vilib

Release 1.0

sunfounder

Dec 23, 2022

CONTENTS

Vilib is a library developed by SunFounder for Raspberry Pi camera.

It contains some practical functions, such as taking pictures, video recording, pose detection, face detection, motion detection, image classification and so on.

CHAPTER

ONE

INSTALL VILIB

Run this command into the terminal to install the vilib library.

```
cd /home/pi/
git clone https://github.com/sunfounder/vilib.git
cd vilib
sudo python3 install.py
```

CHAPTER

TWO

EXAMPLE

2.1 Display

You can view the pictures captured by the Raspberry Pi camera in the browser.

Run the Code

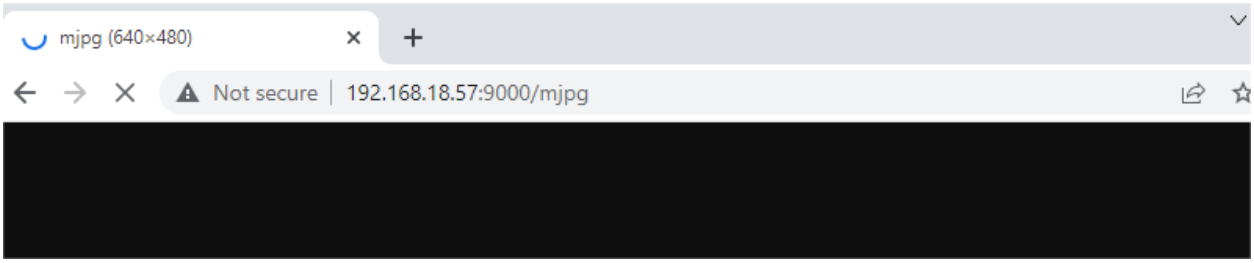
```
cd /home/pi/vilib/examples
sudo python3 display.py
```

View the Image

After the code runs, the terminal will display the following prompt:

```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

Then you can enter `http://<your IP>:9000/mjpg` in the browser to view the video screen. such as: `http://192.168.18.113:9000/mjpg`



Code

```
from vilib import Vilib

def main():
    Vilib.camera_start(vflip=False,hflip=False) # vflip:vertical flip,␣
↪hflip:horizontal Flip
    # local:local display, web:web display
    # when local=True, the image window will be displayed on the system desktop
    # when web=True, the image window will be displayed on the web browser at http://
↪localhost:9000/mjpg
```

(continues on next page)

(continued from previous page)

```
    Vilib.display(local=True,web=True)
    print('\npress Ctrl+C to exit')

if __name__ == "__main__":
    main()
```

How it works?

The content involved in this article is exactly the basic function of the `vilib` library. We have already *Install vilib*.

What you need to focus on is the following:

```
from vilib import Vilib
```

All functions related to computer vision are encapsulated in this library.

```
Vilib.camera_start(vflip=True,hflip=True)
```

Let the camera module enter the working state. If you modify the two parameters of this function to `False`, the screen will be flipped horizontally/vertically.

```
Vilib.camera_close()
```

Stop the camera module.

```
Vilib.display(local=True,web=True)
```

Allows you to see the picture taken by the camera module.

- Its parameter `local=True` is used to open the viewfinder in the Raspberry Pi desktop, which is suitable for remote desktop or the situation where a screen is provided for the Raspberry Pi.
- The parameter `web=True` allows you to view the image through the browser, which is the method suggested in this article. It is suitable for the situation where your PC and Raspberry Pi are connected to the same local area network.

2.2 Take Photo

You can use Raspberry Pi to take a photo and store it in `/home/pi/Pictures/vilib/photos`.

Run the Code

```
cd /home/pi/vilib/examples
sudo python3 take_photo.py
```

After the program runs: 1. You can enter `http://<Your Raspberry Pi IP>:9000/mjpg` in the browser (such as chrome) to view the viewfinder screen. 2. Type `q` in Terminal and press Enter to take a photo.

Code

```
import time
from vilib import Vilib

def main():
    Vilib.camera_start(vflip=False,hflip=False)
    Vilib.display(local=True,web=True)
```

(continues on next page)

(continued from previous page)

```
    path = "/home/pi/Pictures/vilib/photos"

    while True:
        if input() == 'q':
            _time = time.strftime("%y-%m-%d_%H-%M-%S", time.localtime())
            Vilib.take_photo(str(_time),path)
            print("The photo save as:%s/%s.jpg"%(path, _time))
            time.sleep(0.1)


if __name__ == "__main__":
    main()
```

How it works?

What you need to focus on is the following:

```
Vilib.take_photo(photo_name=str(_time),path=path)
```

As the name suggests, this function takes pictures. The first parameter is the name of the generated image file, and the second parameter is the path where the file is located. They can all be customized.

2.3 Record Video

This example allows us to record a video.

Here you will use two windows at the same time: One is Terminal, you can type `q` to record/pause/continue recording, type `e` to stop recording, and type `g` to exit shooting . If the program has not been terminated after exiting the shooting, please type `ctrl+c`. Another browser interface, after the program runs, you will need to enter `http://<Your Raspberry Pi IP>:9000/mjpg` in the PC browser (such as chrome) to view the viewfinder screen.

Run the Code

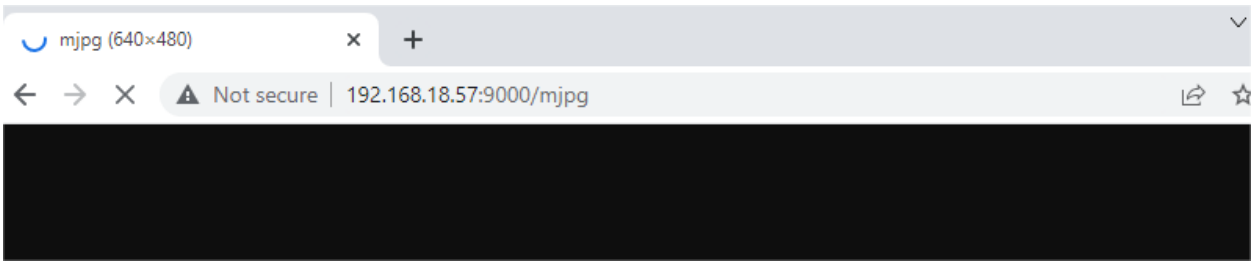
```
cd /home/pi/vilib/examples
sudo python3 record_video.py
```

View the Image

After the code runs, the terminal will display the following prompt:

```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

Then you can enter `http://<your IP>:9000/mjpg` in the browser to view the video screen. such as: `http://192.168.18.113:9000/mjpg`



Code

```
from time import sleep,strftime,localtime
from vilib import Vilib
import readchar

manual = '''
Press keys on keyboard to control recording:
    Q: record/pause/continue
    E: stop
    ESC: Quit
'''

def print_overwrite(msg,  end='', flush=True):
    print('\r\033[2K', end='',flush=True)
    print(msg, end=end, flush=True)

def main():
    rec_flag = 'stop' # start,pause,stop
    vname = None
    Vilib.rec_video_set["path"] = "/home/pi/Videos/" # set path

    Vilib.camera_start(vflip=False,hflip=False)
    Vilib.display(local=True,web=True)
    sleep(0.8)  # wait for startup

    print(manual)
    while True:
        # read keyboard
        key = readchar.readkey().lower()
        # start,pause
        if key == 'q':
            key = None
            if rec_flag == 'stop':
                rec_flag = 'start'
                # set name
                vname = strftime("%Y-%m-%d-%H.%M.%S", localtime())
                Vilib.rec_video_set["name"] = vname
                # start record
                Vilib.rec_video_run()
                Vilib.rec_video_start()
                print_overwrite('rec start ...')
            elif rec_flag == 'start':
                rec_flag = 'pause'
                Vilib.rec_video_pause()
                print_overwrite('pause')
            elif rec_flag == 'pause':
                rec_flag = 'start'
                Vilib.rec_video_start()
```

(continues on next page)

(continued from previous page)

```
                print_overwrite('continue')
        # stop
        elif key == 'e' and rec_flag != 'stop':
            key = None
            rec_flag = 'stop'
            Vilib.rec_video_stop()
            print_overwrite("The video saved as %s%s.avi"%(Vilib.rec_video_set["path
↪"],vname),end='\n')
        # quit
        elif key == readchar.key.CTRL_C or key in readchar.key.ESCAPE_SEQUENCES:
            Vilib.camera_close()
            print('\nquit')
            break

        sleep(0.1)

if __name__ == "__main__":
    main()
```

How it works?

Parameters related to recording include the following:

- `Vilib.rec_video_set["path"]` The address where the video is saved
- `Vilib.rec_video_set["name"]` The name of the saved video

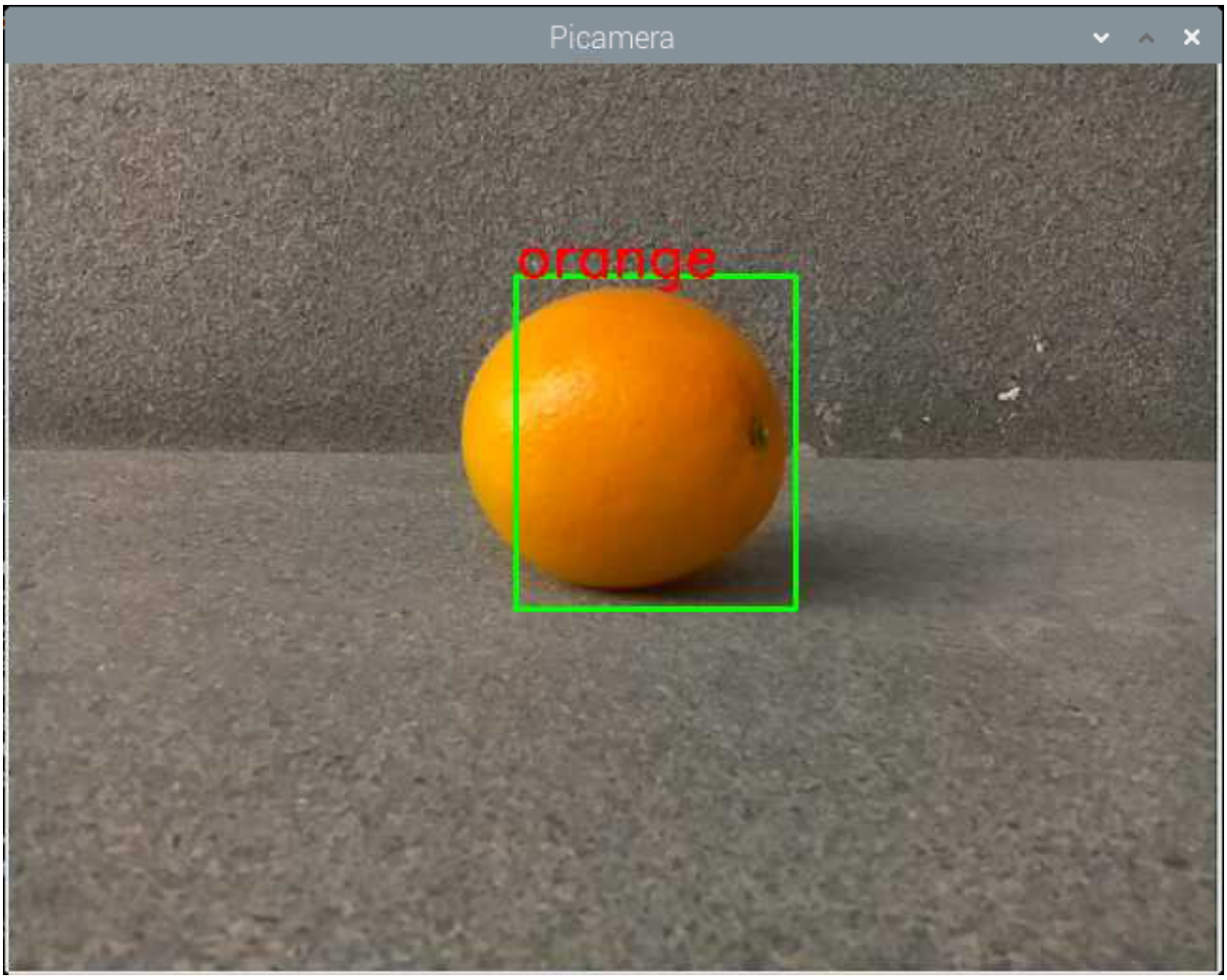
Functions related to recording include the following:

- `Vilib.rec_video_run()` Start recording
- `Vilib.rec_video_pause()` Pause recording
- `Vilib.rec_video_start()` Continue recording
- `Vilib.rec_video_stop()` Stop recording

2.4 Color Detection

Say a color and mark it in the field of view. This is not difficult for most humans, because we have been trained in this way since we were young.

For computers, thanks to deep learning, such tasks can also be accomplished. In this project, there is an algorithm that can find a certain color (6 kinds in total), such as finding "orange".



Run the Code

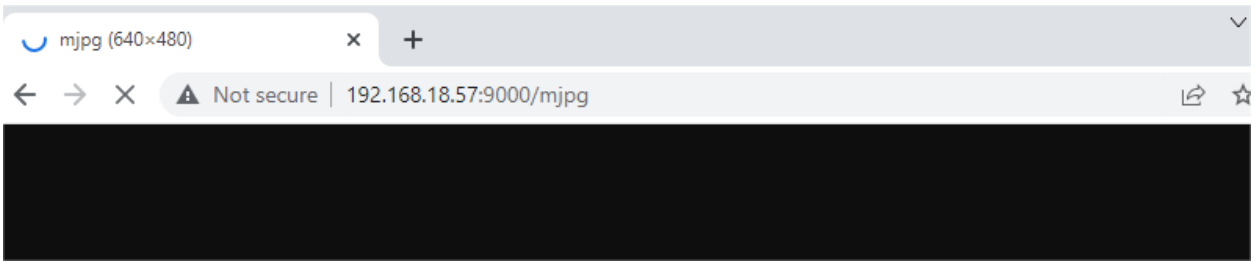
```
cd /home/pi/vilib/examples
sudo python3 color_detection.py
```

View the Image

After the code runs, the terminal will display the following prompt:

```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

Then you can enter `http://<your IP>:9000/mjpg` in the browser to view the video screen. such as: `https://192.168.18.113:9000/mjpg`



Code

```
#!/usr/bin/env python3
from vilib import Vilib
from time import sleep


def main():

    Vilib.camera_start(vflip=False,hflip=False) #
    Vilib.display(local=True,web=True)
    Vilib.color_detect(color="red")  # red, green, blue, yellow , orange, purple
    sleep(1)
    # Vilib.detect_obj_parameter['color_x']    # Maximum color block center
↪coordinate x
    # Vilib.detect_obj_parameter['color_y']    # Maximum color block center
↪coordinate x
    # Vilib.detect_obj_parameter['color_w']    # Maximum color block pixel width
    # Vilib.detect_obj_parameter['color_h']    # Maximum color block pixel height
    # Vilib.detect_obj_parameter['color_n']    # Number of color blocks found

    while True:
        n = Vilib.detect_obj_parameter['color_n']
        print("%s color blocks are found"%n, end=',', flush=True)
        if n != 0:
            w = Vilib.detect_obj_parameter['color_w']
            h = Vilib.detect_obj_parameter['color_h']
            print("the maximum color block pixel size is %s*%s"%(w,h))
        else:
            print('') # new line
        sleep(0.5)



if __name__ == "__main__":
    main()
```

How it works?

The first thing you need to pay attention to here is the following function. These two functions allow you to start the camera.

```
Vilib.camera_start(vflip=True,hflip=True)
Vilib.display(local=True,web=True)
```

Functions related to "color detection":

- `Vilib.color_detect(color)` : For color detection, only one color detection can be performed at the same time. The parameters that can be input are: `"red"`, `"orange"`, `"yellow"`, `"green"`, `"blue"`, `"purple"`

- `Vilib.color_detect_switch(False)` : Switch OFF color detection

The information detected by the target will be stored in the `detect_obj_parameter = Manager().dict()` dictionary.

In the main program, you can use it like this:

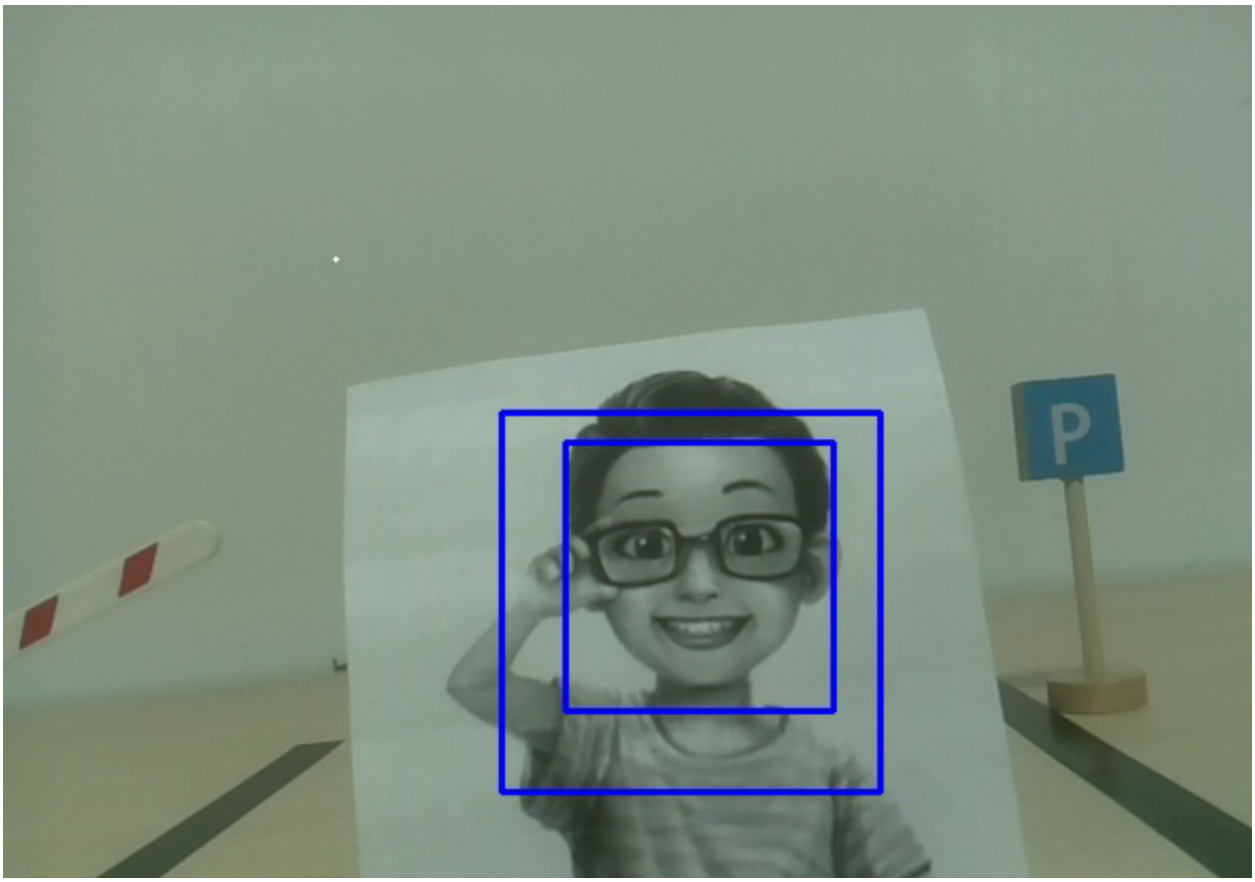
```
Vilib.detect_obj_parameter['color_x']
```

The keys of the dictionary and their uses are shown in the following list:

- `color_x`: the x value of the center coordinate of the detected color block, the range is 0~320.
- `color_y`: the y value of the center coordinate of the detected color block, the range is 0~240.
- `color_w`: the width of the detected color block, the range is 0~320.
- `color_h`: the height of the detected color block, the range is 0~240.
- `color_n`: the number of detected color patches.

2.5 Face Detection

Face detection is the first step in building a face recognition system. It will uniformly calibrate the face according to the key points in the face (such as the position of the eyes, the position of the nose, the position of the mouth, the contour points of the face, etc.) and mark the area containing the face. This technology is widely used in security and other scenarios.



Run the Code

```
cd /home/pi/pan-tilt-hat/examples
sudo python3 face_detection.py
```

View the Image

After the code runs, the terminal will display the following prompt:

```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

Then you can enter `http://<your IP>:9000/mjpg` in the browser to view the video screen. such as: `https://192.168.18.113:9000/mjpg`

Code

```
from vilib import Vilib
from time import sleep

def main():
    Vilib.camera_start(vflip=False,hflip=False)
    Vilib.display(local=True,web=True)
    Vilib.face_detect_switch(True)
    sleep(1)

    while True:
        n = Vilib.detect_obj_parameter['human_n']
        print("%s faces are found."%n)
        sleep(1)

if __name__ == "__main__":
    main()
```

How it works?

Functions related to human face detection:

- `Vilib.face_detect_switch(True)` : Switch ON/OFF face detection

The keys of the dictionary and their uses are shown in the following list:

- `human_x`: the x value of the center coordinate of the detected human face, the range is 0~320
- `human_y`: the y value of the center coordinate of the detected face, the range is 0~240
- `human_w`: the width of the detected human face, the range is 0~320
- `human_h`: the height of the detected face, the range is 0~240
- `human_n`: the number of detected faces

2.6 Read QR Code

The full name of QR Code is Quick Response Code, which is a coding method. It can store more information and represent more data types than the traditional Bar Code.

As a new information storage, transmission and identification technology, QR Code has attracted the attention of many countries in the world since its birth. The United States, Germany, Japan and other countries have not only applied QR Code technology to the management of various documents by public security, diplomatic, military and other departments, but also applied QR Code to the management of various reports and bills by customs, taxation and other departments , the management of commodity and cargo transportation by commercial and transportation departments, the management of postal parcels by the postal department, and the automated management of industrial production lines in the industrial production field.

Run the Code

```
cd /home/pi/vilibt/examples
sudo python3 qr_coder_read.py
```

View the Image

After the code runs, the terminal will display the following prompt:

```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
```

(continues on next page)

(continued from previous page)

```
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

Then you can enter `http://<your IP>:9000/mjpg` in the browser to view the video screen. such as: `https://192.168.18.113:9000/mjpg`

Code

```
from vilib import Vilib
from time import sleep


def main():
    Vilib.camera_start(vflip=False,hflip=False)
    Vilib.display(local=True,web=True)
    Vilib.qrcode_detect_switch(True)

    while True:
        text = Vilib.detect_obj_parameter['qr_data']
        print(text)
        sleep(0.5)

if __name__ == "__main__":
    main()
```

How it works?

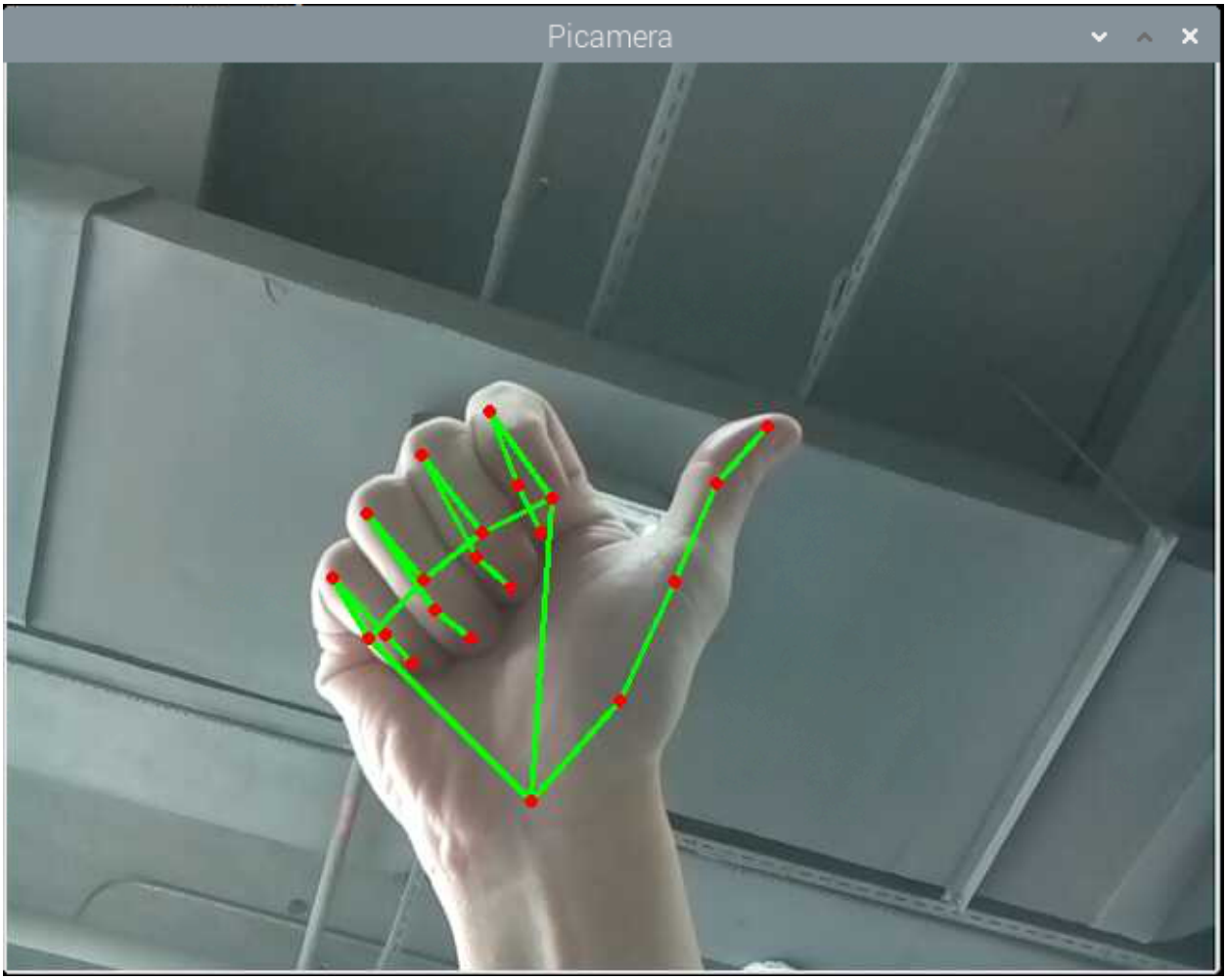
Functions related to human face detection:

- `Vilib.qrcode_detect_switch(False)` : Switch ON/OFF QR code detection, Returns the decoded data of the QR code.

2.7 Hands Detection

Hand detection is a research hotspot in the field of human-computer interaction, allowing people to issue instructions to the computer without the mouse and keyboard. For example, use your fingers to command the robot, or play a guessing game with the robot, and so on.

In this project, you will see that the hand in front of the camera is recognized and the coordinates of the index finger will be printed.



Run the Code

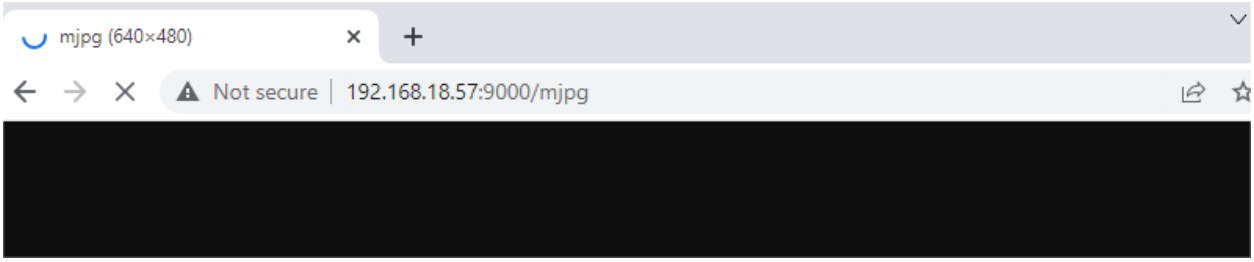
```
cd /home/pi/vilib/examples
sudo python3 hands_detection.py
```

View the Image

After the code runs, the terminal will display the following prompt:

```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

Then you can enter `http://<your IP>:9000/mjpg` in the browser to view the video screen. such as: `https://192.168.18.113:9000/mjpg`



Code

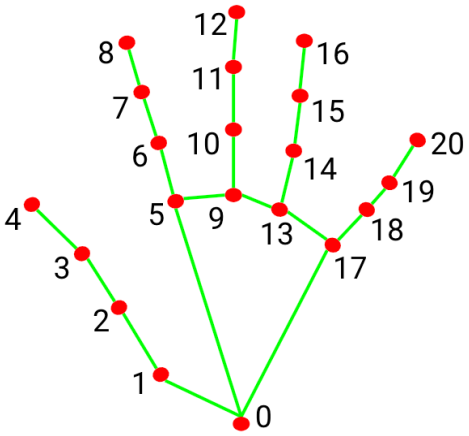
```
#!/usr/bin/env python3
from time import sleep
from vilib import Vilib

def main():
    Vilib.camera_start(vflip=False,hflip=False)
    Vilib.display(local=True,web=True)
    Vilib.hands_detect_switch(True)

    # while True:
    #     print(Vilib.detect_obj_parameter['hands_joints']) # Print finger joint␣
↪coordinates
    #     sleep(0.5)

if __name__ == "__main__":
    main()
```

How it works?

This function has been encapsulated in the `vilib` library, execute `Vilib.hands_detect_switch(True)` to start hand detection.

The information detected by the target will be stored in the `Vilib.detect_obj_parameter['hands_joints']`.

The three-dimensional coordinates of the 21 joints of the hand are stored here, such as the `joints[8]` printed in this article, that is, the coordinates of the index finger. The serial numbers of all joints are shown in the figure below.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

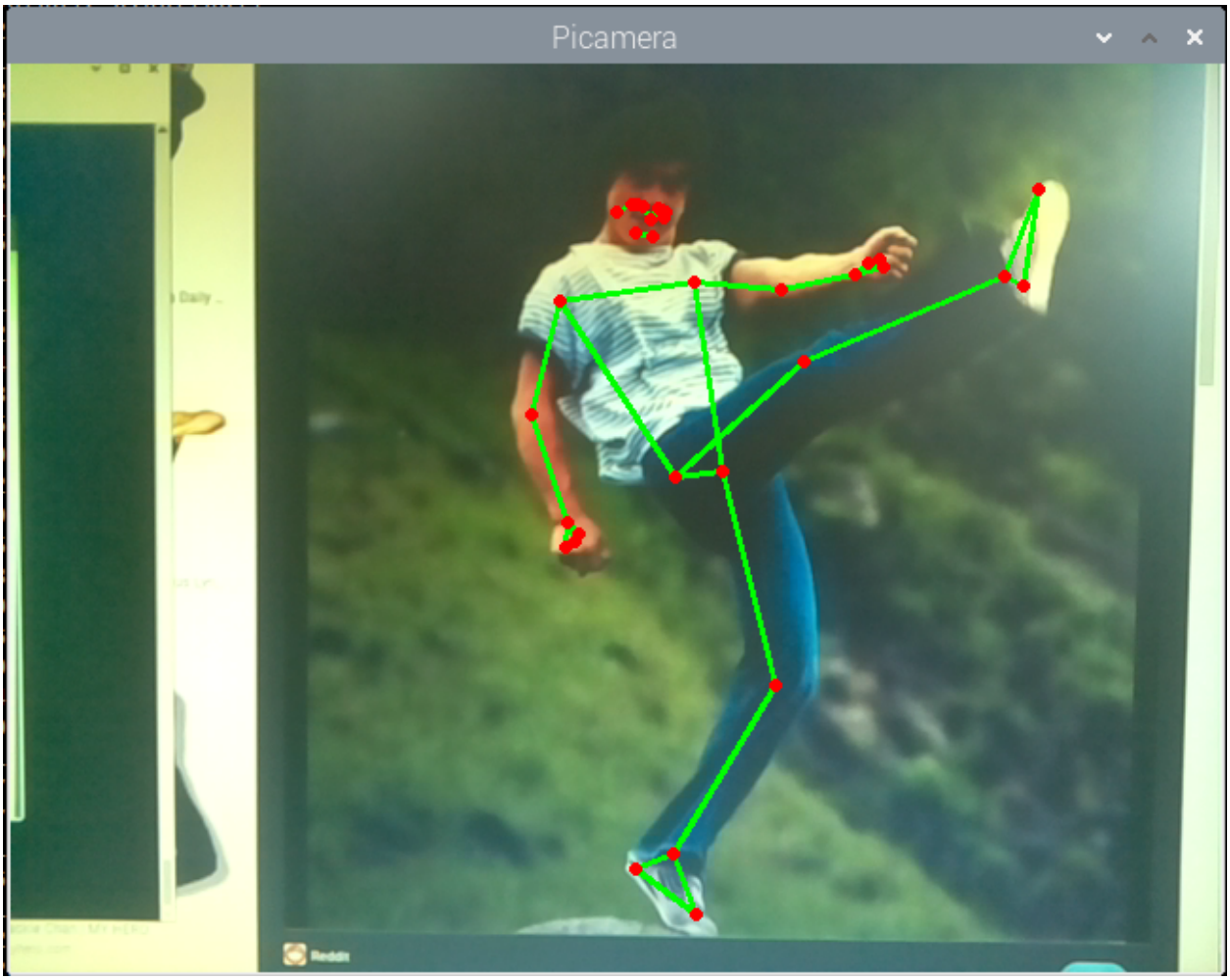
The coordinates of these joints are composed of [x,y,z]. x and y are normalized to 0.0 ~ 1.0 by the width and height of the image (rather than the specific pixel position), the upper left corner of the screen is [0.0, 0.0], and the lower right corner is [1.0, 1.0]. The z coordinate represents the depth, and the wrist depth is the origin. The smaller the value, the closer the landmark is to the camera. The size of z uses roughly the same ratio as x.

This feature is based on MediaPipe Google, please see more knowledge in Hands - MediaPipe .

2.8 Pose Detection

Like hand detection, posture detection is also part of the field of human-computer interaction. It can be used to judge action instructions, identify dances, quantify exercises and other scenes.

In this project, you will see that the human body in front of the camera is recognized and the coordinates of the left wrist are printed.



Run the Code

```
cd /home/pi/vilib/examples
sudo python3 pose_detection.py
```

View the Image

After the code runs, the terminal will display the following prompt:

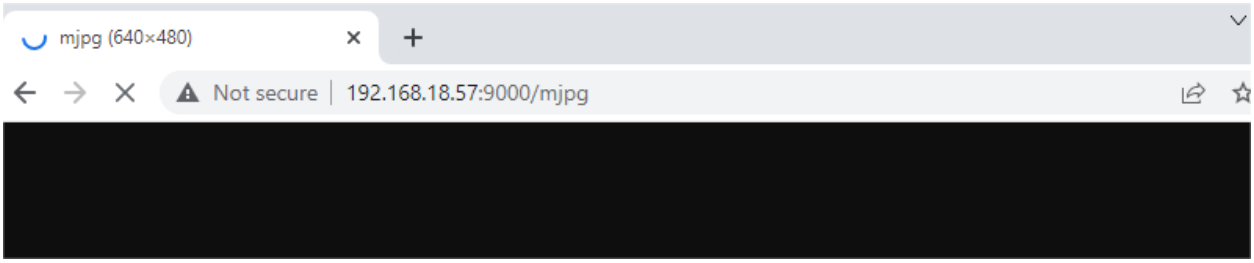
```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
```

(continues on next page)

(continued from previous page)

```
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

Then you can enter `http://<your IP>:9000/mjpg` in the browser to view the video screen. such as: `https://192.168.18.113:9000/mjpg`



Code

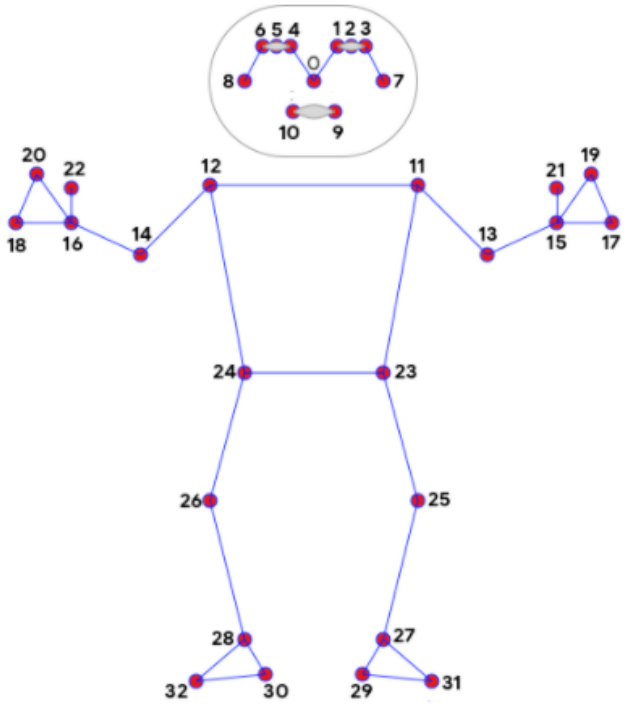
```python
#!/usr/bin/env python3
from vilib import Vilib


def main():

    Vilib.camera_start(vflip=False,hflip=False)
    Vilib.display(local=True,web=True)
    Vilib.pose_detect_switch(True)

if __name__ == "__main__":
    main()
```

How it works?

This function has been encapsulated in the `vilib` library, execute `Vilib.pose_detect_switch(True)` to start gesture detection.

The information detected by the target will be stored in the `Vilib.detect_obj_parameter['body_joints']`.

The three-dimensional coordinates of the 33 joints of the hand are stored here, such as the `joints[15]` printed in this article, which is the coordinates of the left wrist. The serial numbers of all joints are shown in the figure below.



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

The coordinates of these joints are composed of [x,y,z,visiblity]. x and y are normalized to 0.0 ~ 1.0 by the width and height of the image (rather than the specific pixel position), the upper left corner of the screen is [0.0, 0.0], and the lower right corner is [1.0, 1.0].

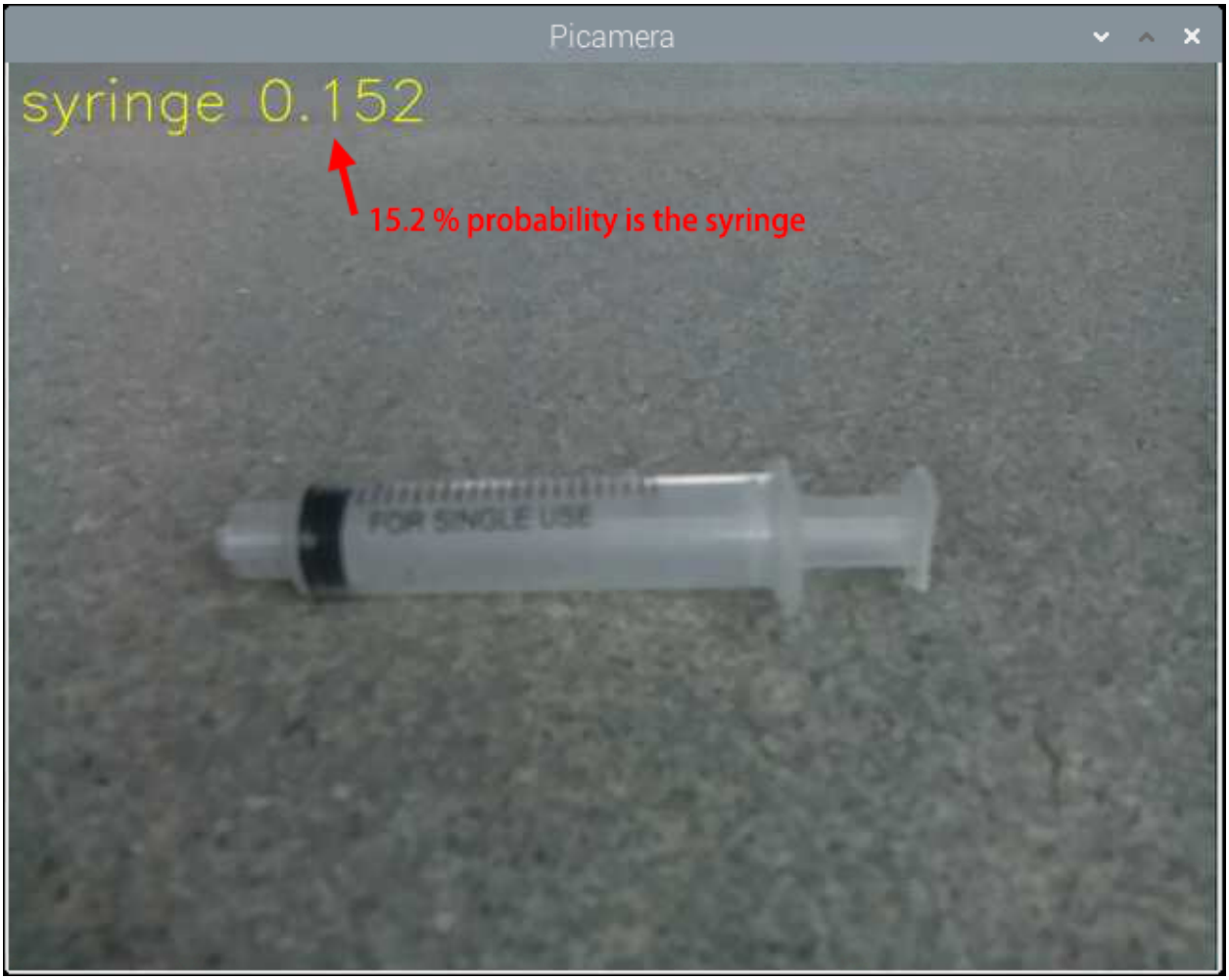
The z coordinate represents the depth, with the crotch (between 23 and 24) depth as the origin. The smaller the value, the closer the landmark is to the camera. The size of z uses roughly the same ratio as x. visibility is a value indicating the possibility that the joint is visible (existing and not occluded) in the image, and its range is between 0.0 and 1.0.

This feature is based on MediaPipe Google, please see more knowledge in Pose - MediaPipe .

2.9 Image Classification

When you see a rabbit, you will directly say its name. The computer will tell you that 91% of it may be a rabbit, 4% of it may be a cat, 3% of it may be a dog, and 2% of it may be something else.

We use image classification to reveal the cognitive model of computer vision. Image classification is a basic task. A trained model is used to identify images representing various objects, and then the images are assigned to specific tags to classify the images.



Run the Code

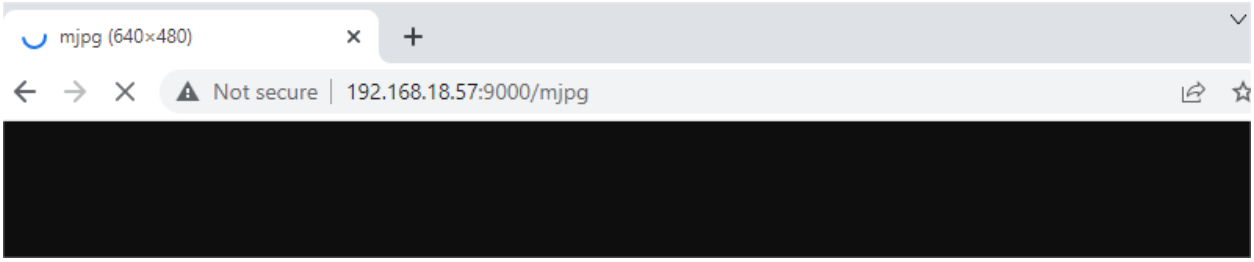
```
cd /home/pi/vilib/examples
sudo python3 image_classification.py
```

View the Image

After the code runs, the terminal will display the following prompt:

```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

Then you can enter `http://<your IP>:9000/mjpg` in the browser to view the video screen. such as: `https://192.168.18.113:9000/mjpg`



Code

```
from vilib import Vilib

def main():
    Vilib.camera_start(vflip=False,hflip=False)
    Vilib.display(local=True,web=True)
    Vilib.image_classify_set_model(path='/home/pi/vilib/mobilenet_v1_0.25_224_quant.
↪tflite')
    Vilib.image_classify_set_labels(path='/home/pi/vilib/labels_mobilenet_quant_v1_
↪224.txt')
    Vilib.image_classify_switch(True)

if __name__ == "__main__":
    main()
```

How it works?

The use of this feature is very simple. You only need to pay attention to the following three lines of code:

- `Vilib.image_classify_set_model(path)` : Load the trained model file.
- `Vilib.image_classify_set_labels(path)` : Load the corresponding label file.
- `Vilib.image_classify_switch(True)` : Start the image classifier.

Here, we directly use Tensorflow pre-trained model, which is an image classification model that includes thousands of objects. You can open the label file (`/home/pi/vilib/labels_mobilenet_quant_v1_224.txt`) to see which objects are included.

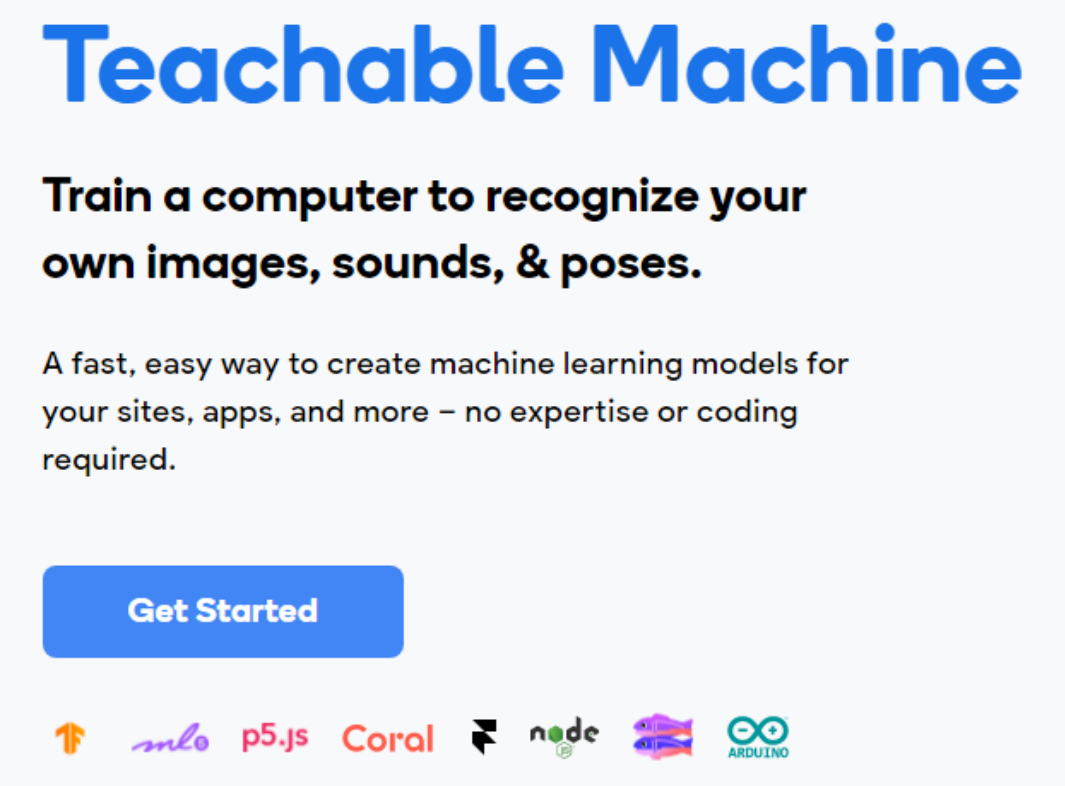
In addition to the built-in models in this article, you can also download the pre-trained image classification model on TensorFlow Hub. It should be noted that these models may not be suitable for your project, please use them as appropriate.

If you want to try to create your own model. We strongly recommend that you use Teachable Machine. It is a web-based tool that allows everyone to create machine learning models quickly, easily, and accessible. Please click Get start on the webpage to start training your model.

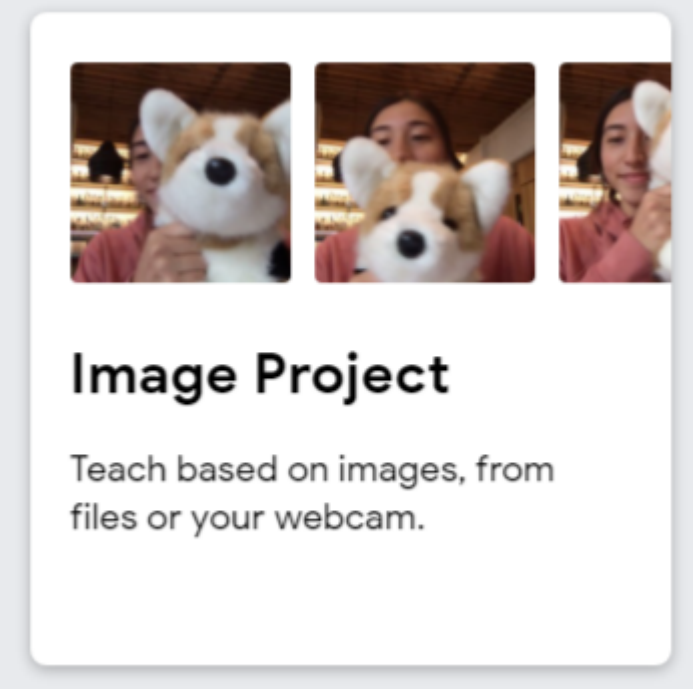
Note: Raspberry Pi may not be able to use Teachable Machine smoothly. You will need to prepare a PC or laptop equipped with a camera.

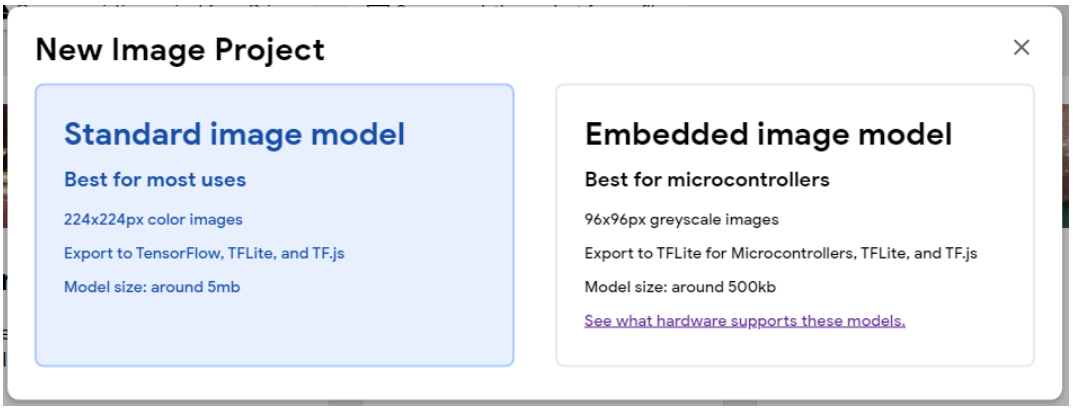
Model Training

1. Open Teachable Machine, you will see an obvious Get Start on the web page, click on it.



2. Select Image Project (Audio Project and Pose Project are not applicable here). You will be prompted to choose Standard image model or Embedded image model. The former has a higher accuracy rate and the latter has a faster speed. We recommend choosing the first one.

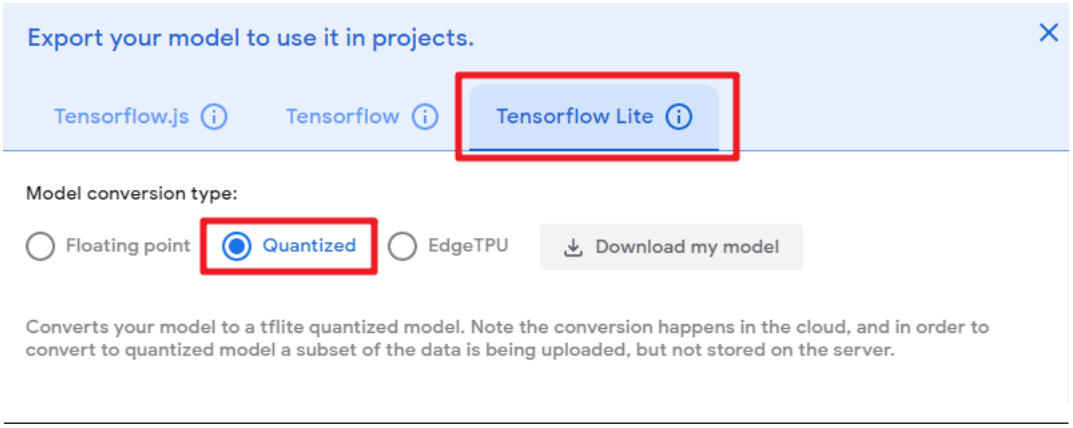




3. Train the model. Teachable Machine provides a detailed video step-by-step explanation, please see:

Note: The video after 0:55 is the content of the other two projects and is not applicable here.

Note: The export settings applicable to this project are shown in the figure:



---

4. Unzip the downloaded zip file, you will be able to see the *model file* and *label file*, their formats are `.tflite` and `.txt` respectively. Use **Filezilla Software** to copy them to the `/home/pi/vilib/` directory of the Raspberry Pi.

5. Modify the two lines of the sample code in this article, and change them to your model and label.

```
Vilib.image_classify_set_model(path='/home/pi/vilib/your_model.tflite')
Vilib.image_classify_set_labels(path='/home/pi/vilib/your_label.txt')
```

6. Re-run the example. It will recognize the objects in your training model.