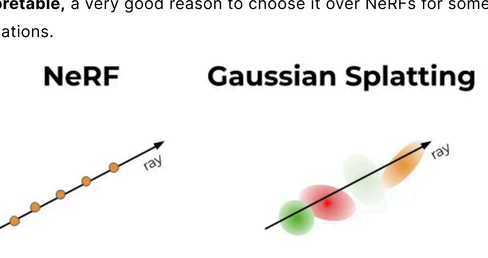
DLCV-HW4

- Student ID: B09611007

3D Novel View Synthesis

1-1. Try to explain 3D Gaussian Splatting in my own words

- In Gaussian splatting, a 3D world is represented with a set of many 3D points. Each point is a 3D Gaussian with **its own unique parameters that are fitted per scene** such that renders of this scene match closely to the known images.
- Its simple and explicit representation makes Gaussian splatting particularly **interpretable,** a very good reason to choose it over NeRFs for some applications.



1-2. Compare 3D Gaussian Splatting with NeRF (pros & cons)

- **NeRF (Ray Tracing)**
  - **Pros:** High quality (reflection, refraction, shadow, texture)
  - **Cons:** Complex & slow
- **Gaussian Splatting (Rasterization)**
  - **Pros:** Fast & easy
  - **Cons:** Not realistic as NeRF

1-3. Which part of 3D Gaussian Splatting is the most important you think? Why?

- There **isn't any neural network,** a scene is essentially just a set of 3D rotated and stretched ellipsoids in space
- It uses Gaussian functions to represent points in space, which allows for **more efficient computation**

2. Describe the implementation details of your 3D Gaussian Spaltting for the given dataset. You need to explain your ideas completely.

- Environment setup

```
conda create -n gs python=3.7
conda activate gs

pip install torch==1.12.1+cu116 torchvision==0.13.1+cu116 torchaudio=
pip install -r requirements.txt

pip install submodules/diff-gaussian-rasterization
pip install submodules/simple-knn
pip install submodules/fused-ssim
```

- Training (train.py (http://train.py))

```
train/
    images/   # 59 images
    sparse/0/  # 59 camera poses
        cameras.txt
        images.txt
        points3D.ply # SFM points
```

- Rendering (render.py (http://render.py))

```
3d_gaussian_optimized.ply # optimized points

public_test/
    images/   # 50 validation images
    sparse/0/  # 50 camera poses
        cameras.txt
        images.txt
```

- Evaluation (grade.py (http://grade.py))

```
python3 grade.py output/renders output/gt
```

3. Given novel view camera pose, your 3D gaussians should be able to render novel view images. Please evaluate your generated images and ground truth images with the following three metrics (mentioned in the 3DGS paper).

- Performance on the public testing set

Setting	PSNR	SSIM	LPIPS (vgg)	# of gaussians
pos_lr=0.002	34.5	0.968	0.065	13414
pos_lr=0.008	36.0	0.974	0.055	13414
pos_lr=0.02	**36.5**	**0.978**	0.048	13414

  - Common params: resolution=1/2, white background, iterations=30000, feature_lr=0.0025, scale_lr=0.005, opacity_lr=0.05, rotation_lr=0.001, densification (default)

- Metrics
  - **PSNR (Peak Signal-to-Noise Ratio):** the similarity between the rendered image and the ground truth by comparing pixel values, with higher values indicating better quality.
  - **SSIM (Structural Similarity Index):** the perceptual quality of an image by considering luminance, contrast, and structure, aiming to mimic human visual perception for image similarity.
  - **LPIP (Learned Perceptual Image Patch Similarity) (vgg):** the perceptual similarity between images by using VGG.

4. Instead of initializing from SFM points, try to train 3D gaussians with random initializing points.

- Method to initialize 3D gaussians

```
def randomize_point_3d():
    # Random coordinates (x, y, z)
    x = random.uniform(-10, 10)
    y = random.uniform(-10, 10)
    z = random.uniform(-10, 10)

    # Random normals (nx, ny, nz) and normalize
    nx = random.uniform(-1, 1)
    ny = random.uniform(-1, 1)
    nz = random.uniform(-1, 1)

    norm = (nx**2 + ny**2 + nz**2)**0.5
    if norm > 0:
        nx /= norm
        ny /= norm
        nz /= norm

    # Random colors (red, green, blue)
    red = random.randint(0, 255)
    green = random.randint(0, 255)
    blue = random.randint(0, 255)

    print(x, y, z, nx, ny, nz, red, green, blue)

    return (x, y, z, nx, ny, nz, red, green, blue)

# Generate 13,414 random points
point_3d_data = []
for _ in range(13414):
    point_3d_data.append(randomize_point_3d())
```

- Training with random initializing points

Init. points	PSNR	SSIM	LPIPS (vgg)	# of gaussians
SFM (Structure from Motion)	36.5	0.978	0.048	13414
Random	**35.8**	**0.974**	**0.056**	13414

Reference

[1] https://arxiv.org/abs/2308.04079 (https://arxiv.org/abs/2308.04079)
[2] https://www.youtube.com/watch?v=UxP1ruyFOAQ (https://www.youtube.com/watch?v=UxP1ruyFOAQ)
[3] https://towardsdatascience.com/a-comprehensive-overview-of-gaussian-splatting-e7d570081362 (https://towardsdatascience.com/a-comprehensive-overview-of-gaussian-splatting-e7d570081362)