Deep Learning for Computer Vision

HW4

電子所 ICS 組, R13943015, 張根齊

Problem: 3D Novel View Synthesis

1. Please explain:

a. Try to explain 3D Gaussian Splatting in your own words

Gaussian Splatting 是給定一場景，訓練出該場景的 3D 模型(立體模型，非 prediction model)的技術。 Gaussian Splatting 的 3D model representation 類似於 point cloud 都是 explicit，但是相對於 point cloud 由多個點組成，gaussian splatting 的 3D 模型是由多個橢圓體組成。而不同於 NeRF，gaussian splatting rendering 的方法是將場景中的物體由近到遠向 camera 投影(splat)，看會影響圖片中的那些像素。

b. Compare 3D Gaussian Splatting with NeRF (pros & cons)

pros:

1. 相較於 NeRF，3DGS 不管在 training phase 或是 rendering phase 都比較快，因為 gaussian 橢圓球體其不需要透過 NN 來得到場景資訊，且 3DGS 的 rasterization 不用像 NeRF 的 ray tracing 需處理複雜光射線。

2. 因為不需要透過 NN 的特性，3DGS 視覺化簡單，相較於 NeRF，可以較輕鬆的直接使用於遊戲引擎上。

cons:

1. 相較於 NeRF 的 ray tracing，3DGS 的 rasterization render 出的結果在真實性較差，在反射、折射、陰影、材質的效果也沒有 ray tracing 來的好。

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

我覺得最重要的部分是採用 gaussian 橢圓球體作為 3D 立體建模，因為 explicit 的表示不僅讓 3DGS 在 render 時不需要過 NN 來獲取場景資訊，以達到 real-time rendering，用 gaussian

橢圓球體表示場景也更好的應用於遊戲引擎等視覺化應用。

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

Training:

訓練階段和官方 gaussian-splatting 的訓練一模一樣，沒有對 train.py 做更動。訓練大致可以分為以下幾個步驟，一開始先載入 TA 提供的 SfM points 作為訓練場景的初始值，接著從 training set 中隨機選取 camera 視角，對 guassian 橢球建立的場景做 rendering，並將 render 得到的圖片和 training set 的 ground truth 圖片取 loss(L1 & SSIM)、更新 guassian 橢球參數，在訓練過程中，會透過 densify/pruning 來增多/減少 guassian 橢球數目。

在訓練階段時，我將 densification 的 iteration 限制在 100 ~ 3000 中，預設是 500~15000。另外使用白色背景，並且將 densification 的 threshold(scene extent)提高到 0.1，也提高 SSIM loss 的權重，並且增加訓練迭代次數到 60000。

Training command:

```
python3 ./gaussian-splatting/train.py -s ./dataset/train -m checkpoint_model/model_v16 --densify_from_iter 100 --percent_dense 0.1 -w --densify_until_iter 3000 --lambda_dssim 0.5 --iterations 60000
```

Rendering:

Rendering 基本也是完全和官方一樣，只有做一些改動。3DGS 將場景用多個 gaussian 橢球表示，每個橢球包含位置、透明度、顏色、縮放和旋轉等參數，在 render 時會給定一組 camera 視角，用這組視角將每個 gaussian 橢球都從 3D 轉換到 2D 空間上，並用 rasterization 將橢球合成到圖片上。而以下是改動部分:

render.py:

在 render.py 原本會在 model_path 產生 gt 資料夾放 ground truth 圖片，且也產生 renders 資料夾放生成的 render 圖片，且圖片名為 00000.png ~ xxxxx.png。本次作業要求將生成圖片放置於指定資料夾，不需要 gt 資料夾，並且圖片名應和 image.txt 中 image name 相同，所以改動如下。

```python
def render_set(model_path, name, iteration, views, gaussians, pipeline, background, train_test_exp, separate_sh):
    #render_path = os.path.join(model_path, name, "ours_{}".format(iteration), "renders")
    #gts_path = os.path.join(model_path, name, "ours_{}".format(iteration), "gt")
    render_path = os.path.join(args.output_folder) # save redering images to specified output_folder path

    makedirs(render_path, exist_ok=True)
    #makedirs(gts_path, exist_ok=True)
    #print(views[0].image_name) # 127460000.png
    for idx, view in enumerate(tqdm(views, desc="Rendering progress")):
        rendering = render(view, gaussians, pipeline, background, use_trained_exp=train_test_exp, separate_sh=separate_sh)["render"]
        #gt = view.original_image[0:3, :, :]

        if args.train_test_exp:
            rendering = rendering[..., rendering.shape[-1] // 2:]
            #gt = gt[..., gt.shape[-1] // 2:]

        #torchvision.utils.save_image(rendering, os.path.join(render_path, '{0:05d}'.format(idx) + ".png"))
        #torchvision.utils.save_image(gt, os.path.join(gts_path, '{0:05d}'.format(idx) + ".png"))
        torchvision.utils.save_image(rendering, os.path.join(render_path, view.image_name))# save redering images using filenames from image.txt
```

dataset_readers.py:

在 render 時，並不會用到用於訓練的初始值(SfM points: points3D.ply)，且在 private set 和 public set 也不會給 points3D.ply，所以會出錯，所以將讀取 SfM 初始值的方式改為下圖，在 render 時就將 pcd 設為 None，不會影響 render 結果。

```python
def readColmapSceneInfo(path, images, depths, eval, train_test_exp, llffhold=8):
    ply_path = os.path.join(path, "sparse/0/points3D.ply")
    bin_path = os.path.join(path, "sparse/0/points3D.bin")
    txt_path = os.path.join(path, "sparse/0/points3D.txt")

    # original code:
    """
    if not os.path.exists(ply_path):
        print("Converting point3d.bin to .ply, will happen only the first time you open the scene.")
        try:
            xyz, rgb, _ = read_points3D_binary(bin_path)
        except:
            xyz, rgb, _ = read_points3D_text(txt_path)
        storePly(ply_path, xyz, rgb)
    try:
        pcd = fetchPly(ply_path)
    except:
        pcd = None
    """

    if not os.path.exists(ply_path):
        try:
            print("Converting point3d.bin to .ply, will happen only the first time you open the scene.")
            try:
                xyz, rgb, _ = read_points3D_binary(bin_path)
            except:
                xyz, rgb, _ = read_points3D_text(txt_path)
            storePly(ply_path, xyz, rgb)
        except:
            # When rendering, public test and private test folder has no points3D.ply ==> set pcd = None
            # And we dont need points3D.ply(SfM points for training initialization) during redering phase
            pass
    try:
        pcd = fetchPly(ply_path)
    except:
        pcd = None
```

camera_utils.py:

這部分和上部分類似，在 render 時也不需要 gt images，為了防止錯誤，若 images/ 資料夾不存在就創建 861*478 的黑色圖片(TA 提供的 dataset 圖片大小)，同樣因為 render 不會用到，所以結果不會變。

```python
def loadCam(args, id, cam_info, resolution_scale, is_nerf_synthetic, is_test_dataset):
    # original code:
    """
    image = Image.open(cam_info.image_path)
    """
    try:
        image = Image.open(cam_info.image_path)
    except:
        # if dataset has only ./sparse/0/ and has no ./images, just use black image(render result is unchanged)
        default_size = (861, 478)  # (width, height)
        image = Image.fromarray(np.zeros((default_size[1], default_size[0], 3), dtype=np.uint8))

```

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). Try to use at least three different hyperparameter settings and discuss/analyze the results.

- Please report the PSNR/SSIM/LPIPS on the public testing set.
- Also report the number of 3D gaussians.
- Different settings such as learning rate and densification interval, etc.

Setting	hyperparameters	PSNR	SSIM	LPIPS(vgg)	Number of 3D gaussian
A	densify_from_iter 100 densify_until_iter 15000 percent_dense 0.1 white_background lambda_dssim 0.2 iterations 30000	36.035	0.974	0.093	287543

B	densify_from_iter 100 **densify_until_iter 3000** percent_dense 0.1 white_background lambda_dssim 0.2 iterations 30000	37.342	0.978	0.079	248269
C	densify_from_iter 100 densify_until_iter 3000 percent_dense 0.1 white_background **lambda_dssim 0.5** iterations 30000	37.638	0.980	0.072	463955
D	densify_from_iter 100 densify_until_iter 3000 percent_dense 0.1 white_background lambda_dssim 0.5 **iterations 60000**	38.286	0.981	0.068	460518

由 A set 和 B set 可以發現減少 densification 的 iteration 次數可以增加 PSNR, SSIM 並且減少 LPIPS，代表生成的圖像更接近 ground truth，並且也可以發現 Number of 3D gaussian 減少，這是因為 densification 的次數減少所導致。而 C set 將 SSIM loss 的權重由 0.2 增加至 0.5，在 3 個 metric 上都有更好的表現，並且也大幅提升 Number of 3D gaussian。D set 相較於 C set 又多訓練 30000 個 iteration，達到更好的 result。

- You also need to explain the meaning of these metrics.

1. PSNR (Peak Signal-to-Noise Ratio)

用 pixel 差異評估原始圖像和生成圖像的相似度。單位為分貝 (dB)，值越高，代表重建圖像的誤差越小，越像原圖。缺點是只考慮 pixel 差異，忽略人眼感知特性。

2. SSIM (Structural Similarity Index Measure)

用於衡量圖像的結構相似性，模擬人眼對亮度、對比度和結構的感知。範圍為 0 ~ 1，值越接近 1，代表圖像品質越好。

3. LPIPS (Learned Perceptual Image Patch Similarity)

用 VGG 計算兩張圖像的感知差異。值越低，代表圖像在感知上越相似。優勢是比 PSNR 和 SSIM 更符合圖像的感知質量（用學出來的 feature，而非 pixel 或簡單結構）。

4. Instead of initializing from SFM points [dataset/sparse/points3D.ply], please try to train your 3D gaussians with random initializing points.

- Describe how you initialize 3D gaussians

透過觀察助教提供的 points3D.ply，發現 gaussians 橢球 X, Y, Z 的最大值、最小值分別為 117、-66，所以我 X, Y, Z 我是從(120, -70)隨機取值。並且發現助教提供的 points3D.ply 的 gaussians 橢球 R, G, B 數值都很接近，所以我控制 R, G, B 相差不超過 5，error 則是如助教提供的 points3D.ply 設 0，points 初始數量為 15000(助教提供的 points3D.ply 為 13414 點)。

```python
def create_random_points3D(file_path, num_points=15000):
    """ create random points3D.txt using COLMAP format"""
    with open(file_path, 'w') as file:
        # write the header
        file.write("# 3D point list with the following format:\n")
        file.write("# POINT3D_ID, X, Y, Z, R, G, B, ERROR, TRACK[] as (IMAGE_ID, POINT2D_IDX)\n")

        for point_id in range(1, num_points + 1):
            # random (X, Y, Z)
            x, y, z = [random.uniform(-70.0, 120.0) for _ in range(3)]

            # random (R, G, B): close to each other
            r = random.randint(5, 250)
            g = r + random.randint(-5, 5)
            b = r + random.randint(-5, 5)
            # Ensure RGB values are within valid bounds (0-255)
            r, g, b = [max(0, min(255, v)) for v in (r, g, b)]

            error = 0.0

            # Write the point data to the file
            file.write(f"{point_id} {x:.6f} {y:.6f} {z:.6f} {r} {g} {b} {error:.1f}\n")
```

- Compare the performance with that in previous question.

Setting	hyperparameters	PSNR	SSIM	LPIPS(vgg)	Number of 3D gaussian
E	densify_from_iter 100 densify_until_iter 3000 percent_dense 0.1 white_background lambda_dssim 0.5 iterations 60000 **use random initial points**	34.273	0.952	0.121	349225

相較於 D set 使用 SfM 作為 initial points，E set 採用 random initial points，和 D set 相比 E set 的 3 個 metric 都表現得更差，但 Number of 3D gaussian 比較少。

Reference:

1. Chatgpt

https://chatgpt.com/

2. AI 甘安捏

https://www.youtube.com/watch?v=LLSLSRgoIjY

https://www.youtube.com/watch?v=KPO8BpzY0LA

https://www.youtube.com/watch?v=UxP1ruyFOAQ

3. gaussian-splatting

https://github.com/graphdeco-inria/gaussian-splatting