalgesic.h"

@property (strong, nonatomic) VideoAnalgesic *videoAnalgesic;

self.captureManager = [VideoAnalgesic captureManager];

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

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

[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

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

return image to draw to screen

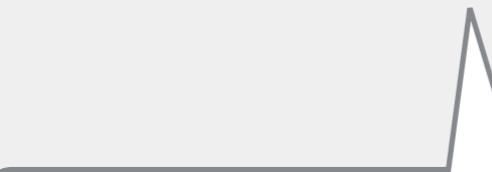
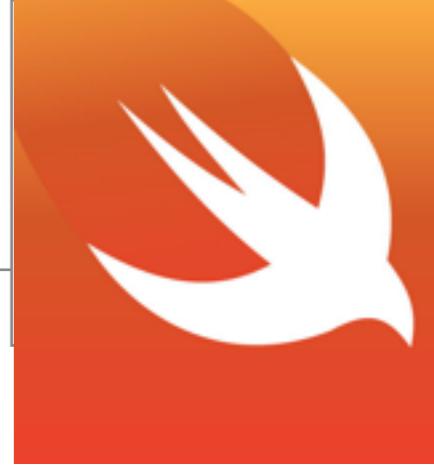


image from camera sent in block

```
__block CIFilter *filter = [CIFilter filterWithName:@"CISepiaTone"];  
[self.captureManager setProcessBlock:^(CIImage *cameraImage){  
  
    [filter setValue:cameraImage forKey:kCIInputImageKey];  
    CIImage* output = filter.outputImage;  
    return output;  
  
}];
```

how to program this?



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

declare

```
var videoAnalgesic: VideoAnalgesic! = nil

self.videoAnalgesic = VideoAnalgesic.sharedInstance

if !self.videoAnalgesic.isRunning{
    self.videoAnalgesic.start()
}

self.videoAnalgesic.setPreset( AVCaptureSessionPresetMedium )
self.videoAnalgesic.toggleCameraPosition()
self.videoAnalgesic.setCameraPosition( AVCaptureDevicePosition.Front )

if self.videoAnalgesic.isRunning{
    self.videoAnalgesic.stop()
    self.videoAnalgesic.shutdown()
}
```

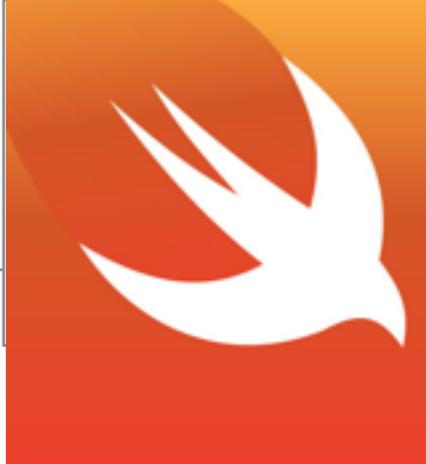
init

start

options

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

```
// setup a block to perform any processing
self.videoAnalgesic.setProcessingBlock()
{ (inputImage:CIImage) -> (CIImage) in
    return inputImage
}
```

image from camera passed in

return image to draw to screen

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

// setup a block to perform any processing
self.videoAnalgesic.setProcessingBlock()
{ (inputImage:CIImage) -> (CIImage) in
    filter.setValue(inputImage, forKey: "inputImage")
    return filter.outputImage
}
```

video process demo

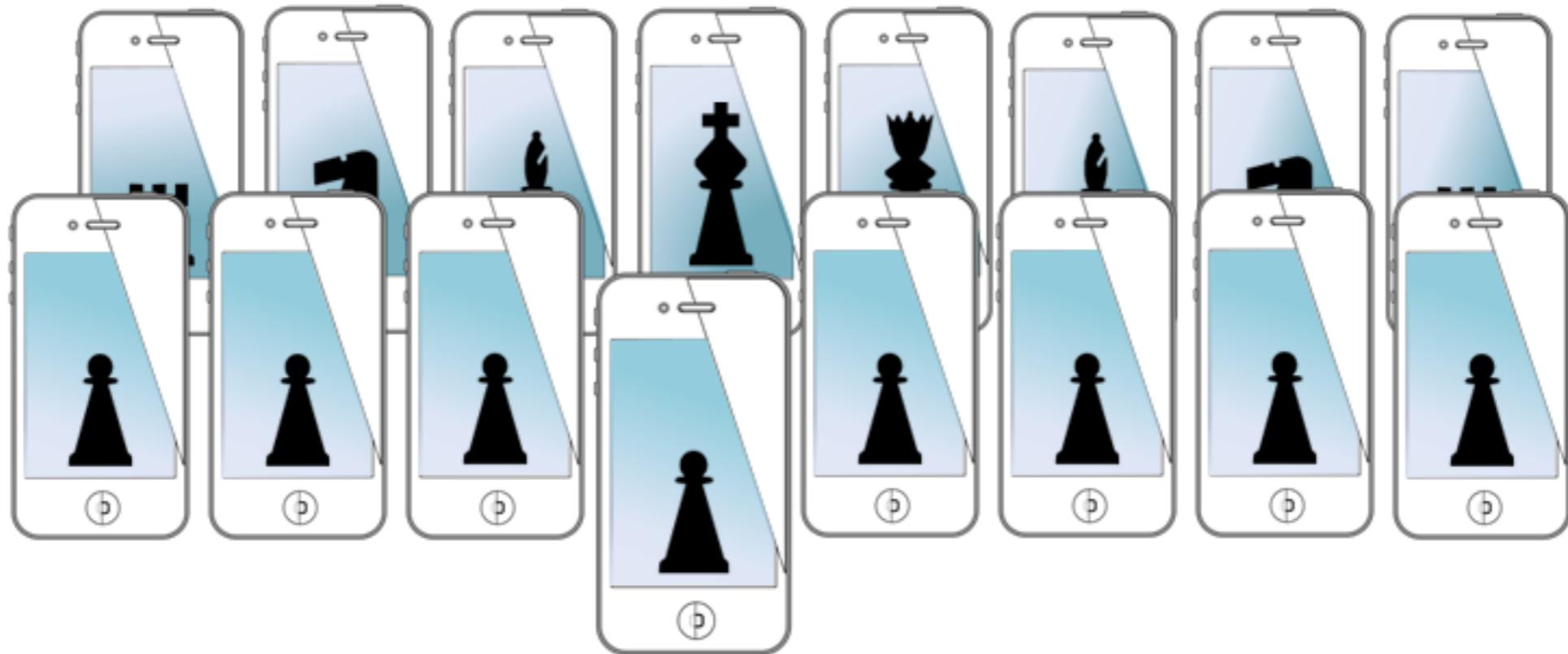
- ImageLab++



for next time...

- filters
- faces
- blood flow physiology
- then on to the next in-class assignment!!
 - computer vision with OpenCV
 - generic operations
 - tracking

MOBILE SENSING LEARNING



CSE5323 & 7323
Mobile Sensing and Learning

week 6: core image and image processing

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