CME 466
Design of an Advanced Digital System

Shahim Vedaei, Omid Yaghoobian and Khan Wahid

Fall 2021 (Created), Winter 2023 (Updated)

Pi Camera and OpenCV

1. Pi Camera:

Ref: https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/0

Figure 1 shows a raspberry pi and Pi Camera. The Pi Camera is a cheap and great camera for using with RPI. You might also be able to use a normal USB webcam as well. However, as the RPI OS has an abstract Linux kernel, it might need a driver to work. The RPI OS is constantly working on its kernel to support more and more devices.

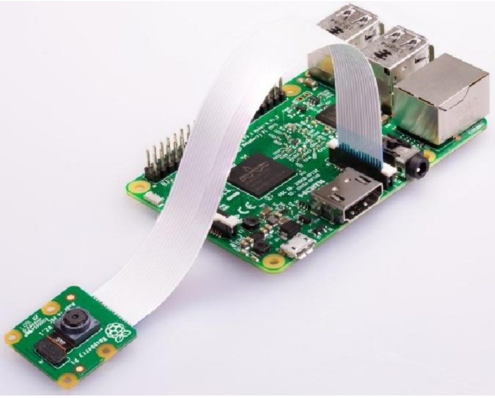
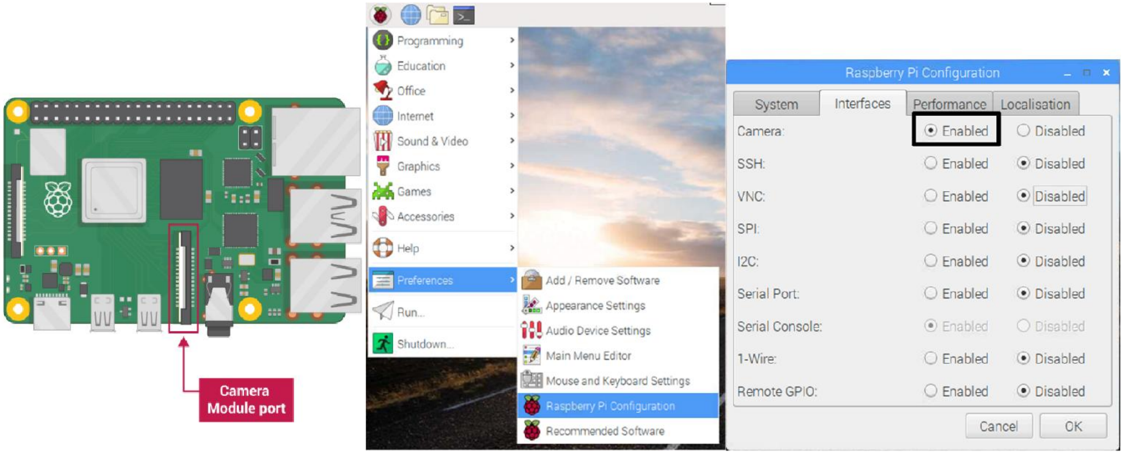


Fig 1. Connect Pi Camera to RPI

On the RPI a separate connector is provided to work with Pi Camera. Connect the Pi Camera using this connector. Then, enable the camera setting in configuration menu, as shown on Figure 2.

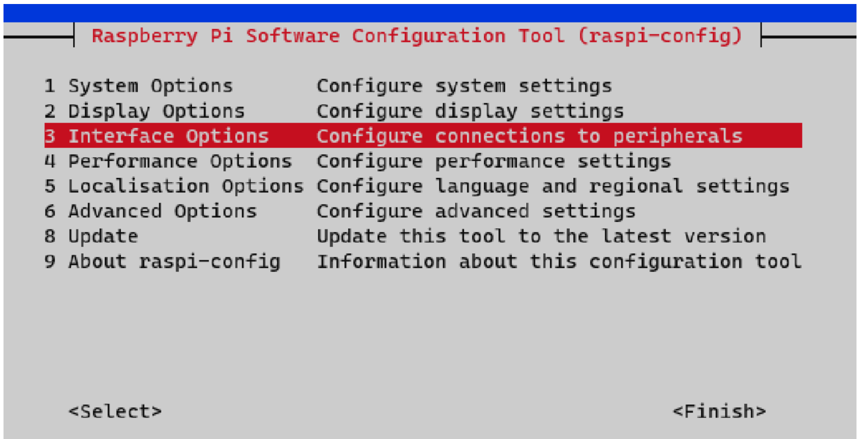


Camera:	● Enabled	○ Disabled
SSH:	○ Enabled	● Disabled
VNC:	○ Enabled	● Disabled
SPI:	○ Enabled	● Disabled
I2C:	○ Enabled	● Disabled
Serial Port:	○ Enabled	● Disabled
Serial Console:	● Enabled	○ Disabled
1-Wire:	○ Enabled	● Disabled
Remote GPIO:	○ Enabled	● Disabled

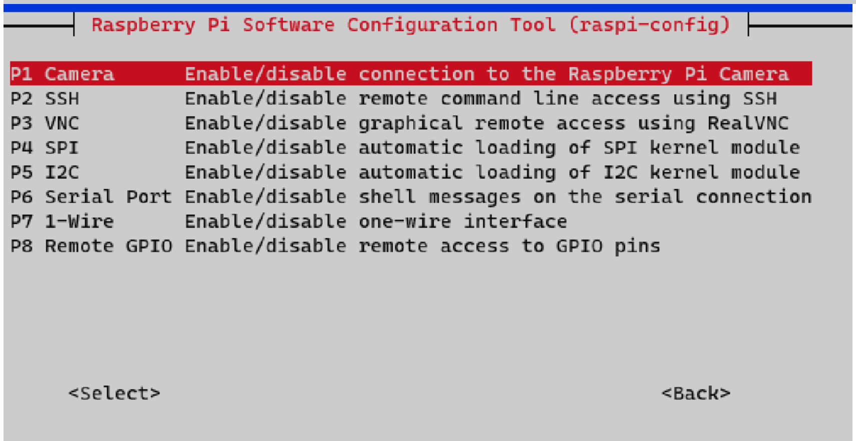
Fig 2. Pi Camera setup

You can also use ssh to enable rpi camera:

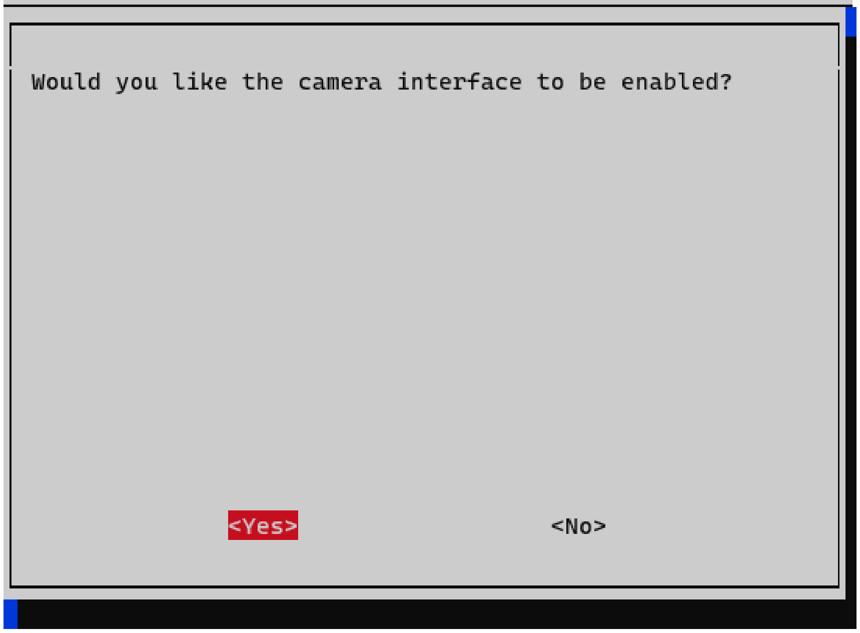
```
sudo raspi-config
```



```
Raspberry Pi Software Configuration Tool (raspi-config)

 1 System Options       Configure system settings
 2 Display Options      Configure display settings
 3 Interface Options    Configure connections to peripherals
 4 Performance Options  Configure performance settings
 5 Localisation Options Configure language and regional settings
 6 Advanced Options     Configure advanced settings
 8 Update               Update this tool to the latest version
 9 About raspi-config   Information about this configuration tool

  <Select>                                            <Finish>
```



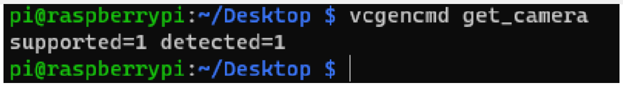
```
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera      Enable/disable connection to the Raspberry Pi Camera
P2 SSH         Enable/disable remote command line access using SSH
P3 VNC         Enable/disable graphical remote access using RealVNC
P4 SPI         Enable/disable automatic loading of SPI kernel module
P5 I2C         Enable/disable automatic loading of I2C kernel module
P6 Serial Port Enable/disable shell messages on the serial connection
P7 1-Wire      Enable/disable one-wire interface
P8 Remote GPIO Enable/disable remote access to GPIO pins

     <Select>                                          <Back>
```



```
Would you like the camera interface to be enabled?

                  <Yes>                      <No>
```

Check the status of camera using:

```
vcgencmd get_camera
```

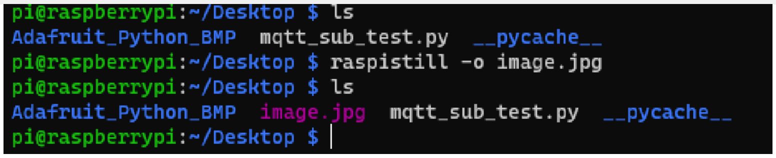


```
pi@raspberrypi:~/Desktop $ vcgencmd get_camera
supported=1 detected=1
pi@raspberrypi:~/Desktop $
```

Supported shows whether rpi support a camera or not. Detected shows whether a camera is detected or not.

Check easily whether your camera works correctly or not. You can also use other image formats such as png.

```
raspistill -o image.jpg
```



```
pi@raspberrypi:~/Desktop $ ls
Adafruit_Python_BMP  mqtt_sub_test.py  __pycache__
pi@raspberrypi:~/Desktop $ raspistill -o image.jpg
pi@raspberrypi:~/Desktop $ ls
Adafruit_Python_BMP  image.jpg  mqtt_sub_test.py  __pycache__
pi@raspberrypi:~/Desktop $
```

Picamera is installed on raspian distro by default. You don't need to install. For the cases you need to install, you can install picamera as:

```
sudo pip install picamera
```

To utilize opencv along side picamera, we need to install picamer[array], since in opencv the images will be represented as Numpy arrays.

```
pip install "picamera[array]"
```

1.1. Use PiCamera using builtin picamera

```
# picam recording

from picamera import PiCamera
from time import sleep

camera = PiCamera()
camera.start_preview()

sleep(5)
camera.capture('image2.jpg')

#camera.start_recording('recorded.h264')
#camera.wait_recording(10)
#camera.stop_recording()

camera.stop_preview()
```

You can change the image setting (e.g., change resolution, frame rate, etc.) and add image effects (e.g., change brightness, contrast, add text, etc.).
More info: https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/7

2. OpenCV:

One easy way to capture pictures or videos from camera is to use OpenCV library.

2.1. Install OpenCV

First, update the system

```
sudo apt-get update
```

Install dependancies

```
sudo apt install libgl1-mesa-glx
```

Install OpenCV

```
sudo pip3 install opencv-python
```

2.2. Take pictures from a camera

The first thing we need to do is to create the VideoCapture class that is responsible for capturing the frames from the camera. When passing the index 0, the OpenCV library knows which driver to use to get the images from the camera.

```
import cv2

cap = cv2.VideoCapture(0)

# Capture frame
ret, frame = cap.read()
if ret:
		cv2.imwrite('image.jpg', frame)

cap.release()
```

If there is an error running above code, you need to install some dependencies. You can use the link below https://stackoverflow.com/questions/53347759/importerror-libcblas-so-3-cannot-open-shared-object-file-no-such-file-or-dire

You also can easily install the below dependencies:

```
pip3 install opencv-python
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
```

2.3. Read images from a source file

```
import numpy as np
import cv2

img = cv2.imread('/path_to_image/my.jpg', 0)  #0 for gray, 1 for color

cv2.imshow('image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2.4. Show a picture using matplotlib

```
import cv2
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(10, 10))
ax.grid(False)

im=cv2.imread('./my_images.jpg')
plt.imshow(cv2.cvtColor(im, cv2.COLOR_BGR2RGB))
plt.show()
```

2.5. How to crop an image

```
import cv2

img = cv2.imread("lenna.png")

y=0
x=0
h=100
w=200

crop_img = img[y:y+h, x:x+w]
cv2.imshow("cropped", crop_img)
cv2.waitKey(0)
```

2.6. Convert an image to grayscale

Formula: https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

2.7. Resize an image

```
scale_percent = 60 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)

# resize image
resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
```

2.8. How filters are working

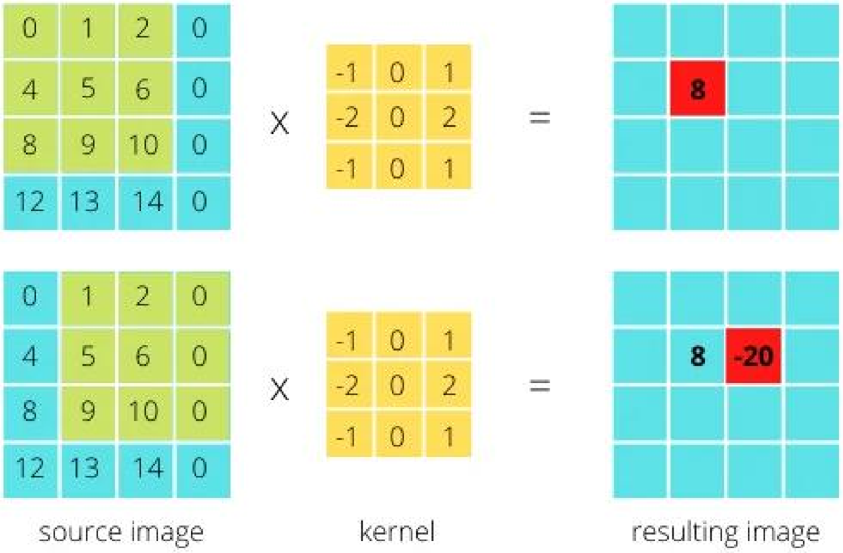


Fig 3. How filters are working

Syntax: *filter2D (src, dst, ddepth, kernel)*

Parameters:

- ***Src –*** *The source image to apply the filter on.*
- ***Dst –*** *Name of the output image after applying the filter*

- ***Ddepth –*** *Depth of the output image [-1 will give the output image depth as same as the input image]*
- ***Kernel –*** *The 2d matrix we want the image to convolve with.*

```
import cv2
import numpy as np

img = cv2.imread("HeliView.jpg")
img = cv2.resize(img, (0, 0), None, .25, .25)

gaussianBlurKernel = np.array(([[1, 2, 1], [2, 4, 2], [1, 2, 1]]), np.float32)/9
sharpenKernel = np.array(([[0, -1, 0], [-1, 9, -1], [0, -1, 0]]), np.float32)/9
meanBlurKernel = np.ones((3, 3), np.float32)/9

gaussianBlur = cv2.filter2D(src=img, kernel=gaussianBlurKernel, ddepth=-1)
meanBlur = cv2.filter2D(src=img, kernel=meanBlurKernel, ddepth=-1)
sharpen = cv2.filter2D(src=img, kernel=sharpenKernel, ddepth=-1)

horizontalStack = np.concatenate((img, gaussianBlur, meanBlur, sharpen), axis=1)

cv2.imwrite("Output.jpg", horizontalStack)

cv2.imshow("2D Convolution Example", horizontalStack)

cv2.waitKey(0)
cv2.destroyAllWindows()
```