Hiwonder TurboPi Custom Face Tracking

Manual

Contents

[Part 1. Face Tracking Specific face 2](#_Toc192452488)

[1. Opencv Tutorial 2](#_Toc192452489)

[2. Create your own dataset 2](#_Toc192452490)

[3. Training our face recognition model 3](#_Toc192452491)

[4. Run face recognition model on Hiwonder 6](#_Toc192452492)

[5. Transfer the files to Hiwonder 8](#_Toc192452493)

[6. Run the New FaceTracking Function on Hiwonder. 10](#_Toc192452494)

## Part 1. Face Tracking Specific face

### Opencv Tutorial

Now we are familiar with Hiwonder and play with it a lot. The next step is we are going to use the Hiwonder hardware and code to create our own car which can track our own face.

The Opencv Tutorial can help us get familiar with Opencv library and know all the things we need next. Here is a tutorial link you can use:

**OpenCV with Python in 4 Hours**

Video link: <https://www.youtube.com/watch?v=oXlwWbU8l2o&t=114s>

The Github code of Tutorial: <https://github.com/jasmcaus/opencv-course>

### Create your own dataset

To train our own model, we first need a suitable dataset. You can download the dataset from the GitHub repository for the OpenCV course: <https://github.com/jasmcaus/opencv-course> .

**Navigate to the Resource → face → train** folder, which serves as the training dataset.

A screenshot of a computer

Description automatically generated

Next, create a new folder named after yourself within the train folder and add several photos of your face in **JPG format**. A screenshot of a computer

Description automatically generated

Ensure the photos include a variety of angles and lighting conditions to improve the model's robustness.

### Training our face recognition model

We will use OpenCV to create our first face recognition model.

The Python script for training (self\_training\_py) can be found at the link:

Part 10. Face Tracking Specific face files

https://drive.google.com/drive/folders/1yQc3ZrAOUB72UqF8yJErDqKeev3qj14Z?usp=sharing

You have two options:

1. **Write your own training script**
2. **Use the provided Python script from the link.** If you choose to use the script from the link, make sure to replace the place holder name "Your name" with the actual name that matches your training folder to ensure proper dataset labeling.

* I**mport** the required libraries: **OpenCV and NumPy**, as these are the only libraries we will be using.

A close up of white text

Description automatically generated

* List the names of trained people: If we use the resources from the **OpenCV with Python in 4 Hours** GitHub repository, we will have access to your own photos along with several photos of celebrities.A screen shot of a computer code

  Description automatically generated
* To create the directory for the training dataset and define all the labels we will use. To detect faces, we will use the **CascadeClassifier**, an OpenCV face detection algorithm.

The **CascadeClassifier** is a traditional object detection algorithm in OpenCV that uses Haar or LBP features combined with a cascade of boosted classifiers (Adaboost) to detect objects like faces efficiently. Unlike deep learning, it relies on manually engineered features and simpler models, making it less robust and accurate but faster and lightweight for specific tasks.

The necessary **haarcascade\_frontalface\_default.xml** file for face detection can be found in the GitHub repository under the **Section #3 → Face folder.**

* Next, Set the width of the images to 480 pixels, as the Hiwonder camera operates at a resolution of 640 × 480 pixels. This ensures compatibility with the camera's output.

A screen shot of a computer program

Description automatically generated

We will create a function to prepare the training dataset by detecting faces in images, extracting their regions of interest (ROI), and storing both the ROIs and corresponding labels.

* Combine the path for each person's image: Use a function to iterate through the dataset directory and access the images for each person.
* Read and resize the image: Ensure all images are resized to a fixed width of 480 pixels to match the camera's resolution.
* Convert the image to grayscale and reduce noise: Grayscale conversion simplifies processing, and adding a blur helps reduce noise for more accurate face detection.
* Detect faces using PID: Use the Haar Cascade classifier to detect faces in the images. Parameters like scaleFactor=1.1 and minNeighbors=4 are good starting points:

scaleFactor: Controls how much the image size is reduced at each scale during detection.

minNeighbors: Specifies how many neighbor rectangles each candidate face must have to be considered a valid detection. Adjust these values as needed for your dataset.

* Extract the ROI for each detected face: Loop through all detected faces, extract their ROIs, and store them along with the corresponding labels. This ensures that only the face regions are used for training.

A screen shot of a computer program

Description automatically generated

* Call the **create train()** function to prepare the training dataset and convert the features and labels lists into NumPy arrays for compatibility with the face recognition model.
* Create the Local Binary Patterns Histograms (LBPH) face recognizer to train the model. After training, save the trained model along with the features and labels arrays to a file.

You can choose to save the file to the default path or specify a custom location. To determine where the file is saved, use the os.getcwd() function, which will display the current working directory.A screen shot of a computer program

Description automatically generated

### Run face recognition model on Hiwonder

For now, we can modify the existing face tracking functionality with our new trained model to track our own face!

The Function we are going to use and modify on Google drive -- <https://drive.google.com/drive/folders/1Y9FdmRe_h6JPQ0ggJe8bi1GDT4IsESSV> TurboPi – TurboPi RPI 4B Version Source Code – TurboPi – Functions – FaceTracking.py

A screenshot of a computer

Description automatically generated

It is the Face Tracking Program which we are quite familiar with. The face tracking program is face detection based on mediapipe. Now we are going to use the opencv to detect and track our own face.

* Open the FaceTracki5ng.py. Save the file name as FaceTracking\_opencv.py.

A screenshot of a computer

Description automatically generated

* Replace mediapipe face Detection Initialization replace by OpenCV haar\_cascade.

A screenshot of a computer program

Description automatically generated

* Change the run function with haar\_cascade face detection.

A screen shot of a computer code

Description automatically generated

* Detect face faces in camera. Label the detected face with a confidence score. If the detected face is either not yours or the system is uncertain about it being yours, only draw a box around the face with a label. If the system is confident it is your face (indicated by a **lower confidence score**), it will actively track the face.

A screen shot of a computer program

Description automatically generated

After rewriting these two parts of the face tracking function, we can run the new face tracking function on our Hiwonder to see if it works or not.

The function we modified and referenced can be found at the link:

Part 10. Face Tracking Specific face files

<https://drive.google.com/drive/folders/1yQc3ZrAOUB72UqF8yJErDqKeev3qj14Z?usp=sharing>

### Transfer the files to Hiwonder

Now we have all the files we are going to use. To transfer the documents to Hiwonder by using WinSCP. How to use WinSCP please check: [WinSCP download and connection](#_WinSCP_download_and).

Transfer all the documents we have to Hiwonder function folder including

* haar\_face.xml
* face\_trained.yml (the files after you trained with your own dataset)
* FaceTracking\_opencv.py

The files you referenced can be found at the link:

Part 10. Face Tracking Specific face files

<https://drive.google.com/drive/folders/1yQc3ZrAOUB72UqF8yJErDqKeev3qj14Z?usp=sharing>

* Open WonderPi APP press and hold the car icon to show the IP address of Hiwonder.

A screen shot of a device information

Description automatically generated

* Open WinSCP and input the Host name (Device IP), User name(pi) and password(raspberry).

A screenshot of a computer

Description automatically generated

* Open folder TurboPi on Raspberry Pi – Function

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Drag **haar\_face.xml, face\_trained.yml, FaceTracking\_opencv.py** into function folder.

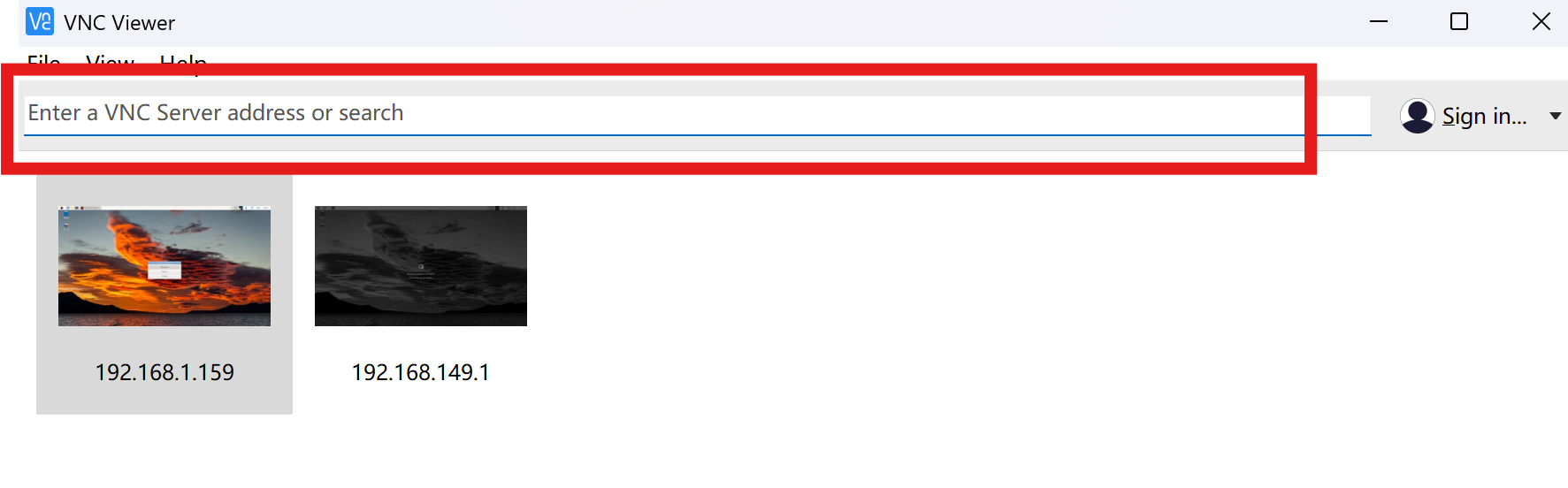
A screenshot of a computer

Description automatically generated

### Run the New FaceTracking Function on Hiwonder.

After we finish transferring the documents we need into Hiwonder, we can use VNC viewer to connect to Hiwonder. If you forget how to use VNC viewer, please review [Part 6. Remote Desktop Installation and Connection](#_Part_5._Remote).

* Open VNC viewer and input the IP address in box



* Connect to Hiwonder.

A sunset over water with mountains in the background

Description automatically generated

* Double click or press “Ctrl+Alt+T” to enter the LX terminal.

A black rectangular object with white border

Description automatically generated

* Input command “cd TurboPi/Functions/” and press Enter to enter the catalog where game programs are stored.



* Input command “sudo python3 FaceTracking\_opencv.py”.



* Running the face tracking program.

A screenshot of a computer

Description automatically generated

**The function runs successfully! We have an autonomous car that tracks only our face!**