

Openflexure Object Detection User Guide

Borys Zyto

Last updated August 2023

Introduction

Welcome to the user guide of implementing object detection features into openflexure based microscope. By following the steps below you will be able to implement the machine learning functionalities from scratch. Should you have any insolvable issues please contact me at

borys.zyto@student.manchester.ac.uk

or if you are reading this after June 2024, you should probably try my personal email

borys.zyto@gmail.com

Keep in mind that by the time you are reading this, I am probably no longer working on the EAC experiment, and so I might have limited capabilities.

1 Installing libraries

Performing object detection on the Raspberry Pi relies on three libraries

1. `tflite-runtime`,
2. `numpy`,
3. `opencv2`.

`numpy` and `opencv2` should be already installed, however we must pay special attention to installing `tflite-runtime`. If you have not already done so please install openflexure OS on the Raspberry Pi. You can follow the steps at the official openflexure website. To avoid complications please make sure to install the version with desktop interface.

Having installed the OS you can plug in your pi and connect it to the internet. Next, connect to the pi via ssh from your computer. We choose to do it from the terminal but you could as well use PuTTY.

```
1 $ssh pi@ip_of_the_raspberry
```

You should now get a prompt to type password. The default password is `openflexure`. Once logged in, you have to change user permissions so that

you can install custom libraries into the microscope app. See official openflexure website

```
1 pi@microscope:~$ sudo chmod -R g+w /var/openflexure/  
2 pi@microscope:~$ sudo adduser pi openflexure-ws  
3 pi@microscope:~$ ln -s /var/openflexure/application\  
4 /openflexure-microscope-server/~pi/
```

You should be now able to install libraries onto the server. First, activate the virtual environment.

```
1 pi@microscope:~$ ofm activate
```

You should be now seeing something like

```
1 (OFM Server)pi@microscope:~$
```

Now, it is time to install libraries.

```
1 (OFM Server)pi@microscope:~$ pip3 install --upgrade pip  
2 (OFM Server)pi@microscope:~$ pip3 install tfllite-runtime
```

This may take some time. In your terminal you might see the following output.

```
1 running setup.py bdist_wheel for numpy ...
```

Do not worry if it takes way too long than you expected.

If everything went successfully you should be now able to import the libraries in python. You can check by typing the following commands

```
1 (OFM Server)pi@microscope:~$ python3  
2 ...  
3 >>>import tfllite_runtime  
4 >>>tfllite_runtime.__version__
```

If you get version 2.11.0, or higher, this means that you managed to install the libraries.

2 Working with the scan.py file

scan.py is the file responsible for managing scans- the functionality of interest. We now have one **scan.py** file that we can use to set up single or multi sample scans with or without object detection features. The code is available at my github. Please download the **scan.py** file and the .tflite model. Place the tflite model in the same directory we store **scan.py**. Once you have modified the code according to your needs you should place the **scan.py** file in location

```
1 /var/openflexure/application/  
2 openflexure-microscope-server/openflexure_microscope/  
3 api/default_extensions/scan.py
```

Next restart the server using the command

```
1 (OFM Server)pi@microscope:~$ sudo ofm restart
```

We will now consider two scanning scenarios.

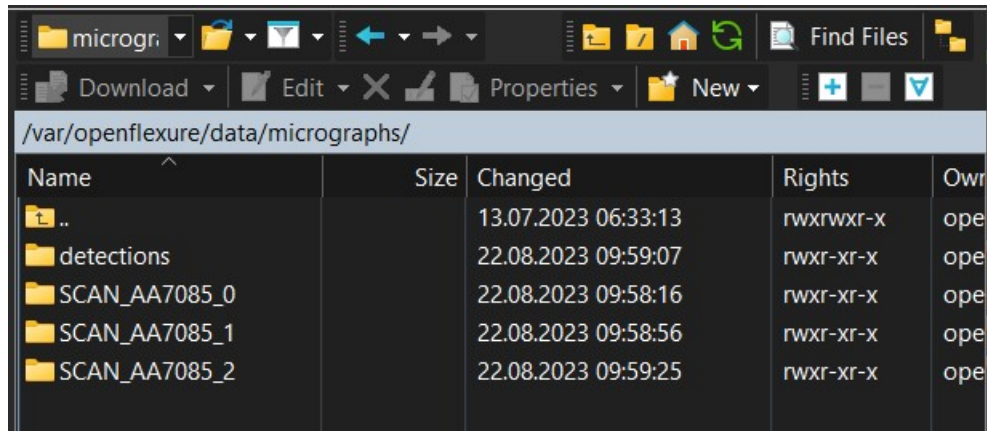
2.1 Scanning with detection

Assume you would like to run a single sample scan of AA7085 alloy with crack detection. First, find lines 677-680 in the `scan.py` script found in my github repository. We then modify the lines accordingly

```
1 scan_array = [["AA7085", (0, 0, 0)]]
2 n_of_scans = 10000
3 detect = True
4 save = True
```

Listing 1: single scan experiment setup

Executing the code with the parameters above will lead to running 10000 scans with detection enabled. Images with cracks detected by the machine learning model will be saved to a `detections` directory. Overall, the directory with images will have the following structure.



Name	Size	Changed	Rights	Owr
..		13.07.2023 06:33:13	rw-rw-r-x	ope
detections		22.08.2023 09:59:07	rw-r-xr-x	ope
SCAN_AA7085_0		22.08.2023 09:58:16	rw-r-xr-x	ope
SCAN_AA7085_1		22.08.2023 09:58:56	rw-r-xr-x	ope
SCAN_AA7085_2		22.08.2023 09:59:25	rw-r-xr-x	ope

Figure 1: Example directory structure

In the `detections` folder, you can find images with cracks in bounding boxes.

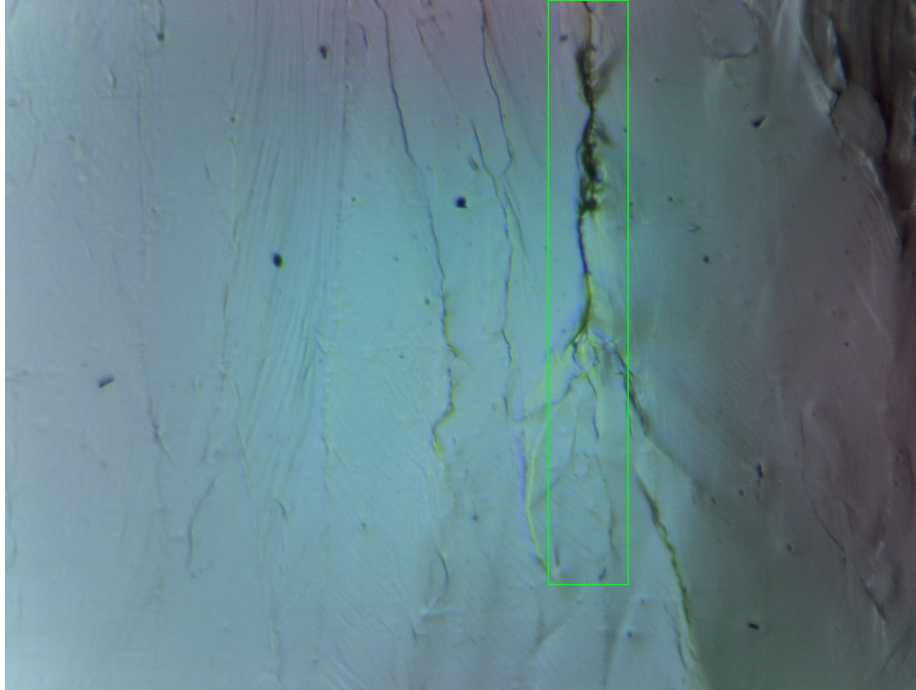


Figure 2: Example output produced by the model

Unfortunately, the images with boxes bounding detected cracks significantly increase the size and number of files stored on a Raspberry Pi. Should you wish to be able to store more data on the Raspberry Pi, you can change the **save** parameter to **False**. In such cases, you can check the model output by looking at the **detections.txt** file. It can be found in all **SCAN_...** directories. Once you open it you will find filenames of images where the model detected cracks, for example

```
AA7085_0_0_0_-31.jpeg  
AA7085_0_0_640_-27.jpeg  
AA7085_0_800_640_-20.jpeg  
AA7085_0_800_0_-80.jpeg
```

The structure of naming jpeg files is
alloy_scan_number_x_y_z.jpeg

2.2 Multi-sample scan

Now, assume you want to scan three samples AA7085, AA7049 and AA7885. Since running three scans generates a lot of data, and detecting cracks is a lengthy process, we discourage using the detection feature on more than one sample scans. Hence, we will run a simple scan without detecting cracks. Again, locate the lines 677-680 and modify them accordingly

```

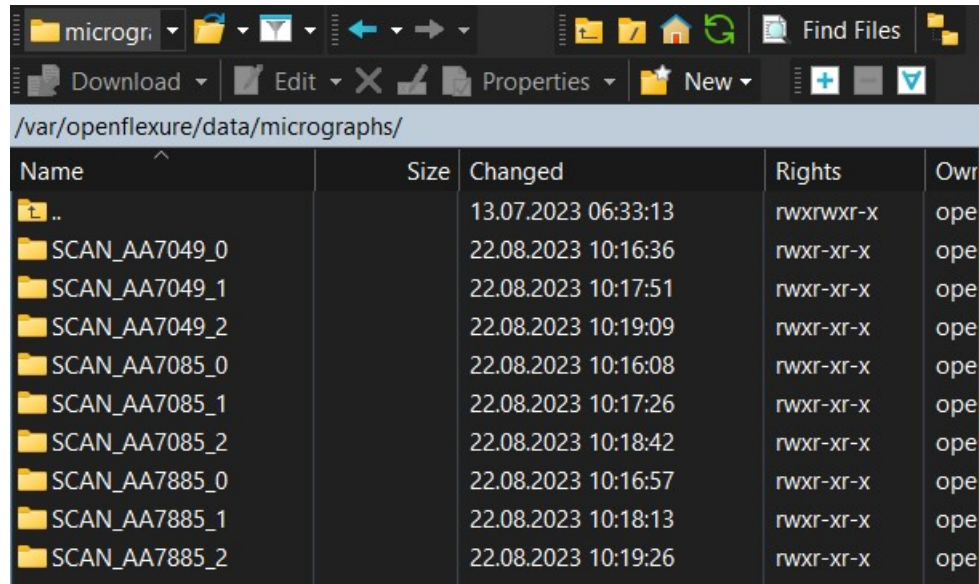
1 scan_array = [["AA7085", (0, 0, 0)], ["AA7049", (1600, 1600
    , 0)], ["AA7885", (3200, 3200, 0)]]
2 n_of_scans = 10000
3 detect = False
4 save = False

```

Listing 2: multi scan experiment setup

The second value in each "alloy array" corresponds to the right bottom corner of the sample.

Executing the code above will result in running 10000 scans with the following directory structure



Name	Size	Changed	Rights	Owr
..		13.07.2023 06:33:13	rw-rw-r--	ope
SCAN_AA7049_0		22.08.2023 10:16:36	rw-r--r--	ope
SCAN_AA7049_1		22.08.2023 10:17:51	rw-r--r--	ope
SCAN_AA7049_2		22.08.2023 10:19:09	rw-r--r--	ope
SCAN_AA7085_0		22.08.2023 10:16:08	rw-r--r--	ope
SCAN_AA7085_1		22.08.2023 10:17:26	rw-r--r--	ope
SCAN_AA7085_2		22.08.2023 10:18:42	rw-r--r--	ope
SCAN_AA7885_0		22.08.2023 10:16:57	rw-r--r--	ope
SCAN_AA7885_1		22.08.2023 10:18:13	rw-r--r--	ope
SCAN_AA7885_2		22.08.2023 10:19:26	rw-r--r--	ope

Figure 3: Multi sample scan directory structure

Object detection features have been disabled and so we do not have to worry about the data storage.

3 Troubleshooting and tips

1. Please make sure that the directory `micrographs` with `SCAN_` folders is empty whenever you start a new scan.
2. If you get `Segmentation Error` anywhere, this means that your `tf-lite-runtime` library is not up to date. Update `pip` and then update the library.
3. Please make sure that you are using the 'coordinates' naming style, when you are setting up the scan.

4. We suggest that you run scanning with detections until you are able to locate the area with cracks, and then you set up a new scan, without detections, on a refined grid.
5. Make sure that the variable `modelpath` in line 34 of `scan.py` gives a real path to the tflite model file.
6. You need the object detection library installed on the Raspberry Pi, no matter if you run the detection or not.

4 Useful Resources

1. https://github.com/bzyto/openflexure/tree/main/openflexure_github
- my github repository with all the code you need in the world
2. <https://gitlab.com/openflexure/openflexure-microscope-server> -
official OpenFlexure repository
3. <https://openflexure.gitlab.io/microscope-handbook/develop-extensions/index.html> - official guide on developing on openflexure software.