3D无限场景的生成

用github来记录了整个项目的进度

https://github.com/Yezili486/Infinite_3Dscene

第一周

图形学最基础知识，包括相机标定，colmap，sfm等方法

三维重建基础

LucidDreamer

论文解读

In this work, we propose a pipeline called LucidDreamer that utilizes Stable Diffusion and 3D Gaussian splatting to create diverse high-quality 3D scenes from various types of inputs such as text, RGB, and RGBD.

1. **局限**：dataset far form real world（特定领域）
    a. 利用stable diffusion生成3D模型，难以保证multi-view consistency
    b. diffusion模型学习：

    diffusion模型基础
2. **Step: dream and alignment**
    a. 投影点云，利用生成模型。根据深度提升三维空间
    b. Alignment algorithm：integrate
    c. 作为3dgs的inital points

简单来说，初始图像投影之后用diffusion模型生成，重新升维到3维空间后，用alignment算法和原先的点云群融合

3. **输入**：初始图像和深度图
4. **输出**：点云
5. **优点**：多场景，配合文字输入，生成更多视角
6. **具体过程**：

a. 初始化点云，利用stable diffusion得到imag和深度图（深度图的生成根据zero-depth the monocular depth estimation model），把二维图像升维到三维

   i. $P_0 = \phi_{2\to3}([I0, D0], K, P0)$

b. 聚合点云：把生成的点云和原来的点云聚合成更大的点云，需要满足一致性

c. dream过程：投影的image会有没有得到的点，因此利用mask来表示填充和没有填充的点，在没有填充的点继续进行stable diffusion和monocular depth的过程

   i. 实际过程中，深度图可能会不一致（因为模型缺陷），所以需要optimal depth

   $I_i = \mathcal{S}(\hat{I}, M_i), \hat{D}_i = \mathcal{D}(I_i), D_i = d_i\hat{D}_i$ 新的图像I，是从旧图像和mask利用S来生成的，depth图也可以用来生成，di是系数来控制

   $d_i = \arg\min_d \left(\sum_{M_i=1} \left\| \phi_{2\to3}([I_i, d\hat{D}_i], K, P_i) - P_{i-1} \right\|_1 \right)$ 这个是用来逼近di

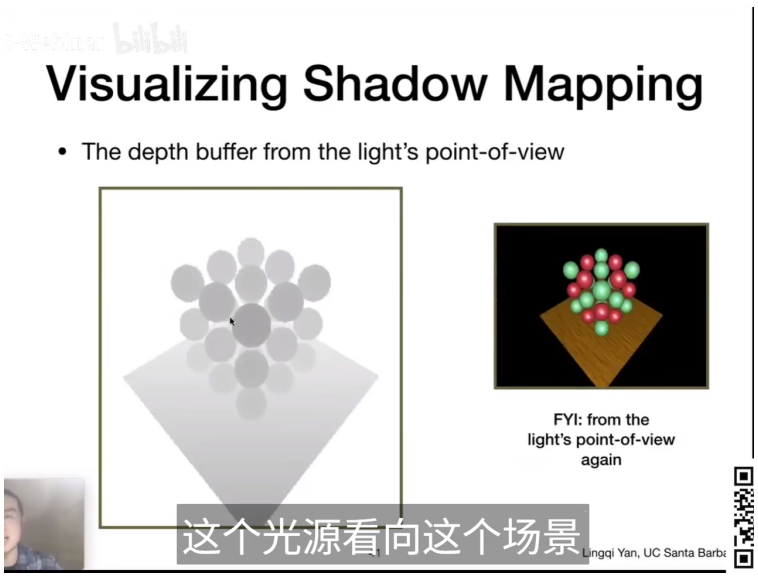
d. alignment：dream的过程是同时生成深度图和新图像，用off the shelf 的深度图生成方法会更好（适配更多的场景，更加的精确）

   i. 需要解决consistency的问题，引入alignment算法。因为没有考虑多个depth之间的关系，直接移动点，让之间smooth。（需要计算vector），naive的移动会导致distort，需要利用差值算法解决

   图形学基础（有差值算法）

   i. 具体的过程：将图片沿着射线移动（因为深度是这个方向），找到最接近Pi-1位置的点，并计算深度改变的幅度（因为移动）

   ii. 对于没有真实图像的地方，用线性差值的方法解决（深度吐的图片）



   iii. 重复整个过程，就可以完成整个深度图的构造（其实整个alignment，就是在找更精确的di）

Algorithm 1: Constructing point cloud

Input: A single RGBD image $[\mathbf{I}_0, \mathbf{D}_0]$
Input: Camera intrinsic $\mathbf{K}$, extrinsics $\{\mathbf{P}_i\}_{i=0}^{N}$
Output: Complete point cloud $\mathcal{P}_N$

$\mathcal{P}_0 \leftarrow \phi_{2\to3}\left([\mathbf{I}_0, \mathbf{D}_0], \mathbf{K}, \mathbf{P}_0\right)$
for $i \leftarrow 1$ **to** N **do**
  $\hat{\mathbf{I}}_i, \mathbf{M}_i \leftarrow \phi_{3\to2}\left(\mathcal{P}_{i-1}, \mathbf{K}, \mathbf{P}_i\right)$
  $\mathbf{I}_i \leftarrow \mathcal{S}\left(\hat{\mathbf{I}}_i, \mathbf{M}_i\right), \hat{\mathbf{D}}_i \leftarrow \mathcal{D}\left(\mathbf{I}_i\right)$
  $d_i \leftarrow 1$
  while *not converged* **do**
    $\tilde{\mathcal{P}}_i \leftarrow \phi_{2\to3}\left(\left[\mathbf{I}_i, d_i\hat{\mathbf{D}}_i\right], \mathbf{K}, \mathbf{P}_i\right)$
    $\mathcal{L}_d \leftarrow \frac{1}{\|\mathbf{M}_i=1\|}\sum_{\mathbf{M}_i=1}\left\|\tilde{\mathcal{P}}_i - \mathcal{P}_{i-1}\right\|_1$
    Calculate $\nabla_d \mathcal{L}_d$
    $d_i \leftarrow d_i - \alpha\nabla_d\mathcal{L}_d$
  end
  $\mathbf{D}_i \leftarrow d_i\hat{\mathbf{D}}_i$
  $\hat{\mathcal{P}}_i \leftarrow \phi_{2\to3}\left([\mathbf{I}_i, \mathbf{D}_i|\mathbf{M}_i = 0], \mathbf{K}, \mathbf{P}_i\right)$
  $\mathcal{P}_i \leftarrow \mathcal{P}_{i-1} \cup \mathcal{W}\left(\hat{\mathcal{P}}_i\right)$
end

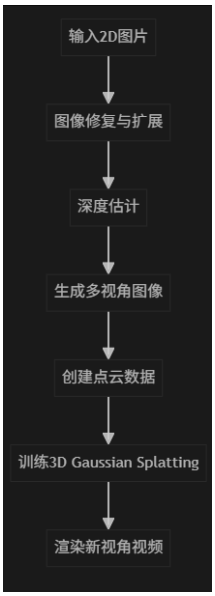
e. 最后就是利用3dgs高斯来完成训练和渲染（整个luciddreammer其实关注的就是初始化点云）
   i. 值得注意的是：高斯损失函数只关注有真实结果的，mask=0的地方不会计入损失函数

高斯学习相关知识

7. **疑问：**
   a. 相机的内参和外参如何得到：The camera intrinsic matrix and the extrinsic matrix of I0 are denoted as K and P0, respectively. For the case where I0 and D0 are generated from the diffusion model, we set the values of K and P0 by convention regarding the size of the image.
   b. 高感知质量是什么：high perceptual quality
   c. 训练的时候添加M照片是为了什么（实验经验吗）：For the images to train the model, we use additional M images as well as (N + 1) images for generating the point cloud, since the initial (N + 1) images are not sufficient to train the network for generating the plausible output. The M new images and the masks are generated by reprojecting from the point cloud PN by a new camera sequence of length M, denoted as PN+1, ..., PN+M.
   d. alingnment移动点，为什么就可以解决一致性的问题，直接使用差值不可以么

8. **和项目的关系**：luciddreammer是利用stable diffusion和3dgs的整体框架，可以利用stable diffusion和深度图来提供多视角，新的三维场景。但是我们的项目是想要做超分，相当于并不是实现横向的新场景添加，是要实现纵向的超分效果。提供了一种思路，就是利用stable生成点云和alignment进行结合。因为lucciddreamer本身有提供框架和代码，所以在luciddreamer上结合power of ten的思路，可以尝试实现。
9. **发现**：lucciddreamer可以处理好，深度之间的关系，一旦处理好深度之间的关系，纵深感就会给出一种类似超分的感觉，但是依然没有实现，没有做到在图片中图片的效果。整个最大的贡献，就是保证了通过单张图片和提示来完成的3D场景的合成，也就是多视角的一致性，通过深度估计和几何投影完成了这样的效果

代码复现与理解



1. 前几天主要是在进行luciddreamer的环境配置，周二开始进行代码调试，修改相应的参数来观察对生成速度和质量的影响
    a. 我尝试自定义输入图像，发现就是复杂纹理区域生成的效果非常的差，可能是因为遮挡关系处理的不够准确
    b. 文件结构：
        i. model.py是3dgs模型定义
        ii. dreamer.py是文本到图像生成模块
        iii. alignment.py:是多视角对齐优化
        iv. `renderer.py是` 3d渲染
    c. 参数调整：gaussian_args是调整高斯球数量，密度。ptimization `是` 修改学习率和迭代次数
2. 之后尝试对luciddreamer来进行一些代码的修改
    a. 在dreamer.py中添加了预处理函数，这样可以增强输入图像的细节

b. 修改renderer.py理的渲染参数，从64变成了128

i. self.num_samples_per_ray=128

3. 3D信息提取（单视角到多视角）

```
depth_curr = self.d(image_curr)  # ZoeDepth深度估计
```

a. 这里使用 zoedepth模型直接估计深度了，有了深度图就可以方便之后渲染那

4. dream过程

```
for i in range(1, len(render_poses)):
    # 1. 将已有的3D点投影到新视角
    pts_coord_cam2 = R.dot(pts_coord_world) + T
    pixel_coord_cam2 = np.matmul(K, pts_coord_cam2)

    # 2. 生成部分图像（已知区域）
    image2 = interp_grid(pixel_coord_cam2, pts_colors, grid)

    # 3. 用SD修复未知区域
    image_curr = self.rgb(
        prompt=prompt,
        image=image2,
        mask_image=mask2  # 标记需要修复的区域
    )
```

a. 对于未知的3D区域，直接通过stable diffusion来进行生成，而深度就是通过她所谓的优化算法，来对准到现有的几何

b. 对于alignment的算法

```
# 深度尺度优化 - 解决不一致性
sc = torch.ones(1).float().requires_grad_(True)  # 学习尺度参数
trans3d = torch.tensor([[sc,0,0,0], [0,sc,0,0], [0,0,sc,0], [0,0,0,1]])

# 最小化新旧点云距离 - 对应论文中的对齐目标
loss = torch.mean((torch.tensor(pts_coord_world[:,valid_idx]).float()
                   - coord_world2_trans)**2)
```

```
# 检测mask边界变化 - 对应论文中的|∇M |
mask_hf = np.abs(mask2[:H-1, :W-1] - mask2[1:, :W-1]) + \
          np.abs(mask2[:H-1, :W-1] - mask2[:H-1, 1:])
border_valid_idx = np.where(mask_hf[round_coord_cam2[1],
round_coord_cam2[0]] == 1)[0]
```

代码块

```
# 计算相机射线方向 - 对应论文中的射线约束
vector_camorigin_to_campixels = coord_world2_trans.detach().numpy() -
camera_origin_coord_world2
vector_camorigin_to_pcdpixels = pts_coord_world[:,valid_idx[border_valid_idx]]
- camera_origin_coord_world2

# 沿射线移动的深度补偿系数
compensate_depth_coeff = np.sum(vector_camorigin_to_pcdpixels *
vector_camorigin_to_campixels, axis=0) / \
                         np.sum(vector_camorigin_to_campixels *
vector_camorigin_to_campixels, axis=0)
```

代码块

```
# 对M =0区域进行线性插值 - 对应论文描述
masked_pixels_xy = np.stack(np.where(1-mask2), axis=1)[:, [1,0]]
new_depth_linear, new_depth_nearest = interp_grid(pixel_cam2,
compensate_depth, masked_pixels_xy), \
                                      interp_grid(pixel_cam2, compensate_depth,
masked_pixels_xy, method='nearest')
new_depth = np.where(np.isnan(new_depth_linear), new_depth_nearest,
new_depth_linear)
```

代码块

```
# 组合新旧点云 - 对应论文中的P    = P     ∪ W(P   )
new_pts_coord_world2 = (np.linalg.inv(R).dot(new_warp_pts_coord_cam2) -
                        np.linalg.inv(R).dot(T)).astype(np.float32)
# 累积到全局点云中
pts_coord_world = np.concatenate((pts_coord_world, new_pts_coord_world2),
axis=1)
```

alignment之所以能够解决，就是在处理因为视角不同导致深度估计不同，所以会出现不一致性的现象，所以他沿着射线方向来对这个深度图进行优化，让两个深度估计最佳接近，这样可以确定视角。同时新的视角也迭代完成，所以可以保证每个视角的深度误差都不会那么的大。

```
# 检测边界变化
boundary = |∇mask| > 0

# 在边界附近进行约束插值
depth_smooth = interpolate(depth_known, positions=boundary_pixels)
```

边界变化过大的话，需要对他们重新进行差值处理，这样能够让边界丝滑

5. 一个有趣的点就是，整个luccidreamer相机标准化的过程采用的是nerf的相机标准，这样也能和3DGS进行互通

Power of ten

论文解读

We achieve this through a joint multi-scale diffusion sampling approach that encourages consistency across different scales while preserving the integrity of each individual sampling process.

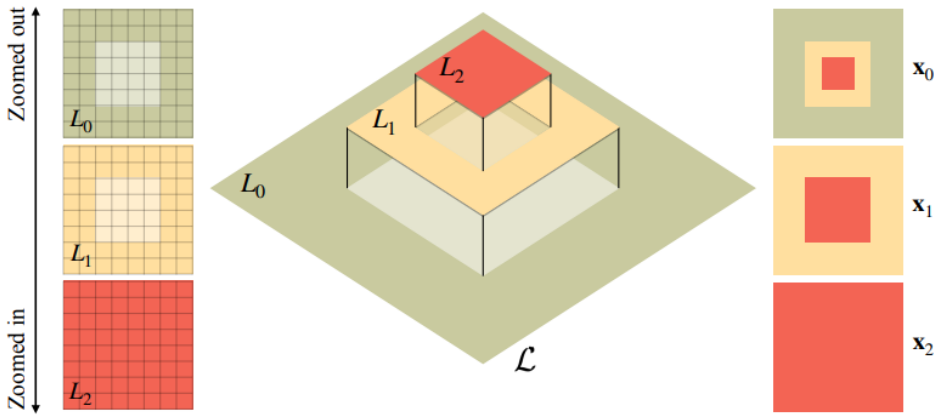
our method enables deeper levels of zoom than traditional super-resolution methods that may struggle to create new contextual structure at vastly different scales

1. **优点**：相比传统的超分结构，在contextual上，上下文结构上具有更好的效果。现有的文生图模型，无法实现在zoom level上的consistent
    a. 传统的超分仅仅依靠图像信息，因此无法再deeper上实现细节，但是我们有text promot可以解决这个问题
    b. 每个scale上的plausible image，同时每个scale之间又是连续
    c. 传统的超分也可以实现zoom in和zoom out的效果，但是上下文关系薄弱，递归的时候会产生问题
    d. 同时生成整个consistent sequence
    e. 总的来说，我希望在zoom in和zoom out过程中，整个语音可以被保持
2. **难点**：zoom in的问题是语义semantic：因为我们生成的图片之间应该要有很多的过度关系（比如一个人的手掌放大会有皱纹）

3. **输入**：text提示词（会包括不同的scale的提示词）
4. **过程**：
   a. 先介绍了diffusion模型生成图像的过程，逐步添加高斯噪声来生成图片（train），如果train完了，就需要利用denooise的过程，来得到干净的图像
      i. 我对stable diffusion模型的学习 diffusion模型的核心原理就是预测噪声，利用噪声消除来得到
   b. 目标：利用生成模型生成一组图片，图片之间的在不同的 zoom level上（大图片经过裁剪），能够保持consist

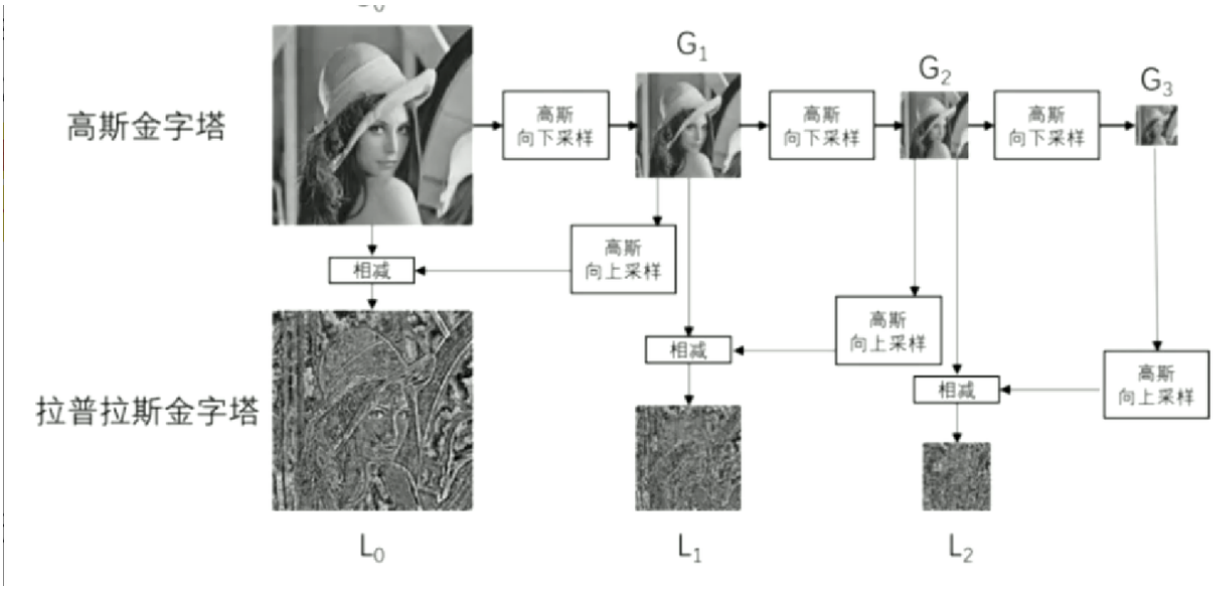
> consistent way. This means that the image $\mathbf{x}_i$ at any specific zoom level p_i, should be consistent with the center $H/p \times W/p$ crop of the zoomed-out image $\mathbf{x}_{i-1}$.

   c. 因此提出了两个方法：multi-scale joint sampling和 zoom stack representation
      i. Zoom stack帮助我们在不同的level来渲染图片。iamge rendering保证了在不同的zoom level，都可以在overlap的地方实现一致性
         1. Image rendering：将图像进行下采样（降低分辨率），同时用zero-pad进行填充，然后利用高斯核进行迭代替换，这样可以保证图像的连续性

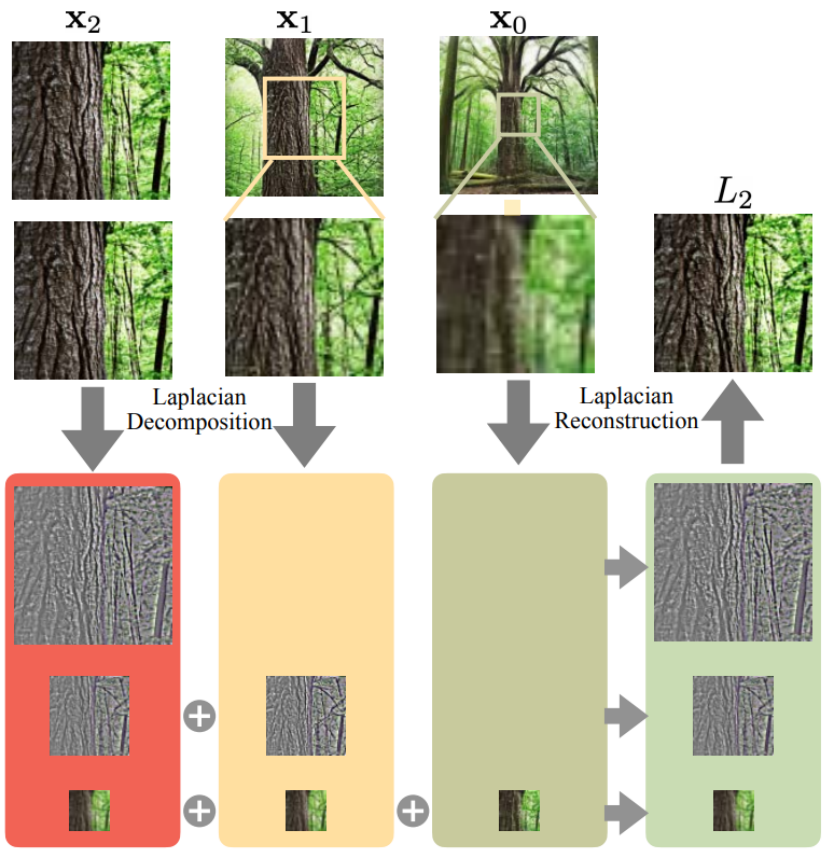


         2. Noise image：这里保证了语义的连续性，因为本质上diffusion是VAE和DDPM的集合，是DDPM在语义层面的扩散，VAE确保了语义过程的实现。这里就实现了和刚刚image rendering类似的操作，将独立的噪声打包成zoom-consistant，这个过程和image rendering中，对zoom stack的打包类似，同样有高斯核的过滤。
   d. Multi-resolution blending：
      i. 把多个zoom level整合成一个stack来渲染
      ii. 因为在不同的zoom level形成的图像不具有连续性，一种方法就是直接朴素平均，但是朴素平均会导致信息的丢失。
      iii. 解决这个问题的方法是 multi resolution blending，用拉普拉斯金字塔来融合不同的 frequency band（这里设计到一个知识，图像金字塔）

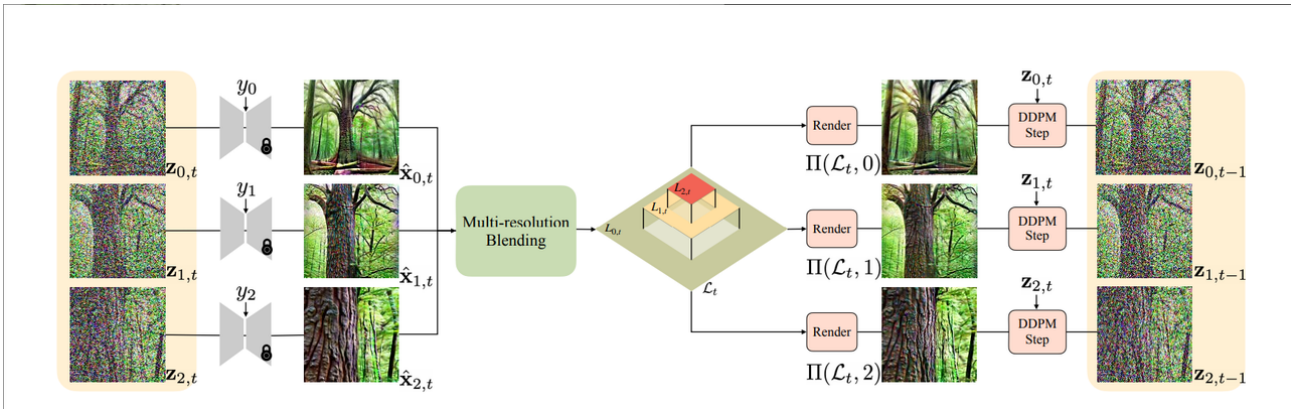
1. 简单的下采样和上采样会导致信息的丢失，图像金字塔有高斯金字塔和拉普拉斯金字塔。高斯金字塔就是简单的模糊并且下采样
2. 拉普拉斯金字塔在高斯金字塔的基础上，进行上采样，同时用高斯金字塔的结果减去同一层，就可以得到拉普拉斯金字塔（关注的事差异）



3. 拉普拉斯金字塔就可以保存每一层的信息，在这里，生成的每个照片已经有连续性，而且存在分辨率大小的问题，因此把每个图片都看做拉普拉斯金字塔的一层，做模糊处理同时得到拉普拉斯金字塔，就可以得到保留的信息，这样不同层的信息就可以保留



4. 每个zoom level的image都形成自己的拉普拉斯金字塔，把细节进行相加，利用相加的细节在进行重建，来保持图像的一致性



e. 整个过程的简要概述，噪声图片先用diffusion model来生成一系列的图片，然后对这些图片采用multi-resolution，也就是融合，融合得到的 zoom stack再次重新渲染，保证更好的一致性之后再重新用DDPM进行训练，这样DDPM的结果就可以保持一致性

   i. 这样是一个DDPM的一个步骤，相当于我在去噪声的时候，对每个噪声进行这样一步的处理

   ii. 等全部的噪声处理完，我就可以得到干净的图片

> - 从当前zoom stack $\mathcal{L}_t$ 渲染出一致的噪声图像 $\mathbf{z}_{i,t}$ 和噪声 ϵ_i（使用渲染函数Pi_image和Pi_noise，Algorithm 1：这涉及下采样更高层并掩码融合，确保低频一致）。
> - 将 $\mathbf{z}_{i,t}$ 和对应提示 y_i 输入预训练扩散模型，预测噪声 $\hat{\epsilon}_{i,t-1}$，从而计算估计的干净图像 $\hat{\mathbf{x}}_{i,t-1}$。（此时，每个尺度的 $\hat{\mathbf{x}}_{i,t-1}$ 可能在重叠区域不一致。）

f. 最终就可以完成这样的构造（不仅可以用文字生成，也可以直接用图片生成）

5. **疑问**：

   a. 在noise rendering过程中，因为对zoom level存在下采样，DDPM又要求是高斯分布，是如何保证每次噪声都是高斯分布的

   b. 同样是在noise rendering过程中，noise image之间相互独立，是如何把他们打包成一个zoom-consiste noise。（文章中提到是和image rendering类似的操作，但是噪音是如何和图像做到一样的操作的）

   c. 我在image rendering 的时候已经做到了一致性，为什么后面还需要multi-resolution利用拉普拉斯金字塔来保持一致性

6. **和项目的关系**：将这个文章的思想和luciddreamer结合在一起，基本上可以很好的完成无限放大3D的效果。因为lucid利用text promot可以预测出新图在三维空间的相机位置，而power of ten的思想，多重融合的角度可以让三维中的某一个视角完成超分。

   a. 问题的难点在于三维的一致性如何保持，也就是在zoom的过程中，luciddreamer生成的周边图像也依然需要保持一致性。（多视角的一致性）

   b. 但是已经可以解决固定视角的交互性问题，对于单视角来说，利用multi-resolution bending的思想，在生成新的超分图像的同时，利用luciddreeam确定新的点云的位置（有了新图像，可以确定新的深度图），那么这个点云就可以用作高斯渲染，完成单视角的固定

c. 所以现在的难点就是在三维多视角下，保持一致性。

LocalNeRF: Efficient Neural Radiance Fields with Local Sampling (NeurIPS 2023)

论文解读

主要提到了nerf的局部细节方法，用来增强3dgs的细节，因为目前阶段主要是想要实现3D的缩放效果，还没有考虑具体的细节优化，所以并没有想过多参考

ZoeDepth: Zero-Shot Transfer by Combining Relative and Metric Depth (CVPR 2023)

论文解读

这个主要是luciddreammer中提到的，用来解决depth的论文，可以很好的处理遮挡关系，如果有细节可以在考虑根据这个来进行优化

Denoising Diffusion Probabilistic Models

论文解读

最最最基本的diffusion模型，介绍了降噪和去噪的思想，看了很多相关的视频了解了其基本的原理，现在再来看看原文，方便理解整个diffusion的过程

Diffusion models [53] are latent variable models of the form $p_\theta(\mathbf{x}_0) := \int p_\theta(\mathbf{x}_{0:T})\, d\mathbf{x}_{1:T}$, where $\mathbf{x}_1, \ldots, \mathbf{x}_T$ are latents of the same dimensionality as the data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$. The joint distribution $p_\theta(\mathbf{x}_{0:T})$ is called the *reverse process*, and it is defined as a Markov chain with learned Gaussian transitions starting at $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$:

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \qquad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{1}$$

1. 第一步是反向过程，也就是从纯噪声开始，我们可以发现Pxt是符合高斯分布的
2. 根据条件概率，我们每次根据当前的一步来计算前一步
   a. 因为高斯噪声算是最后的一步了，这样就可以实现去噪的过程，因此最终可以回归到X0
   b. 用连乘来表示这个去噪的过程，每一步去噪就是第二个公式
   c. 每次预测下一步，都是用高斯分布来预测，而且高斯分布的参数都是可以学习的

What distinguishes diffusion models from other types of latent variable models is that the approximate posterior $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$, called the *forward process* or *diffusion process*, is fixed to a Markov chain that gradually adds Gaussian noise to the data according to a variance schedule $\beta_1, \dots, \beta_T$:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \qquad q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{2}$$

1. 第二部就是扩散过程，也就是我们增加噪声的过程，直到变成一个高斯噪声
    a. 也就是上面的反过程，所以只要调换一下条件的顺序就可以了
    b. 因为噪声是我们人为添加的，所以只要我们设置对应的高斯系数，就可以得到新的图片

Training is performed by optimizing the usual variational bound on negative log likelihood:

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] = \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] =: L \tag{3}$$

1. 优化的时候参考ELBO，也就是所谓的变分下界
    a. 这个ELBO我在看视频里面有更详细的解释，所以我这里不做过多阐述
    b. 简单的理解就是更加逼近高斯分布

Second, to represent the mean $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$, we propose a specific parameterization motivated by the following analysis of L_t. With $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$, we can write:

$$L_{t-1} = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2\right] + C \tag{8}$$

1. 在训练的过程中，我们直接用噪声当做损失函数，而不是照片本体，因为像素的分布是更加品骏，更加清晰

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t\mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

在训练的过程中，随机抽取一个噪声和时间步，让模型预测这个时间步的噪声，这样来做就是对噪声进行训练

擦除的过程，就是同样是随机取一个时间步，预测出此时的噪声，就可以对应的擦除

High-Resolution Image Synthesis with Latent Diffusion Models

To enable DM training on limited computational resources while retaining their quality and flexibility, we apply them in the latent space of powerful pretrained autoencoders

论文解读

Latent diffusion models和传统的DM相比，是直接在语义空间来进行操作，符合power of ten中提到的思想，所以可以被拿来利用和学习，这种 cross -attention layer的做法能够让整个diffusion模型更加的灵活和强大