Getting started with Open3d and PyMeshLab

We recommend using Open3d for visualization and feature computation and pymeshlab for mesh preprocessing. In this tutorial, we will explain how to get started using these packages.

Installation on Windows/Mac/Linux

Start by installing Python on your system if you have not done so already. Open3d does not support python version 3.13 at the moment, so use version 3.12 or older.

Download and install the open3d and pymeshlab python packages by opening a command prompt/shell/powershell window and running:

```
pip install open3d
pip install pymeshlab
```

(if you plan to work within a virtual environment, be sure to create- and activate it before installing open3d and pymeshlab).

Verification

To verify installation success for open3d, run

```
python -c "import open3d as o3d; print(o3d.__version__)"
```

You should see the installed version number in the console (like `0.19.0` for example).
To our knowledge, there is no equivalent command in pymeshlab. However, if you are able to run the code sample in the next section successfully, installation must have been successful.

'Hello World' code sample

We have provided a very basic code sample with which you can load an example object and view it in open3d's UI. The code sample is called 'helloworld.py' (stored in the same folder as this PDF file) and can be run using a python interpreter that has open3d and pymeshlab installed. The window displayed on the right should pop up.

Implementation explanation

In this basic 'helloworld' script, we first load a mesh using a given mesh path. This yields an Open3d MeshSet object which we can use to visualize the mesh, query basic features such as vertex positions, and manipulate the mesh (e.g. through translations or rotations).

After loading the mesh, we compute its vertex normals. This is necessary for rendering using smooth shading (see exercise 1 in the exercises section).

Next, the mesh is visualized by calling 'draw_geometries' and passing in a list of geometry objects (in this case only the single mesh we created). The default visualization option is smoothshade, but you may change the vis_option variable to view alternative ways of rendering the mesh as well as accessories such as lines that visualize the world axis system (see exercise 2).

If you are able to run this sample, you are ready to proceed to the next step (although we recommend taking a look at the exercises below to familiarize yourself with the libraries and the shape database).

Troubleshooting

If you encounter any problems with installation and/or running the code sample, please check open3d's *getting started* page, try a google search for the specific error you're getting, and/or ask other groups if they've had the same problem and if so how they fixed it.

Exercises

Exercise 1

Try commenting out the line
`mesh.compute_vertex_normals()`
in the helloworld.py script.
What do you expect will happen if you run the code sample?

→Now run the sample. What do you see? Is it what you expected? If not, why do you think we get this result?

Exercise 2

Try the different visualization techniques available in the script, by changing the value of the `vis_option` variable in the line `vis_option = "smoothshade"` from *"smoothshade"* to one of the other options ("*wireframe_on_shaded*", "*wireframe*", "*world_axes*").

→Run the script. What do you see? For what purposes do you think each of these rendering techniques might be useful?

Exercise 3

The lines

```
vertices = np.asarray(mesh.vertices)
triangles = np.asarray(mesh.triangles)
```

show you how to obtain a numpy array containing the vertices, respectively triangles of the mesh. Use these arrays to find out how many vertices and triangles are in the mesh by adding additional code.

→Make a scatter plot to show the relationship between the number of vertices/triangles and the object's file size in KB. What do you notice?