Module 2: Computer Vision Assignment

Practice Assignment: 请完成 Task1 和 Task2.

Written Assignment：请完成Task1.

Bonus(占总评1分)：请完成 Task1 和 Task2 中所有“可选”的步骤

Task 1

使用 Omnidata (https://github.com/EPFL-VILAB/omnidata/tree/main) 提供的预训练模型进行深度和法线估算。

- 克隆 repo Omnidata (https://github.com/EPFL-VILAB/omnidata/tree/main)，并准备python环境：

```
git clone git@github.com:EPFL-VILAB/omnidata.git
cd omnidata/omnidata_tools/torch
conda create -n omnidata -y python=3.8
source activate omnidata
pip install -r requirements.txt
```

- 通过下面的命令下载预训练模型：

```
pip install gdown
mkdir -p pretrained_models
gdown '1Jrh-bRnJEjyMCS7f-WsaFlccfPjJPPHI&confirm=t' -O ./pretrained_models/ # omnidata depth (v2)
gdown '1wNxVO4vVbDEMEpnAi_jwQObf2MFodcBR&confirm=t' -O ./pretrained_models/ # omnidata normals (v2)
```

如果无法连接到模型 URL，请点击 此处 (https://cloud.tsinghua.edu.cn/d/88a344983a9843399b21/) 手动下载。

- 将预训练的模型用于不同领域的图像，如照片/绘画、真实/合成图像、带/不带背景的图像、室内/室外场景等。需要从不同领域选择至少 5 张图像，其中至少包括一幅模型预测结果不佳的图像。运行模型的命令如下：

```
python demo.py --task $TASK --img_path $PATH_TO_IMAGE_OR_FOLDER --output_path $PATH_TO_SAVE_OUTPUT
### Example
python demo.py --task normal --img_path assets/demo/test1.png --output_path assets/
```

注意： 该库的原始 torch 版本较低，可能与你的 GPU CUDA 版本不匹配。如果出现以下错误 "RuntimeError: CUDA error: no kernel image is available for execution on the device CUDA kernel errors might be asynchronously reported at some other API call, so the stack trace below might be incorrect." 请安装对应的 CUDA 版本的 torch。

- 比较不同图像上的结果，分析该模型在某些类型的图像上表现良好或不佳的原因。
- 根据上述结果和你的分析，讨论应该如何改进模型性能？
- 找到一个优于 Ominidata 模型的单目深度估计模型，并简要概述这项工作如何提高深度估计任务的性能。
- （可选）尝试你找到的模型，并将您的图像结果与 Ominidata 的模型进行比较，尤其是效果不佳的图像。

你的报告应包括：

- 实验结果（image, normal, depth）。(15 分）
- 对结果的分析。(15 分）
- 改进性能的可能途径。(10 分）
- 更好的深度估计模型的概述。(10 分）

Task 2

从多视角 RGBD 重建 Mesh。

- 给定多视角 RGBD（图像和深度），补全 `hw.ipynb` 中的代码，以重建物体的 Mesh。TSDF 流程已经编写完成。只需完成相机参数（包括 pose 和 intrinsic）和 RGBD 图像的读取。RGB 图像和深度图像分别在 `color` 和 `depth` 文件夹下，intrisic 和 pose 都存储在 `camera.json` 文件中。请注意相机坐标系和图像格式及形状要求。
- **注意**：读取depth图像后，需要将其数值缩小1000倍进行单位转换。
- **注意**：修正相机坐标问题。如果你发现结果不正确，这可能是相机坐标系不匹配造成的。你需要将相机位姿绕 x 轴旋转 180 度来解决这个问题。在 3D 视觉研究中，相机位姿和坐标系是一个需要注意的常见问题，相关的 bug 不熟练的情况下可能需要几天才能找到。
- 尝试 TSDF 的不同超参数，包括视图数量、voxel_size、depth_truc 和 sdf_truc。了解这些超参数对最终结果的影响。
- (可选）使用 bender (https://www.blender.org/) 可视化并比较重建网格与真实网格，并在报告中截图汇报。
- (可选）安装pytorch3D，计算重建误差，并在报告中汇报。
- **注意**：定量计算重建效果（Chamfer Distance）时需要使用 pytorch3D 库，该库安装较复杂，安装方式如下（请先确保 torch 等其他库已经安装）： 直接从 github 安装

```
pip install "git+https://github.com/facebookresearch/pytorch3d.git"
```

先本地 clone 再安装

```
git clone https://github.com/facebookresearch/pytorch3d.git
cd pytorch3d && pip install -e .
```

你的报告应包括：

- 补全的 ipynb 文件以及可视化的结果。
- 读取带有相机参数的 RGBD 图像并重建mesh，但结果错误。(20 分）
- 解决摄像机坐标问题并重建正确的mesh。(15 分）
- 不同超参数下的结果比较分析。(15 分）