limhpone / computervision-final-prep

Code Issues Pull requests Actions Projects Wiki Security In

computervision-final-prep / lab / Lab 12 (3D Vision)-20251128 / open3d_example.py

limhpone 3D Vision- lab 12 daadd1a · 2 hours ago

351 lines (289 loc) · 10.5 KB

Code Blame Raw

```python
# ----------------------------------------------------------------------------
# -                        Open3D: www.open3d.org                            -
# ----------------------------------------------------------------------------
# Copyright (c) 2018-2024 www.open3d.org
# SPDX-License-Identifier: MIT
# ----------------------------------------------------------------------------

import open3d as o3d
import numpy as np
import os
import shutil
import sys
import zipfile
from os import listdir, makedirs
from os.path import exists, isfile, join, splitext, dirname, basename
import re
from warnings import warn
import json
import open3d as o3d
import copy

if (sys.version_info > (3, 0)):
    pyver = 3
    from urllib.request import Request, urlopen
else:
    pyver = 2
    from urllib2 import Request, urlopen


def edges_to_lineset(mesh, edges, color):
    ls = o3d.geometry.LineSet()
    ls.points = mesh.vertices
    ls.lines = edges
    ls.paint_uniform_color(color)
    return ls


def get_plane_mesh(height=0.2, width=1):
    mesh = o3d.geometry.TriangleMesh(
        vertices=o3d.utility.Vector3dVector(
            np.array(
                [[0, 0, 0], [0, height, 0], [width, height, 0], [width, 0, 0]],
                dtype=np.float32,
            )),
        triangles=o3d.utility.Vector3iVector(np.array([[0, 2, 1], [2, 0, 3]])),
    )
    mesh.compute_vertex_normals()
    return mesh


def get_non_manifold_edge_mesh():
    verts = np.array(
        [[-1, 0, 0], [0, 1, 0], [1, 0, 0], [0, -1, 0], [0, 0, 1]],
        dtype=np.float64,
    )
    triangles = np.array([[0, 1, 3], [1, 2, 3], [1, 3, 4]])
    mesh = o3d.geometry.TriangleMesh()
    mesh.vertices = o3d.utility.Vector3dVector(verts)
    mesh.triangles = o3d.utility.Vector3iVector(triangles)
    mesh.compute_vertex_normals()
    mesh.rotate(
        mesh.get_rotation_matrix_from_xyz((np.pi / 4, 0, np.pi / 4)),
        center=mesh.get_center(),
    )
    return mesh


def get_non_manifold_vertex_mesh():
    verts = np.array(
        [
            [-1, 0, -1],
            [1, 0, -1],
            [0, 1, -1],
            [0, 0, 0],
            [-1, 0, 1],
            [1, 0, 1],
            [0, 1, 1],
        ],
        dtype=np.float64,
    )
    triangles = np.array([
        [0, 1, 2],
```

```python
            [0, 1, 3],
            [1, 2, 3],
            [2, 0, 3],
            [4, 5, 6],
            [4, 5, 3],
            [5, 6, 3],
            [4, 6, 3],
        ])
        mesh = o3d.geometry.TriangleMesh()
        mesh.vertices = o3d.utility.Vector3dVector(verts)
        mesh.triangles = o3d.utility.Vector3iVector(triangles)
        mesh.compute_vertex_normals()
        mesh.rotate(
            mesh.get_rotation_matrix_from_xyz((np.pi / 4, 0, np.pi / 4)),
            center=mesh.get_center(),
        )
        return mesh


    def get_open_box_mesh():
        mesh = o3d.geometry.TriangleMesh.create_box()
        mesh.triangles = o3d.utility.Vector3iVector(np.asarray(mesh.triangles)[:-2])
        mesh.compute_vertex_normals()
        mesh.rotate(
            mesh.get_rotation_matrix_from_xyz((0.8 * np.pi, 0, 0.66 * np.pi)),
            center=mesh.get_center(),
        )
        return mesh


    def get_intersecting_boxes_mesh():
        mesh0 = o3d.geometry.TriangleMesh.create_box()
        T = np.eye(4)
        T[:, 3] += (0.5, 0.5, 0.5, 0)
        mesh1 = o3d.geometry.TriangleMesh.create_box()
        mesh1.transform(T)
        mesh = mesh0 + mesh1
        mesh.compute_vertex_normals()
        mesh.rotate(
            mesh.get_rotation_matrix_from_xyz((0.7 * np.pi, 0, 0.6 * np.pi)),
            center=mesh.get_center(),
        )
        return mesh


    def file_downloader(url, out_dir="."):
        file_name = url.split('/')[-1]
        u = urlopen(url)
        f = open(os.path.join(out_dir, file_name), "wb")
        if pyver == 2:
            meta = u.info()
            file_size = int(meta.getheaders("Content-Length")[0])
        elif pyver == 3:
            file_size = int(u.getheader("Content-Length"))
        print("Downloading: %s " % file_name)

        file_size_dl = 0
        block_sz = 8192
        progress = 0
        while True:
            buffer = u.read(block_sz)
            if not buffer:
                break
            file_size_dl += len(buffer)
            f.write(buffer)
            if progress + 10 <= (file_size_dl * 100. / file_size):
                progress = progress + 10
                print(" %.1f / %.1f MB (%.0f %%)" % \
                        (file_size_dl/(1024*1024), file_size/(1024*1024), progress))
        f.close()


    def unzip_data(path_zip, path_extract_to):
        print("Unzipping %s" % path_zip)
        zip_ref = zipfile.ZipFile(path_zip, 'r')
        zip_ref.extractall(path_extract_to)
        zip_ref.close()
        print("Extracted to %s" % path_extract_to)


    def sorted_alphanum(file_list_ordered):
        convert = lambda text: int(text) if text.isdigit() else text
        alphanum_key = lambda key: [convert(c) for c in re.split('([0-9]+)', key)]
        return sorted(file_list_ordered, key=alphanum_key)


    def get_file_list(path, extension=None):
        if extension is None:
            file_list = [path + f for f in listdir(path) if isfile(join(path, f))]
        else:
            file_list = [
                path + f
                for f in listdir(path)
                if isfile(join(path, f)) and splitext(f)[1] == extension
            ]
        file_list = sorted_alphanum(file_list)
        return file_list

```

```python

def add_if_exists(path_dataset, folder_names):
    for folder_name in folder_names:
        if exists(join(path_dataset, folder_name)):
            path = join(path_dataset, folder_name)
            return path
    raise FileNotFoundError(
        f"None of the folders {folder_names} found in {path_dataset}")


def read_rgbd_image(color_file, depth_file, convert_rgb_to_intensity, config):
    color = o3d.io.read_image(color_file)
    depth = o3d.io.read_image(depth_file)
    rgbd_image = o3d.geometry.RGBDImage.create_from_color_and_depth(
        color,
        depth,
        depth_scale=config["depth_scale"],
        depth_trunc=config["depth_max"],
        convert_rgb_to_intensity=convert_rgb_to_intensity)
    return rgbd_image


def get_rgbd_folders(path_dataset):
    path_color = add_if_exists(path_dataset, ["image/", "rgb/", "color/"])
    path_depth = join(path_dataset, "depth/")
    return path_color, path_depth


def get_rgbd_file_lists(path_dataset):
    path_color, path_depth = get_rgbd_folders(path_dataset)
    color_files = get_file_list(path_color, ".jpg") + \
            get_file_list(path_color, ".png")
    depth_files = get_file_list(path_depth, ".png")
    return color_files, depth_files


def make_clean_folder(path_folder):
    if not exists(path_folder):
        makedirs(path_folder)
    else:
        shutil.rmtree(path_folder)
        makedirs(path_folder)


def check_folder_structure(path_dataset):
    if isfile(path_dataset) and path_dataset.endswith(".bag"):
        return
    path_color, path_depth = get_rgbd_folders(path_dataset)
    assert exists(path_depth), \
            "Path %s is not exist!" % path_depth
    assert exists(path_color), \
            "Path %s is not exist!" % path_color


def write_poses_to_log(filename, poses):
    with open(filename, 'w') as f:
        for i, pose in enumerate(poses):
            f.write('{} {} {}\n'.format(i, i, i + 1))
            f.write('{0:.8f} {1:.8f} {2:.8f} {3:.8f}\n'.format(
                pose[0, 0], pose[0, 1], pose[0, 2], pose[0, 3]))
            f.write('{0:.8f} {1:.8f} {2:.8f} {3:.8f}\n'.format(
                pose[1, 0], pose[1, 1], pose[1, 2], pose[1, 3]))
            f.write('{0:.8f} {1:.8f} {2:.8f} {3:.8f}\n'.format(
                pose[2, 0], pose[2, 1], pose[2, 2], pose[2, 3]))
            f.write('{0:.8f} {1:.8f} {2:.8f} {3:.8f}\n'.format(
                pose[3, 0], pose[3, 1], pose[3, 2], pose[3, 3]))


def read_poses_from_log(traj_log):
    import numpy as np

    trans_arr = []
    with open(traj_log) as f:
        content = f.readlines()

        # Load .log file.
        for i in range(0, len(content), 5):
            # format %d (src) %d (tgt) %f (fitness)
            data = list(map(float, content[i].strip().split(' ')))
            ids = (int(data[0]), int(data[1]))
            fitness = data[2]

            # format %f x 16
            T_gt = np.array(
                list(map(float, (''.join(
                    content[i + 1:i + 5])).strip().split()))).reshape((4, 4))

            trans_arr.append(T_gt)

    return trans_arr


flip_transform = [[1, 0, 0, 0], [0, -1, 0, 0], [0, 0, -1, 0], [0, 0, 0, 1]]


def draw_geometries_flip(pcds):
    pcds_transform = []
    for pcd in pcds:
```

```python
            pcd_temp = copy.deepcopy(pcd)
            pcd_temp.transform(flip_transform)
            pcds_transform.append(pcd_temp)
        o3d.visualization.draw_geometries(pcds_transform)


def draw_registration_result(source, target, transformation):
    source_temp = copy.deepcopy(source)
    target_temp = copy.deepcopy(target)
    source_temp.paint_uniform_color([1, 0.706, 0])
    target_temp.paint_uniform_color([0, 0.651, 0.929])
    source_temp.transform(transformation)
    source_temp.transform(flip_transform)
    target_temp.transform(flip_transform)
    o3d.visualization.draw_geometries([source_temp, target_temp])


def draw_registration_result_original_color(source, target, transformation):
    source_temp = copy.deepcopy(source)
    target_temp = copy.deepcopy(target)
    source_temp.transform(transformation)
    source_temp.transform(flip_transform)
    target_temp.transform(flip_transform)
    o3d.visualization.draw_geometries([source_temp, target_temp])


class CameraPose:

    def __init__(self, meta, mat):
        self.metadata = meta
        self.pose = mat

    def __str__(self):
        return 'Metadata : ' + ' '.join(map(str, self.metadata)) + '\n' + \
            "Pose : " + "\n" + np.array_str(self.pose)


def read_trajectory(filename):
    traj = []
    with open(filename, 'r') as f:
        metastr = f.readline()
        while metastr:
            metadata = list(map(int, metastr.split()))
            mat = np.zeros(shape=(4, 4))
            for i in range(4):
                matstr = f.readline()
                mat[i, :] = np.fromstring(matstr, dtype=float, sep=' \t')
            traj.append(CameraPose(metadata, mat))
            metastr = f.readline()
    return traj


def write_trajectory(traj, filename):
    with open(filename, 'w') as f:
        for x in traj:
            p = x.pose.tolist()
            f.write(' '.join(map(str, x.metadata)) + '\n')
            f.write('\n'.join(
                ' '.join(map('{0:.12f}'.format, p[i])) for i in range(4)))
            f.write('\n')


def initialize_opencv():
    opencv_installed = True
    try:
        import cv2
    except ImportError:
        pass
        print("OpenCV is not detected. Using Identity as an initial")
        opencv_installed = False
    if opencv_installed:
        print("OpenCV is detected. Using ORB + 5pt algorithm")
    return opencv_installed
```