



limhpone / computervision-final-prep



&lt;&gt; 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



```
1  # -----
2  # -                               Open3D: www.open3d.org           -
3  # -----
4  # Copyright (c) 2018-2024 www.open3d.org
5  # SPDX-License-Identifier: MIT
6  # -----
7
8  import open3d as o3d
9  import numpy as np
10 import os
11 import shutil
12 import sys
13 import zipfile
14 from os import listdir, makedirs
15 from os.path import exists, isfile, join, splitext, dirname, basename
16 import re
17 from warnings import warn
18 import json
19 import open3d as o3d
20 import copy
21
22 if (sys.version_info > (3, 0)):
23     pyver = 3
24     from urllib.request import Request, urlopen
25 else:
26     pyver = 2
27     from urllib2 import Request, urlopen
28
29
30 def edges_to_lineset(mesh, edges, color):
31     ls = o3d.geometry.LineSet()
32     ls.points = mesh.vertices
33     ls.lines = edges
```

```
34     ls.paint_uniform_color(color)
35     return ls
36
37
38     def get_plane_mesh(height=0.2, width=1):
39         mesh = o3d.geometry.TriangleMesh(
40             vertices=o3d.utility.Vector3dVector(
41                 np.array(
42                     [[0, 0, 0], [0, height, 0], [width, height, 0], [width, 0, 0]],
43                     dtype=np.float32,
44                 )),
45             triangles=o3d.utility.Vector3iVector(np.array([[0, 2, 1], [2, 0, 3]])),
46         )
47         mesh.compute_vertex_normals()
48         return mesh
49
50
51     def get_non_manifold_edge_mesh():
52         verts = np.array(
53             [[-1, 0, 0], [0, 1, 0], [1, 0, 0], [0, -1, 0], [0, 0, 1]],
54             dtype=np.float64,
55         )
56         triangles = np.array([[0, 1, 3], [1, 2, 3], [1, 3, 4]])
57         mesh = o3d.geometry.TriangleMesh()
58         mesh.vertices = o3d.utility.Vector3dVector(verts)
59         mesh.triangles = o3d.utility.Vector3iVector(triangles)
60         mesh.compute_vertex_normals()
61         mesh.rotate(
62             mesh.get_rotation_matrix_from_xyz((np.pi / 4, 0, np.pi / 4)),
63             center=mesh.get_center(),
64         )
65         return mesh
66
67
68     def get_non_manifold_vertex_mesh():
69         verts = np.array(
70             [
71                 [-1, 0, -1],
72                 [1, 0, -1],
73                 [0, 1, -1],
74                 [0, 0, 0],
75                 [-1, 0, 1],
76                 [1, 0, 1],
77                 [0, 1, 1],
78             ],
79             dtype=np.float64,
80         )
81         triangles = np.array([
82             [0, 1, 2],
```

```
83         [0, 1, 3],
84         [1, 2, 3],
85         [2, 0, 3],
86         [4, 5, 6],
87         [4, 5, 3],
88         [5, 6, 3],
89         [4, 6, 3],
90     ])
91     mesh = o3d.geometry.TriangleMesh()
92     mesh.vertices = o3d.utility.Vector3dVector(verts)
93     mesh.triangles = o3d.utility.Vector3iVector(triangles)
94     mesh.compute_vertex_normals()
95     mesh.rotate(
96         mesh.get_rotation_matrix_from_xyz((np.pi / 4, 0, np.pi / 4)),
97         center=mesh.get_center(),
98     )
99     return mesh
100
101
102  ✓ def get_open_box_mesh():
103     mesh = o3d.geometry.TriangleMesh.create_box()
104     mesh.triangles = o3d.utility.Vector3iVector(np.asarray(mesh.triangles)[: -2])
105     mesh.compute_vertex_normals()
106     mesh.rotate(
107         mesh.get_rotation_matrix_from_xyz((0.8 * np.pi, 0, 0.66 * np.pi)),
108         center=mesh.get_center(),
109     )
110     return mesh
111
112
113  ✓ def get_intersecting_boxes_mesh():
114     mesh0 = o3d.geometry.TriangleMesh.create_box()
115     T = np.eye(4)
116     T[:, 3] += (0.5, 0.5, 0.5, 0)
117     mesh1 = o3d.geometry.TriangleMesh.create_box()
118     mesh1.transform(T)
119     mesh = mesh0 + mesh1
120     mesh.compute_vertex_normals()
121     mesh.rotate(
122         mesh.get_rotation_matrix_from_xyz((0.7 * np.pi, 0, 0.6 * np.pi)),
123         center=mesh.get_center(),
124     )
125     return mesh
126
127
128  ✓ def file_downloader(url, out_dir="."):
129     file_name = url.split('/')[-1]
130     u = urlopen(url)
131     f = open(os.path.join(out_dir, file_name), "wb")
```

```
132     if pyver == 2:
133         meta = u.info()
134         file_size = int(meta.getheaders("Content-Length")[0])
135     elif pyver == 3:
136         file_size = int(u.getheader("Content-Length"))
137     print("Downloading: %s " % file_name)
138
139     file_size_dl = 0
140     block_sz = 8192
141     progress = 0
142     while True:
143         buffer = u.read(block_sz)
144         if not buffer:
145             break
146         file_size_dl += len(buffer)
147         f.write(buffer)
148         if progress + 10 <= (file_size_dl * 100. / file_size):
149             progress = progress + 10
150             print(" %.1f / %.1f MB (%.0f %%) " % \
151                   (file_size_dl/(1024*1024), file_size/(1024*1024), progress))
152     f.close()
153
154
155     def unzip_data(path_zip, path_extract_to):
156         print("Unzipping %s" % path_zip)
157         zip_ref = zipfile.ZipFile(path_zip, 'r')
158         zip_ref.extractall(path_extract_to)
159         zip_ref.close()
160         print("Extracted to %s" % path_extract_to)
161
162
163     def sorted_alphanum(file_list_ordered):
164         convert = lambda text: int(text) if text.isdigit() else text
165         alphanum_key = lambda key: [convert(c) for c in re.split('([0-9]+)', key)]
166         return sorted(file_list_ordered, key=alphanum_key)
167
168
169     def get_file_list(path, extension=None):
170         if extension is None:
171             file_list = [path + f for f in listdir(path) if isfile(join(path, f))]
172         else:
173             file_list = [
174                 path + f
175                 for f in listdir(path)
176                 if isfile(join(path, f)) and splitext(f)[1] == extension
177             ]
178             file_list = sorted_alphanum(file_list)
179         return file_list
180
```

```
181
182 ✓ def add_if_exists(path_dataset, folder_names):
183     for folder_name in folder_names:
184         if exists(join(path_dataset, folder_name)):
185             path = join(path_dataset, folder_name)
186             return path
187     raise FileNotFoundError(
188         f"None of the folders {folder_names} found in {path_dataset}")
189
190
191 ✓ def read_rgbd_image(color_file, depth_file, convert_rgb_to_intensity, config):
192     color = o3d.io.read_image(color_file)
193     depth = o3d.io.read_image(depth_file)
194     rgbd_image = o3d.geometry.RGBDImage.create_from_color_and_depth(
195         color,
196         depth,
197         depth_scale=config["depth_scale"],
198         depth_trunc=config["depth_max"],
199         convert_rgb_to_intensity=convert_rgb_to_intensity)
200     return rgbd_image
201
202
203 def get_rgbd_folders(path_dataset):
204     path_color = add_if_exists(path_dataset, ["image/", "rgb/", "color/"])
205     path_depth = join(path_dataset, "depth/")
206     return path_color, path_depth
207
208
209 ✓ def get_rgbd_file_lists(path_dataset):
210     path_color, path_depth = get_rgbd_folders(path_dataset)
211     color_files = get_file_list(path_color, ".jpg") + \
212         get_file_list(path_color, ".png")
213     depth_files = get_file_list(path_depth, ".png")
214     return color_files, depth_files
215
216
217 ✓ def make_clean_folder(path_folder):
218     if not exists(path_folder):
219         makedirs(path_folder)
220     else:
221         shutil.rmtree(path_folder)
222         makedirs(path_folder)
223
224
225 ✓ def check_folder_structure(path_dataset):
226     if isfile(path_dataset) and path_dataset.endswith(".bag"):
227         return
228     path_color, path_depth = get_rgbd_folders(path_dataset)
229     assert exists(path_depth), \
```

```
230         "Path %s is not exist!" % path_depth
231     assert exists(path_color), \
232         "Path %s is not exist!" % path_color
233
234
235     def write_poses_to_log(filename, poses):
236         with open(filename, 'w') as f:
237             for i, pose in enumerate(poses):
238                 f.write('{ } {} {} \n'.format(i, i, i + 1))
239                 f.write('{0:.8f} {1:.8f} {2:.8f} {3:.8f} \n'.format(
240                     pose[0, 0], pose[0, 1], pose[0, 2], pose[0, 3]))
241                 f.write('{0:.8f} {1:.8f} {2:.8f} {3:.8f} \n'.format(
242                     pose[1, 0], pose[1, 1], pose[1, 2], pose[1, 3]))
243                 f.write('{0:.8f} {1:.8f} {2:.8f} {3:.8f} \n'.format(
244                     pose[2, 0], pose[2, 1], pose[2, 2], pose[2, 3]))
245                 f.write('{0:.8f} {1:.8f} {2:.8f} {3:.8f} \n'.format(
246                     pose[3, 0], pose[3, 1], pose[3, 2], pose[3, 3]))
247
248
249     def read_poses_from_log(traj_log):
250         import numpy as np
251
252         trans_arr = []
253         with open(traj_log) as f:
254             content = f.readlines()
255
256             # Load .log file.
257             for i in range(0, len(content), 5):
258                 # format %d (src) %d (tgt) %f (fitness)
259                 data = list(map(float, content[i].strip().split(' ')))
260                 ids = (int(data[0]), int(data[1]))
261                 fitness = data[2]
262
263                 # format %f x 16
264                 T_gt = np.array(
265                     list(map(float, (''.join(
266                         content[i + 1:i + 5]).strip().split()))).reshape((4, 4))
267
268                 trans_arr.append(T_gt)
269
270         return trans_arr
271
272
273     flip_transform = [[1, 0, 0, 0], [0, -1, 0, 0], [0, 0, -1, 0], [0, 0, 0, 1]]
274
275
276     def draw_geometries_flip(pcds):
277         pcds_transform = []
278         for pcd in pcds:
```

```
279         pcd_temp = copy.deepcopy(pcd)
280         pcd_temp.transform(flip_transform)
281         pcds_transform.append(pcd_temp)
282     o3d.visualization.draw_geometries(pcds_transform)
283
284
285     def draw_registration_result(source, target, transformation):
286         source_temp = copy.deepcopy(source)
287         target_temp = copy.deepcopy(target)
288         source_temp.paint_uniform_color([1, 0.706, 0])
289         target_temp.paint_uniform_color([0, 0.651, 0.929])
290         source_temp.transform(transformation)
291         source_temp.transform(flip_transform)
292         target_temp.transform(flip_transform)
293         o3d.visualization.draw_geometries([source_temp, target_temp])
294
295
296     def draw_registration_result_original_color(source, target, transformation):
297         source_temp = copy.deepcopy(source)
298         target_temp = copy.deepcopy(target)
299         source_temp.transform(transformation)
300         source_temp.transform(flip_transform)
301         target_temp.transform(flip_transform)
302         o3d.visualization.draw_geometries([source_temp, target_temp])
303
304
305     class CameraPose:
306
307         def __init__(self, meta, mat):
308             self.metadata = meta
309             self.pose = mat
310
311         def __str__(self):
312             return 'Metadata : ' + ' '.join(map(str, self.metadata)) + '\n' + \
313                 "Pose : " + "\n" + np.array_str(self.pose)
314
315
316     def read_trajectory(filename):
317         traj = []
318         with open(filename, 'r') as f:
319             metastr = f.readline()
320             while metastr:
321                 metadata = list(map(int, metastr.split()))
322                 mat = np.zeros(shape=(4, 4))
323                 for i in range(4):
324                     matstr = f.readline()
325                     mat[i, :] = np.fromstring(matstr, dtype=float, sep=' \t')
326                 traj.append(CameraPose(metadata, mat))
327                 metastr = f.readline()
```

```
328     return traj
329
330
331  ✓ def write_trajectory(traj, filename):
332     with open(filename, 'w') as f:
333         for x in traj:
334             p = x.pose.tolist()
335             f.write(' '.join(map(str, x.metadata)) + '\n')
336             f.write('\n'.join(
337                 ' '.join(map('{0:.12f}'.format, p[i])) for i in range(4)))
338             f.write('\n')
339
340
341  ✓ def initialize_opencv():
342     opencv_installed = True
343     try:
344         import cv2
345     except ImportError:
346         pass
347     print("OpenCV is not detected. Using Identity as an initial")
348     opencv_installed = False
349     if opencv_installed:
350         print("OpenCV is detected. Using ORB + 5pt algorithm")
351     return opencv_installed
```