搭建开发环境的步骤：

1. 安装python开发环境（Pytorch imageio imageio_ffmpeg ConfigArgParse）
2. 安装ImageMagicK命令行
3. 拍摄不同角度的多视角图像并配置目录
4. 安装ColMap，并将多视角图像计算得到相机位姿
5. 保存ColMap生成的模型，并修改配置目录
6. 将ColMap转换为LLFF格式
7. 制作NeRF的启动配置文件
8. 开始训练：

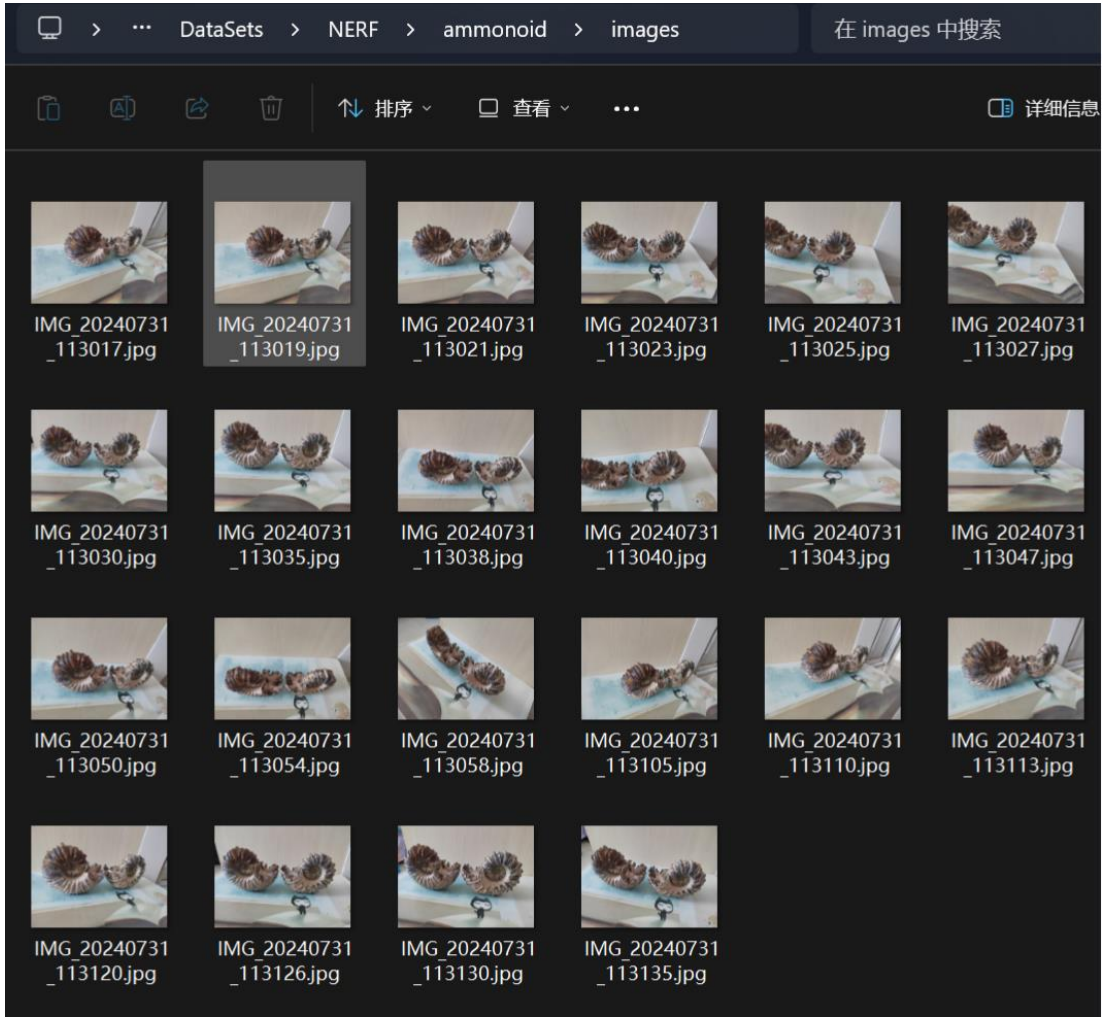
D:/Software/Python38/python.exe run_nerf.py --config configs/mydata.txt

1. 安装python开发环境

安装Python依赖库：
Pytorch imageio imageio_ffmpeg ConfigArgParse

2. 拍摄不同角度的多视角图像

拍摄不同角度的多张图片：



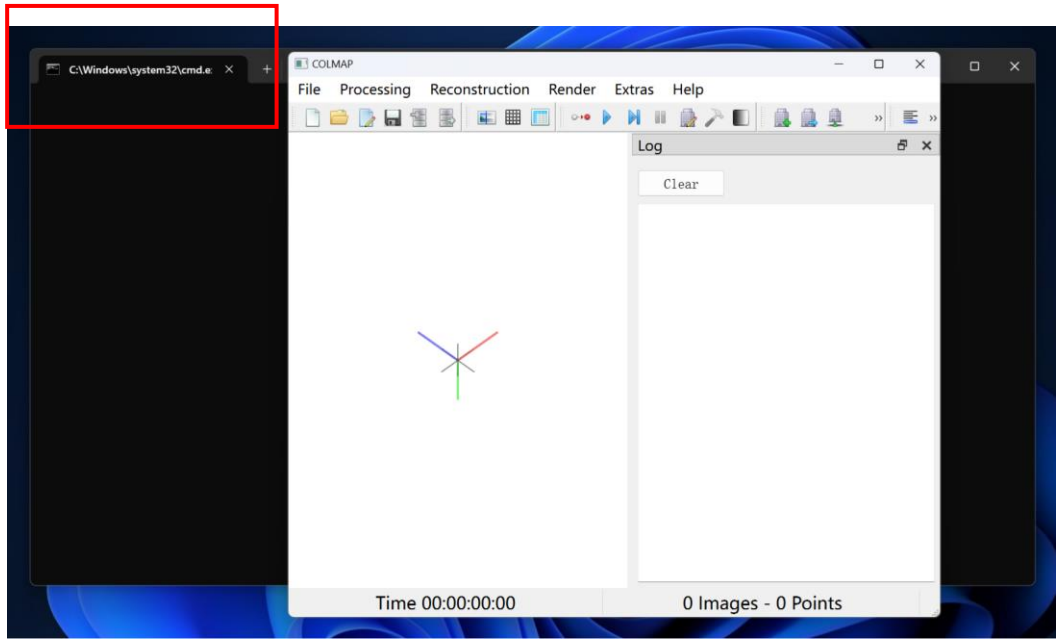
3. 安装ColMap，并将多视角图像计算得到相机位姿

在下面的地址下载ColMap

https://github.com/colmap/colmap/releases

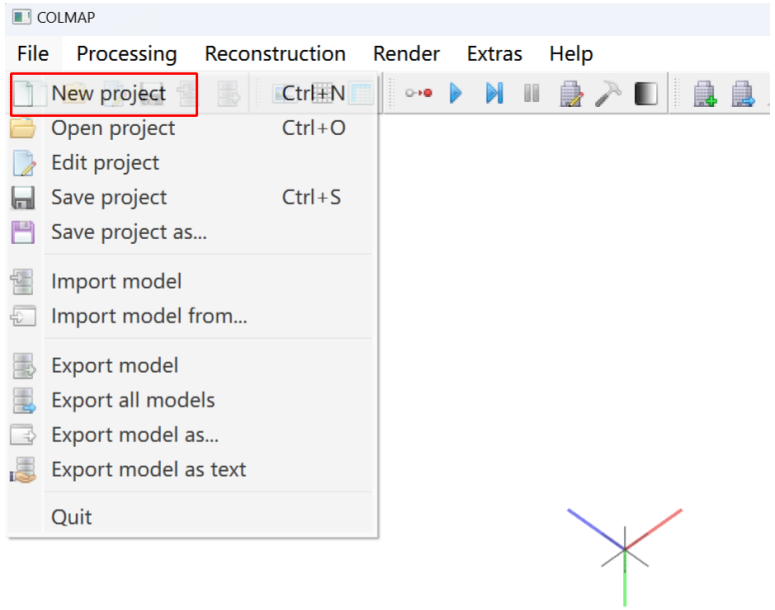
下载好以后解压，点击.bat文件即可运行：

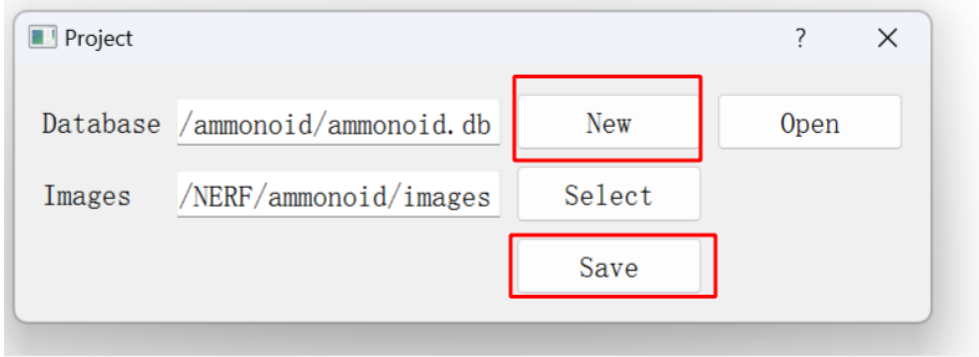
运行后，终端不要关闭：



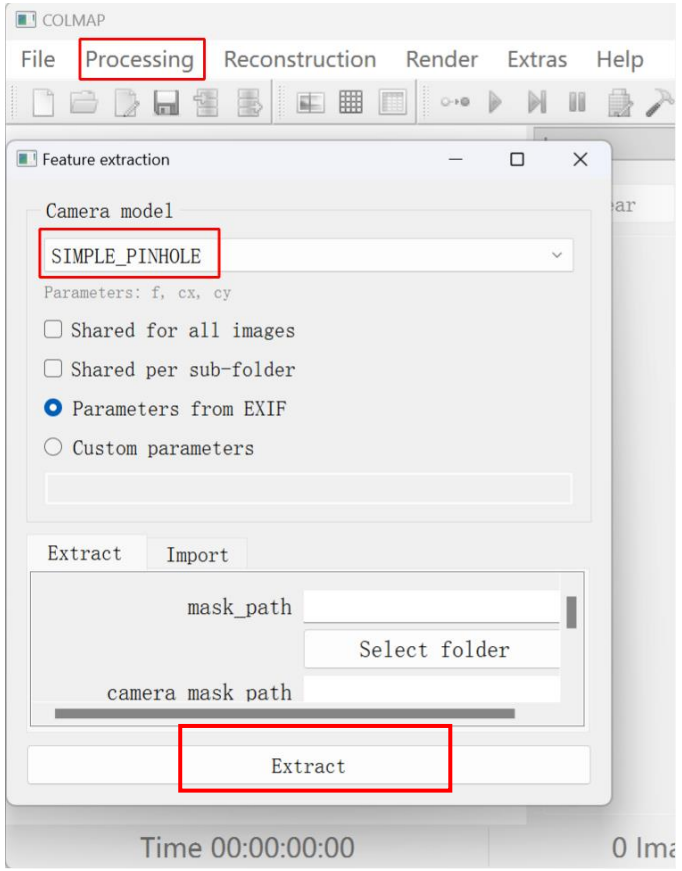
3. 安装ColMap，并将多视角图像计算得到相机位姿

点击菜单栏File，新建工程。
然后在工程中新建DataBase。
然后选择图像文件所在目录。之后点击Save。



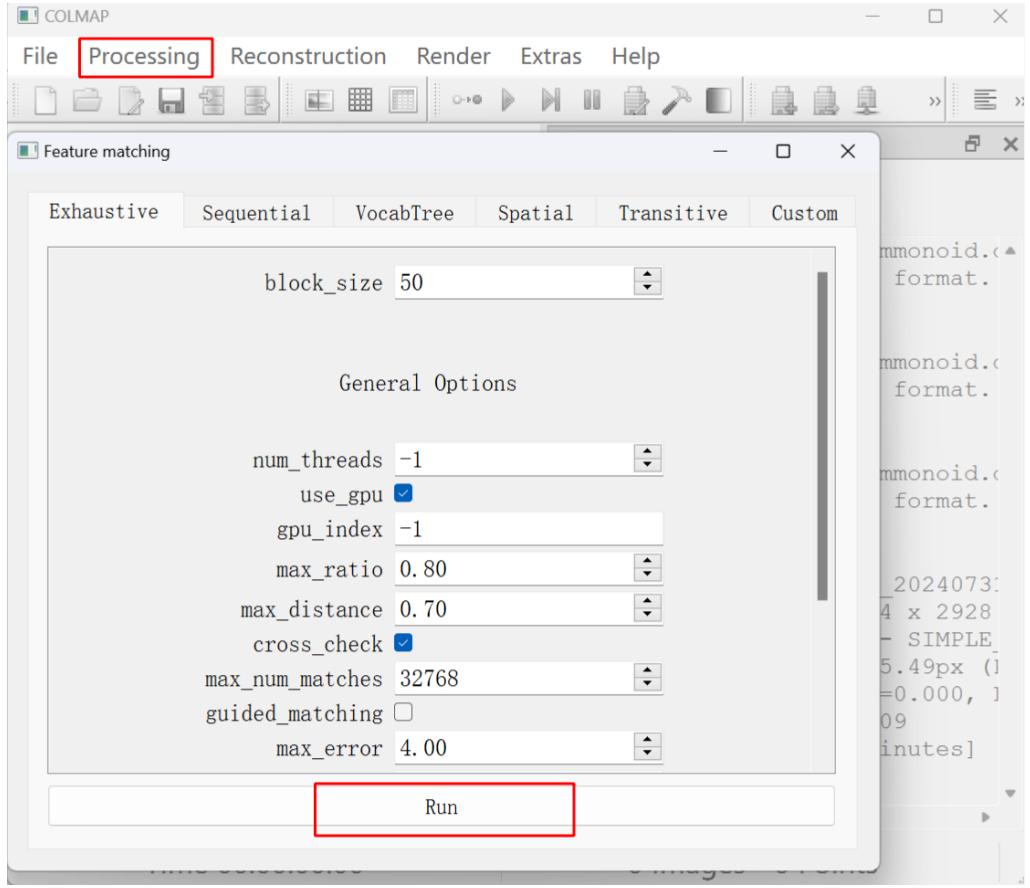


点击菜单栏Processing，选择特征抽取。
然后选择针孔相机来抽取特征即可。

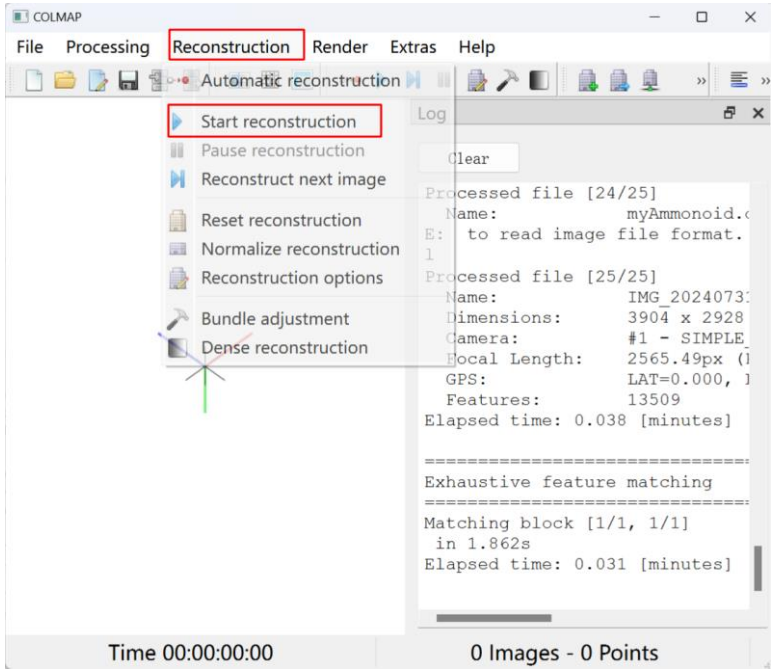


3. 安装ColMap，并将多视角图像计算得到相机位姿

再次点击菜单栏Processing，选择特征匹配。
用默认的参数即可，然后进行匹配。

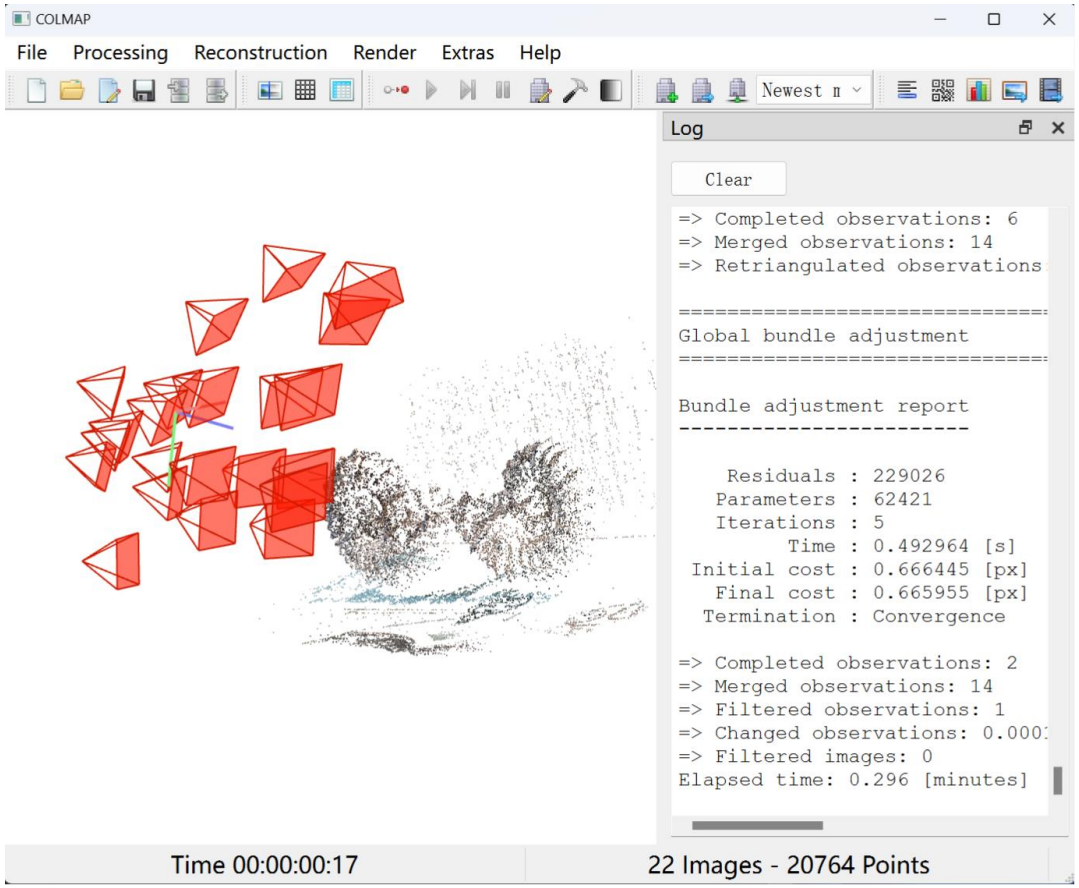


点击重建，开始重建：

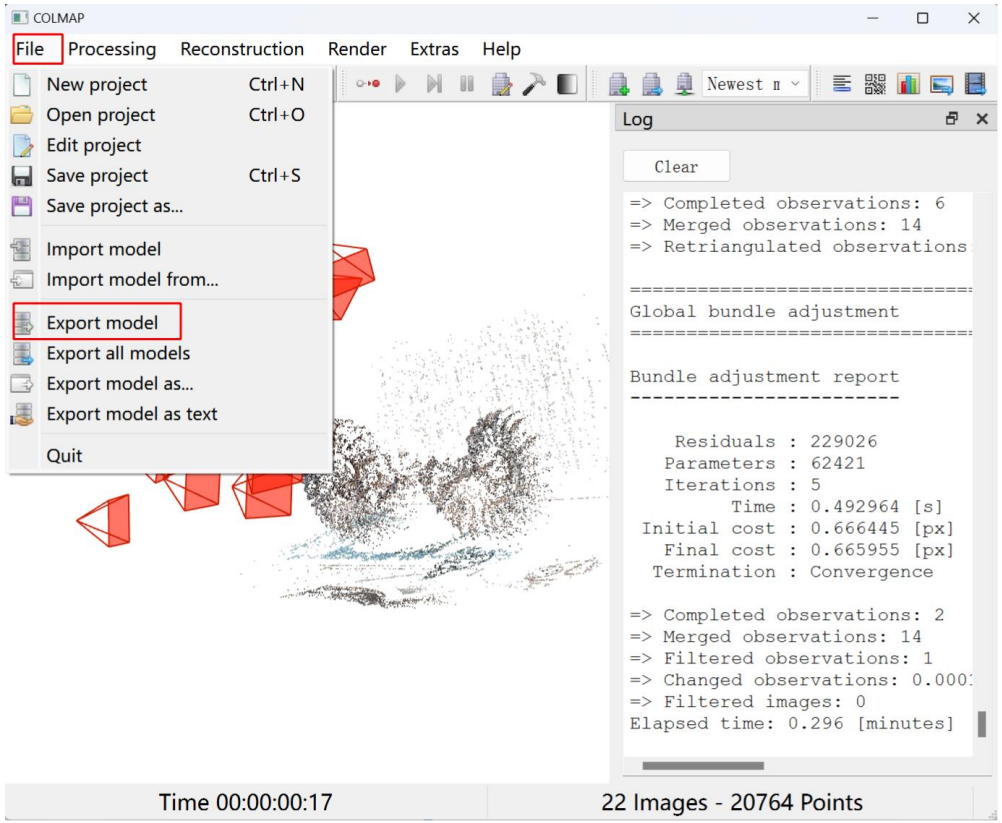


3. 安装ColMap，并将多视角图像计算得到相机位姿

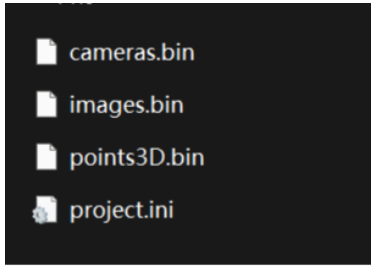
重建得到稀疏点云和相机位置：



点击菜单栏File， 导出模型。位置是在images所在目录下新建sparse/0/

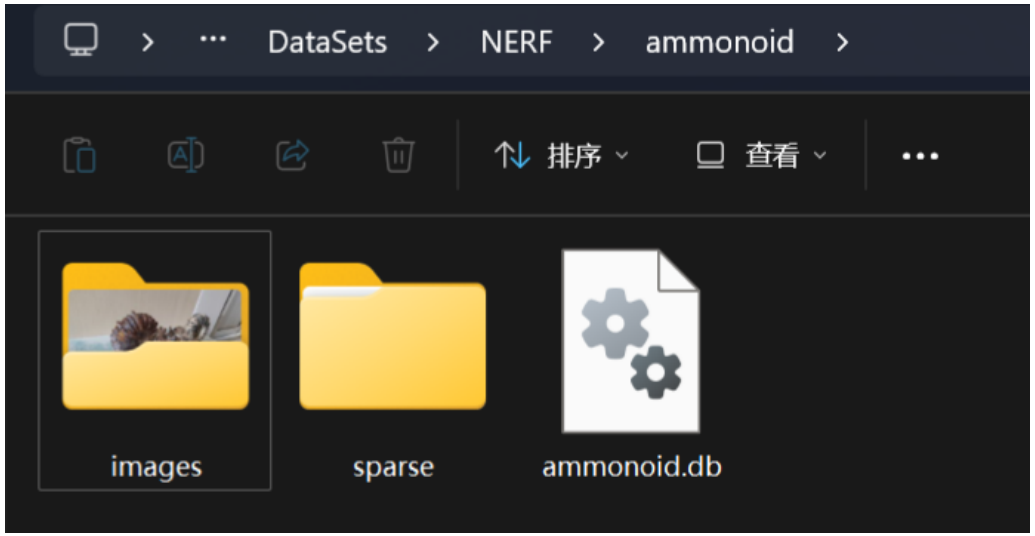


会导出这几个文件：



3. 安装ColMap，并将多视角图像计算得到相机位姿

当前工程结构:



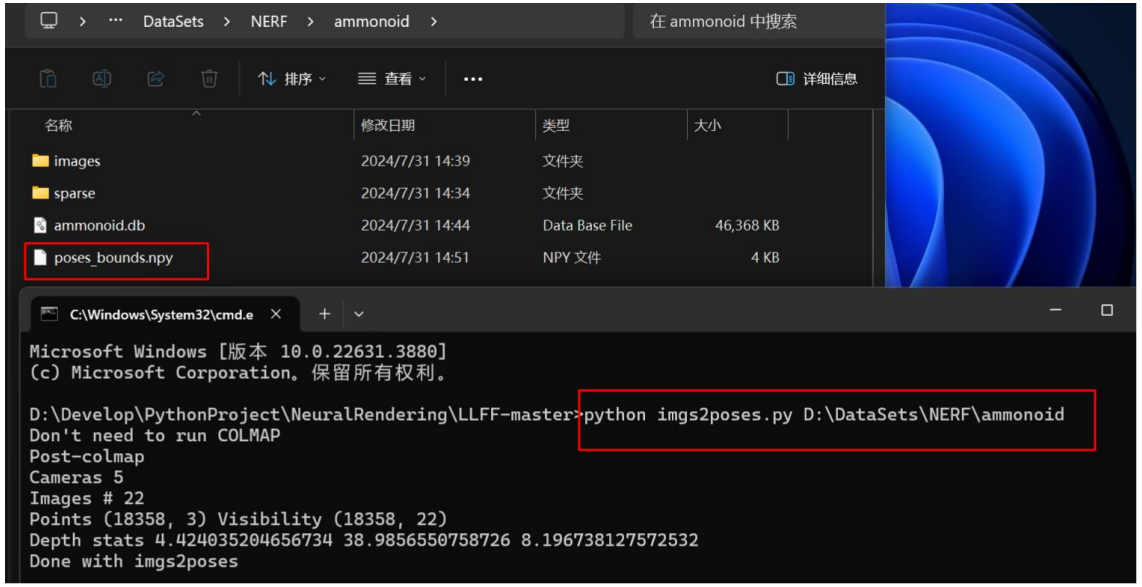
存放图像文件的目录一定要叫images，否则后面转LLFF格式时会报错。

4. 将ColMap转换为LLFF格式

从下面的网址下载LLFF工程并解压:

https://github.com/Fyusion/LLFF

然后使用Python在该工程目录下运行
python imgsposes.py 你的sparse文件夹所在目录



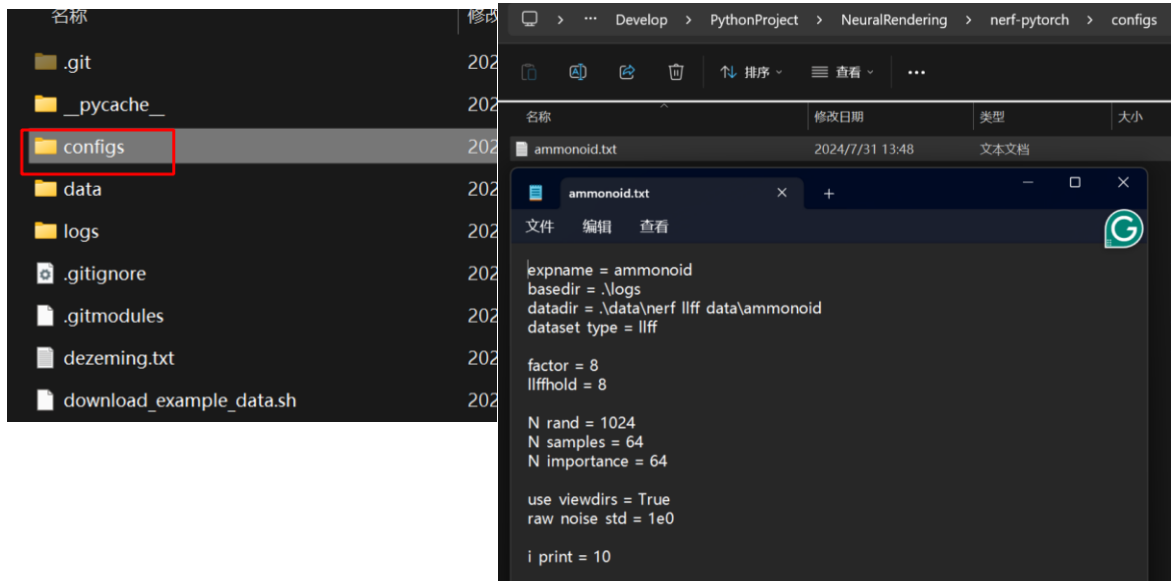
得到poses_bounds.npy文件

开始训练的步骤:

1. 制作启动配置文件
2. 开始训练

1. 制作启动文件

把之前的图像拷贝到data目录下的nerf_llff_data目录下。
在源码configs下可以找到配置文件。



修改配置文件的相关信息，然后调用：
python run_nerf.py --config configs/ammonoid.txt

也可以直接将这些配置信息修改到源码中parse参数的dafault参数值上，然后直接启动。

```
1 usage  SirinL *
def config_parser():

    import configargparse
    parser = configargparse.ArgumentParser()
    parser.add_argument('--config', is_config_file=True,
                        help='config file path')
    parser.add_argument("--expname", type=str, default='ammonoid',
                        help='experiment name')
    parser.add_argument("--basedir", type=str, default='.\logs\\',
                        help='where to store ckpts and logs')
    parser.add_argument("--datadir", type=str, default='.\data\\nerf_llff_data\\ammonoid',
                        help='input data directory')
```

1. 制作启动文件

用原尺寸图像训练可能太大，因此这里使用下采样。为参数：

```
## llff flags
parser.add_argument("--factor", type=int, default=16,
                    help='downsample factor for LLFF images')
```

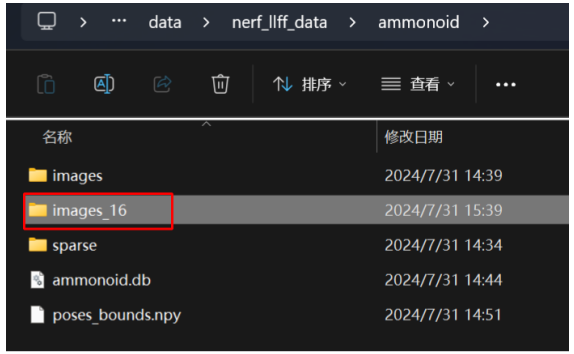
但是要提前安装好ImageMagicK这个工具，并且设置其环境变量。
https://imagemagick.org/script/download.php

ImageMagicK会自带命令行工具，如果没有独立的命令行工具，意味着封装到了magick.exe。此时需要把load_llff.py中的下面的代码进行修改：

load_llff.py × run_nerf.py

```
54            continue
55
56        print('Minifying', r, basedir)
57
58        os.makedirs(imgdir)
59        print('----------------------------------------------------')
60        print(imgdir)
61        print('----------------------------------------------------\n')
62        # 复制图片文件到新的目录
63        # check_output('cp {}/* {}/'.format(imgdir_orig, imgdir), shell=True)
64        check_output('copy {}\* {}'.format(imgdir_orig.replace('/','\\'), imgdir.replace('/','\\')), shell=True)
65
66        ext = imgs[0].split('.')[-1]
67        args = ' '.join(['magick mogrify', '-resize', resizearg, '-format', 'png', '*.{}'.format(ext)])
68        print(args)
69        os.chdir(imgdir)
70        check_output(args, shell=True)
71        os.chdir(wd)
72
```

这样运行时会创建缩小的图像：



2. 开始训练

```
Removed duplicates
Done
Loaded image data (183, 244, 3, 22) [183.         244.         169.05274159]
Loaded .\data\nerf_llff_data\ammonoid 4.475204041410084 35.75301955448777
recentered (3, 5)
[[ 1.0000000e+00  1.5623099e-10 -4.2785651e-09  2.9802322e-08]
 [-1.5623099e-10  1.0000000e+00  3.5654710e-10  5.4186042e-09]
 [ 4.2785651e-09 -3.5654710e-10  1.0000000e+00 -9.4825570e-09]]
Data:
(22, 3, 5) (22, 183, 244, 3) (22, 2)
HOLDOUT view is 3
Loaded llff (22, 183, 244, 3) (120, 3, 5) [183.      244.      169.05273] .\data\nerf_llff_data\ammonoid
Auto LLFF holdout, 8
DEFINING BOUNDS
NEAR FAR 0.0 1.0
Found ckpts []
get rays
done, concats
shuffle rays
done
Begin
TRAIN views are [ 1  2  3  4  5  6  7  9 10 11 12 13 14 15 17 18 19 20 21]
TEST views are [ 0  8 16]
VAL views are [ 0  8 16]
  0%|          | 10/200000 [00:03<16:03:39,  3.46it/s][TRAIN] Iter: 10 Loss: 0.062467314302921295  PSNR: 15.049487113952637
  0%|          | 20/200000 [00:06<15:47:52,  3.52it/s][TRAIN] Iter: 20 Loss: 0.05400983244180679  PSNR: 15.68204116821289
  0%|          | 30/200000 [00:09<15:50:38,  3.51it/s][TRAIN] Iter: 30 Loss: 0.053749438375234604  PSNR: 15.707589149475098
  0%|          | 40/200000 [00:11<15:49:04,  3.51it/s][TRAIN] Iter: 40 Loss: 0.05311670899391174  PSNR: 15.757442474365234
```