

Xcode Tutorial for Computer Vision Apps

Siddhant Wadhwa
SCS'17

Quick demo:

How to set up an Xcode project

- Basic use of storyboard
- Embedding C++ code into Objective C files
- Using the built-in simulator
- Basic OpenCV image formatting

Web Resources:

<https://www.youtube.com/watch?v=lszcQrCNvMk>

https://github.com/slucy-cs-cmu-edu/Intro_iOS_OpenCV/blob/master/Intro_iOS_OpenCV/ViewController.mm

Drawing with OpenCV

(Avoid it at all costs)

```

- (cv::Mat)cvMatFromUIImage:(UIImage *)image
{
    CGColorSpaceRef colorSpace = CGImageGetColorSpace(image.CGImage);
    CGFloat cols = image.size.width;
    CGFloat rows = image.size.height;

    cv::Mat cvMat(rows, cols, CV_8UC4); // 8 bits per component, 4 channels (color channels + alpha)

    CGContextRef contextRef = CGContextCreate(cvMat.data, // Pointer to data
                                                cols, // Width of bitmap
                                                rows, // Height of bitmap
                                                8, // Bits per component
                                                cvMat.step[0], // Bytes per row
                                                colorSpace, // Colorspace
                                                kCGImageAlphaNoneSkipLast |
                                                kCGBitmapByteOrderDefault); // Bitmap info flags

    CGContextDrawImage(contextRef, CGRectMake(0, 0, cols, rows), image.CGImage);
    CGContextRelease(contextRef);

    return cvMat;
}

```

```

-(UIImage *)UIImageFromCVMat:(cv::Mat)cvMat
{
    NSData *data = [NSData dataWithBytes:cvMat.data length:cvMat.elemSize()*cvMat.total()];
    CGColorSpaceRef colorSpace;

    if (cvMat.elemSize() == 1) {
        colorSpace = CGColorSpaceCreateDeviceGray();
    } else {
        colorSpace = CGColorSpaceCreateDeviceRGB();
    }

    CGDataProviderRef provider = CGDataProviderCreateWithCFData((__bridge CFDataRef)data);

    // Creating CGImage from cv::Mat
    CGImageRef imageRef = CGImageCreate(cvMat.cols,                                     //width
                                       cvMat.rows,                                   //height
                                       8,                                             //bits per component
                                       8 * cvMat.elemSize(),                       //bits per pixel
                                       cvMat.step[0],                               //bytesPerRow
                                       colorSpace,                                  //colorspace
                                       kCGImageAlphaNone|kCGBitmapByteOrderDefault, // bitmap info
                                       provider,                                     //CGDataProviderRef
                                       NULL,                                         //decode
                                       false,                                        //should interpolate
                                       kCGRenderingIntentDefault                    //intent
                                       );

    // Getting UIImage from CGImage
    UIImage *finalImage = [UIImage imageWithCGImage:imageRef];
    CGImageRelease(imageRef);
    CGDataProviderRelease(provider);
    CGColorSpaceRelease(colorSpace);

    return finalImage;
}

```

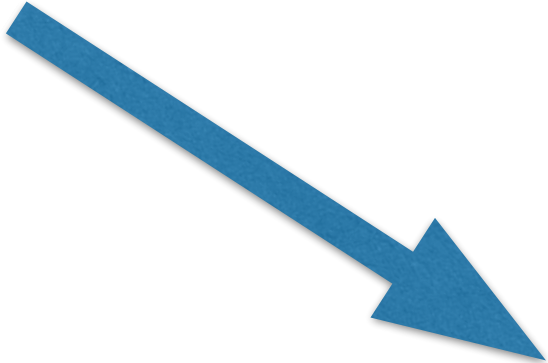
Find alternatives!

- UIImage \longleftrightarrow cv::Mat conversions

- cv::cvtColor



Use GPUImage library



Try manually forming the struct by prefixing metadata headers.

Accessing the built-in Cameras

Here's the OpenCV way

```
// 4. Initialize the camera parameters and start the camera (inside the App)
photoCamera_ = [[CvPhotoCamera alloc] initWithParentView:liveView_];
photoCamera_.delegate = self;

// This chooses whether we use the front or rear facing camera
photoCamera_.defaultAVCaptureDevicePosition = AVCaptureDevicePositionFront;

// This is used to set the image resolution
photoCamera_.defaultAVCaptureSessionPreset = AVCaptureSessionPreset640x480;

// This is used to determine the device orientation
photoCamera_.defaultAVCaptureVideoOrientation = AVCaptureVideoOrientationPortrait;

// This starts the camera capture
[photoCamera_ start];
```

```
//=====
// To be compliant with the CvPhotoCameraDelegate we need to implement these two methods
- (void)photoCamera:(CvPhotoCamera *)photoCamera capturedImage:(UIImage *)image
{
    [photoCamera_ stop];
    resultView_.hidden = false; // Turn the hidden view on

    // You can apply your OpenCV code HERE!!!!
    // If you want, you can ignore the rest of the code base here, and simply place
    // your OpenCV code here to process images.
    cv::Mat cvImage; UIImageToMat(image, cvImage);
    cv::Mat gray; cv::cvtColor(cvImage, gray, CV_RGBA2GRAY); // Convert to grayscale
    cv::GaussianBlur(gray, gray, cv::Size(5,5), 1.2, 1.2); // Apply Gaussian blur
    cv::Mat edges; cv::Canny(gray, edges, 0, 50); // Estimate edge map using Canny edge detector
    UIImage *resImage = MatToUIImage(edges);

    // Special part to ensure the image is rotated properly when the image is converted back
    resultView_.image = [UIImage initWithCGImage:[resImage CGImage]
                                     scale:1.0
                                     orientation: UIImageOrientationLeftMirrored];
    [takephotoButton_ setHidden:true]; [goliveButton_ setHidden:false]; // Switch visibility of buttons
}
- (void)photoCameraCancel:(CvPhotoCamera *)photoCamera
{
}
..
```

Taken from :

https://github.com/slucy-cs-cmu-edu/Intro_iOS_Camera/blob/master/Intro_iOS_Camera/ViewController.mm

But the OpenCV cvPhotoCameraDelegate protocol limits you to 30fps, in order to capture frames at up to 120fps (iPad air 2) or 240fps (iphone 6/6+) you could :

1) Create custom capture sessions as defined in the AVFoundation framework pre-installed on all iOS devices, here's a link to documentation that really helped me :

https://developer.apple.com/library/mac/documentation/AudioVideo/Conceptual/AVFoundationPG/Articles/04_MediaCapture.html

2) Or easier : use a wrapper (MyVideoCamera.h/.mm) written by a student of 16-423 last semester:

<https://github.com/wwd7086/CorrelationTrackIOS/tree/master/MosseTracker>

Quick demo:

Time Profiling your app to discover memory leaks/inefficient sections of the code

- How to use the built in time profiler with Xcode.

Web Resources:

<https://www.youtube.com/watch?v=wQmflj8jOvg>

Other tips and tricks

Dont draw ON images, add layers onto views

Minimizes memory accesses and writes

```
// add layer which bounding box is drawn on
-(void)addOverlayRect{
    self->rectOverlay=[CAShapeLayer layer];
    self->rectOverlay.fillColor=[UIColor colorWithRed:0.1 green:0.1 blue:0.1 alpha:0.1].CGColor;
    self->rectOverlay.lineWidth = 2.0f;
    self->rectOverlay.lineCap = kCALineCapRound;
    self->rectOverlay.strokeColor = [[UIColor redColor] CGColor];
    [self.view.layer addSublayer:self->rectOverlay];
}

// redraw the rectangle based on the drags' positions
-(void)refreshRect{
    std::array<CGPoint, 4> x;
    bp::Matrix33f I = bp::Matrix33f::Identity();
    if(hasTemplate){
        x = RectToPoints(templateBox, result.T.data());
    }
    else{
        x = RectToPoints(templateBox_scaled, I.data());
    }
    //NSLog(@"Entering refreshRect");
    UIBezierPath * rectPath=[UIBezierPath bezierPath];

    //NSLog(@"Point 1 = %d\n",x[0]);

    //Rectangle drawing
    [rectPath moveToPoint:x[0]];
    [rectPath addLineToPoint:x[1]];
    [rectPath addLineToPoint:x[2]];
    [rectPath addLineToPoint:x[3]];
    [rectPath addLineToPoint:x[0]];

    self->rectOverlay.path=rectPath.CGPath;
}
```

(Just an example of how you could do it)

Grand Central Dispatch

Prevent task from being executed in the background
Push tasks to the front of the queue on the main thread

Example:

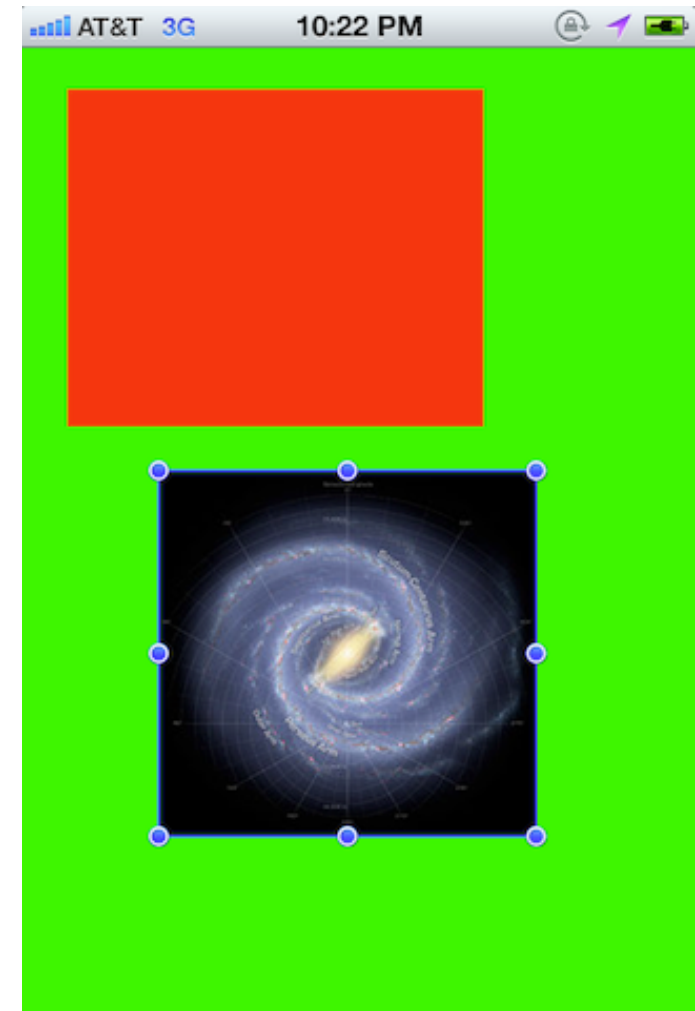
```
dispatch_sync(dispatch_get_main_queue(), ^{  
    // Add code body here  
});
```

spUserResizableView

Check it out if you need an elegant UI for drawing touch gesture-resizable bounding boxes

Link:

[https://github.com/
spoletto/
SPUserResizableView](https://github.com/spoletto/SPUserResizableView)



What I'm currently
working on

What I'm currently working on:

A demo for the Local Binary
Feature based image
alignment framework -
BitPlanes (developed under
Prof. Lucey's guidance)

Link to project(v0) report:

[https://github.com/
siddhantwadhwa/ios-driving-
assitant/blob/master/
final_project_report.pdf](https://github.com/siddhantwadhwa/ios-driving-assitant/blob/master/final_project_report.pdf)

16-423 Final Project Report : Local Binary Feature based image alignment app

[click to view project demo](#)

Siddhant Wadhwa
School of Computer Science
Carnegie Mellon University
siddhantwadhwa@cmu.edu

Abstract

This project serves as a technical prototype for image tracking based apps that I wish to pursue in the near future. It demonstrates the viability and performance of an implementation of the Lukas-Kanade algorithm using local binary features (instead of the usual pixel intensities), suited to perform optimally on mobile devices.

1. Introduction

Initially, I had det out to build a *Driving assistant* app that could be run on iOS devices mounted as dashboard cameras facing the front of the car, to offer safety warnings such as when the driver is not maintaining a safe distance with the car in front, or if the car is veering outside the lane. Exploring the various image alignment techniques, I realized that most conventional image representations are ill-suited to execution on mobile devices. So instead, I decided to pivot to working on this technical challenge of implementing a fast, mobile-friendly and efficient image tracking application.

I started working with Hatem Alismail, of the Robotics Institute at Carnegie Mellon University, to build a prototype application that made use of **BitPlanes**^[1], a framework that Hatem was authoring (along with Brett Browning and Simon Lucey, also of the Robotics Institute). Bitplanes was just the right fit for my driving assistant app-like ideas: a fast and robust image alignment framework well suited to the constraints of mobile devices.

2. Background

As described in "Bit-Planes: Dense Subpixel Alignment of Binary Descriptors."^[1], the bitplanes framework uses a descriptor called *bit-planes*, which is an adaptation of the simplest binary descriptor : LBP^[2] descriptor (explained in Figure 1). The bit-planes descriptor is designed to work

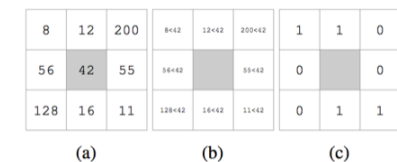


Figure 1. ^[1]The LBP descriptor is obtained by performing pixel comparisons in a fixed order and converting the binary string to a decimal value. In (a) the center pixel is in a 3 3 neighborhood is highlighted, and compared to its neighbors as shown in (b). Finally, the descriptor is obtained by combining the results of each comparison in (c) into a single scalar descriptor. *Figure taken from [1]*

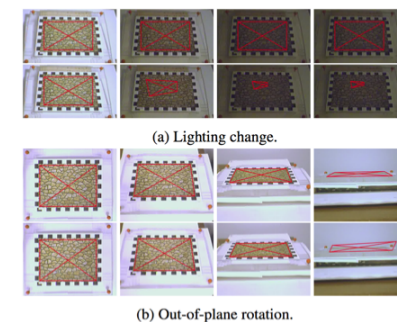


Figure 2. ^[1]Tracking results using the Bricks dataset [3]. The top row of each figure shows the performance of bit-planes, while the bottom row shows classical intensity-based LK. *Figure taken from [1]*

with the Lukas Kanade tracking algorithm to minimize the least squares distance between the template and the input image.

Thanks!

Link to demo 1 code and this presentation:

[https://github.com/siddhantwadhwa/
iOS_computer_vision_tutorial](https://github.com/siddhantwadhwa/iOS_computer_vision_tutorial)

