**Vision Subsystem Documentation**

**A step-by-step guide to getting OpenCV and Aruco working in Visual Studio as well as the RaspberryPi**

**This is quite a tedious process. If you ever get stuck, reach out to me:**

[**suhail.prasathong@gmail.com**](mailto:suhail.prasathong@gmail.com)

[**sxp9646@rit.edu**](mailto:sxp9646@rit.edu)

**Setting up Aruco (entire platform) on the Pi:**

**Instructions for setting up a New Raspberry Pi OS and OpenCV**

1. Go to http://www.raspberrypi.org/ to install the latest OS (Raspbian).
2. Remember, anytime you want to a fresh install back to the original OS, reboot and hold “Shift” during power up.
3. For fresh OS install, you get a chance to change the configuration (or change it anytime by entering sudo raspi-config).
   1. Make sure you enabled SSH and the camera (per the table below)
   2. Set your Timezone
   3. I read somewhere else you might want to give the GPU more memory like 256 for OpenCV but I have not tried this yet.
4. login as pi, password raspberry or whatever
5. startx
6. Get WiFi working (gaines, WPA2-Personal(PSK), TKIP, PSK is our home password).

NOTE – IT MIGHT BE WISE TO USE ETHERNET CONNECTION TO DO ALL THIS INSTALL STUFF. COULD BE 10X FASTER! ALSO, IF WIFI CHIP FAILS IN THE MIDDLE OF A LOAD IS A BUMMER.

1. If desired, add remote access so you can work on the Raspberry Pi from you normal PC. http://www.raspberrypi.org/documentation/remote-access/vnc/README.md

|  |  |
| --- | --- |
| **Explanation** | **Command** |
| He said to make sure you have both ssh and the camera enabled. I did this in step 2 above. | If you failed to do this in step 2 above, then could use: sudo raspi-config |
|  |  |
|  |  |
| I am skipping the first part of this where he needs to do an SSH to his pi. If this install fails, go back at look at this again. Perhaps this is needed if you do the remote access step above (step 8). | Skipping for now but it has things like ping HOSTNAME which for me HOSTNAME=raspberrypi unless you changed the hostname. This gives the IP address for the SSH. |
| This gets rid of pre-existing old versions of opencv | sudo apt-get remove libopencv\* |
| sudo apt-get autoremove |
| Make sure everything is up to date. | sudo apt-get update |
| This upgrades so you have a clean file to work with. | sudo apt-get upgrade |
|  | sudo rpi-update  sudo reboot |
| Install Devel tools | sudo apt-get install build-essential cmake cmake-curses-gui pkg-config |
| Install required libraries (you will run into errors with missing libraries, just install those as well) | sudo apt-get install \  libjpeg-dev \  libtiff5-dev \  libjasper-dev \  libpng12-dev \  libavcodec-dev \  libavformat-dev \  libswscale-dev \  libeigen3-dev \  libxvidcore-dev \  libx264-dev \  libgtk2.0-dev |
| **Video4Linux** Python manages Raspberry's camera by the way of picamera module. You can use this module also in OpenCV but you have to grab images to numpy.array then map the array to OpenCV Mat.  To use the standard grabbing loop cv2.VideoCapture(0) with raspicam the Video4Linux driver is needed.   1. Check prerequisites (with sudo raspi-config):    1. Enable the camera    2. Set large memory for gpu\_mem (In Advance Options > Memmory Split set 128min) |  |
| Install v4l | sudo apt-get -y install libv4l-dev v4l-utils |
| Enable kernel | sudo modprobe bcm2835-v4l2 |
| Decide what version of OpenCV you want. | sudo mount /dev/your-dev-name /home/pi/usbmem mkdir /home/pi/usbmem/opencv cd /home/pi/usbmem/opencv wget https://github.com/opencv/opencv/archive/3.2.0.zip -O opencv\_source.zip wget https://github.com/opencv/opencv\_contrib/archive/3.2.0.zip -O opencv\_contrib.zip |
|  |  |
| This is your cmake you need for your compilation and the build essential. | sudo apt-get install build-essential cmake pkg-config (all one line) |
| Java (jdk – Java Development kit) | sudo apt-get install default-jdk ant |
| Gtk development. I am not sure, but I think this is GTK+, or the GIMP Toolkit, is a multi-platform toolkit for creating graphical user interfaces. Offering a complete set of widgets, GTK+ is suitable for projects ranging from small one-off tools to complete application suites. | sudo apt-get install libgtkglext1-dev |
| Bison is a general-purpose parser generator that converts an annotated context-free grammar into a deterministic LR or generalized LR (GLR) parser employing LALR (1) parser tables. | sudo apt-get install bison |
| He said you could use Qt5 but he using Qt4 and I have used Qt4 in the past successfully. | sudo apt-get install qt4-dev-tools libqt4-dev libqt4-core libqt4-gui |
| v4l is video for Linux. It will say you already have it. | sudo apt-get install v4l-utils |
| He says this is a good IDE if you need one. I have used it in the past and liked it. | sudo apt-get install qtcreator |
|  |  |
| Now time to compile OpenCV. First go to where you downloaded the compressed OpenCV file. Do a dir to see it. Then unzip it. | unzip opencv-3.2.0.zip |
| Once it unzips, it creates a directory. Move into that directory. | cd opencv-3.2.0 |
| Now create a build directory. | mkdir build |
| Change into the build directory. | cd build |
| Configure using cmake. CMake is a cross-platform free software program for managing the build process of software using a compiler-independent method.  Notice, that by adding the -D WITH\_QT=ON, the highgui module will use QT instead of GTK. If you want to go with GTK just remove this. For more info on options, look at the CMakeLists.txt file. When you are happy with the configuration, you can start compiling:  This will take about 9 minutes. It will configure to make sure you have everything ready. You should look for errors here in the output.  **PLEASE CHECK THE OPENCV VIDEO 11 FROM THE PLAYLIST** [**https://www.youtube.com/watch?v=fIpTks0G2m0&t=355s**](https://www.youtube.com/watch?v=fIpTks0G2m0&t=355s)  **This will tell you all the flags needed to get aruco to work**  INSTEAD OF TYPING ALL THAT ON THE RIGHT, YOU MIGHT TRY THIS SOMETIME:  Something of interest. I think you can just type cmake .. or ccmake .. and then press a ‘c’ and you will get a screen showing the installation flags. You then can use your up and down arrows and change flags using the return key. Then I think you press ‘g’ to generate. | cmake -D CMAKE\_BUILD\_TYPE=RELEASE -D INSTALL\_C\_EXAMPLES=ON –D INSTALL\_PYTHON\_EXAMPLES=ON -D BUILD\_EXAMPLES=ON -D WITH\_QT=ON -D CMAKE\_INSTALL\_PREFIX=/usr/local -D WITH\_OPENGL=ON -D WITH\_V4L=ON –D BUILD\_NEW\_PYTHON\_SUPPORT=ON -D WITH\_TBB=ON .. |
| This will start building them. This will also take a long time. About 10 to 12 hours. It will say 100% built when done. | make -j4 |
| Now you are ready to install. If you forget to put sudo, it will do it all and then say you lack sufficient privileges. So don’t forget sudo. | sudo make all  Sudo ld config |
| To configure. | sudo ldconfig |
| Now open this existing file. | sudo nano /etc/bash.bashrc |
| There is a large “if statement” near the bottom of the file. It is to test if the “command-not-found” package is installed. Go after that if statement. NOT inside of it. This is probable just the end of the file but I cannot be sure from the youtube video. In any case, AFTER this if statement, type in the two lines shown at the right.  Ctrl-X to exit, it will ask “Save modified buffer…”, so select Y for yes and agree with the filename and exit the editor. | PKG\_CONFIG PATH=$PKG\_CONFIG\_PATH:/usr/local/lib/pkgconfig  export PKG\_CONFIG\_PATH |
| This was not in the video but important (he added a note later). Type exit at the command prompt to exit the terminal. Then reopen it so the changes to the bashrc file take. | exit |
|  |  |
| NOW DO A TEST. PLUG IN CAMERA AND GO INTO SAMPLES DIRECTORY FOR C (I THINK YOU CAN ALL DO THIS FOR C++ EXAMPLES IN THE CPP DIRECTORY) |  |
|  |  |
|  |  |
|  |  |
|  |  |

Additionally, here are some other helpful links and videos to help you through the installation process:

<http://pklab.net/?id=392&lang=EN>

<https://www.youtube.com/watch?v=HmDkXMe5_RA>

**Test the Aruco code:**

\*The Aruco generation, recognition etc has been built out in the main.cpp (https://goo.gl/9ifQpT) file

\*Run code by first:

cd into the folder where your code and Camera Calibration file is located (if you go through the videos you’ll know what the Camera Calibration file is)

Then run

|  |
| --- |
| sudo modprobe bcm2835-v4l2 |

This is meant to instantiate one of the video libraries. The library as it stands has issues and doesn’t self-instantiate.

then

|  |
| --- |
| g++ -o Aruco main.cpp -lopencv\_highgui -lopencv\_core -lopencv\_videoio -lopencv\_imgcodecs -lopencv\_aruco -lopencv\_calib3d |

Aruco - this is what my executable is called. If your executable is different, change it

main.cpp - this is what I called my main file. Change it if yours is different

The rest are opencv dependencies that you need to pass in for this to work

Lastly, execute your code:

|  |
| --- |
| ./Aruco |