Raspberry Pi Computer Vision Setup

Table of Contents

[I. Python and OpenCV Compatibility Issues 2](#_heading=h.tyjcwt)

[II. Python Installation & Set-up 3](#_heading=h.3dy6vkm)

[III. OpenCV Installation & Set-up 5](#_heading=h.1t3h5sf)

[IV. Pip Libraries, Requirements Installation 7](#_heading=h.4d34og8)

[V. Camera Module Set-up 9](#_heading=h.2s8eyo1)

[VI. OpenCV & Camera Module Compatibility 11](#_heading=h.17dp8vu)

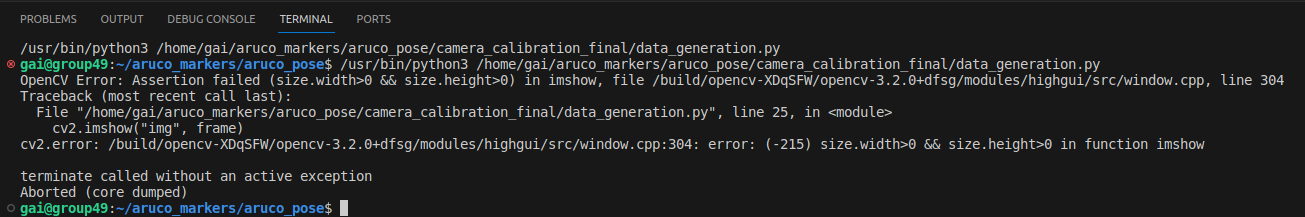
## **Python and OpenCV Compatibility Issues**

Before proceeding with the installation of Python and OpenCV on the Raspberry Pi 4 (4GB RAM), users may encounter a common compatibility issue. Specifically, improper installation may lead to code execution problems when working with OpenCV. Therefore, it is crucial to thoroughly understand the installation requirements for Python and OpenCV, as outlined in this section.

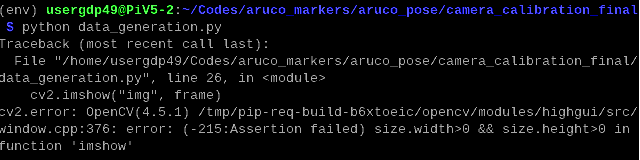
Furthermore, it is important to note the differences in implementing OpenCV between the Jetson Nano and Raspberry Pi platforms. For detailed instructions on setting up OpenCV on the Jetson Nano, please refer to the separate documentation provided in [link to the document].

#### Compatibility Issue

For clarity, it is important to recognize the close relationship between Python and OpenCV versions in terms of compatibility. Incompatibility between these versions can lead to numerous issues when attempting to run OpenCV on the device. Some common errors that may arise include the Cv2.error: (-215: Assertion failed) size.width>0 && size.height>0 in function ‘imshow’ as supported in the Figure 1 and 2 below.



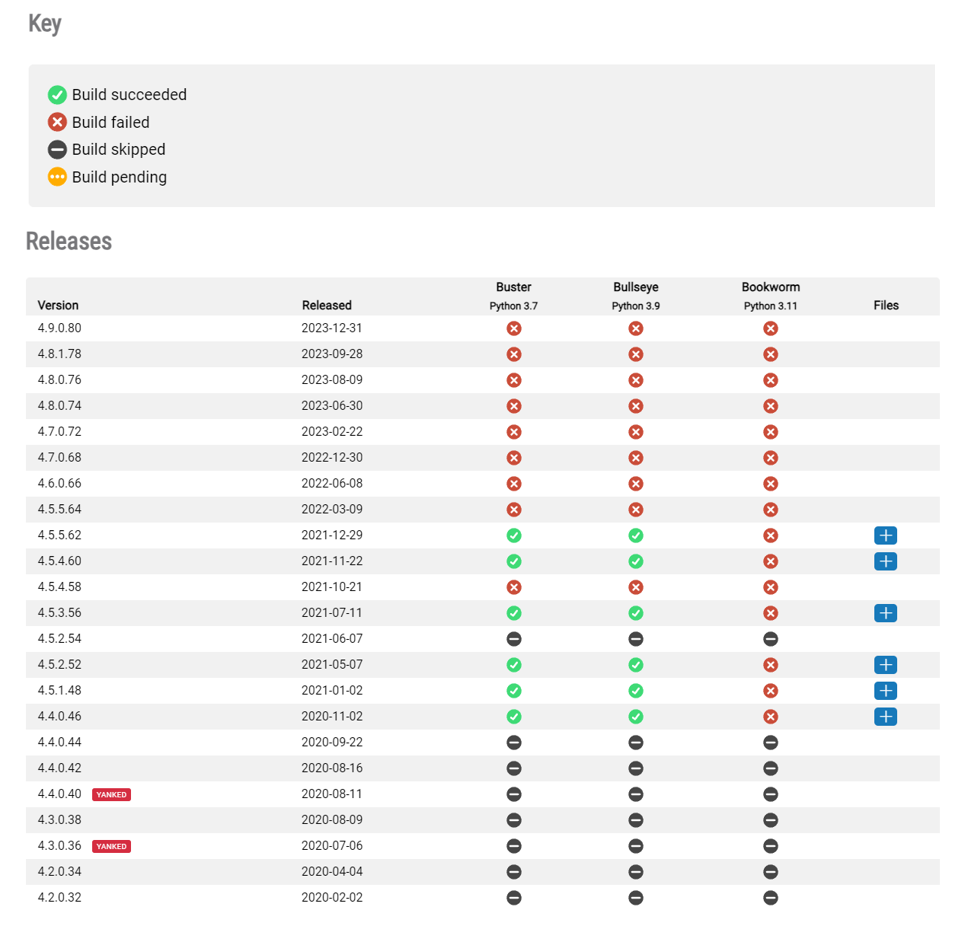
*Figure 1: cv2.error example 1*



*Figure 1: cv2.error example 2*

#### Solution

Below is a comprehensive list of Python and OpenCV compatibility versions. It is important to note that any of the listed compatible versions below (indicated by a green tick) can be utilized. However, for our specific case, we have chosen Python Bullseye 3.9 and OpenCV 4.5.3.56. The rationale behind this decision will be covered in the subsequent sections titled ***'I. Python Installation & Setup***' and 'OpenCV Installation & Setup'.



*Figure 3: Table comparing the compatibility between Python and OpenCV versions* [1]

## **Python Installation & Set-up**

As discussed in the preceding section titled ***'I. Python and OpenCV Compatibility Issue'***, this section outlines the comprehensive procedure for installing a specific Python version, namely Bullseye, onto the device. It is worth noting that while Python 3.9.4 will be installed, any versions within the 3.9 series should function seamlessly as well.

There exist various methods to install a specific Python version, including direct downloads from the official website source. However, I have personally found that installing via Pyenv is one of the simplest and most efficient methods available.

#### Pyenv Installation

Pyenv serves as a lightweight and robust Python version management tool. It facilitates effortless switching between multiple Python versions installed on a system, thereby enabling smooth management of diverse Python environments.

It is important to emphasize that Pyenv is typically not pre-installed on a Raspberry Pi. Therefore, the initial step involves opening the terminal and executing the following commands to install Pyenv:

| **Command** | **Description** |
| --- | --- |
| curl https://pyenv.run | bash | Run the easy installer |
| sudo nano ~/.bashrc | Add pyenv to .bashrc:  Edit the .bashrc with the command |
| export PATH="$HOME/.pyenv/bin:$PATH"  eval "$(pyenv init --path)"  eval "$(pyenv virtualenv-init -)" | Add the following three lines to the bottom of the .bashrc file: |
| exec $SHELL | Restart the terminal |
| sudo apt-get install --yes libssl-dev zlib1g-dev libbz2-dev libreadline-dev libsqlite3-dev llvm libncurses5-dev libncursesw5-dev xz-utils tk-dev libgdbm-dev lzma lzma-dev tcl-dev libxml2-dev libxmlsec1-dev libffi-dev liblzma-dev wget curl make build-essential openssl | Install system packages |
| pyenv update | Update pyenv |

#### Python (Specific Version) Installation

Once Pyenv installation is successfully completed, reopen the terminal and proceed by executing the following commands to install Python 3.9.4 (in our specific case):

| **Command** | **Description** |
| --- | --- |
| pyenv install --list | Check python versions available to install |
| pyenv install 3.9.4 | Install specific python version |
| Cd <folder name> | Cd to the specific folder |
| pyenv local 3.9.4  pyenv global 3.9.4 (for all folders) | Set python version to 3.9.4. |

Following the steps described above, we should now have the compatible version of Python – 3.9.4 installed in our Raspberry Pi.

## **OpenCV Installation & Set-up**

As discussed in the preceding section titled ***'I. Python and OpenCV Compatibility Issue'***, this section outlines the process to successfully install a compatible OpenCV version for the **Raspberry Pi 4 (4GB RAM).** It is worth noting that while OpenCV 4.5.3.56 will be installed, any versions which are compatible with Python 3.9 should function seamlessly as well.

It is imperative to acknowledge that various methods have been explored to achieve successful installation of OpenCV, such as methods which involve cmake etc. While most methods do not encounter issues during installation, challenges may arise when executing code utilizing OpenCV. A comprehensive list of attempted methods, including those that were unsuccessful for various reasons, is provided in this document. [attach link]

In this section, I will exclusively focus on the successful and optimal method. The following commands are necessary to ensure the successful installation of OpenCV 4.5.3.56:

#### Expanding the filesystem and update/upgrade sudo

First, we need to expand the filesystem size and update/upgrade sudo to ensure smooth installation by toggling the following commands:

| **Command** | **Description** |
| --- | --- |
| sudo raspi-config | Toggle on to “6 Advanced Options” 🡪 “A1 Expand Filesystem” 🡪 Reboot |
| sudo apt-get update && sudo apt-get upgrade | Upgrade and update sudo |
| Python -V | Check the python version installed  Version: 3.9.4 |

#### Creating a virtual environment (Optional)

Virtual environments are better choice when you are running multiple different kind of applications on the similar device. However, considering if you are using the Raspberry Pi for only 1 application then there is no need of virtual environment. Furthermore, the Pyenv implementation also acts as a virtual environment, hence this section is just for full documentation purposes.

| **Commands** | **Description** |
| --- | --- |
| mkdir aruco\_markers | Create a project directory for the ArUco markers detection |
| cd aruco\_markers | Cd to the specific folder |
| pip install virtualenv | Pip install virtualenv library |
| virtualenv env | Create a virtual environment, named “env” |
| Source/env/bin/activate | Activate the virtual environment |

#### Installing OpenCV

The final step would be to install the specific version of OpenCV 4.5.3.56 in the device by executing the following commands:

| **Commands** | **Description** |
| --- | --- |
| sudo apt update | Update sudo before proceeding with the installation |
| pip install --upgrade pip setuptools wheel | Update pip, set-up tools and wheel, which will be used to install prebuilt binaries |
| sudo apt-get install -y libhdf5-dev libhdf5-serial-dev python3-pyqt5 libatlas-base-dev libjasper-dev | Install requirements |
| python -V | Always check the Python version before proceeding. The version should be similar to the specific version that we set for Pyenv, which in our case would be 3.9.4 |

A quick note that there are four different types of OpenCV releases. The main differences are the graphical user interface and extra modules. All of them are official releases:

* **opencv-python**: The main library. GUI components are included (such as cv2.imshow() and extra modules are not included.
* **opencv-python-headless**: Headless version of the main library. Headless means there are no components for GUI. So you will not be able to show interfaces in the graphical user interface of the Raspberry Pi. This can be useful when you use Raspberry Pi OS without a desktop environment (Raspberry Pi OS Lite) for robotics or other projects that won’t require an interface directly from Raspberry Pi.
* **opencv-contrib-python**: The main library plus extra modules that contributed. You can find a list of modules here. I am going to be following this package and I suggest it if you are not low on space or have other reasons not to.
* **opencv-contrib-python-headless**: Headless version of contrib package. The same case with the main headless package, this one does not include GUI modules.

In our case, we will install **‘opencv-contrib-python**’ to ensure the inclusion of all necessary extra modules. This helps prevent any deficiencies in module availability. Additionally, it is crucial to verify the successful installation of the compatible version before executing any computer vision-related code.

To verify the installation, execute the following commands and ensure that the printed OpenCV version is 4.5.3.56:

| **Commands** | **Description** |
| --- | --- |
| pip install opencv-contrib-python==4.5.3.56 | Install the opencv-contrib-python package for version 4.5.3.56 |
| python | Open up Python |
| import cv2 | Import cv2 into the python environment |
| cv2.\_*version\_* | Print out the OpenCV version.   * Version: 4.5.3.56 |

Following the steps described above, we should now have the compatible version of OpenCV – 4.5.3.56 installed in our Raspberry Pi.

## **Pip Libraries, Requirements Installation**

Upon successful installation of OpenCV on the device, several pip libraries/packages are necessary to ensure smooth operations. These include picamera, PyYAML, tqdm, and others. Users have the option to install the requirements file via pip using the following command. This step streamlines the entire process.

| **Commands** | **Description** |
| --- | --- |
| pip install -r requirements.txt | Install all the requirements into your project. |

Certain key packages warrant further explanation due to issues encountered during setup. These will be explained in the subsequent subsections below:

#### Installing Picamera

Before proceeding with the installation of picamera on your device, it is crucial for the user to comprehensively comprehend the distinctions between each library. While they share similar features, it is imperative to determine which one is best suited for our specific application.

* + **Picamera** is a Python library that provides a pure Python interface for controlling Raspberry Pi camera modules. It allows for easy capture of images and video, as well as advanced features like capturing images in different resolutions and formats etc.
  + **Libcamera** is a Linux user space camera library that provides a standardized interface for controlling and configuring camera devices.
  + **Picamera2** is an updated version of the picamera library, designed for Python Bullseye version or later and newer Raspberry Pi models. It offers improved performance, enhanced features, and better compatibility with modern Raspberry Pi devices, but has lots of limitations when implemented with OpenCV. It is also not advised to install Picamera2 via pip due to compatibility of Picamera2 and OS versions.

In our case, we will proceed with the installation of ‘**Picamera**’ library due to its suitability for OpenCV applications and camera module V2 purchased. To install, open up your terminal and type the following commands:

| **Commands** | **Description** |
| --- | --- |
| pip install picamera | Install picamera library |
| pip install picamera[array] | Install picamera library to include array functions |

#### Installing NumPy

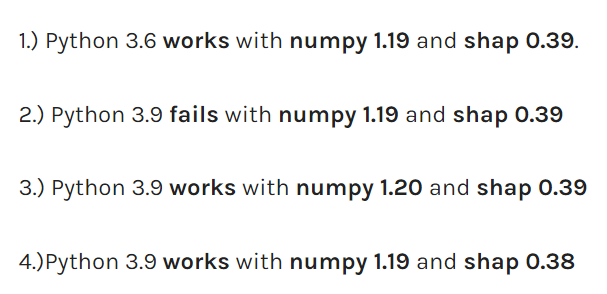
NumPy is a fundamental Python library for handling multidimensional arrays. Typically, users opt for the straightforward installation command 'pip install numpy' to install the latest version by default. However, in our specific application, when executing the code, errors often arise, as demonstrated below:

* libf77blas.so.3: cannot open shared object file: No such file or directory
* ImportError: numpy.core.multiarray failed to import
* Similar errors faced by other users: [2]

Various troubleshooting methods were attempted to address these issues, although not all proved effective. The table below outlines both unsuccessful attempts and the successful method utilized:

| **Commands** | **Description** |
| --- | --- |
| **Failed: Troubleshooting NumPy** [3] | |
| sudo apt-get install libatlas-base-dev | Install missing libraries expected by the self-compiled NumPy (ATLAS is a possible provider of linear algebra). |
| pip3 uninstall numpy  apt install python3-numpy | Uninstall numpy and use the numpy provided by Raspbian. |
| **Failed: Install all dependencies and libraries** [4] | |
| Sudo apt-get install libcblas-dev  Sudo apt-get install libhdf5-dev  Sudo apt-get install libhdf5-serial-dev  Sudo apt-get install libatlas-base-dev  Sudo apt-get install libjasper-dev  Sudo apt-get install libqtgui4  Sudo apt-get install libqt4-test | Install all dependencies |
| **Success: Ensuring NumPy, Python and Shap is compatible** [5] | |
| Pip uninstall numpy  Pip install numpy==<version> | Uninstall numpy and install the specific numpy version which is compatible with Python and Shap. |

In our scenario, based on the snapshot obtained from [5] as seen in Figure 4 below, we will proceed with ‘pip install numpy==1.20’, where we use Python 3.9 and shap 0.39



*Figure 4: Compatibility versions between Python, NumPy and Shap* [5]

#### Summary

Here is a summary of the original and updated versions for Python, OpenCV, and libraries:

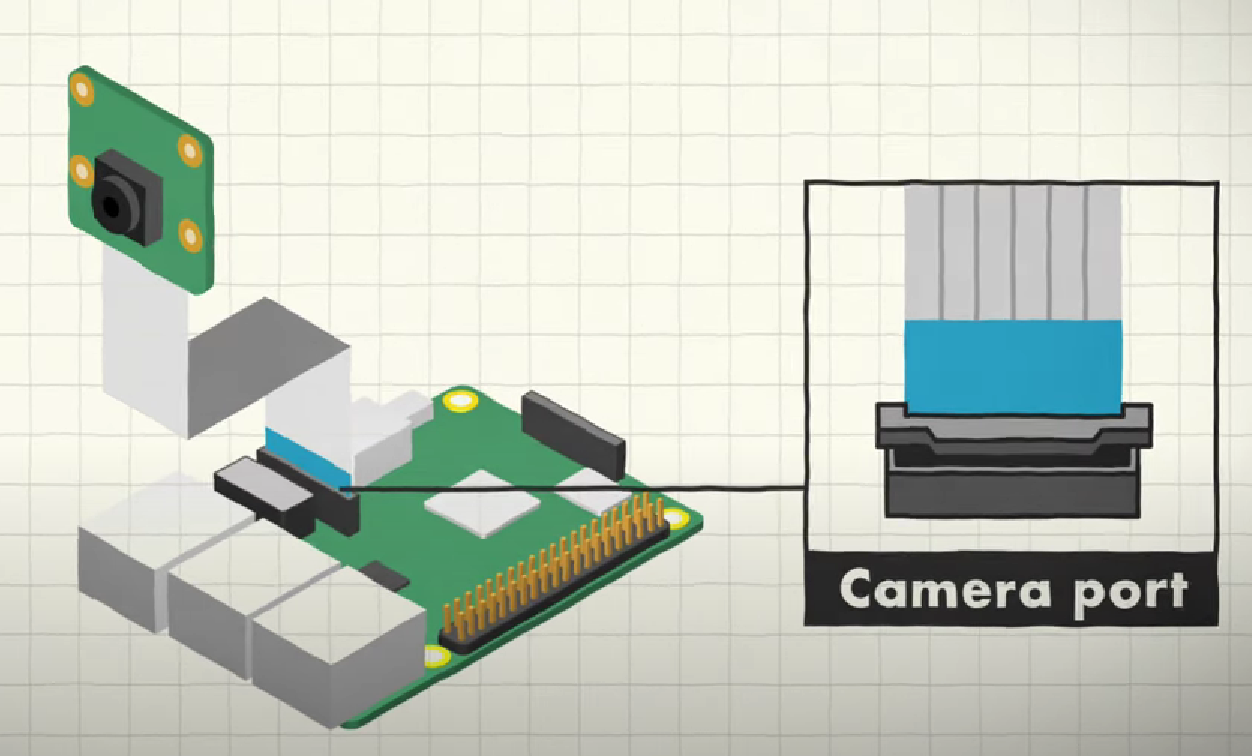
|  | **Original** | **Updated** |
| --- | --- | --- |
| Python | 3.9.2 | 3.9.4 |
| OpenCV | - | 4.5.3.56 |
| Numpy | 1.26.4 | 1.20 |
| Picamera | 1.13 | 1.13 |
| Shap | 0.39 | 0.39 |

## **Camera Module Set-up**

This section outlines the full setting up of a functioning camera module. The camera module we will be using would be the **Raspberry Pi Camera Module V2**.

#### Connecting the Camera Module to Raspberry Pi

Prior to connecting a camera module to the Raspberry Pi, ensure that the power supply is not connected. Lift the plastic tabs of the CSI camera connector slot gently and place the ribbon cable fully into the slot. Once the ribbon cable is in place, gently press down the plastic tabs to lock them in place. Key to note that the blue marking should face the ethernet port as seen in Fig 5 below.



*Figure 5: Orientation of Camera Module Insertion into the Camera Port*

#### Testing the Camera Module

Once the camera module is fully connected, power up the Raspberry Pi and open the terminal window and run the following commands.

| **Commands** | **Description** |
| --- | --- |
| **Option 1: Raspistill** | |
| sudo raspi-config | Open up the configuration settings. |
| Toggle to “Interface Options” and enable connection to the Raspberry Pi Camera | Toggle to enable the camera option. |
| raspistill -o test.jpg | Capture an image and save in the specified directory as “test.jpg”. |
| **Option2: Libcamera** | |
| Libcamera-hello | Opens up the camera module for a few seconds. |
| git clone https://github.com/raspberrypi/picamera2.git | Install and update the picamera2 software in a specified folder named “picamera2” with various functions to be used. |
| cd picamera2/examples  python preview.py | Example function of opening up the camera module. |

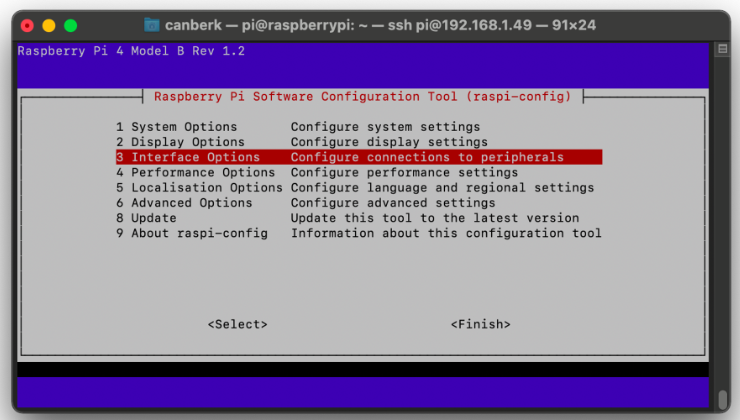
Up till this current step, we have a functioning camera module powered by the Raspberry Pi along with a compatible version of OpenCV being installed in this device. However, further steps will be required in setting-up the code to link OpenCV to this specific camera, which will be covered in the next section below.

## **OpenCV & Camera Module Compatibility**

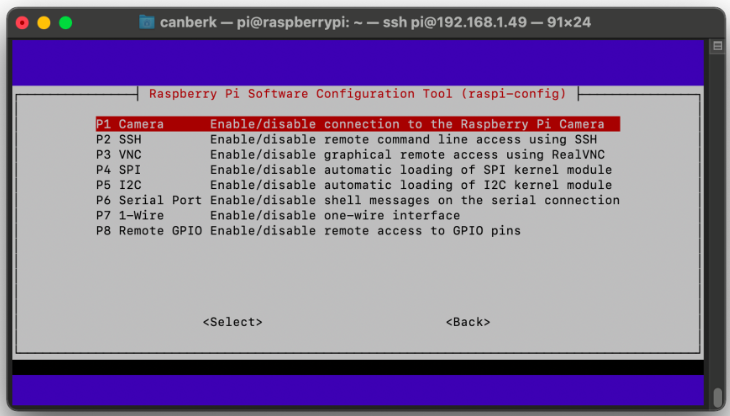
After ensuring the camera module is securely attached and its functionalities are tested, it's imperative to perform the following steps to ensure smooth operations when utilizing OpenCV with the camera module.

#### Enable Camera Legacy

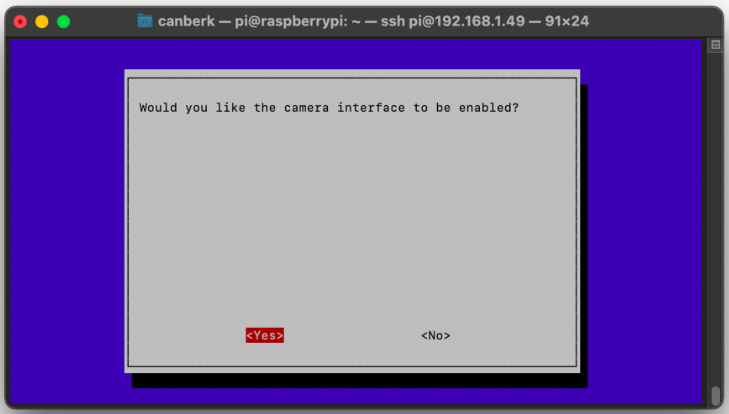
To enable the camera legacy permission for the device, users should open a terminal and enter the command 'sudo raspi-config' to launch the Raspberry Pi software configuration tool. Follow the steps illustrated in the figures below:



*Figure 6: Toggle onto ‘Interface Options’* [6]



*Figure 7: Toggle onto ‘P1 Camera’* [6]



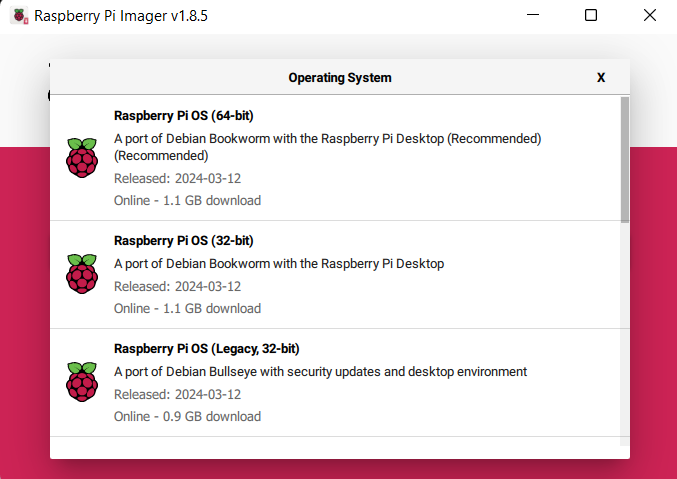
*Figure 8: Enable the camera interface* [6]

Finally, reboot the Raspberry Pi by executing the command 'sudo reboot'. Note that if the option to enable the camera interface 'P1 camera' is missing or not visible, please refer to the next sub-section titled 'Raspberry Pi OS Legacy Issue' for further instructions to resolve this issue.

#### Raspberry Pi OS Legacy Issue

The term "legacy" in "Raspberry Pi OS (legacy 32-bit)" typically refers to an older version or an older ARMv6 architecture that is still supported for compatibility reasons. This version is maintained for compatibility with older Raspberry Pi models, such as the Raspberry Pi 1 and Raspberry Pi Zero, which use ARMv6 processors. On the other hand, newer Raspberry Pi models, such as the Raspberry Pi 2, 3, 4, and later, use ARMv7 or ARMv8 (64-bit) processors.

However, in our scenario where enabling camera legacy is necessary, it is advisable to install the Raspberry Pi OS (Legacy, 32-bit) for simplification purposes, as indicated in the figure below.



*Figure 9: Raspberry Pi Operating System Options*

It is worth noting that while it is possible to run OpenCV without enabling camera legacy, it would entail additional modifications to the code, such as the integration of other libraries like libcamera or potentially picamera2. However, for our specific case, we will adhere to enabling camera legacy and utilizing a compatible version of picamera for conducting computer vision applications.

# References

| [1] | [Online]. Available: https://www.piwheels.org/project/opencv-contrib-python/. |
| --- | --- |
| [2] | [Online]. Available: https://github.com/python-poetry/poetry/issues/6447. |
| [3] | [Online]. Available: https://numpy.org/devdocs/user/troubleshooting-importerror.html. |
| [4] | [Online]. Available: https://stackoverflow.com/questions/53347759/importerror-libcblas-so-3-cannot-open-shared-object-file-no-such-file-or-dire. |
| [5] | [Online]. Available: https://www.pythonpool.com/solved-importerror-numpy-core-multiarray-failed-to-import/. |
| [6] | [Online]. Available: https://singleboardblog.com/install-python-opencv-on-raspberry-pi/. |
| [7] | [Online]. Available: https://stackoverflow.com/questions/53347759/importerror-libcblas-so-3-cannot-open-shared-object-file-no-such-file-or-dire. |