Final Project

1 3D Object Segmentation[40 pt]

Problem 1. In this problem, you will use a network to do object segmentation in a dataset.

1. Task description. The task is to segment objects for all the objects of interest in the scene given the image and depth map. You are required to segment 79 objects classes in the dataset, and each object class forms a single label of segmentation. The input is "xxx_color_kinect.png" and "xxx_depth_kinect.png", and you need to output "xxx_label_kinect.png". The dataset description is provided in part 6.

2. Model requirements. You are **required** to use a network with 3D components. If you only use 2D detection/segmentation networks, your maximum score for the report will be 60%. You need to describe your method in the report. Here are some recommended methods:

   (a) Frustum PointNet: https://arxiv.org/abs/1711.08488

   (b) MVPNet: https://arxiv.org/abs/1909.13603

3. Output format. You need to provide a zip file containing 1,000 images, their names should be "{level}-{scene}-{variant}_label_kinect.png".

4. Evaluation (10 pts). The evaluation metric is the mean Intersection-over-Union (mIoU) among all label categories. Your score of this part is computed by the perfomance of your submission in part 3:

   - 10 points: If you achieve higher than 80 mIoU.
   - 8 points: If you achieve higher than 60 mIoU.
   - 6 points: If you achieve higher than 40 mIoU.
   - 4 points: If you achieve higher than 20 mIoU.
   - 2 points: Make a submission.

5. Codes (10 pts). You need to provide your complete experiment codes. In order to make sure others can understand and run your codes, you should also provide a guidance or documentation containing running environment, running method, and some necessary instructions of your core functions and models.

6. Report (20 pts). The report should explicitly describe your network model and your experiments. Things to include: network architecture, experiment setups, training details, validation and test scores, visualizations and failure sample analysis. You may write more if you have interesting findings.

7. Dataset description. You can download the whole dataset from folder *3DVC_Project* on https://cloud.tsinghua.edu.cn/d/9c5032cf151046598988/. You need to download 5 zip files **training_data_{1-5}.zip** and merge these files to obtain the whole training data.

   In training data, there are 2 folders *data* and *splits*. The *data* folder contains the training data and *splits* is a pre-made train/validation split. You are allowed to create your own train/validation split and ignore *splits* completely. In *split/train.txt* and *split/val.txt*, each line is in the format "{level}-{scene}-{variant}". You can use it to find corresponding data files in the *data* folder. Here we give a detailed description of all files:

   - **{level}-{scene}-{variant}_color_kinect.png**: An RGB image captured from a camera containing the target objects.
   - **{level}-{scene}-{variant}_depth_kinect.png**: A depth image captured from a camera containing the target objects. The depth is in the unit of mm. You need to convert it into m.
   - **{level}-{scene}-{variant}_label_kinect.png**: A segmentation image captured from a camera containing the target objects. The segmentation numbers for objects are from 0 to 78. We do not care about the background numbered from 79 to 81.
   - **{level}-{scene}-{variant}_meta.pkl**: Camera intrinsic.

   In testing data, there are a folder *data* containing 500 pairs of data. You need to provide **{level}-{scene}-{variant}_label_kinect.png** image for each pair of data.

8. Example codes. You can find some example codes in *examples* folders.

   - **read_data.ipynb**: A Jupiter notebook to show how to use the data and also some handy scripts for visualization.
   - **network_example**: An example 3D network architecture (3D UNet with sparse convolution: https://arxiv.org/abs/2007.16100) for 3D segmentation. You can install TorchSparse v1.2.0 (https://github.com/mit-han-lab/torchsparse/tree/v1.2.0) for running the code.
   - **eval.py**: Evaluate some metrics on some data.

9. Note. You may assume that each neural network experiment will take 1 whole day on a GPU, so you should start solving the problem early.