# Notes on Optipos implementation on Raspberry Pi

Login to the RPi using PUTTY.

## Install OpenCV

I followed this guide:

<http://robertcastle.com/2014/02/installing-opencv-on-a-raspberry-pi/>

Commands that were executed:

sudo apt-get update  
sudo apt-get upgrade

sudo apt-get -y install build-essential cmake cmake-curses-gui pkg-config libpng12-0 libpng12-dev libpng++-dev libpng3 libpnglite-dev zlib1g-dbg zlib1g zlib1g-dev pngtools libtiff4-dev libtiff4 libtiffxx0c2 libtiff-tools libeigen3-dev

sudo apt-get -y install libjpeg8 libjpeg8-dev libjpeg8-dbg libjpeg-progs ffmpeg libavcodec-dev libavcodec53 libavformat53 libavformat-dev libgstreamer0.10-0-dbg libgstreamer0.10-0 libgstreamer0.10-dev libxine1-ffmpeg libxine-dev libxine1-bin libunicap2 libunicap2-dev swig libv4l-0 libv4l-dev python-numpy libpython2.6 python-dev python2.6-dev libgtk2.0-dev

The guide had OpenCV version 2.4.8, but this has been changed to 2.4.9 below, since that was the version used on Windows.

wget http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.9/opencv-2.4.9.zip/download opencv-2.4.9.zip

Also, for some reason the wget command did not manage to save into the file opencv-2.4.9.zip. Therefore, the zip file ended up in a file called download. An extra step was added to rename that file:

mv download opencv-2.4.9.zip

unzip opencv-2.4.9.zip

After unzipping, the zip file should be removed to save space. This command was added:

rm opencv-2.4.9.zip

cd opencv-2.4.9

mkdir release

cd release

ccmake ../

If the above command does not work, something went wrong in installing cmake and ccmake. In that case, do the following, and then try the above command again:

sudo apt-get install cmake

sudo apt-get install cmake-curses-gui

Inside ccmake, press ‘c’ to configure. Here, different settings can be changed, but I did not change any. Press ‘c’ again to finish configuring, and then press ‘g’ to generate the Makefile

make

The build process takes many hours, but if it is aborted, it can be continued later by typing make again.

sudo make install

## Build Optipos

The next step is to build Optipos on the Raspberry Pi so that it can use a webcam, just as on Windows.

mkdir ~/Optipos

Transfer the Optipos source files to the RPi, putting them in the ~/Optipos directory. Then compile, using:

g++ Optipos.cpp OptiposLib.cpp -o Optipos -I/usr/local/include/opencv -DRPI -lopencv\_imgproc -lopencv\_ml -lopencv\_core -lopencv\_highgui -L/home/pi/opencv-2.4.9/release/lib

As an alternative, for the OptiposCam application, replace Optipos with OptiposCam in this command.

To allow the loader to find the shared libraries, the paths must be set. Create a file called /etc/ld.so.conf.d/opencv.conf:

sudo nano /etc/ld.so.conf.d/opencv.conf

Enter the following line in the text editor:

/home/pi/opencv-2.4.9/release/lib

Then run

sudo ldconfig -v

Add the path of these libraries to the LD\_LIBRARY\_PATH:

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/home/pi/opencv-2.4.9/release/lib

Run the program using this command, which should produce a printout of the command line options:

./Optipos –h

Create a file called map.txt in the Optipos directory, containing (for example):

// Map file for test

// Ceiling height [m]:

2.65

// Marker size [m]:

0.08

// Markers (one per row, with x [m], y [m], orientation [degrees], and type [1..36])

0.0 0.0 0.0 16

Note that this file must be Unix format, it is not possible to copy it from MS Windows.

Also create a file called settings.txt in the same directory, containing (for example):

// Settings file for RPi with raspicam

// Camera field of view [degrees]

40

// Camera offset x [m]

-0.021

// Camera offset y [m]

0

// Hue [0..720], saturation [0..100], and value [0..100] min and max for color 1

0 80 20 100 30 100

// Hue [0..720], saturation [0..100], and value [0..100] min and max for color 2

190 270 50 100 30 100

// Hue [0..720], saturation [0..100], and value [0..100] min and max for color 3

280 359 40 100 30 100

// Settings for the Canny edge detection: threshold1, threshold2 (normally = 3 \* threshold1), and Sobel kernel size

20 60 3

// Threshold for the findMarkers filtering

3

// Threshold for the selectAndRankMarkers filtering

0.2

Connect the webcam, and use the following command to see that it is recognized:

lsusb

(There should also be a device present under /dev/video0, which can be checked by ls /dev. If not, try rebooting the RPi with the webcam connected.)

Try the webcam using:

fswebcam image.jpg -S 2

Now try to run the program with the webcam connected:

./Optipos -c 1 –s settings.txt -m map.txt

(It is not totally predictable which camera to use; sometimes it works with –c 0 instead.)

Most likely, the program is running at around 1 Hz or so, but with a lag of about 5 s. The lag is due to the video buffer.

To optimize compilation, add the –O3 flag.

## Raspicam using V4L2 driver

An alternative approach to installing the libraries above is to instead install the V4L2 driver as described here: <http://www.linux-projects.org/modules/sections/index.php?op=viewarticle&artid=14>

<http://www.linux-projects.org/modules/sections/index.php?op=viewarticle&artid=16>. Then it becomes possible to use OpenCV directly, just as with a web cam.

wget <http://www.linux-projects.org/listing/uv4l_repo/lrkey.asc> && sudo apt-key add ./lrkey.asc

Add the following line to the file /etc/apt/sources.list using sudo nano /etc/apt/sources.list:

deb <http://www.linux-projects.org/listing/uv4l_repo/raspbian/> wheezy main

sudo apt-get update

sudo apt-get install uv4l uv4l-raspicam

sudo apt-get install uv4l-raspicam-extras

The driver can be tested with the following commands:

uv4l --driver raspicam --auto-video\_nr --width 640 --height 480 --encoding jpeg

dd if=/dev/video0 of=snapshot.jpeg bs=11M count=1

This should produce an image called snapshot.jpeg in the current working directory.

A utility program can be used to change settings of the driver. Get it by:

The program can be compiled with:

g++ Optipos.cpp -o Optipos -I/usr/local/include/opencv -I/usr/local/include/raspicam -DRPI -lopencv\_imgproc -lopencv\_ml -lopencv\_core -lopencv\_highgui -L/home/pi/opencv-2.4.9/release/lib -L/usr/lib/uv4l/uv4lext/armv6l -luv4lext -Wl,-rpath,'/usr/lib/uv4l/uv4lext/armv6l' –O3

Run it using:

./Optipos -c 1 -f 40 -x 0.021 -y 0.0 -m map.txt

(If it does not work, it is possible that the driver has to be started manually, using the command below. Try that, and then run it again using the above command.)

pkill uv4l

uv4l --driver raspicam --auto-video\_nr --encoding yuv420 --width 640 --height 480

To reduce lag, use the flag

--frame-buffers 2

## Profiling

To install gprof, do the following:

sudo apt-get install binutils

Compile the program with the added flag –pg for profiling, and optionally –g for debug mode, which gives more detailed information (but can have different behavior than the release mode compilation).

## OptiposCam on Raspberry Pi

For embedded application use, a smaller application called OptiposCam has been created, which does not include any visualization output. It can be compiled using:

g++ OptiposCam.cpp OptiposLib.cpp -o OptiposCam -I/usr/local/include/opencv -DRPI -lopencv\_imgproc -lopencv\_ml -lopencv\_core -lopencv\_highgui -L/home/pi/opencv-2.4.9/release/lib –O3

The application does not rely on anything else than the Raspicam command line program raspistill (so make sure that raspistill is installed). It can be triggered as follows:

(raspistill –width 640 –height 480 -t 999999999 -s -o /dev/shm/optiposimage.jpg &); ./OptiposCam `pgrep raspistill` settingsRaspicam.txt map.txt 100 ./Images &

Ensure that the ./Images directory exists prior to giving the command. (If the second last command line argument is set to 0, the last one can be set to any value, it will not be used.)

# Old stuff

## Add Raspicam API

(This section is obsolete, use approach in next section instead.)

This part is based on the following instruction which provides a raspicam api: <http://www.uco.es/investiga/grupos/ava/node/40>

First download the files from <http://sourceforge.net/projects/raspicam/files/>? and then unpack and compile the necessary software:

unzip raspicam-0.1.1.zip

cd raspicam-0.1.1

mkdir build

cd build

cmake ..

make

sudo make install

sudo ldconfig

Test the set up by running the following program:

raspicam\_cv\_test

Now compile the Optipos application using the following command:

g++ Optipos.cpp -o Optipos -I/usr/local/include/opencv -I/usr/local/include/raspicam -DRPI -lopencv\_imgproc -lopencv\_ml -lopencv\_core -lopencv\_highgui -lraspicam -lraspicam\_cv -lmmal -lmmal\_util -L/home/pi/opencv-2.4.9/release/lib -L/opt/vc/lib –O3

## Streaming video from RPi

Note: The below was an attempt to stream the processed images from the RPi. Unfortunately, I was unable to create the output video stream.

Install the VLC player for windows on the PC from http://get.videolan.org/vlc/2.1.5/win32/vlc-2.1.5-win32.exe.

To stream the video image from RPi, install the following package:

sudo apt-get install vlc

Verify that the streaming works by executing:

raspivid --width 640 --height 480 -o - -t 0 |cvlc -v stream:///dev/stdin --sout '#standard{access=http,mux=ts,dst=:8554}' :demux=h264

To view the video stream, open VLC, type CTRL+N, and enter the following address: http://193.10.66.47:8554/