## This tutorial is taken and modified from the following website

http://thinkrpi.wordpress.com/opencv-and-pi-camera-board/

The code necessary for this can be found in the folder along with this tutorial

STEP 1

It’s quite easy to install your new Pi Camera. Installation procedure is very well described on raspberrypi fondation website here : <http://www.raspberrypi.org/archives/3890>

Unfortunatly, cases are not today designed for the camera cable and your new toy.  Hard to say, but I did a hole in my nice white plastic case for cable path.  
Once your webcam is installed, test it with this command (show pictures till 10 seconds)

raspistill -t 10000

STEP 2

sudo apt-get install rpi-update

sudo rpi-update

do the above steps to upgrade the firmware

sudo apt-get update

MMAL library and raspivid/raspistill source code are found in Userland folder[(@GitHub, here)](https://github.com/raspberrypi/userland). First of all, we need to compile the whole package before doing anything else with OpenCV.

1. get source code (zip file)  here : <https://github.com/raspberrypi/userland>
2. unzip the file and copy the directory under **/opt/vc**
3. go to opt/vc and type : sed -i ‘s/if (DEFINED CMAKE\_TOOLCHAIN\_FILE)/if (NOT DEFINED CMAKE\_TOOLCHAIN\_FILE)/g’ makefiles/cmake/arm-linux.cmake
4. create a build directory and compile (it takes a while)

sudo mkdir build  
cd build  
sudo cmake -DCMAKE\_BUILD\_TYPE=Release ..  
sudo make  
sudo make install

Binary should be under **/opt/vc/bin**

Go to /opt/vc/bin and test one file typing : **./raspistill -t 3000**

At this stage, you should be able to modify this software to include OpenCV calls.  Congratulation ! Now, all nexts steps are piece of cakes….

STEP 3

As an example, we will create camcv, a soft strongly inspired from raspistill. It will allow us later to modify source code and play with OpenCV.

1. create a new folder in your home directory and copy all raspicam apps source code.

cd  
mkdir camcv  
cd camcv  
cp -r /opt/vc/userland/host\_applications/linux/apps/raspicam/\*  .  
mv RaspiStill.c camcv.c  
Perhaps a chmod is necessary (sudo chmod 777 camcv.c)

cp /gl\_scenes/\* .

2. remove then content of CMakeLists.txt and replace with :  
cmake\_minimum\_required(VERSION 2.8)  
project(camcv)  
SET(COMPILE\_DEFINITIONS -Werror)  
include\_directories(/opt/vc/userland/host\_applications/linux/libs/bcm\_host/include)  
include\_directories(/opt/vc/userland/interface/vcos)  
include\_directories(/opt/vc/userland)  
include\_directories(./gl\_scenes)  
include\_directories(.)  
include\_directories(/opt/vc/userland/interface/vcos/pthreads)  
include\_directories(/opt/vc/userland/interface/vmcs\_host/linux)  
include\_directories(/opt/vc/userland/interface/khronos/include)  
include\_directories(/opt/vc/userland/interface/khronos/common)  
add\_executable(camcv RaspiCamControl.c RaspiCLI.c RaspiPreview.c camcv.c RaspiTex.c RaspiTexUtil.c tga.c gl\_scenes/teapot.c gl\_scenes/models.c gl\_scenes/square.c gl\_scenes/mirror.c gl\_scenes/yuv.c gl\_scenes/sobel.c)  
target\_link\_libraries(camcv /opt/vc/lib/libmmal\_core.so /opt/vc/lib/libmmal\_util.so /opt/vc/lib/libmmal\_vc\_client.so /opt/vc/lib/libvcos.so /opt/vc/lib/libbcm\_host.so /opt/vc/lib/libGLESv2.so /opt/vc/lib/libEGL.so)

3. delete CMakeFiles directory if it exits  
4. Compile & test  
cmake .  
make   
./camcv -t 1000

STEP 4

Install both dev lib and python lib. My soft is C-written. Anyway, Python is still usefull for small scripts. I recommend to install it.

sudo apt-get update  
sudo apt-get install libopencv-dev  
sudo apt-get install python-opencv

The face recognition API is called libfacerec-0.04. All information and doc [can be found on this excellent website.](https://github.com/bytefish/libfacerec/blob/master/README.md)

Download the zip file here.<https://github.com/bytefish/libfacerec/zipball/v0.04>

I unzip it on my mac and transfer the whole directory on my rpi.

Go on the directory and just compile it using

cmake .  
make

Of course, OpenCV must be already installed in your Pi. To do this, just follow[the third step of my previous post “Magic Mirror”](http://thinkrpi.wordpress.com/2013/04/05/step-3-install-softwares-for-webcam-and-computer-vision/)

1. Modify your CMakeFiles.txt to include OpenCV library

cmake\_minimum\_required(VERSION 2.8)

project( camcv )

SET(COMPILE\_DEFINITIONS -Werror)

#OPENCV

find\_package( OpenCV REQUIRED )

#except if you’re pierre, change the folder where you installed libfacerec

#optional, only if you want to go till step 6 : face recognition

link\_directories( /home/pi/facedetect/bytefish-libfacerec-e1b143d )

include\_directories(/opt/vc/userland/host\_applications/linux/libs/bcm\_host/include)

include\_directories(/opt/vc/userland/interface/vcos)

include\_directories(/opt/vc/userland)

include\_directories(/opt/vc/userland/interface/vcos/pthreads)

include\_directories(/opt/vc/userland/interface/vmcs\_host/linux)

add\_executable(camcv RaspiCamControl.c RaspiCLI.c RaspiPreview.c camcv.c)

target\_link\_libraries(camcv /opt/vc/lib/libmmal\_core.so /opt/vc/lib/libmmal\_util.so /opt/vc/lib/libmmal\_vc\_client.so /opt/vc/lib/libvcos.so /opt/vc/lib/libbcm\_host.so /home/pi/facedetect/bytefish-libfacerec-e1b143d/libopencv\_facerec.a ${OpenCV\_LIBS})

2. Recompile. Should be ok. No change (of course!) since you didn’t modify your source code  
make  
./camcv

Actually, this was a pretty easy step !

STEP 5

In this step, we will modify our camcv code to

* remove the preview display provided by MMAL layer
* copy the camera buffer to a CvMat object
* link CvMat to a IplImage object and display it
* do some cleaning to remove all useless code (for us)

1. Download the camcv.c file here and note following comments/change :<http://raufast.org/download/camcv.c>

\*\*\*\*\*\* Actually this step is optional - I just took the camcv.c file, replaced it in place of the last one and went to next step.

Lines 61+ : add OpenCV Includes  
// \*\*\* PR : ADDED for OPENCV  
#include <cv.h>  
#include <highgui.h>

Line 156 : modify init values for test (size of file)  
// \*\*\* PR : modif for demo purpose : smaller image  
state->timeout = 1000; // 5s delay before take image  
state->width = 320;//2592;  
state->height = 200; //1944;

Line 230+ : in static void encoder\_buffer\_callback function. This is the core of the modification. This function is a callback, call to get the image in the queue. buffer contains the picture from the camera.

// \*\*\* PR : OPEN CV Stuff here !  
// create a CvMat empty structure, with size of the buffer.  
CvMat\* buf = cvCreateMat(1,buffer->length,CV\_8UC1);  
  
// copy buffer from cam to CvMat  
buf->data.ptr = buffer->data;

// decode image (interpret jpg)  
IplImage \*img = cvDecodeImage(buf, CV\_LOAD\_IMAGE\_COLOR);  
  
// we can save it !  
cvSaveImage(“foobar.bmp”, img,0);  
// or display it  
cvNamedWindow(“camcvWin”, CV\_WINDOW\_AUTOSIZE);  
cvShowImage(“camcvWin”, img );  
cvWaitKey(0);

Line 711/726/823 : we remove the native preview window (replaced by opencv window)  
// \*\*\* PR : we don’t want preview  
camera\_preview\_port = NULL;

// PR : we don’t want preview  
// status = connect\_ports(camera\_preview\_port, preview\_input\_port, &state.preview\_connection);

// mmal\_connection\_destroy(state.preview\_connection);

2. In the main function of the c file do the following changes

” else if (!raspipreview\_create(&state.preview\_parameters)) ” line in main to  
” else if ( (status = raspipreview\_create(&state.preview\_parameters)) != MMAL\_SUCCESS) ”

3. compile, run and check if your “foobar.bmp” file is created and if a nice window shows your picture taken by Pi Cam ! (press key to stop)

STEP 6

In this step, we will learn how to diplay a video from the camera board, using OpenCV display (and not the native preview GPU window).  
At the end of this step, you should be able to capture frames from your camera board, and use them directly using OpenCV ! Enjoy, creativity will be your only limit (and perhaps CPU a little bit)

This tuto is based on this file (<http://raufast.org/download/camcv_vid0.c>). Download it and read explanations below. (don’t forget to change the CMakeLists.txt). I found many technicals difficulties to write it,  thanx to Matthieu Tardivon (a brillant student) for his precious hint and help. I appreciate.

We start  from **raspivid.c**(the camera app)  but we need to remove  all useless lines, not linked with capturing frames.

We  delete  
- all lines related to the preview component,  
- all lines related to the encoder component.  
- all lines related to inline command parsing and picture info…

We change :  
- add the callback directly to the video\_port  (line 286)  
- create and attach the pool (to get/send message)  to the video port… (line 320)  
- change format encoding to ENCODING\_I420 in line (268) (instead of OPAQUE)

**Result** : the callback is called with the right FPS (around 30fps/s) during the capture. (FPS without OpenCV treatment).  
The **Buffer**variable contains the raw YUV I420 frame which needs to be converted in a RGB format to be used with OpenCV.  
To do it, understand the I420 format : read some cryptic pages like  <http://en.wikipedia.org/wiki/YUV> and <http://www.fourcc.org/yuv.php>

I wrote few lines to convert the picture in the callback function (line 141)  
- read the buffer and copy it by parts in 3 differents IplImage starting with Y component (full size), continue with U (half size) and finish with V component (half size)  
- merge the 3 IplImage (YUV) into one (line 170)  
- convert with the right color space (RGB)  (line 171)  
- and display it !

Warning ! : cvMerge, cvCvtColor are slow functions. If you want to increase FPS rate, you can stay with gray picture (the first Y channel). You’ll double your FPS doing that.  (parameter**graymode=1**, line 124). Line 118 set the**timeout** variable : it’s the period to capture (ms)

* 320×240 color : FPS = 27,2
* 320×240 gray : FPS = 28,6
* 640×480 color : FPS = 8
* 640×480 gray : FPS = 17

Copy the file in the pi taken from the above link.

again change

” else if (!raspipreview\_create(&state.preview\_parameters)) ” line in main to  
” else if ( (status = raspipreview\_create(&state.preview\_parameters)) != MMAL\_SUCCESS) ”

go to CMakeLists.txt file and change camcv.c to camcv\_vid0.c

Then run ./camcv and you will get video streaming. The quality is poor because it is around 2fps. reading the comments on this page to try to improve it. Will do it later.

http://thinkrpi.wordpress.com/2013/05/22/opencvpi-cam-step-6-video/