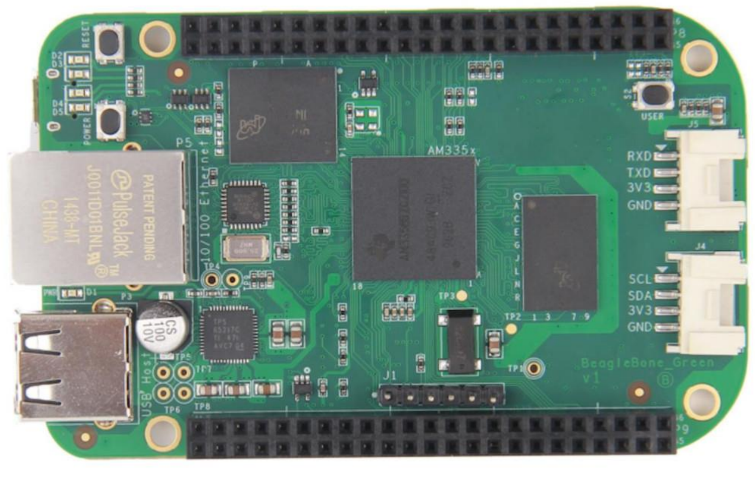
Capturing and Processing Webcam Video with the BeagleBone Green and OpenCV



By : (John) Sang Hyo Kim, Jasdeep Jassal, Adrian Li, Samuel Kim

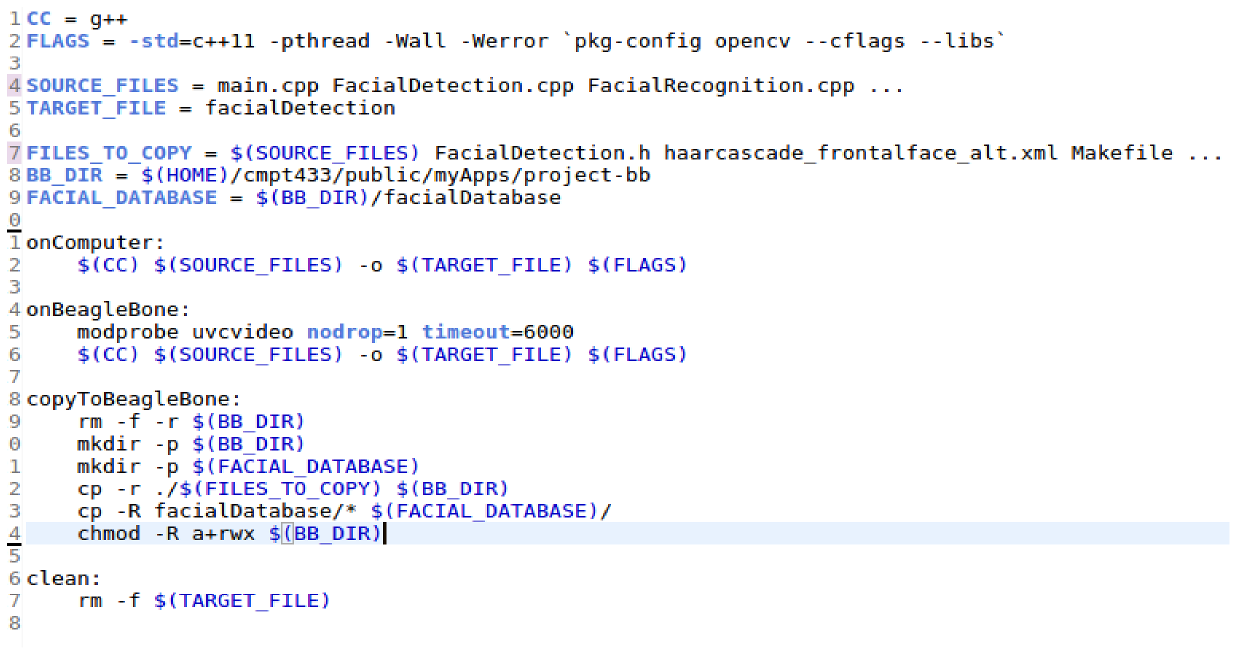
1. Plug your webcam into the BeagleBone’s USB port.
2. Verify that the webcam is connected by typing `ls /dev/video*` in the root directory of the BeagleBone. You should see `/dev/video0` if the webcam is connected successfully.

   Troubleshooting :

   If the webcam is not being detected, try :

   - Replugging the webcam
   - Rebooting the BeagleBone
   - Verify that the USB port on the BeagleBone is working properly by plugging in a different USB device
   - Unplug any other extra peripherals connected to the BeagleBone that might take up power

3. For this tutorial, it is assumed that you are not importing any extra libraries for the project. This is important because the BeagleBone Green comes with OpenCV 2.4.9 pre-installed. Therefore, the code can be compiled natively on the BeagleBone, by copying all project files to the BB and compiling on the BB. Here is a sample makefile :



```
CC = g++
FLAGS = -std=c++11 -pthread -Wall -Werror `pkg-config opencv --cflags --libs`

SOURCE_FILES = main.cpp FacialDetection.cpp FacialRecognition.cpp ...
TARGET_FILE = facialDetection

FILES_TO_COPY = $(SOURCE_FILES) FacialDetection.h haarcascade_frontalface_alt.xml Makefile ...
BB_DIR = $(HOME)/cmpt433/public/myApps/project-bb
FACIAL_DATABASE = $(BB_DIR)/facialDatabase

onComputer:
	$(CC) $(SOURCE_FILES) -o $(TARGET_FILE) $(FLAGS)

onBeagleBone:
	modprobe uvcvideo nodrop=1 timeout=6000
	$(CC) $(SOURCE_FILES) -o $(TARGET_FILE) $(FLAGS)

copyToBeagleBone:
	rm -f -r $(BB_DIR)
	mkdir -p $(BB_DIR)
	mkdir -p $(FACIAL_DATABASE)
	cp -r ./$(FILES_TO_COPY) $(BB_DIR)
	cp -R facialDatabase/* $(FACIAL_DATABASE)/
	chmod -R a+rwx $(BB_DIR)

clean:
	rm -f $(TARGET_FILE)

```

Note :

- note the chmod command, as the BeagleBone requires permissions to access and execute files after they have been copied. The -R flag recursively applies the chmod to all files within the given directory.
- you can compile on your host PC as well as the BeagleBone using the same compile command. For testing purposes, you might want to compile for the host PC and connect the webcam to the host PC.

4. **Optional** – Installing OpenCV and related dependencies to compile code for host PC.

1) Install dependencies

```
sudo apt-get -y install libopencv-dev build-essential cmake git libgtk2.0-dev
pkg-config python-dev python-numpy libdc1394-22 libdc1394-22-dev libjpeg-dev
libpng12-dev libtiff4-dev libjasper-dev libavcodec-dev libavformat-dev
libswscale-dev libxine-dev libgstreamer0.10-dev libgstreamer-plugins-
base0.10-dev libv4l-dev libtbb-dev libqt4-dev libfaac-dev libmp3lame-dev
libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev
libxvidcore-dev x264 v4l-utils unzip
```

2) Download OpenCV 3.0.0

```
mkdir opencv //create directory to store opencv files

cd opencv

wget https://github.com/Itseez/opencv/archive/3.0.0-alpha.zip -O opencv-
3.0.0-alpha.zip

unzip opencv-3.0.0-alpha.zip
```

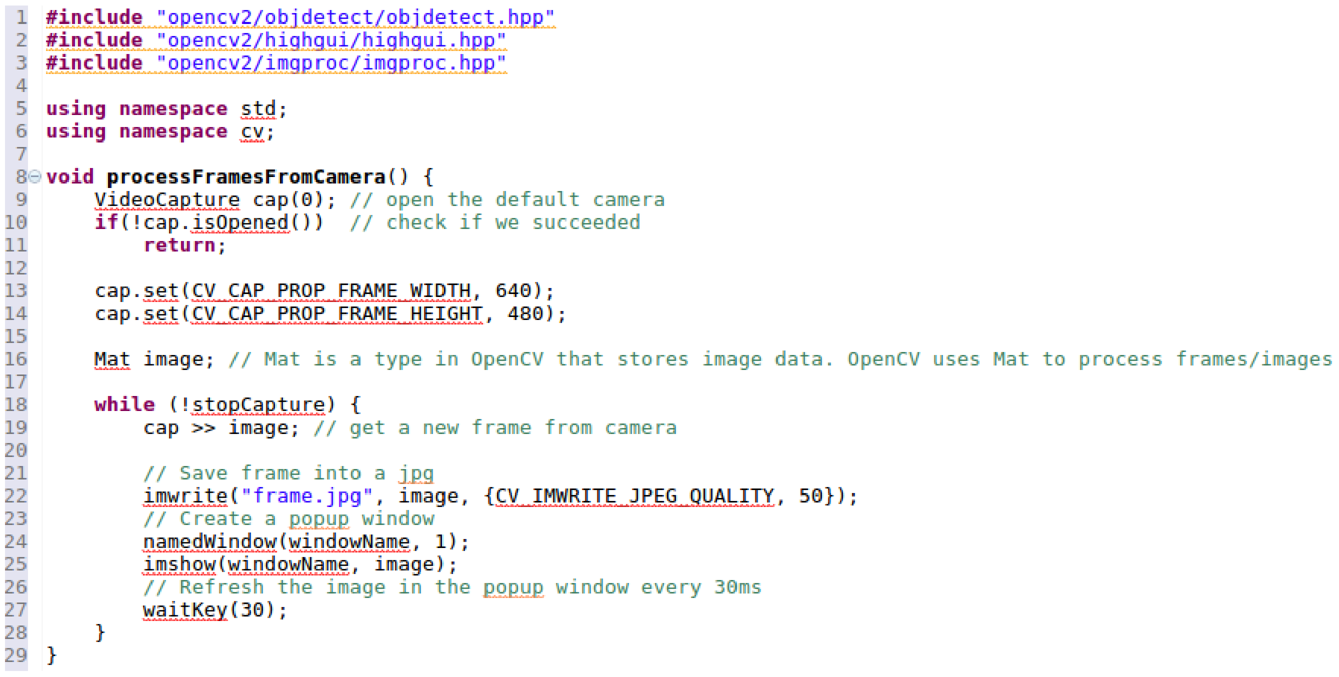
3) Install OpenCV

```
cd opencv-3.0.0-alpha

mkdir build

cd build

cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D
WITH_TBB=ON -D WITH_V4L=ON -D WITH_QT=ON -D WITH_OPENGL=ON ..

make -j $(nproc)

sudo make install
```

4) Finish the Installation

```
sudo /bin/bash -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'

sudo ldconfig
```

5. To capture frames from the camera using OpenCV, use the following sample code as a guide :



```
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"

using namespace std;
using namespace cv;

void processFramesFromCamera() {
    VideoCapture cap(0); // open the default camera
    if(!cap.isOpened())  // check if we succeeded
        return;

    cap.set(CV_CAP_PROP_FRAME_WIDTH, 640);
    cap.set(CV_CAP_PROP_FRAME_HEIGHT, 480);

    Mat image; // Mat is a type in OpenCV that stores image data. OpenCV uses Mat to process frames/images

    while (!stopCapture) {
        cap >> image; // get a new frame from camera

        // Save frame into a jpg
        imwrite("frame.jpg", image, {CV_IMWRITE_JPEG_QUALITY, 50});
        // Create a popup window
        namedWindow(windowName, 1);
        imshow(windowName, image);
        // Refresh the image in the popup window every 30ms
        waitKey(30);
    }
}
```

6. OpenCV features a vast array of functions that can be utilized to process images and frames. Using the Mat data type, you can process the image to detect a face. Details on how to do facial detection using OpenCV can be found here :

   http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html