**Automatic\_FissionTrack\_identification**

This code uses the deep learning method to identify and count fission tracks.

The source codes are available for downloading at the link: (https://github.com/andyzz613121/Automatic\_FissionTrack\_identification)

Contact: sc20184@zju.edu.cn; royang1985@zju.edu.cn; 11938030@zju.edu.cn; [21938015@zju.edu.cn](mailto:21938015@zju.edu.cn)

School of Earth Sciences, Zhejiang University, Hangzhou, China,

# Requirements

This code is based on Python & Pytorch, and requires GPU support.

Before using this code:

## Environment

Please make sure the GPU environment is configured correctly:

1. **CUDA** (version of 11.1)
2. **cuDNN** (version of 8.1.1)

## Packages:

Please install the following packages, including:

1. **Anaconda**, which is an open source distribution of Python and including lots of science packages and their dependencies. **Please install the Anaconda with Python version of 3.6.5.**

Install the following packages by ‘**pip install ……**’

1. **Numpy** (version of 1.19.5).
2. **GDAL** (version of 3.0.2)
3. **opencv-python** (version of 4.5.4.58)
4. **scikit-image** (version of 0.17.2)
5. **torch** (version of 1.8.0+cu111)
6. **torchaudio** (version of 0.8.0)
7. **torchvision** (version of 0.9.0+cu111)

# Folder structure

The folder is organized as follow, where ‘A’ denotes images of spontaneous tracks, ‘M’ denotes images of induced tracks, ‘R’ denotes images under reflected light.

## FT\_Data: The images for training and testing (need to be count)

**┠─training** *(The data used for training)*　　**┠─A** *(The training data of A)*  
　　　**┠─image\_patch** *(Training images)*  
　　　**┠─label\_patch** *(Training labels of semantic segmentation)*  
　　　**┠─edge\_label\_patch**  *(Training labels of edge extraction)*  
　　　**┠─raw\_big\_image** *(The raw images and labels)*  
　　**┠─M**　*(The training data of M, similar as A)*  
　　**┠─train\_A\_Seg.csv**　 *(The training image filenames of A for semantic segmentation)*  
　　**┠─train\_M\_Seg.csv**　 *(The training image filenames of M for semantic segmentation)*  
　　**┠─train\_HED (Contains\_FT).csv**　 *(The training image filenames of A and M for boundary extraction, where all the images are all contain fission tracks)*

**┠─testing**　 *(The data used for testing and counting)*  
　　**┠─B1**　*(The sample that need to be count)*  
　　　**┠─image**　 *(The images of fission tracks)*  
　　　　**┠─A**  
　　　　**┠─M**  
　　　　**┠─R**  
　　　**┠─area\_raster**　 *(The images of area boundary)*  
　　　**┠─HED** *(The images of multi-scale boundary)*  
　　　**┠─Seg** *(The images of semantic segmentation)*  
　　　**┠─result** *(The results of counting)*  
　　**┠─B2**　*(similar as B1)*  
　　**┠─B3**　*(similar as B1)*

**┠─manual\_counting** *(The manual identified results)*

## FT\_Edge: The multi-scale boundary extraction

**┠─FT\_Edge**  
　**┠─dataset**　 *(The data augmentation and the dataset class of pytorch)*  
　**┠─model**　 *(HED model)*  
　**┠─pretrained**　 *(Pretrained model parameters)*  
　**┠─result**　 *(The final trained model)*  
　**┠─Train\_HED\_FT.py**　*(Train the HED model)*  
　**┠─Test\_HED\_FT.py**　 *(Using the HED model to generate boundary of new image)*

## FT\_Segmentation: The semantic segmentation

**┠─FT\_Segmentation**  
　**┠─dataset**　 *(The data augmentation and the dataset class of pytorch)*  
　**┠─model**　 *(HED model)*  
　**┠─network**　 *(Semantic segmentation network)*  
　**┠─pretrained**　 *(Pretrained model)*  
　**┠─result**　 *(The final trained model, including: The trained HED model after integrating with segmentation& The trained semantic segmentation model)*  
　**┠─loss.py**　*(The loss for semantic segmentation)*  
　**┠─Train\_Segmentation\_FT.py**　*(Train the Segmentation together with HED model)*  
　**┠─Test\_Segmentation\_FT.py**　 *(Using the trained model to generate refined semantic segmentation image of new image)*

## FT\_Counting: Count fission tracks

**┠─FT\_Counting**　  
　**┠─Config**　 *(Configuration parameter)*  
　　**┠─A\_config.ini**　 *(Configuration parameter of A)*  
　　**┠─M\_config.ini**　 *(Configuration parameter of M)*

**┠─Segimage\_ALL**  *(Put all Seg images into this folder to calculate Ta)*  
　**┠─Area\_Threshold.py** *(Compute the area threshold)*  
　**┠─Calculation.py**　 *(Called by Identification\_A.py and Identification\_M.py)*  
　**┠─Identification\_A.py** *(Identify and count fission tracks of A)*  
　**┠─Identification\_M.py** *(Identify and count fission tracks of M)*

# Workflow

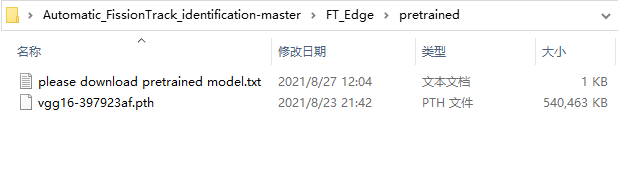
## (1) Prepare the data

1. Download the data: [**https://pan.baidu.com/s/1U-lx0BhDDVZBlAjY-f94gg**](https://pan.baidu.com/s/1U-lx0BhDDVZBlAjY-f94gg)**password：wj9n.**
2. Prepare the training and label image patches and put them into the corresponding folders.
3. Prepare the testing images (images needed to be count) and put them into the corresponding folders.

## (2) Multi-scale boundary extraction

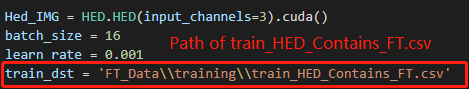
1. **Download the pretrained model.**

(<https://download.pytorch.org/models/vgg16-397923af.pth>)

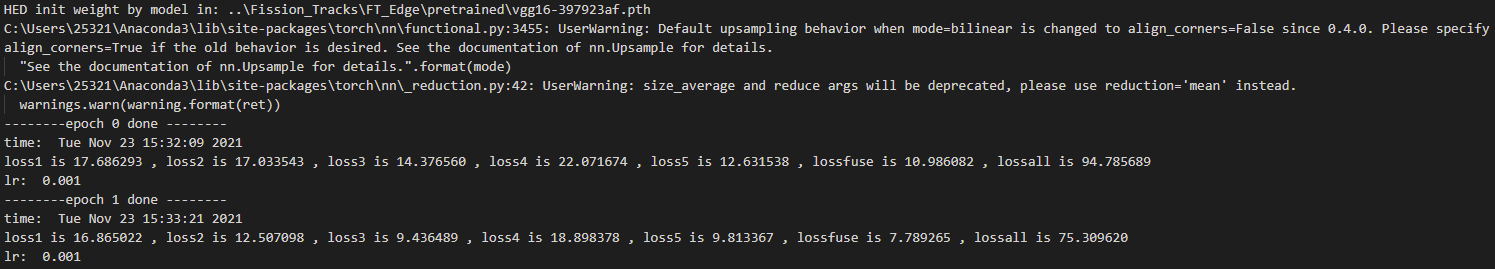


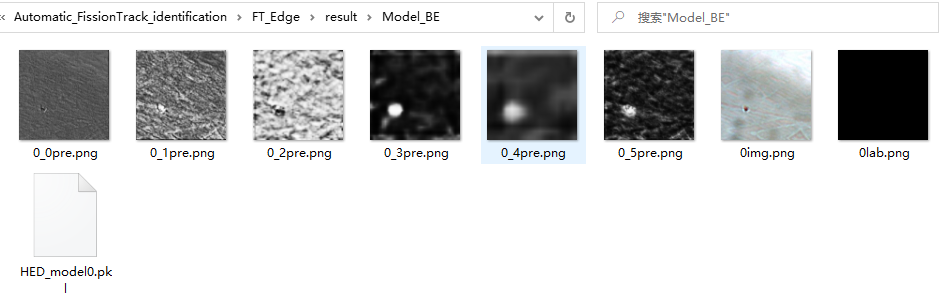
1. **Train HED model by running the ‘Train\_HED(Contains\_FT).py’.**

The training images are from: ‘train\_HED(Contains\_FT).csv’ where images are all contain fission tracks in ‘FT\_Data\\training\\’.



The code will output the training information and save the models as follow:





We also provided a trained model can be used directly at:

https://doi.org/10.5281/zenodo.5783272

1. **Predict multi-scale boundary by running the ‘Test\_HED\_FT.py’.**

The input image filename: ‘FT\_Data\\testing\\XX\\ image\\A(M)\\’



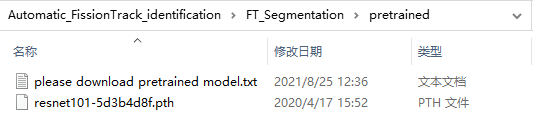
The output image filename: ‘FT\_Data\\testing\\XX\\HED\\ A(M)\\’.



## (3) Semantic segmentation

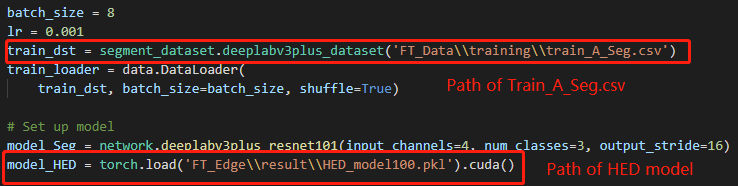
1. **Download the pretrained model.**

(<https://download.pytorch.org/models/resnet101-5d3b4d8f.pth>)



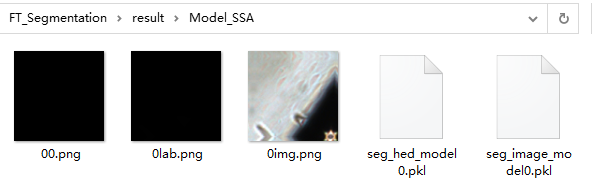
1. **Train semantic segmentation network by running the ‘Train\_Segmentation\_FT.py’.** (Needs an input HED model, so the HED should be trained before).

The filenames of training images are from: ‘train\_M\_Seg.csv’ or ‘train\_A\_Seg.csv’ file in ‘FT\_Data\\training\\’.



The code will output the training information and save the models as follow:





We also provided a trained model can be used directly at:

https://doi.org/10.5281/zenodo.5783272

1. **Predict the refined semantic segmentation image by running the ‘Test\_Segmentation\_FT.py’**

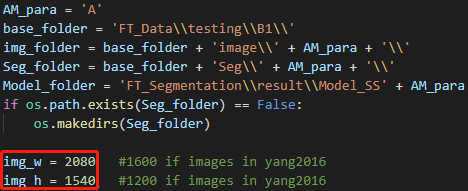
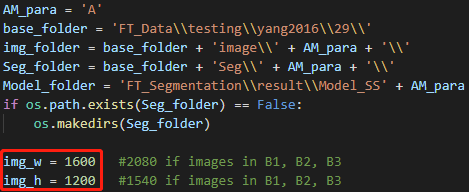
The input image filename is ‘FT\_Data\\testing\\XX\\ image\\A(M)\\’



The output image filename is ‘FT\_Data\\testing\\XX\\Seg\\ A(M)\\’.



Note that the image sizes of B1, B2, B3 and yang2016 are different, so they should be changed.

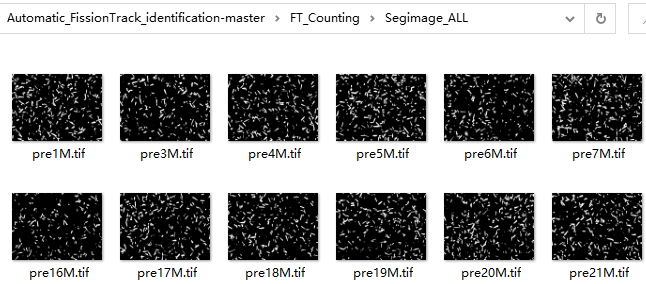
 

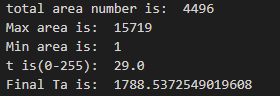
## (4) Counting

1. **Compute the area threshold**

Put the semantic segmentation images of all test images into the folder: ‘FT\_Counting\\Segimage\_ALL\\’

Run ‘Area\_Threshold.py’ to compute the area threshold for M.





1. **Adjust the area threshold in config folder.**

(The other parameters in config files can also be adjusted)

1. **Counting by running the ‘Identification\_A.py’ or ‘Identification\_M.py’.** (Final step)

The result is saved in ‘FT\_Data\\testing\\XX\\result\\A(M)’, with an identification image, a excel table and a shape file.



