July 31, 2015

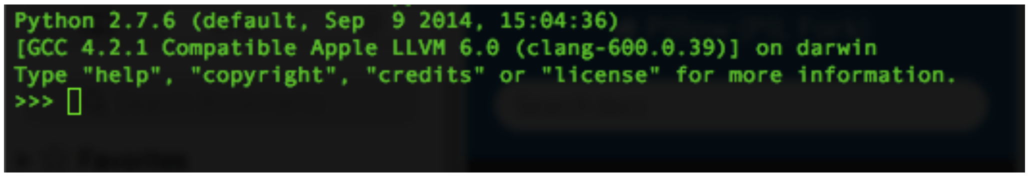
ImageWatch for Xcode Using LLDB

v. 0.3

This document describes how to install and use ImageWatch for **Xcode** using **LLDB**.

1. Python Setup and Documentation

Note that ImageWatch debugger is fully based on the Python API of LLDB. Such API support only the native Python on the OSX. If another version of Python (e.g., installed with Anaconda) is present in the system, it has to be uninstalled in order to be able to use ImageWatch. Check the python version with the terminal command **python**, which should produce the following output.



```
Python 2.7.6 (default, Sep  9 2014, 15:04:36)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.39)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

For documentation about LLDB refer to this link. For documentation about LLDB for Python refer to this link, and for the LLDB Python API this other link. Another interesting page is Fabian Guerra's introduction to LLDB Python Scripting (link).

This project is largely inspired by the GDB-ImageWatch project by Renato Garcia (link).

2. Install Python packages: Pillow and Opencv

- Install Pillow for the system python (note that LLDB is compatible only with the system python)

  brew install pillow

  (if not working try: **easy_install Pillow**)
- Check that pillow is correctly installed by typing in the python terminal

  import PIL
- Install OpenCV packages for python and check that is it possible to import it by typing in the python terminal

  import cv2

3. Allowing LLDB debugging in Python

- Add **~/.pythonrc** file on your machine.
- Add the following code to pythonrc (or use the provided file and rename it by adding a . at the beginning of the name)

  import sys

  sys.path = sys.path + ["/Applications/Xcode.app/Contents/SharedFrameworks/LLDB.framework/Resources/Python"]
- Add the following code to ~/.bash_profile

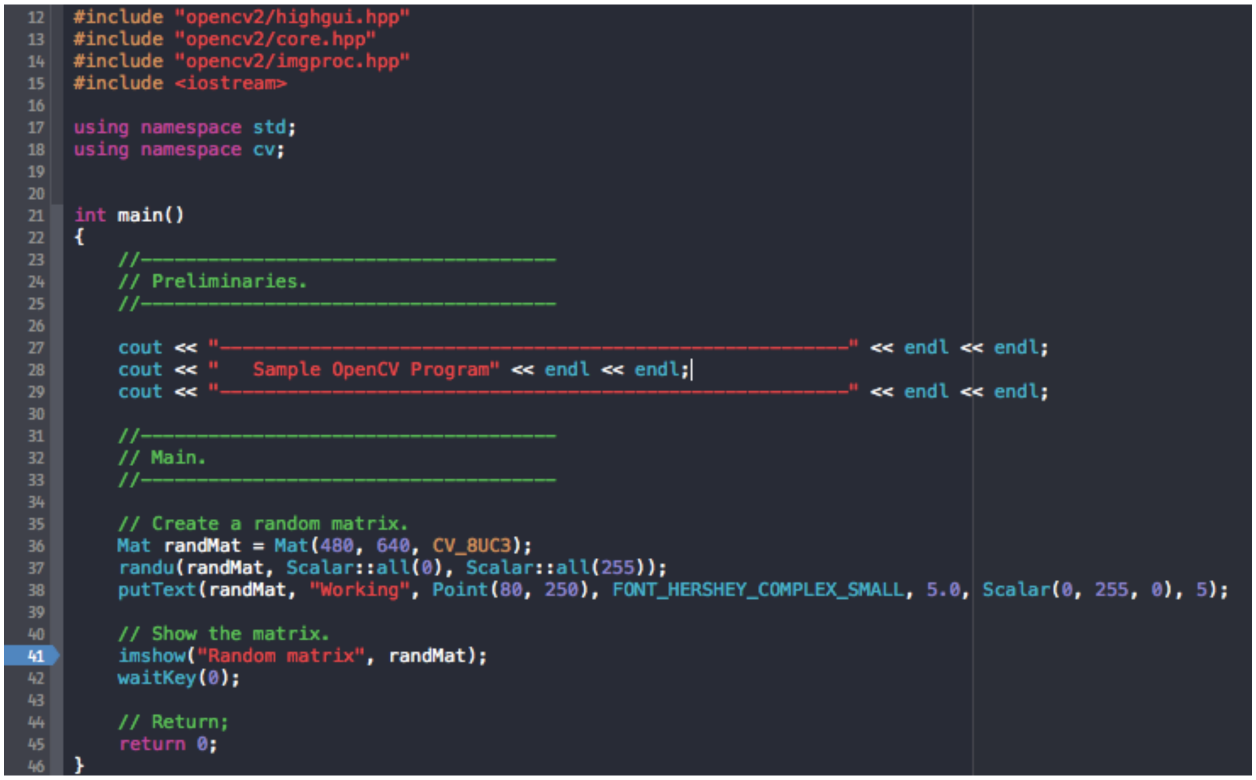
  export PYTHONSTARTUP="$HOME/.pythonrc"

4. Setup LLDB

- Create the folder **~/lldb**
- Place the provided python script **iw.py** in **~/lldb**
- Place the provided python script **iw_visualizer.py** in **~/lldb**
- Create the subfolder **~/lldb/iw_temp** (this is a folder containing the stored images)
- Add the the provided **lldbinit** file as **~/.lldbinit** (renaming by adding a . at the beginning of the file). This file is imported by default by lldb and it loads the **iw.py** file in the command set of lldb.

5. Call ImageWatch from lldb

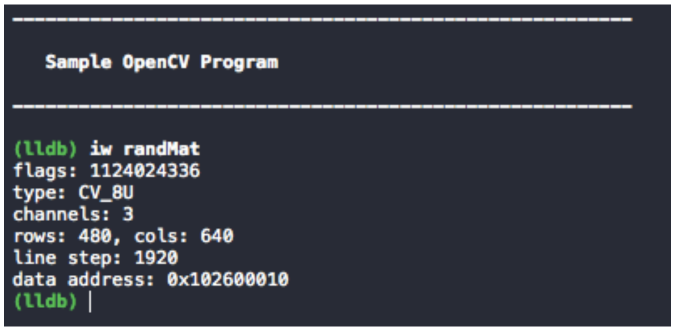
- In order to be able to call ImageWatch while debugging it is necessary to setup an Xcode **breakpoint** at the location of the code that is desired to be debugged. For example:



```
#include "opencv2/highgui.hpp"
#include "opencv2/core.hpp"
#include "opencv2/imgproc.hpp"
#include <iostream>

using namespace std;
using namespace cv;


int main()
{
    //-----------------------------------
    // Preliminaries.
    //-----------------------------------

    cout << "--------------------------------------------------------" << endl << endl;
    cout << "   Sample OpenCV Program" << endl << endl;
    cout << "--------------------------------------------------------" << endl << endl;

    //-----------------------------------
    // Main.
    //-----------------------------------

    // Create a random matrix.
    Mat randMat = Mat(480, 640, CV_8UC3);
    randu(randMat, Scalar::all(0), Scalar::all(255));
    putText(randMat, "Working", Point(80, 250), FONT_HERSHEY_COMPLEX_SMALL, 5.0, Scalar(0, 255, 0), 5);

    // Show the matrix.
    imshow("Random matrix", randMat);
    waitKey(0);

    // Return;
    return 0;
}
```

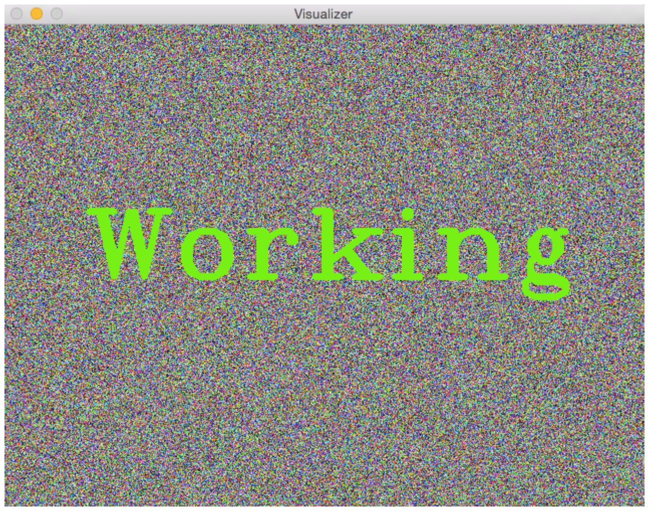
- Run the application **with debugging information**.
- Call ImageWatch when the function is stopped at the breakpoint with the command

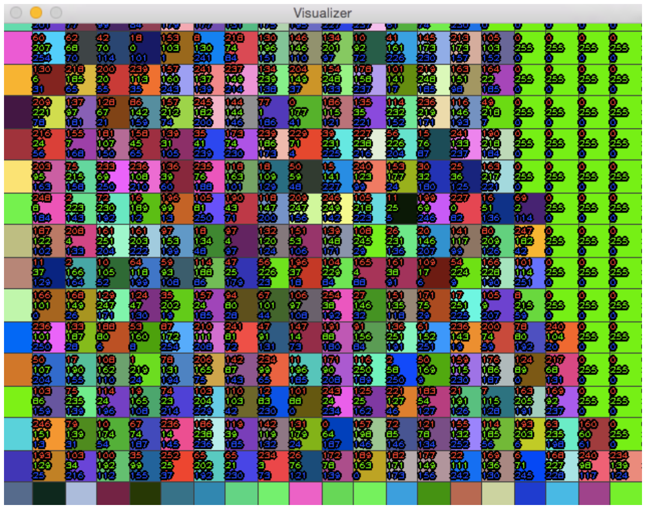
iw matName

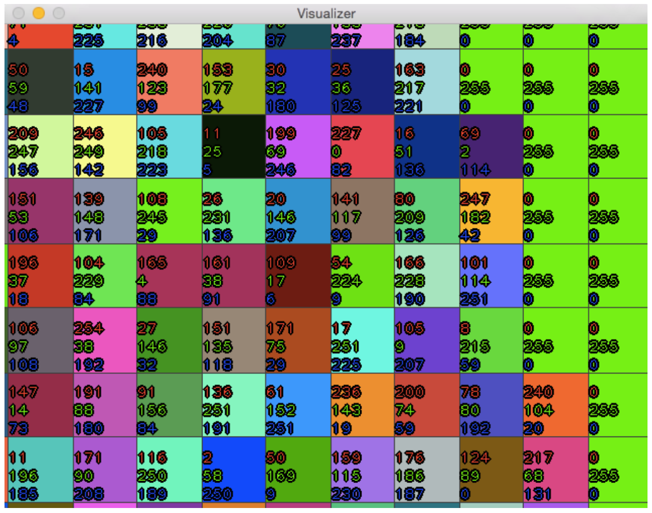


```
--------------------------------------------------------

   Sample OpenCV Program

--------------------------------------------------------

(lldb) iw randMat
flags: 1124024336
type: CV_8U
channels: 3
rows: 480, cols: 640
line step: 1920
data address: 0x102600010
(lldb) 
```

- The image get stored in the folder **~/lldb/iw_temp** with the name **matName_hh_mm_ss.png**
- The python script **iw_visualizer.py** is called, displaying the desired image. **ZoomIn** is implemented with the mouse l**eft click** and **ZoomOut** with the **right click**. If enough zoom is provided, RGB values are displayed with the pixels. In the following figure, a test image is shown at different zoom levels.







- Also, a text for importing the image in Matlab and removing images in the temp folder is copied in the clipboard and can be just pasted in Matlab to have an image with the same name of the original Mat.

6. Future steps

- LLDB
    - Extend to different types of cv::Mat, including float and double values
    - Extend to vectors of cv::Point, vectors of int, float, …
    - Extend to vectors of cv::Mat
    - Dump all the quantities in the current system state.
- Visualization
    - Have the possibility to maximize the contrast in the visualized images.
    - Display the current location of the mouse pointer in the images.
    - Extend to FMG
    - Include it in a more evolved GUI
    - Add possibility to draw lines, rectangles, and use them to get statistics on the images (min, max, median values and histogram)
    - Show preview of all the quantities that are available in the folder, thumbnails
    - Link views
- Longer term
    - Implement a quick look for cv::Mat data type