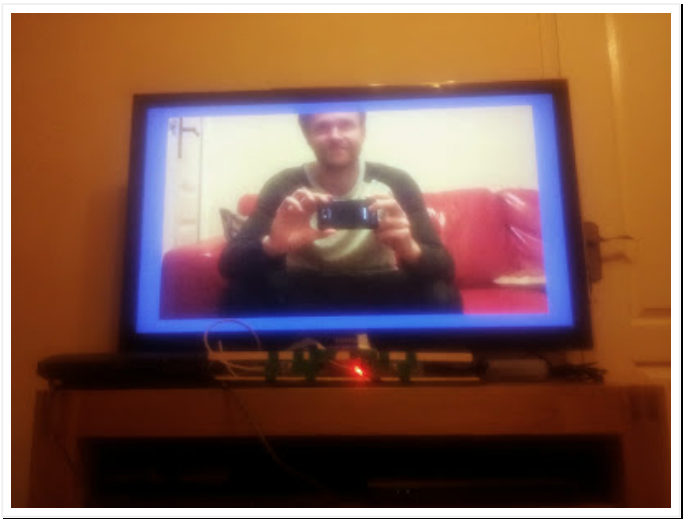
My Robot Blog

Saturday, 26 October 2013

An Efficient And Simple C++ API for the Rasperry Pi Camera Module

For the past few days I've been messing around with my new raspberry pi camera modules (see earlier blog posts for excessive details) and part of that has involved putting together a nice and easy to use api to access the camera in c++ and read its frames. This post is a guide to installation, an overview of the very simple api and a description of the sample application.



One word of caution - as with any tinkering there is always a chance something will go wrong and result in a dead pi. If this worries you, back up first. I didn't bother, but I didn't have anything on there I was worried about losing!

Installation

Make sure you're on a recent Raspberry Pi build, and have a working Camera!

I'm assuming at this point you've got a camera module and it's working. If you've not set it up yet you may need to update your raspberry pi (depends when you bought it). I won't go over this process as it's been described 100 times already, but here's a link to get you going just in case:

http://www.raspberrypi.org/archives/3890

Once all is up and running type:

```
raspivid -t 10000
```

That should show you the raspberry pi video feed on screen for 10 seconds.

Get CMake

If you haven't already got it, you'll need cmake for building just about anything:

```
sudo apt-get install cmake
```

Download and install the latest 'userland-master'

This is the bit of the raspberry pi OS that contains the code for the camera applications and the various libraries they use. At time of writing it

isn't supplied as part of the install, so you need to download, build and install it manually. To do so:

Download the latest userland-master.zip from here

Unzip it into your */opt/vc* directory. You should now have a folder called */opt/vc/userland-master* with various folders in it such as *"host_applications"* and *"interfaces"*.

Change to the */opt/vc/userland-master* folder, then build it with the following commands:

```
sudo mkdir build
cd build
sudo cmake -DCMAKE_BUILD_TYPE=Release ..
sudo make
sudo make install
```

Test everything worked by running raspivid again. You may see some different messages pop up (I got some harmless errors probably due to the build being so recent), but the crucial thing is that you still get the video feed on screen.

Download and build the PiCam API/Samples

The api and samples can all be downloaded here:

http://www.cheerfulprogrammer.com/downloads/picamtutorial/picamdemo.zip

Extract them into a folder in your home directory called 'picamdemo'. You should have a few cpp files in there, plus a make file and some shaders.

Change to the folder and build the application with:

```
cmake .
make
```

Then run the sample with

```
./picamdemo
```

If all goes well you should see some text like this:

```
Compiled vertex shader simplevertshader.glsl:
<some shader code here>

Compiled fragment shader simplefragshader.glsl:
<some shader code here>

mmal: mmal_vc_port_parameter_set: failed to set port parameter 64:0:ENOSYS
mmal: Function not implemented
Init camera output with 512/512
Creating pool with 3 buffers of size 1048576
Init camera output with 256/256
Creating pool with 3 buffers of size 262144
Init camera output with 128/128
Creating pool with 3 buffers of size 65536
Init camera output with 64/64
Creating pool with 3 buffers of size 16384
Camera successfully created
Running frame loop
```

And your tv should start flicking between various resolutions of the camera feed like this:

(Edit - I've had some reports of the blogger you-tube link not working. You can see the full video here on proper you tube: http://www.youtube.com/watch?v=9bWJBSNxeXk)

The API (and what it does!)

PiCam is designed to be very simple but also useful for image processing algorithms. Right now it lets you:

- Start up the camera with a given width, height and frame rate
- Specify a number of 'levels'. More on that later.
- Choose whether to automatically convert the camera feed to RGBA format

Basic Initialisation

All this is done just by calling *StartCamera* and passing in the right parameters. It returns a pointer to a CCamera object as follows:

```
CCamera* mycamera = StartCamera(512,512,30,1,true);
```

That's a 512x512 image at 30hz, with 1 level and rgba conversion enabled.

Reading

Once started you can extract frames from the camera by calling *ReadFrame* and passing in a buffer:

```
char mybuffer[512*512*4]
mycamera->ReadFrame(0,mybuffer,sizeof(mybuffer));
```

ReadFrame will return the number of bytes actually read, or -1 if there was an error. An error occurs either when there is no data available or your buffer is not large enough.

In addition to *ReadFrame* there are 2 functions: *BeginReadFrame* and *EndReadFrame*. These slightly more advanced versions are shown in the demo, and allow you to be more efficient by locking the actual camera buffer, using it, then releasing it. Internally *ReadFrame* is implemented using these functions.

Shutting down

Once done, call 'StopCamera'

Levels

In image processing it is often useful to have your data provided at different resolutions. Expensive operations need to be performed on low res images to run at a good frame rate, but you may still want higher res versions around for other operations or even just showing on screen. The PiCam api will do this for you automatically (for up to 3 additional levels). If we modify the *StartCamera* call to this:

```
CCamera* mycamera = StartCamera(512,512,30,4,true);
```

The system will automatically generate the main image plus an additional 3 down-sampled ones (at half res, quarter res and 1/8th res). These are then accessed by specifying a level other than 0 in the call to *ReadFrame* (or *BeginReadFrame*):

```
mycamera->ReadFrame(0,mybuffer,sizeof(mybuffer)); //get full res frame
mycamera->ReadFrame(1,mybuffer,sizeof(mybuffer)); //get half res frame
mycamera->ReadFrame(2,mybuffer,sizeof(mybuffer)); //get quarter res frame
mycamera->ReadFrame(3,mybuffer,sizeof(mybuffer)); //get 1/8th res frame
```

RGBA Conversions

For most purposes you'll want the data in a nice friendly RGBA format, however if you actually want the raw YUV data feed from the camera, specify false as the last parameter to StartCamera and no conversions will be done for you.

The demo application

The picamdemo application consists of the core camera code as these files:

- camera.h/camera.cpp
- cameracontrol.h/cameracontrol.cpp
- mmalincludes.h

A very simple opengl graphics api (which you are welcome to use/modify/change in any way you please):

- graphics.h/graphics.cpp

And the main demo app itself:

- picam.cpp

Which looks like this:

```
#include <stdio.h>
#include <unistd.h>
#include "camera.h"
#include "graphics.h"

#define MAIN_TEXTURE_WIDTH 512
#define MAIN_TEXTURE_HEIGHT 512

char tmpbuff[MAIN_TEXTURE_WIDTH*MAIN_TEXTURE_HEIGHT*4];

//entry point
int main(int argc, const char **argv)
{
    //should the camera convert frame data from yuv to argb automatically?
    bool do_argb_conversion = true;

    //how many detail levels (1 = just the capture res, >1 goes down by halves, 4 max)
    int num_levels = 4;

    //init graphics and the camera
    InitGraphics();
    CCamera* cam = StartCamera(MAIN_TEXTURE_WIDTH, MAIN_TEXTURE_HEIGHT,30,num_levels,do_argb_conversion);

    //create 4 textures of decreasing size
    GfxTexture textures[4];
    for(int texidx = 0; texidx < num_levels; texidx++)
```

```
        textures[texidx].Create(MAIN_TEXTURE_WIDTH >> texidx, MAIN_TEXTURE_HEIGHT >> texidx);

    printf("Running frame loop\n");
    for(int i = 0; i < 3000; i++)
    {
        //pick a level to read based on current frame (flicking through them every 30 frames)
        int texidx = (i / 30)%num_levels;

        //lock the chosen buffer, and copy it directly into the corresponding texture
        const void* frame_data; int frame_sz;
        if(cam->BeginReadFrame(texidx,frame_data,frame_sz))
        {
            if(do_argb_conversion)
            {
                //if doing argb conversion just copy data directly
                textures[texidx].SetPixels(frame_data);
            }
            else
            {
                //if not converting argb the data will be the wrong size so copy it in
                //via a temporary buffer just so we can observe something happening!
                memcpy(tmpbuff,frame_data,frame_sz);
                textures[texidx].SetPixels(tmpbuff);
            }
            cam->EndReadFrame(texidx);
        }

        //begin frame, draw the texture then end frame (the bit of maths just fits the image to the screen while maintaining as
        BeginFrame();
        float aspect_ratio = float(MAIN_TEXTURE_WIDTH)/float(MAIN_TEXTURE_HEIGHT);
        float screen_aspect_ratio = 1280.f/720.f;
        DrawTextureRect(&textures[texidx],-aspect_ratio/screen_aspect_ratio,-1.f,aspect_ratio/screen_aspect_ratio,1.f);
        EndFrame();
    }

    StopCamera();
}
```

That's the full code for exploiting all the features of the api. It is designed to loop through each detail level and render them in turn. At the top of the main function you will find a couple of variables to enable argb or change level count, and higher up you can see the frame size settings.

Questions? Problems? Comments?

I'm happy to answer any questions, hear any comments, and if you hit issues I'd like to fix them. Either comment on this blog or email me (wibble82@hotmail.com) with a sensible subject like 'pi cam problem' (so it doesn't go into the junk mail box!).

p.s. right at the end, here's a tiny shameless advert for my new venture - http://www.happyrobotgames.com/no-stick-shooter. If you like my writing, check out the dev blog for regular updates on my first proper indie title!

Posted by Chris Cummings at 22:32

73 comments:

chad 27 October 2013 at 11:09

http://www.youtube.com/v/9bWJBSNxeXk?version=3&f=user_uploads&c=google-webdrive-0&app=youtube_gdata

video does not play form me in blog or on youtube???

tried recent chrome and firefox

Reply

Chris Cummings 27 October 2013 at 13:30

Curious - the link works for me. Can you see it here: http://www.youtube.com/v/9bWJBSNxeXk

Reply

Chris Cummings 27 October 2013 at 13:34

Or here: http://www.youtube.com/watch?v=9bWJBSNxeXk

Reply

JBeale 31 October 2013 at 00:42

Followed instructions, but the make failed as follows:

[25%] Building CXX object CMakeFiles/picamdemo.dir/picam.cpp.o
In file included from /opt/vc/include/interface/vcos/vcos_assert.h:149:0,
from /opt/vc/include/interface/vcos/vcos.h:114,
from /opt/vc/include/interface/vmcs_host/vc_dispmanx.h:33,
from /opt/vc/include/bcm_host.h:46,
from /home/pi/picamdemo/mmalincludes.h:9,
from /home/pi/picamdemo/camera.h:3,
from /home/pi/picamdemo/picam.cpp:3:
/opt/vc/include/interface/vcos/vcos_types.h:38:33: fatal error: vcos_platform_types.h: No such file or directory
compilation terminated.
make[2]: *** [CMakeFiles/picamdemo.dir/picam.cpp.o] Error 1
make[1]: *** [CMakeFiles/picamdemo.dir/all] Error 2
make: *** [all] Error 2

Reply

Replies

JBeale 31 October 2013 at 05:38

Nevermind- problem was my /opt/vc/userland should have been named /opt/vc/userland-master.

Reply

Chris Cummings 31 October 2013 at 08:38

Yup that's the one. For anyone else hitting the issue, the alternative to changing the folder name is to change the references to userland-master in CMakeLists.txt to point to your own folder.

Reply

Anonymous 5 November 2013 at 01:25

Nice. How and where could I start adding OpenCV functions to do processing on a single still picture?

Reply

Chris Cummings 5 November 2013 at 08:10

Hi there

I was planning on releasing a new, better and faster version of the api along with some better samples and an example of hooking it up to opencv. This'll show an actual program using it that can be copied, so if you can wait till the end of the week it'll be detailed in full. In the meantime, you could piece it together from this web site: http://thinkrpi.wordpress.com/2013/05/22/opencv-and-camera-board-csi/

It is referring to getting it working with the raspivid/raspistill, which you can largely ignore. However step4 of that guide details how to install opencv, link it into an application and convert a block of memory into a usable opencv matrix. You would need to adapat that step to read the data from my camera api (returned in BeginReadFrame) and copy that into an opencv matrix.

If you can hold off for a few days, my updated api and samples should be up by the end of the week.

Reply

Replies

Anonymous 5 November 2013 at 16:22

Thanks so much, that will be really helpful. I did initially use the link you provided but the coding is done in C, and I had many issues relating it to my C++ processing code. I am using the Pi and Pi Camera for a project and my deadlines are coming up :(I have a good

idea how to use opencv to process an image (I could use the same steps provided from the other link which creates a matrix and stores the camera buffer therein). I am currently working on it, if you would be soo kind as to give me a little head start, could you please tell me which file/files I edit such that it simply takes one picture (not video) that i can use to process. Another thing, the above code displays an upside down picture, how can I fix that? Thank you in advance!

Chris Cummings 5 November 2013 at 17:24

Ok - well what I'll do is get on with the polishing off this next api. Send me your email (I'm at wibble82@hotmail.com) and I'll send over an early version of the code when its done, before I try and do the full write up. Should be either today or tomorrow.

Unknown 9 December 2013 at 03:21

hi,I have the same question with opencv in c++.And I have sent you an email,please check for that.I really appreciate for your help.By the way,my email is m0121010@stmail.cgu.edu.tw.Thanks a lot.

Brett Maeyer 4 September 2014 at 12:54

Hi
I have been struggling with implementing you API with OpenCV. Any information regarding how to do this with C++ would be very helpful. The above mentioned website did not help very much sadly.
Thanks

Reply

Anonymous 13 November 2013 at 02:39

Hi here
I followed the instructions,but failed there
pi@raspberrypi ~/picamdemo $ make
-- Configuring done
-- Generating done
-- Build files have been written to: /home/pi/picamdemo
Scanning dependencies of target picamdemo
make[2]: Warning: File `picam.cpp' has modification time 2.2e+06 s in the future
[25%] Building CXX object CMakeFiles/picamdemo.dir/picam.cpp.o
[50%] Building CXX object CMakeFiles/picamdemo.dir/camera.cpp.o
[75%] Building CXX object CMakeFiles/picamdemo.dir/cameracontrol.cpp.o
/home/pi/picamdemo/cameracontrol.cpp: In function ‘int raspicamcontrol_set_shutter_speed(MMAL_COMPONENT_T*, int)’:
/home/pi/picamdemo/cameracontrol.cpp:550:78: error: ‘MMAL_PARAMETER_SHUTTER_SPEED’ was not declared in this scope
make[2]: *** [CMakeFiles/picamdemo.dir/cameracontrol.cpp.o] Error 1
make[1]: *** [CMakeFiles/picamdemo.dir/all] Error 2

Reply

Chris Cummings 13 November 2013 at 08:09

Oh dear - I wonder if the latest userland has removed that MMAL parameter. I'll take a look tonight.

Reply

Chris Cummings 14 November 2013 at 09:03

Hi - apologies - I've been really busy lately and not been able to work on the api for a few days. I'll hopefully get back to it soon. You could probably fix it by commenting out the code in the function in cameracontrol.cpp, or even better comparing it to the latest code in the raspicameracontrol.cpp in the camera demos in userland. If not, I'll get to releasing the next version of the api next week hopefully.

Reply

Jiman 20 November 2013 at 14:52

check if you have the latest version of the "interface library": https://github.com/raspberrypi/userland/blob/master/interface/mmal/mmal_parameters_camera.h

Reply

Uwe Reinersmann 21 November 2013 at 18:45

Hi Cris,
thank you for the great API. I have a Question:
It is possible to make a Object for Qt4? Example a Containter for publish the video and gui elements on the same desktop?

Reply

Anonymous 27 November 2013 at 05:29

Chris, great looking API, that seems to accomplish most, if not all, of what I need, with a minimum of code. Awesome! While you're continuing to cleanup the API, can you tell me what components I need to link with my own app in order to utilize your API? I just need to get frames from the camera, so I can push them into an SDL_Surface, where I'll do all my display work in my current app. I'm guessting camera.cpp/h and cameracontrol.cpp/h? What about mmalincludes.h? Thanks.

Reply

Chris Cummings 28 November 2013 at 13:37

Hi there. If you look in the cmakelists.txt file you can see the include paths and libraries. You won't need the opencv libs included (in fact, they shouldn't be there anyway!), but you'll probably need the rest - include directories and linked libraries. In addition to the correct include paths (to userland) and libraries, you'll need camera.cpp/h, cameracontrol.cpp/h and mmalincludes.h.

Reply

kylemallory 28 November 2013 at 16:35

Thanks Chris. I ended up keeping all the opencv libs. I'll probably need a few of them before i'm done anyway. I got it to compile/link, and its working great. A couple of questions:

1) I can capture 1920x1080, but the image is black. Lower resolutions work, fine. Higher resolution fail with an mmal error:
mmal: mmal_vc_port_parameter_set: failed to set port parameter 64:0:ENOSYS
mmal: Function not implemented
Init camera output with 2592/1944
Creating pool with 3 buffers of size 20238336
mmal: mmal_vc_port_enable: failed to enable port vc.ril.resize:out:0(RGBA): ENOMEM
mmal: mmal_port_enable: failed to enable port vc.ril.resize:out:0(RGBA)(0x16a2970) (ENOMEM)
Failed to set video buffer callback
Failed to initialize output 0
Camera init failed

ENOMEM tells me its running out of memory. How do I capture higher resolutions?

2) is there any way to (or is there a distinction in) capturing video vs stills? My intent it to do facial detection (not recognition). I can do the detection on a lower res, but I would like to trim and store the highest resolution available. It seems from the Cam specs, I can go a bit higher resolution if I do stills vs video.

3) how can I control the rest of the camera parameters? I see in cameracontrol.h the raspicamcontrol_set/get functions, but it's not clear where I get the requisite MMAL_COMPONENT_T *camera).

Reply

Chris Cummings 28 November 2013 at 18:51

Hi kyle.

In answer to your first question, It probably would run out of memory at that resolution - mmal has a fair few buffers allocated internally, and the camera pi creates a tripple buffered layer, so when you start creating huge images that demand 20MB buffers, you run out pretty quick! If you want more you'll need to assign more memory to the gpu, which I beleive is possible to do using the raspi config tool, although I've not tried it. I'm also not convinced it'd work at all though, as the api uses the video port of the camera component which I think is limited to 1080p. The performance costs of copying that amount of data around and analysing it would set you down to a very poor frame rate anyway.

On the second question, I've not tried it, but I am aware there's a stills port on the mmal camera component which may be usable at the same time as the video port (which the camera api hooks up to). I've not written any code to do it though.

For your third question, I haven't hooked any of them up nicely but it should be easy to do. You will need to add some functions to the CCamera class that call those set/get functions by passing the CameraComponent member of CCamera.

Reply

Replies

kylemallory 2 December 2013 at 20:47

Thanks Chris. I got my answers. The key being that your API is using the "video" modes, which, yes, I believe is limited to 1080p. I guess I need to look into extending your API to support the stills mode. I think this is how raspistill works-- preview is done in video mode, but just before it records the output image, it jumps to 'still' mode. The give-away is that the image composition of the final output image is wider than the preview, because its using more pixels on the sensor. Though thinking that through, it won't matter then-- if I'm "previewing" at 1280x720, and then jump to 2880x1800 (or whatever), the "subject matter" will still be in the central 1280x720 region, since the video/preview API is cropping the sensor, as opposed to sampling the entire sensor area and scaling down to 1280x720.

CCamera::CameraComponent is perfect. Thanks!

Reply

Uwe Reinersmann 29 November 2013 at 12:30

Hi Cris,
thank you for your response, it is not verry helpful.

Reply

Replies

Chris Cummings 29 November 2013 at 13:42

And that kind of passive aggressive demand is rather rude. As it happens, I have been thinking over your request. I am working on a version 2 of the api which will provide OpenGL textures, and I believe qt has support for drawing them. In future though, I would request that you avoid further requests for support until you can be more polite and patient.

Reply

Anonymous 3 December 2013 at 15:46

Very nice work. Saved me a lot of time. Actually I dug already into the MMAL API using raspistill as an example. The lack of documentation is just frustrating. So this little library is exactly what I need.

Reply

Anonymous 19 December 2013 at 18:28

Hi Chris,
When do you think you'll finish the updated version of this API? In the mean time, I suppose that I'll just follow the instructions in the above comments to get this to work with OpenCV.

Reply

Replies

Chris Cummings 9 January 2014 at 19:25

Hi. Sorry for the delay getting back to you. I've been a bit delayed due to holidays and work so haven't had much time to work on the api. I am going to attack it in the coming months but I can't really give any guaruntees as I never know when I'll have some free time.

Reply

Anonymous 9 January 2014 at 18:38

When trying to build picademo I get this

pi@raspberrypi ~/David/picamdemo $ cmake .
CMake Error at CMakeLists.txt:4 (find_package):
By not providing "FindOpenCV.cmake" in CMAKE_MODULE_PATH this project has
asked CMake to find a package configuration file provided by "OpenCV", but
CMake did not find one.

Could not find a package configuration file provided by "OpenCV" with any
of the following names:

OpenCVConfig.cmake

opencv-config.cmake

Add the installation prefix of "OpenCV" to CMAKE_PREFIX_PATH or set "OpenCV_DIR" to a directory containing one of the above files. If "OpenCV" provides a separate development package or SDK, be sure it has been installed.

-- Configuring incomplete, errors occurred!

Any ideas?

Dave

Reply

Replies

Chris Cummings 9 January 2014 at 19:24

Hi there. You should be able to remove the opencv module from the build - its not needed. Alternatively a quick search online should show you how to download the opencv libraries for the raspberry pi. The latest version isn't available, but the one before it is.

Reply

Anonymous 7 February 2014 at 16:01

There seems to be a failure at the point where I'm supposed to download userland-master.zip. The file doesn't seem to be on github any more. Do you have updated instructions? Alternate locations to get the file from? Thanks.

Reply

Anonymous 7 February 2014 at 16:06

Nevermind - I don't use github that often and the link to the zip is not very obvious. The biggest section of the page is dedicated to downloading and browsing individual files and the zip download button is partially hidden off the right of the page. I'd recommend changing the above link to point directly to the zip download instead of the project page - or maybe it's just me ;-)

Reply

catherine 18 February 2014 at 20:47

Hi Chris, I was wondering if you could spare a minute to look at what I think is a basic issue I'm having with your API... My source code and explanation of the problem is here:

http://www.raspberrypi.org/forums/viewtopic.php?f=33&t=69874

Thanks muchly!

Reply

Groujers 1 March 2014 at 23:52

Hi. Thanks for the awesome api !
One question though,
Why cant I change the resolution to 640*480 for example ?
Are there any rules regarding the image resolution ?

Reply

Replies

Chris Cummings 15 April 2014 at 20:38

There are a few restrictions on the resolution the camera supports, but I beleive there's also a bug somewhere in my code with the conversion to rgb. It makes certain assumptions about the layout of the YUV buffer, which aren't correct for every resolution. I later found out that each block of the YUV data is 16 byte aligned, but my code didn't account for that which meant that certain resolutions (ones that didn't result in YUV blocks of multiples of 16 bytes) didn't convert properly. In theory it should be entirely possible though.

Reply

Didi 18 March 2014 at 08:06

Thanks for the demo. Very nicely done.
However, I cannot use this code in a real time frame analysis-control project.
MMAL is too big and the C++ code is too cryptic.
You seem to be a good programmer. Did you find the routines to directly control the camera?

Reply

Replies

Chris Cummings 15 April 2014 at 20:39

I did dig into the OpenMAx library, which is as close as you can get without having access to the none disclosure agreements and drivers that the pi team have. My experience is that mmal is there to make it easier for you though - it hides a lot of the complexities of OpenMax. Obviously mmal is still pretty advanced, which is why I decided to write my wrapper to make it a bit easier.

Reply

Chris Cummings 18 March 2014 at 08:38

Hi there

Using the routines directly to control the camera would effectively be going direct to OMX, which is far more complex and hairy than using mmal (that's exactly why the pi team used mmal for it). The purpose of my code is to provide you access to that mmal library without having to worry about the innards.

So simple answer, mmal is the simplest way to control the camera, and using my api should be very easy to use. There's no simpler way to do it unfortunately - real time camera processing just isn't a simple job, though I've tried to make it as simple as possible with my api!

-Chris

Reply

Dominik 20 March 2014 at 08:26

Does this API has any license ?

Reply

Replies

Chris Cummings 20 March 2014 at 09:18

Just the license to use it for whatever you like, in any way you like! I don't take any responsibility for any issues you hit etc etc, so if it inadvertently sets off a nuclear explosion or something it's your problem, but other than that, use it for whatever you like. It's nice to be credited if you feel I should be, but that's entirely your call :)

Reply

Romel Valencia 13 April 2014 at 19:44

Hi I'm getting the same error as the user below, and his problem was userland should have been userland-master. I am using userland-master but get the same error? Any suggestions?

JBeale31 October 2013 00:42
Followed instructions, but the make failed as follows:

[25%] Building CXX object CMakeFiles/picamdemo.dir/picam.cpp.o
In file included from /opt/vc/include/interface/vcos/vcos_assert.h:149:0,
from /opt/vc/include/interface/vcos/vcos.h:114,
from /opt/vc/include/interface/vmcs_host/vc_dispmanx.h:33,
from /opt/vc/include/bcm_host.h:46,
from /home/pi/picamdemo/mmalincludes.h:9,
from /home/pi/picamdemo/camera.h:3,
from /home/pi/picamdemo/picam.cpp:3:

/opt/vc/include/interface/vcos/vcos_types.h:38:33: fatal error: vcos_platform_types.h: No such file or directory
compilation terminated.
make[2]: *** [CMakeFiles/picamdemo.dir/picam.cpp.o] Error 1
make[1]: *** [CMakeFiles/picamdemo.dir/all] Error 2
make: *** [all] Error 2

Reply
Replies

JBeale31 October 2013 05:38
Nevermind- problem was my /opt/vc/userland should have been named /opt/vc/userland-master.

Reply

Replies

Chris Cummings 15 April 2014 at 20:41

Very sorry but I can't off the top of my head think of why that'd happen. Presumably something about your directory structure is different from mine and thus it can't find the file. I'd start by starting again, and following each step very carefully. If the userland master code is extracted and built at the right place it should work fine.

Reply

eried 22 April 2014 at 20:07

Hey Chris, how hard would it be to switch raspistill to use the video port for the process signal capture mode? Any hint?

Reply

Replies

Chris Cummings 26 August 2014 at 10:00

Hi eried - sorry for the slow reply. I suspect it wouldn't be too hard in theory, but would require a fair understanding of the mmal layer that raspistill is coded in.

Reply

Dominik 2 May 2014 at 15:33

Hi
Is there a possibility to create a raw make file of camera API (without graphics).

Regards

Reply

Replies

Chris Cummings 26 August 2014 at 10:09

Hi Dominik. Sorry for the slow reply. The camera api itself should be entirely independent of graphics - I just use it for displaying textures. If you replace my main.cpp with your own, and don't include graphics.cpp/h you should find you can remove all use of opengl.

Reply

Dominik 3 May 2014 at 16:27

Hi,

I have a problem. I am using StartCamera, and then ReadFrame and it returns 0 to me so no bytes where copied. Is there someone that got same problem, and solved it ?

Regards

Reply

Diego Andres Vierma Chata 18 July 2014 at 19:50

Hey Cris,

thanks for your post, great explanation, but i need help in something, i see that you have your raspicam inverted, and im trying to figure out how to invert the image so i dont have to use it upside-down, do you think you can tell me where i can change that? thanks

Reply

Replies

Chris Cummings 26 August 2014 at 10:13

Hi Diego. I don't think the images that come out of the camera aren't actually inverted - I just invert them when rendering opengl so they appear the right way up! If however you did want to invert them, it'd be a case of getting the pitch of a row of pixels (the width * 4 bytes for argb) then iterating over each row, just flipping the top ones for the bottom ones.

Reply

Linus S 12 August 2014 at 12:57

Thank you for this API Chris, it really is quiet helpful. I've read that you were planning to update this API to make it better and faster, is that something you still have in mind? Anyhow I appreciate all the effort you put into this API it has helped me a lot, take all the time you need if you're going to update this API, thank you once again for the effort you put into this!

Reply

Replies

Chris Cummings 26 August 2014 at 09:58

Hi Linus - glad to help :)

I'm not working on it right now, as my real job is taking up lots of time - I wrote this one on my holidays, so might well add to it next time I'm off work! I do however intend to get it up onto github as an opensource project, so hopefull others can improve it without a dependency on my holiday allowance :)

-Chris

Reply

Anonymous 7 September 2014 at 12:23

Hi Chris,
Thanks for sharing this extremely useful piece of software !
I have a question regarding resolution : If I'm correct, ReadFrame will return raw data taken from the camera without compression, right ? pixels in the buffer will correspond to physical pixels, encoded either in YUV 420 or ARGB depending on the boolean flag in ReadFrame. If we specify a lower than max resolution while creating the CCamera object, does the image gets cropped, or is there a kind of mapping from max to specified resolution ?
Can you please share you understanding of these questions ?
Then, can you give me a hint on how to proceed to make a compressed image (jpeg ?) from a frame obtained with ReadFrame, since I want to send still images taken at intervals with the Raspberry cam on the network, without taking too much bandwith (uncompressed image is easily a few Megs ...)
Thanks a lot
Oliver

Reply

Replies

Chris Cummings 7 September 2014 at 14:14

Hi Oliver

Yes you're correct - ReadFrame returns uncompressed data from the camera, and depending on the bool you pass in, it'll either be in YUV420 (which is what naturally comes from the camera system internally) or get preconverted to RGBA.

The resolution you specify on initialisation is passed to the mmal camera layer to tell the camera at what resolution to record in, so it is effectively being downsampled, not cropped. The camera doesn't work with absolutely any resolution though, so you might have to experiment a bit.

For the compressed image, you'd need to take the RGBA data and use an image compression library to do it. If you look at my next blog on gpu accelerated camera processing you'll see it contains a small png file encoder which gets used in GfxTexture::Save. There's lots of libraries available to do png/jpg conversions, but this one's definitely the smallest and simplest I've managed to find - just 1 source file you include in your project.

-Chris

Anonymous 8 September 2014 at 14:25

Hello Chris,

Thanks a lot for the very fast answer.

Following your hints I could succesfully compress my image in jpeg.

I want now to make calculation on the image buffer returned by ReadFrame, in order to calculate the average intensity in a given rectangle in the image. for this, I need to access the pixel values from the buffer. I use a 1920x1080 image size. Can you indicate where to look to find how the data is organized in the buffer. Is there a header with image information and is the data aligned to some block size ?
I can make use either of the YUV format or the RGBA format.

Best regards

Oliver

Reply

solderspot 30 October 2014 at 19:18

HI Chris,

Thanks so much for putting this code together. Helped me a lot to break the ice on all of this.

Just to let you know, I modified your code to do a quick bench test of some OpenCV color thresholding I'm developing for a robotics application. I wanted to see what kind of frame rates I could get from the Pi.

I have the test code up on GitHub: https://github.com/solderspot/PiCamCVTest

One of the changes I made that might be of interest was removing the userland-master dependency.

Again, thanks for the help.

All the best,
Tom.

Reply

Replies

Anonymous 19 February 2015 at 20:25

Hi Tom,
I'm curious on what frame rate you achieved with your test.
Cheers,

Wally Bkg 1 June 2015 at 00:56

I got 17-12 fps on a Pi2 with the defaults using the github code.

Any insights as to why the video is inverted compared to rapsivid and raspisill captures?

Wally Bkg 1 June 2015 at 00:58

should be 17 -20 fps, looked right when I clicked publish :(

solderspot 31 January 2016 at 21:22

Running the code on my Pi, which is over clocked at 900Mhz, I get the following results:

IMAGE SIZE 160×160 320×320 640×640
no image processing 40 fps 30 fps 15 fps
image processing 20 fps 10 fps 3 fps

Reply

nick 20 February 2015 at 22:07

Hi Chris,
thx a lot for this API! Great code!
I have a question for you, when I use bool do_argb_conversion = false then the display of the image is wrong. I get a rectangular section at the bottom of the screen and the image is repeated 4 times. Any idea how I can get it to display correctly?
Thx
Nicola

Reply

Anonymous 12 September 2015 at 17:57

Hello,

First of all, good work. But i encountered a strange problem: regardless how i set the resolution, the image quality remains the same:

640x320: http://upload-pictures.de/bild.php/91004,im0OOP5M.jpg

720p: http://upload-pictures.de/bild.php/91005,im1OO79W.jpg

Hardware: Raspberry PI 2, NoIr camera. Works perfectly fine with raspistill.....

Any Ideas?

Reply

SergioM 10 November 2015 at 16:24

Hi Chris! this looks great, but i was trying to download the code at the link in the post and its no logger available, could you please upload it again? thanks in advance! regards!

Reply

Chris Cummings 11 November 2015 at 08:29

BROKEN Link! Hey guys, really sorry but I just noticed that link has broken (my domain expired while I lwan on holiday!). I'll try and get it fixed soon. In the meantime there's a more recent blog post showing where a friendly chap uploaded it to github.

Sorry for the inconvenience and thx for trying picam.

- chris

Reply

Chris Cummings 2 December 2015 at 12:08

Link should be working again!

Reply

Dinushka Wijesuriya 1 January 2016 at 05:44

Im getting a error when running picamdemo. Can you help me?

picamdemo: /root/picamdemo/graphics.cpp:194: bool GfxShader::LoadVertexShader(const char*): Assertion `f' failed.

Reply

Replies

Dinushka Wijesuriya 1 January 2016 at 06:32

My compiler version is (Debian 4.6.3-14+rpi1) 4.6.3.. I realize that problem is with assert() command in LoadVertexShader(). Is this a problem with compiler?

quyền 10 September 2016 at 10:10

This comment has been removed by the author.

quyền 10 September 2016 at 10:11

I get the same error like this, can you tell me know if it fix!!!!!

many thanks

Reply

alii 5 February 2016 at 10:11

This comment has been removed by the author.

Reply

Johannes Merkert 22 April 2016 at 16:22

Hi Chris,
thank you for your work. The example works like a charm and I could easily integrate your API in my project.

I want to set exposure, white balance, ISO, etc. manually so the image does not change when the lighting conditions change. Woh is that possible with your API?

Reply

Unknown 14 June 2016 at 11:34

hi... chris please help me out!!
how do we increase the fps??

Reply

Anonymous 2 September 2016 at 17:22

What is the purpose of resizer component? Can I warp image using resizer? Can it take transform matrix? Bit new at this..

Reply

Anonymous 3 January 2017 at 10:12

Hi Chris,
I have installed the library with no issues.
But I can't read image.
My code is as below

```
CCamera* cam = StartCamera(640, 480,60,1,true);
char mybuffer[640 * 480 * 4];
int ret = cam->ReadFrame(0, mybuffer, sizeof(mybuffer));
cout << " ret " << ret << endl;
Mat img(480, 640, CV_8UC4,mybuffer);
imwrite("img.jpg", img);
```

What could be wrong?
Is it possible to have 60Hz fps using the library?
How to change to Gray image from RGB after reading the image?

Your library is very useful and thanks for sharing.