http-server配置

1. npm安装http-server

note：

防火墙可能导致npm install失败

可以使用代理

https://stackoverflow.com/questions/28056051/running-npm-behind-a-corporate-firewall-what-do-i-need-to-tell-the-security-tea

```
npm config set proxy http://内网里我的电脑的ip:http代理端口
# 在同一个内网内可以用我的vpn，192.168.162.5:10809
```

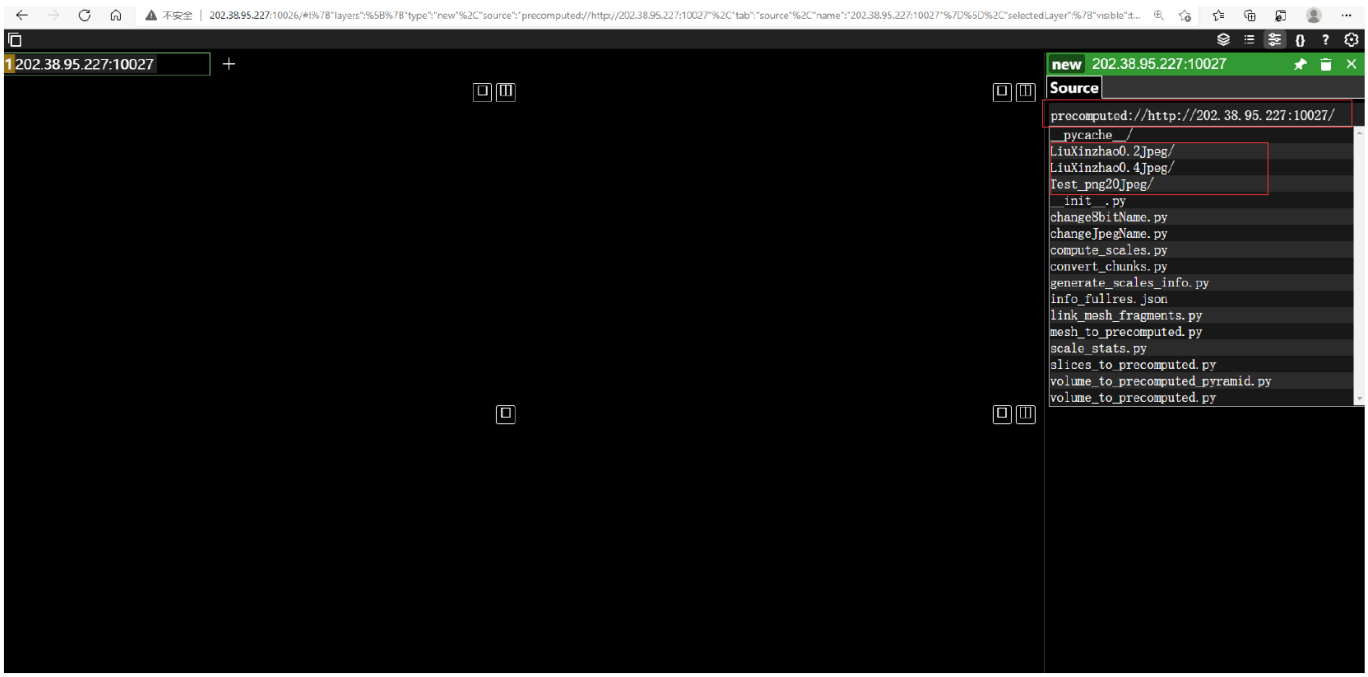
2. 在数据所在目录启动http-server

```
http-server --cors 127.0.0.1  -p your_port
# --cors 127.0.0.1解决跨域问题 port需要用外网开放端口
```

此时如果你的目录下有neuroglancer可以读取的数据，就可以在周洋同学部署的neuroglancer

http://202.38.95.227:10026/ 访问你的数据，source的位置为info文件所在的位置

image放在一个annotation layer（左上角的黄色标签），类型为img，再新建一个annotation layer放segmentation，类型为seg，如果有mesh，双击EM中的一个segment会直接在右边显示mesh，skelenton同理。



可以参考周洋同学的neuroglancer脑数据可视化说明的副本.pdf

准备neuroglancer可以读取的数据

有问题可以参考neuroglancer文档对precomputed格式的描述

https://github.com/google/neuroglancer/tree/master/src/neuroglancer/datasource/precomputed

1. 将EM image和segmentation转化为neuroglancer格式（precomputed）

在upload_IMAGE.py和upload_SEGMENTATION.py中把info的分辨率改为你的数据的voxel分辨率，以及变量im改为numpy array（shape：[x，y，z]）的你的image&seg。

```
pip3 install cloud-volume
python upload_IMAGE.py precomputed://file://your-image-dir
python upload_SEGMENTATION.py precomputed://file://your-segmentation-dir
```

2. 用igneous工具生成mesh（或skelenton）

https://github.com/seung-lab/igneous

```
pip3 install igneous-pipeline
# 将generate_mesh.sh中第一行的路径改为mesh的路径
bash generate_mesh.sh
```

到这里应该就可以启动http-server，打开neuroglancer进行可视化了！有问题问我。可能出现的问题很多，比如浏览器设置导致访问不了http-server（一般换一个浏览器可以解决）。

generate_mesh.sh:

```
segdir="your-segmentation-dir"
# 需要把之前没执行的queue dir都删掉 rm -rf *_queue
# mip是指显示的时候加载的分辨率，会存好几个分辨率，在放缩时加载不同的分辨率（更快），生成mesh也更快。
igneous image downsample --mip 0 --num-mips 4 --queue seg_queue precomputed://file://$segdir
igneous execute -x seg_queue/
# downsample之后是zip格式，arg输入不了None，非常让人无语的bug。解压就好了。
gzip -d $segdir/*/*.gz
# 在mip level 2生成mesh，原分辨率非常非常慢
igneous mesh forge --dir mesh --mip 2 --queue meshforge_queue precomputed://file://$segdir
igneous execute -x meshforge_queue
```

```
igneous mesh merge --dir mesh --magnitude 2 --queue meshmerge_queue precomputed://file://$segdir
igneous execute -x meshmerge_queue
gzip -d $segdir/*/*.gz
```

generate_mesh&skelenton.sh：

```
# mip是指显示的时候加载的分辨率，会存好几个分辨率，在放缩时加载不同的分辨率（更快），生成mesh也更快。
igneous downsample --mip 0 --num-mips 4 --queue seg_queue precomputed://file://your-segmentation-dir
igneous execute -x seg_queue/
# downsample之后是zip格式，暂时没有解决，解压就好了
gzip -d your-segmentation-dir/*/*
# 在mip level 2生成mesh，原分辨率非常非常慢
igneous mesh forge --dir mesh --mip 2 --queue meshforge_queue precomputed://file://your-segmentation-dir
igneous execute -x meshforge_queue
igneous mesh merge --dir mesh --magnitude 2 --queue meshmerge_queue precomputed://file://your-segmentation-dir
igneous execute -x meshmerge_queue
# 此处和AxonEM evaluation的skeletonization参数一致
igneous skeleton forge --mip 2 --scale 4 --const 500 --dust-threshold 100 --queue skelforge_queue precomputed://file://your-segmentation-dir
igneous execute -x skelforge_queue
igneous skeleton merge --queue skelmerge_queue precomputed://file://your-segmentation-dir
igneous execute -x skelmerge_queue
gzip -d your-segmentation-dir/*/*
```

还需要在skeletons的info中指定segment info的位置，add `"segment_properties": '..'`

upload_IMAGE.py

Python L1 (default) ···

```
import argparse
import numpy as np
from cloudvolume import CloudVolume
import tifffile as tf
from joblib import Parallel


def create_image_layer(destination, shape):
```

```
        info = CloudVolume.create_new_info(
                num_channels    = 1,
                layer_type      = 'image',
                data_type       = 'uint8',
                encoding        = 'raw',
                # 修改resolution
                resolution      = [8,8,30],
                voxel_offset    = [0, 0, 0],
                chunk_size      = [64, 64, 16],
                volume_size     = shape
        )

        vol = CloudVolume(destination, compress=False, info=info,
parallel=True)
        print(vol.info)
        vol.commit_info()
        return vol

# python upload.py **source** **destination**
# python upload.py ./test/ precomputed://file://test_output
def main():
        parser = argparse.ArgumentParser('Convert a folder of tif files to
neuroglancer format')
        parser.add_argument('destination', help='Destination path for
precomputed files, pre-pended with precomputed://file://, e.g.
precomputed://file://**path** will write the files to ./**path**')
        args = parser.parse_args()

        raw_path = '/data12T/janechen/mitoEM/human_raw.tiff'
        im = tf.imread(raw_path)
        # 改为numpy array (shape: [x, y, z]) 的你的image
        im = im.transpose((2, 1, 0))
        vol = create_image_layer(args.destination, im.shape)

        vol[:,:,:] = im

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

upload_SEGMENTATION.py

```
import argparse
import numpy as np
from cloudvolume import CloudVolume
```

```
import tifffile as tf
import h5py
from joblib import Parallel


def create_image_layer(destination, shape):
    info = CloudVolume.create_new_info(
        num_channels    = 1,
        layer_type      = 'segmentation',
        data_type       = 'uint32',
        encoding        = 'raw',
        # 修改resolution
        resolution      = [8,8,30],
        voxel_offset    = [0, 0, 0],
        mesh  = 'mesh',
        chunk_size      = [64, 64, 16],
        volume_size     = shape
    )

    vol = CloudVolume(destination, compress=False, info=info, parallel=True)
    print(vol.info)
    vol.commit_info()
    return vol

# python upload.py **source** **destination**
# python upload.py ./test/ precomputed://file://test_output
def main():
    parser = argparse.ArgumentParser('Convert a folder of tif files to
neuroglancer format')
    parser.add_argument('destination', help='Destination path for precomputed
files, pre-pended with precomputed://file://, e.g.
precomputed://file://**path** will write the files to ./**path**')
    args = parser.parse_args()

    raw_path = '/data12T/janechen/mitoEM/result_0-100-0-4096-0-
4096_xy_bc_watershed_instance.h5'
    h5files = h5py.File(raw_path, 'r')
    instances = h5files['vol0']
    im = np.asarray(list(instances.astype('uint32')))
    print(im.shape)
    # 改为numpy array (shape: [x, y, z]) 的你的seg
    im = im.transpose((2, 1, 0))
    vol = create_image_layer(args.destination, im.shape)

    vol[:,:,:] = im

```

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