Structure of RIGA code

configs

load configs

import load_config()
→ Returns dictionary consisting of loaded configuration

main.py
- def main()
- Creation of model:
  - Optimizer
  - Scheduler
  - Data set
  - Data loader
  - Losses
  - Evaluation metrics

Subfolder \test, \train and \val:
- misc
- model
- optim
- data: tdmatch
- train

utils.py
→ def load_config(path): loads config file as dictionary

data

- ´indoor\test´
- ´indoor\train´

→ Large number of files

dataset

- ´indoor\test´
- ´indoor\train´

→ Large number of files

Auch hier könnte eine Anpassung notwendig sein:
1.240: distance = np.sqrt(…)

Auch hier könnte eine Anpassung notwendig sein:
- 1.280: np.linalg.norm(…)
- 1.286: cur_distances = …

common.py
- def get_square_distance_matrix(…)
- def get_batched_square(…)
- def sample_gt_node_correspondence (…)
- def calc_gt_patch_correspondence (…)
- def farthest_point_subsampling(…)
- def collect_local_neighbors(…)
- def point2node_sampling(…)
- def build_ppf_patch(…)
- def calc_ppf_cpu(…)
- def calc_ppf_gpu(…)
- def calc_patch_overlap_ratio(…)

→ and some other functions are implemented here!

dataloader.py
- def get_dataset(config)
- def get_dataloader(dataset, sampler=None, batch_size=1, num_workers=8, shuffle=True, drop_last=True)

pcd_downsampling.py
- class CameraPose

tdmatch.py
- class TDMatchDataset(data.Dataset)
- def __init__(…)
- def __getitem__(…)
- def __len__(…)

modelnet.py
- def download()
- def load_data(partition)
- def translate_pointcloud(pointcloud)
- def jitter_pointcloud(…)
- def farthest_subsample_points(…)
- class ModelNet40(Dataset)
  - def __init__(…)
  - def __getitem__(…)
  - def __len__(self)

Original_tdmatch.py
- def display_inlier_outlier(…)
- class CameraPose
- def read_trajectory(…)
- def write_trajectory(…)
- class PairDataset(data.Dataset)
- class OriginalTDMatchDataset(…)

Original_tdmatch.py
- def display_inlier_outlier(…)
- class CameraPose
- def read_trajectory(…)
- def write_trajectory(…)
- class PairDataset(data.Dataset)
- class OriginalTDMatchDataset(…)

transforms.py
- def uniform_2_sphere(…)
- class SplitSourceRef
- class Resampler
- class FixedResampler(Resampler)
- class RandomJitter
- class RandomCrop
- class RandomTransformSE3
- class RandomTransformSE3_euler(…)
- class RandomRotatorZ(…)
- class ShufflePoints
- class SetDeterministic
- class Dict2DcpList
- class Dict2PointnetLKList

scripts

- benchmark_registration_3dlomatch_c2f.sh
- benchmark_registration_3dlomatch.sh
- benchmark_registration_3dmatch_c2f.sh
- benchmark_registration_3dmatch.sh
- download_data.sh

→ Execution of Python scripts!

snapshot

→ Duplication of a lot of files!!

visualizer

feature_space.py
- def get_color_map(x)
- def mesh_sphere(…)
- def get_colored_point_cloud_feature(…)
- def embed_tsne(data)
- def visualize_feature_space(…)

plot.py
- def draw_distance_geo_feat(…)

visualizer.py
- class Visualizer(object)
  - def __init__(…)
  - def get_random_color()
  - def mesh_sphere(…)
  - def generate_open3d_vis_pcd_from_points(…)
  - def show_alignment(self)
  - def show_correspondences(…)
  - + further visualization functions…

lib

loss.py
- class MetricLoss(…)
  - def __init__(…)
  - def contranstiveLoss(…)
  - def infoNCE(…)
  - def circleLoss(…)
  - def calc_fine_matching_loss(…)
  - def calc_fine_matching_recall(…)
  - def calc_recall(…)
  - def forward(…)

tester.py
- clclass Tester(Trainer)
  - def __init__(self, config)
  - def test(self)
  - def get_trainer(config)

trainer.py
- class Trainer(object)
  - def __init__(self, config)
  - def _snapshot(self, epoch, name=None)
  - def _load_pretrain(…)
  - def _get_lr(…)
  - def stats_dict(self)
  - def stats_meter(self)
  - def inference_one_batch(…)
  - def inference_one_epoch(…)
  - def train(self)
  - def eval(self)

utils.py
- def setup_seed(seed)
- def get_correspondences(…)
- def matching_descriptors(…)
- def square_distance(src, tgt)
- def weighted_procrustes(…)
- def sinkhorn(…)
- def interpolate(weights, points)
- def soft_assignment(…)
- def get_geometric_structure_embeddings(…)
- class AverageMeter(object)
- class Timer(object)
- class Logger(object)

model

loss.py
- class RIGA(nn.Module)
  - def __init__(self, config)
  - def interpolate(…)
  - def forward(…)

modules.py
- class LearnableLogOptimalTransport(nn.Module)
  - def __init__(…)
  - def log_sinkhorn_normalization(…)
  - def forward(…)
  - def __repr__(…)
- class NodeCorrespondenceSelector(…)
  - def __init__(…)
  - def forward(…)

ppftransformer.py
- def MLP(channels: list, do_bn=True)
- class PointNetFeat(nn.Module)
  - def __init__(…)
  - def forward(…)
  - def attention(…)
- class MultiHeadAttention(nn.Module)
  - def __init__(…)
  - def forward(…)
- class CrossAttention(nn.Module)
  - def __init__(…)
  - def forward(…)
- class PPFTrans(nn.Module)
  - def __init__(…)
  - def forward(…)

utils.py
- def get_activation(…)
- class MonteCarloDropout(nn.Module)
  - def __init__(self, p=0.1)
  - def forward(self, x)
- def get_dropout(…)

registration

benchmark_utils.py
- def to_array(tensor)
- def to_tensor(array)
- def to_o3d_feats(embedding)
- def mutual_selection(score_mat)
- def get_inlier_ratio_correspondence(…)
- def get_inlier_ratio(…)
- def ransac_pose_estimation(…)
- def ransac_pose_estimation_correspondences(…)
- def get_scene_split(…)
- def write_est_trajectory(…)

benchmark.py
- def rotation_error(R1, R2)
- def translation_error(t1, t2)
- def computeTransformationErr(…)
- def read_trajectory(…)
- def read_trajectory_info(…)
- def extract_corresponding_trajectors(…)
- def write_trajectory(…)
- def read_pairs(…)
- def evaluate_registration(…)
- def benchmark(…)

evaluate_modelnet_c2f.py
- def benchmark_evaluation(desc)

evaluate_modelnet.py
- def benchmark_evaluation(desc)

evaluate_registration_c2f_rotated.py
- def extract_correspondence(…)
- def benchmark_registration(…)

evaluate_registration_c2f.py
- def extract_correspondence(…)
- def benchmark_registration(…)

evaluate_registration_rotated.py
- def benchmark_registration(…)

evaluate_registration.py
- def benchmark_registration(…)