# HOWTO: Setup XCode 6.1 to work with OpenCV libraries

## Overview

This post demonstrates how to setup your XCode IDE to work with openCV libraries on Yosemite. Initially XCode can be an intimidating environment, but past all the windows and complex sections, it boasts a number of beneficial features that aid and simplify programming – which for newcomers to openCV are a must. Features such as autocomplete and integrated compiler are extremely useful in simplifying compilation, speeding up programming time and clarifying how pieces of code fit together.

**WARNING: This guide assumes you have compiled and installed openCV**
It not, see this guide.

## How To

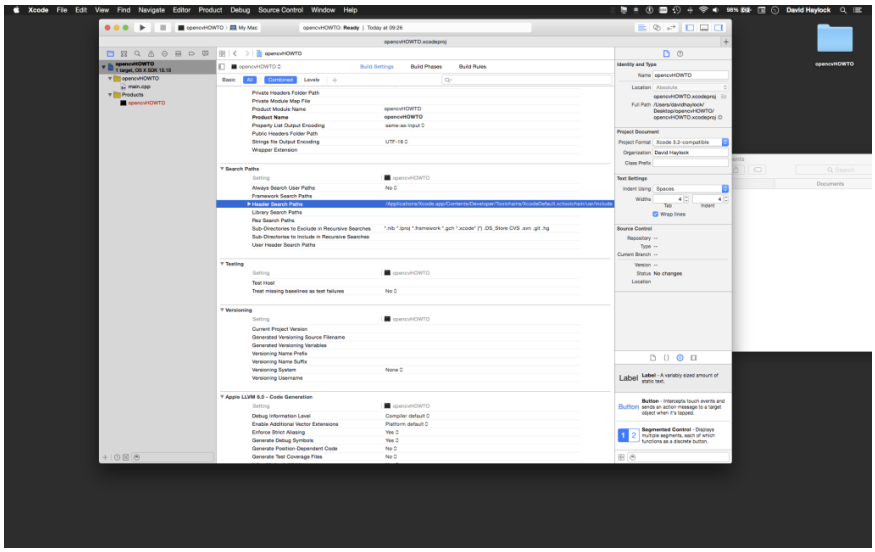
Let's begin.

- Open XCode
- Create New Project
- Under OSX Application > Choose Command Line Tool
- Give your application a name then Save

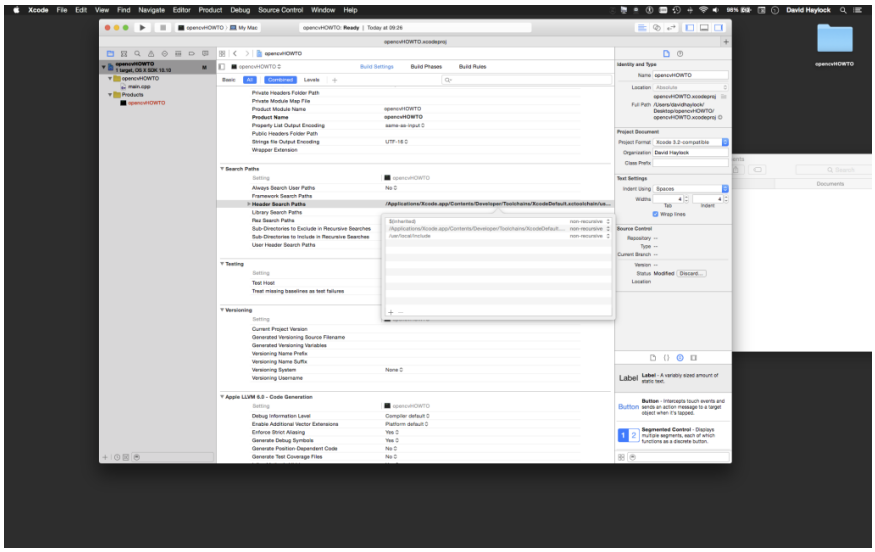
### Assigning the Search Paths

To use the openCV libraries you will need to tell XCode where to find them.

- Click the XCode project file in your inspector (which is the blue icon in the furthest left hand tab). You should now have three tabs in the centre window. `Build Settings | Build Options | Build Rules`
- Click `Build Settings`
- Scroll down until you find ***Search Paths***



- Double Click the ***Header Search Paths*** option, then Click the plus icon
- Type in the following `/usr/local/include`



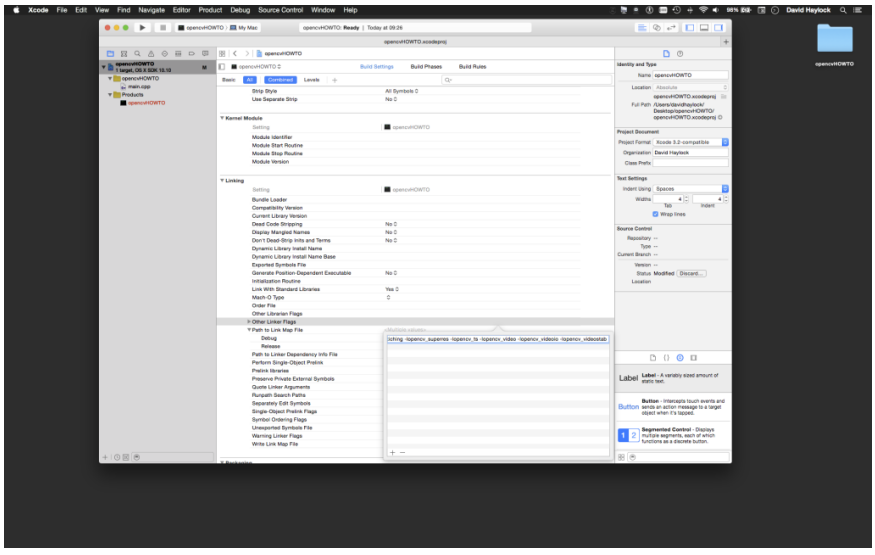
- Next Double Click the ***Library Search Paths*** option, then Click the plus icon
- Type in the following `/usr/local/lib`

## *Linking the Libraries*

Now we need to tell XCode which libraries we want to build against. Still in `Build Settings`.

- Scroll until you find ***Linking***
- Double Click ***Other Linker Flags*** and Click the plus icon
- Copy and Paste the following, then press enter

```
-lopencv_calib3d -lopencv_core -lopencv_features2d -lopencv_flann -lopencv_h
```

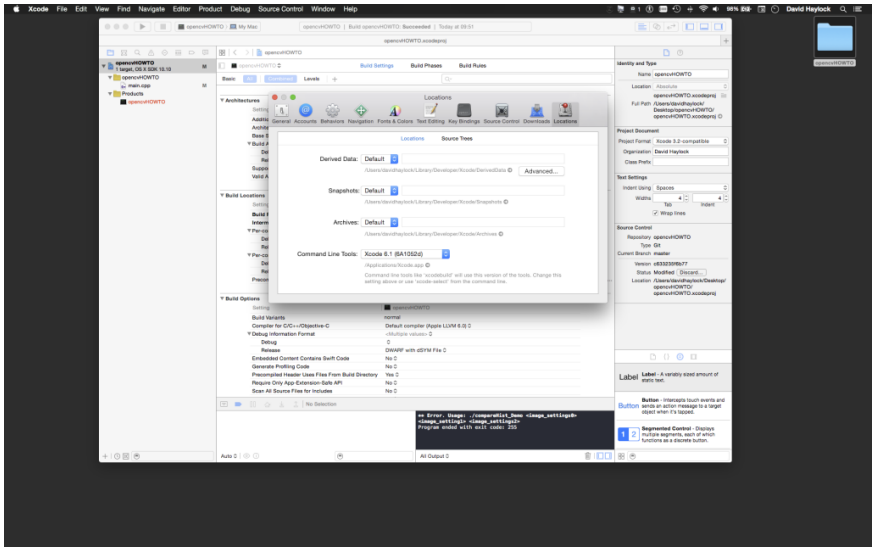


## *Setting the Build Path*

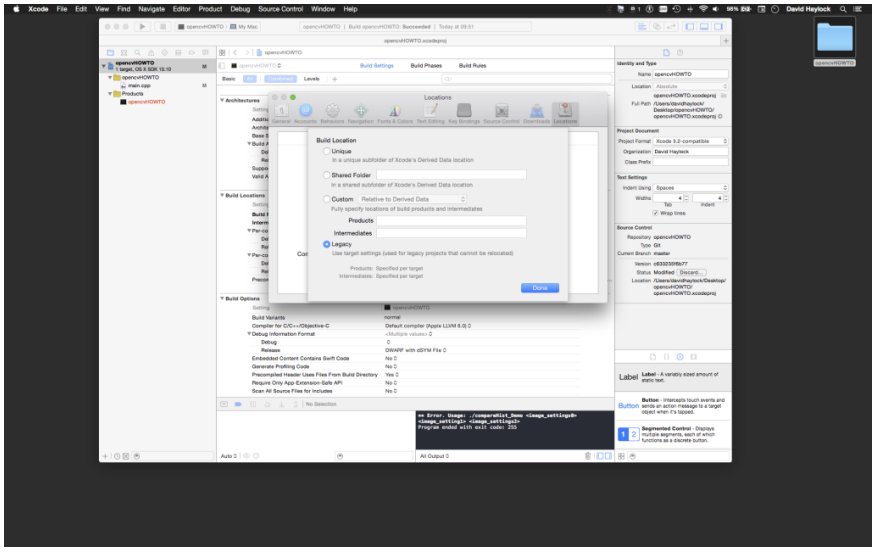
XCode automatically stores the executable file in a derived data folder, we want to change it so that the executable is stored in your project directory.

- Open XCode Preferences

- Select ***Locations Tab***



- Click Advanced
- Change the Location Button from ***Unique*** to ***Legacy***



## *Build your Application*

Now you are ready to build your application. In your main.cpp file, add the following code. This is a sample application from the openCV package.

```
#include <opencv2/imgcodecs.hpp>
#include <opencv2/videoio/videoio.hpp>
#include <opencv2/highgui/highgui.hpp>

#include <iostream>
#include <stdio.h>

using namespace cv;
using namespace std;

//hide the local functions in an anon namespace
namespace {
    void help(char** av) {
        cout << "The program captures frames from a video file, image sequence ((
```

```
        << "Usage:\n" << av[0] << " <video file, image sequence or device number>
        << "q,Q,esc -- quit" << endl
        << "space   -- save frame" << endl << endl
        << "\tTo capture from a camera pass the device number. To find the device
        << "\texample: " << av[0] << " 0" << endl
        << "\tYou may also pass a video file instead of a device number" << endl
        << "\texample: " << av[0] << " video.avi" << endl
        << "\tYou can also pass the path to an image sequence and OpenCV will tre
        << "\texample: " << av[0] << " right%%02d.jpg" << endl;
    }

    int process(VideoCapture& capture) {
        int n = 0;
        char filename[200];
        string window_name = "video | q or esc to quit";
        cout << "press space to save a picture. q or esc to quit" << endl;
        namedWindow(window_name, WINDOW_KEEPRATIO); //resizable window;
        Mat frame;

        for (;;) {
            capture >> frame;
            if (frame.empty())
                break;

            imshow(window_name, frame);
            char key = (char)waitKey(30); //delay N millis, usually long enough t

            switch (key) {
                case 'q':
                case 'Q':
                case 27: //escape key
                    return 0;
                case ' ': //Save an image
                    sprintf(filename,"filename%.3d.jpg",n++);
                    imwrite(filename,frame);
                    cout << "Saved " << filename << endl;
                    break;
                default:
                    break;
            }
        }
        return 0;
    }
}

int main(int ac, char** av) {

    if (ac != 2) {
        help(av);
        return 1;
    }
    std::string arg = av[1];
    VideoCapture capture(arg); //try to open string, this will attempt to open it
    if (!capture.isOpened()) //if this fails, try to open as a video camera, thro
        capture.open(atoi(arg.c_str()));
    if (!capture.isOpened()) {
        cerr << "Failed to open the video device, video file or image sequence!\n
        help(av);
        return 1;
    }
    return process(capture);
}
```

Then hold `CMD + B` (This builds and compiles the application)
Go to your project folder. There should be a new folder called Debug, inside is your executable file.
Open Terminal then navigate to the folder, then simply type `./<yourprojectname> 0`
This should open window and display the camera image.
And there you go!

NOVEMBER 19, 2014 BY DAVID HAYLOCK

Search …