depth_map_processing

May 2, 2025

1 BoXYZ - Depth Map Processing

Process the depth map to create a 3D point cloud and visualize it using Open3D

[1]:
```
!pip install --quiet numpy matplotlib open3d

import os

import numpy as np
import open3d as o3d
import matplotlib.pyplot as plt
from IPython.display import display
```

```
[notice] A new release of pip is
available: 24.3.1 -> 25.1
[notice] To update, run:
pip install --upgrade pip
```

```
Jupyter environment detected. Enabling Open3D WebVisualizer.
[Open3D INFO] WebRTC GUI backend enabled.
[Open3D INFO] WebRTCWindowSystem: HTTP handshake server disabled.
```

[2]:
```
# get pcb from depth map
intrinsics_filename = os.path.join('../..', 'assets', 'intrinsics.npy')
K = np.load(intrinsics_filename)

depth_filename = os.path.join('../..', 'assets', 'one-box.depth.npdata.npy')
xyz = np.load(depth_filename)
print('Depth map shape', xyz.shape)

h, w = xyz.shape
v, u = np.mgrid[0:h, 0:w]

u = u.flatten()
v = v.flatten()
z = xyz.flatten()
valid = z > 0
u = u[valid]
```

```
v = v[valid]
z = z[valid]

fx, fy = K[0, 0], K[1, 1]
cx, cy = K[0, 2], K[1, 2]
x = (u - cx) * z / fx
y = (v - cy) * z / fy

xyz = np.column_stack([x, y, z])

pcb = o3d.geometry.PointCloud()
pcb.points = o3d.utility.Vector3dVector(xyz)
```

```
Depth map shape (1544, 2064)
```

[3]:
```
# add colors to pcb
color_filename = os.path.join('../..', 'assets', 'one-box.color.npdata.npy')
pcb_col = np.load(color_filename)
print('Color data shape', pcb_col.shape)

colors = pcb_col.reshape(-1, 3)
colors = colors / 255.0
pcb.colors = o3d.utility.Vector3dVector(colors)
```

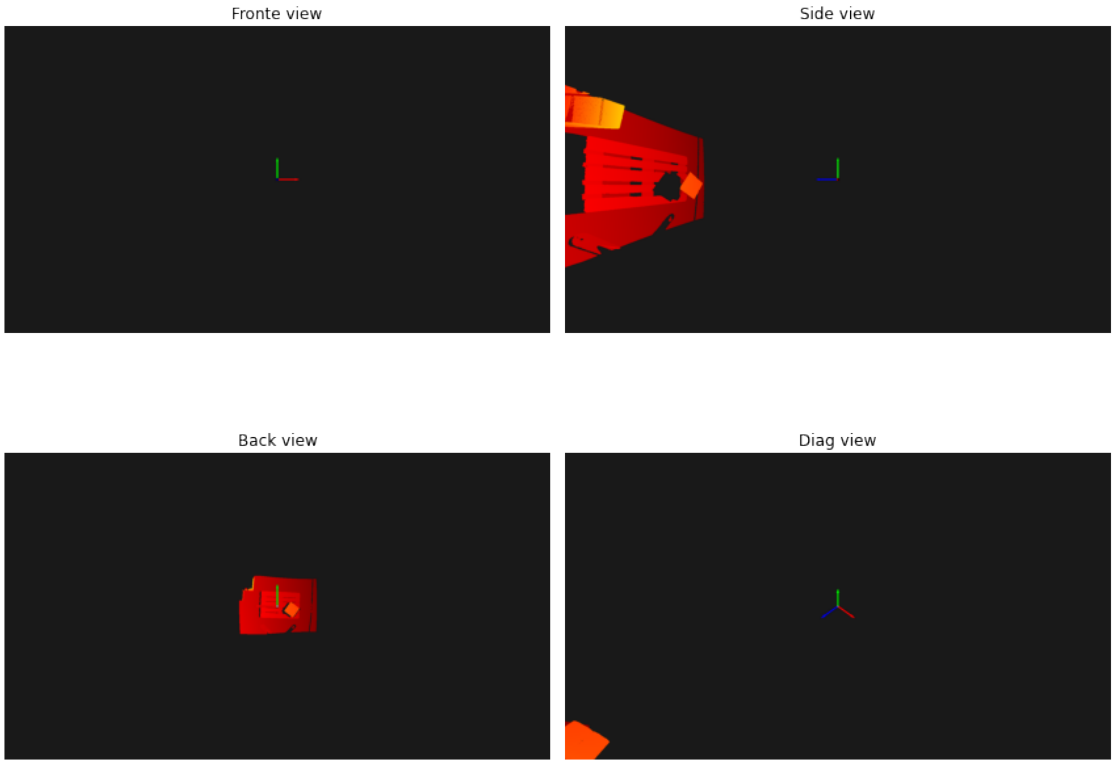
```
Color data shape (1544, 2064)
```

[4]:
```
# is this necessary? leave it for now
# pcb.estimate_normals(search_param=o3d.geometry.
 ↪KDTreeSearchParamHybrid(radius=0.1, max_nn=30))
# pcb.orient_normals_towards_camera_location(camera_location=np.array([0., 0.,␣
 ↪0.]))
```

[5]:
```
# visualize the pcb
def visualize_pcb_with_snapshots(pcb):
    o3d.io.write_point_cloud(os.path.join('../..', 'assets', 'one-box.pcb.
 ↪ply'), pcb)

    vis = o3d.visualization.Visualizer()
    vis.create_window(window_name='Point Cloud Visualization', visible=False)
    vis.add_geometry(pcb)

    # add camera coordinate frame
    camera_frame = o3d.geometry.TriangleMesh.create_coordinate_frame(size=0.3,␣
 ↪origin=[0, 0, 0])
    vis.add_geometry(camera_frame)

    opt = vis.get_render_option()
    opt.background_color = np.array([0.1, 0.1, 0.1])
```

```
    opt.point_size = 2.0

    # setup camera views for snapshots
    snapshots = []
    view_names = ["Fronte view", "Side view", "Back view", "Diag view"]
    view_angles = [
        (np.pi, np.pi, np.pi),
        (0, np.pi/2, 0),
        (0, np.pi, 0),
        (np.pi/4, np.pi/4, 0)
    ]

    # capture snapshots from different angles
    for i, (pitch, yaw, roll) in enumerate(view_angles):
        ctr = vis.get_view_control()
        ctr.change_field_of_view(step=60)
        ctr.set_front([np.sin(yaw) * np.cos(pitch), np.sin(pitch), np.cos(yaw)␣
↪* np.cos(pitch)])
        ctr.set_lookat([0, 0, 0])
        ctr.set_up([0, 1, 0])
        vis.poll_events()
        vis.update_renderer()
        img = vis.capture_screen_float_buffer(do_render=True)
        snapshots.append(np.asarray(img))

    vis.destroy_window()

    vis = o3d.visualization.Visualizer()
    vis.create_window(window_name='Point Cloud Visualization')
    vis.add_geometry(pcb)
    vis.add_geometry(camera_frame)
    opt = vis.get_render_option()
    opt.background_color = np.array([0.1, 0.1, 0.1])
    opt.point_size = 2.0
    vis.run()
    vis.destroy_window()

    # display snapshots
    fig, axes = plt.subplots(2, 2, figsize=(12, 10))
    axes = axes.flatten()

    for i, (img, view_name) in enumerate(zip(snapshots, view_names)):
        axes[i].imshow(img)
        axes[i].set_title(f'{view_name}')
        axes[i].axis('off')

    plt.tight_layout()
```

```
    plt.show()

visualize_pcb_with_snapshots(pcb)
```



[ ]: