### 本项目目的

学习Nerf流程

实现Nerf基于奥比中光RGBD模组

**改为全程使用OpenCV坐标系，不再使用其他坐标系。一方面验证自己的Nerf理解，一方面和SLAM程序对齐。**

A diagram of a projector

AI-generated content may be incorrect.

### 背景

Novel view synthesis指的是对一个场景给定一些input images时，如何获取其他view的视图

有两种：一个是显式三维重建，然后就自然获取了其他view，还一个是不再显式三维重建，直接利用volume rendering来获取新view的每个像素的RGB值，volume render就是模拟光线穿过体数据的采样和累积过程。

A black background with white text

AI-generated content may be incorrect.

T(t)是累积透射率，是该点的volume density, c是color，r(t)=o+td，o是光线原点，d是光线方向

具体以上公式怎么来的，参考下面volume render的链接，非常庞大的章节

**Volume Render**: <https://www.scratchapixel.com/lessons/3d-basic-rendering/volume-rendering-for-developers/intro-volume-rendering.html>

<https://arxiv.org/pdf/2209.02417>

<https://zhuanlan.zhihu.com/p/595117334>

A group of mathematical equations

AI-generated content may be incorrect.

积分肯定是无法直接使用的，必须离散化。把整条光路[0, s]（也就是t\_near, t\_far），划分为N个等距的区间，区间内的

其中，这里是0到t的积分，为了去掉tn前的积分，将T(t)写为

也就得到

假设

代入回总的公式

对T(0->tn)也离散化，得到

最终

对比Nerf论文的公式

A black text on a white background

AI-generated content may be incorrect.

### 训练流程

通俗讲，就是训练一个模型，它的输入是一个射线（direction方向上的采样点xyz+direction方向），输出该采样点的rgb和volume density，一旦有了这条射线上的所有采样点的RGB和volume density，我们就可以根据上面提到volume render方程来计算在图像上，该点的RGB值。

### 数据准备

Colmap：feature extraction，feature matching，start reconstruction

使用统一相机内参，FULL OPENCV，禁止相机内参参与BA优化

A screenshot of a computer

AI-generated content may be incorrect.

参考nerf\_prepare\_dataset.py

### load\_llff\_data

rescale bds

归一化poses，把所有poses的均值放在世界原点，包括translate和rotation

计算render poses，和训练无关，只是为了test的时候出mp4看结果

get\_embedder

为什么需要positional encoding。原论文论述了这一点，deep network通常会biased学习到low frequency的部分，但实际的nerf新视角重建需要network学习到高频部分，一个有效的做法是利用high frequency functions把inputs映射到高维空间去，让network学习高频的变化。

### Nerf

A diagram of a number

AI-generated content may be incorrect.

模型输入为x: 3D点的高维表示，d: 视角的高维表示

模型输出为体素的密度volum density, RGB预测的颜色

### get\_rays\_np

原始实现只有简单的raytrace过程，这里我实现了带畸变的raytrace过程，因为大多数实际相机都有畸变，这里不采用先对图像进行去畸变的操作，也就是保持RGB原图

A screen shot of a computer program

AI-generated content may be incorrect.

### ndc\_rays

<https://yconquesty.github.io/blog/ml/nerf/nerf_ndc.html>

将给定的rays\_o和rays\_d转换到归一化的NDC空间中

首先跟着推文，按原始的OPENGL坐标系推导，随后换到opencv坐标系下

A black background with white text

AI-generated content may be incorrect.

上面透视变换矩阵，把相机坐标系下的点xyz（原本是透视变换）映射到了一个正交体

上面透视变换矩阵怎么来的？首先第四行是为了要求后续做齐次除法时，分母时原来的z

点落在z=-near平面上，做完齐次除法后的xy要与之前xy相同

假设矩阵第三行为，则有

也就是，为了把透视视锥变成平行六面体，代入z=-near/far

A screenshot of a math problem

AI-generated content may be incorrect.

最右边一项：corner to origin, 平移，使得l，b，n对齐原点

中间一项：缩放到2x2x2的cube

最左边一项：平移成[-1,1]的cube上

A screenshot of a math equation

AI-generated content may be incorrect.

注意P是Homogeneous的，乘以-1不影响P的效果，因为Nerf的camera看向z的负向

由于camera frustum都是对称的，则有l=-r, b=-t

A black background with white text

AI-generated content may be incorrect.

于是，当我们有一个点xyz1，和对应的ray: r=o+td在world space下（世界坐标系），我们的NDC投影可以得到，注意这里的t是top，不是ray的t

A black background with white text

AI-generated content may be incorrect.

如何在NDC下表示ray？

A black background with white text

AI-generated content may be incorrect.A black background with white text

AI-generated content may be incorrect.

为什么需要NDC？因为图像的像素中无限depth完全有可能出现，映射到[0,1]NDC空间变成了有界cube

对于实际的camera而言，near就是focal length，far就是无穷大，r和t就是W/2，H/2

A screenshot of a math equation

AI-generated content may be incorrect.

额外的细节，Nerf把ray的起始o点直接放到了near平面上，因为实际采样点只会出现在near平面之后，一旦把world space下的ray转换到NDC下，就可以对t`从0到1进行采样方便。注意这里压缩到NDC空间的是[near, far]这个视锥体，对应的就是[focal, inf]这个区间。

NDC是OPENGL坐标系，也即y向上，x向右，z沿着camera向外，为了转换到opencv坐标系，设

简单验证下，当z=n时，z\_ndc=-1, 当z=f时，z\_ndc=1

当然n=f\_cam, f=inf, r=W/2, t=H/2时

最终opencv坐标系的ndc ray实现参考实际代码

A computer screen shot of a program

AI-generated content may be incorrect.

### 实际渲染render\_rays

<https://yconquesty.github.io/blog/ml/nerf/nerf_rendering.html>

A computer screen with white text

AI-generated content may be incorrect.

前者在depth上均匀采样，后者在d=1/z（disparity）上均匀采样，这样近处会采样更密集，远处采样会更稀疏。

A computer screen with white and blue text

AI-generated content may be incorrect.

沿着ray计算采样点（NDC空间下），在每个区间内随机取点

### raw2outputs

通过MLP输出的采样点的rgb和alpha，重新渲染对应图像

A computer code on a black background

AI-generated content may be incorrect.

求出相邻点的深度差（不是沿着ray，仅z轴方向），同时最后一列增加1e10来表示正无穷



取出MLP输出的RGB值，sigmoid压制到[0,1]，因为网络输出范围负无穷到正无穷

A computer screen shot of a program

AI-generated content may be incorrect.

这段代码在计算论文以及之前都有提到的C(r)的离散计算公式

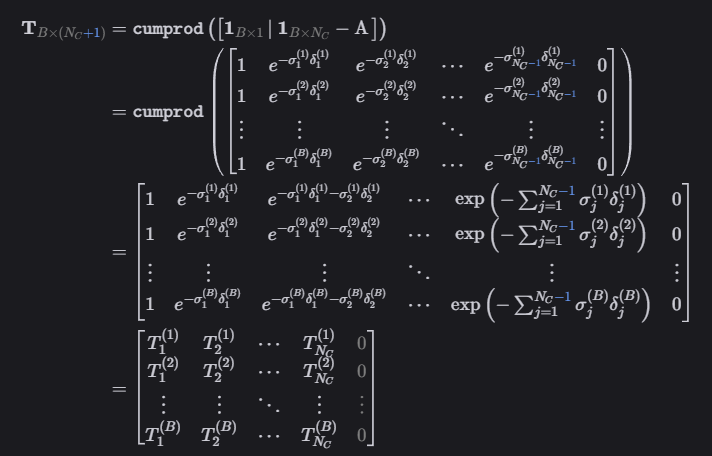
A black text on a white background

AI-generated content may be incorrect.

首先，给MLP输出的volume density加上noise，

计算代码中的alpha

计算weights（就是Ti），这里用了cumprod函数，用1-A获得正的exp，然后横向加到一起



A black background with white text

AI-generated content may be incorrect.A black background with white text

AI-generated content may be incorrect.

直接hadamard product就计算好了weights

最后计算深度图（以weights加权），disp=1/depth，acc map（累积的不透明度）