

# B3M33HRO HW4

## Grasping

### 1 Introduction

You are provided with point clouds from noisy real depth cameras. The individual point clouds are different views on an object on a table. Your task is to combine them to get a full view of a scene and use it to get grasp from two pipelines: **GraspIt!** and **GPD**. Examples of grasps can be seen in Figure 1.

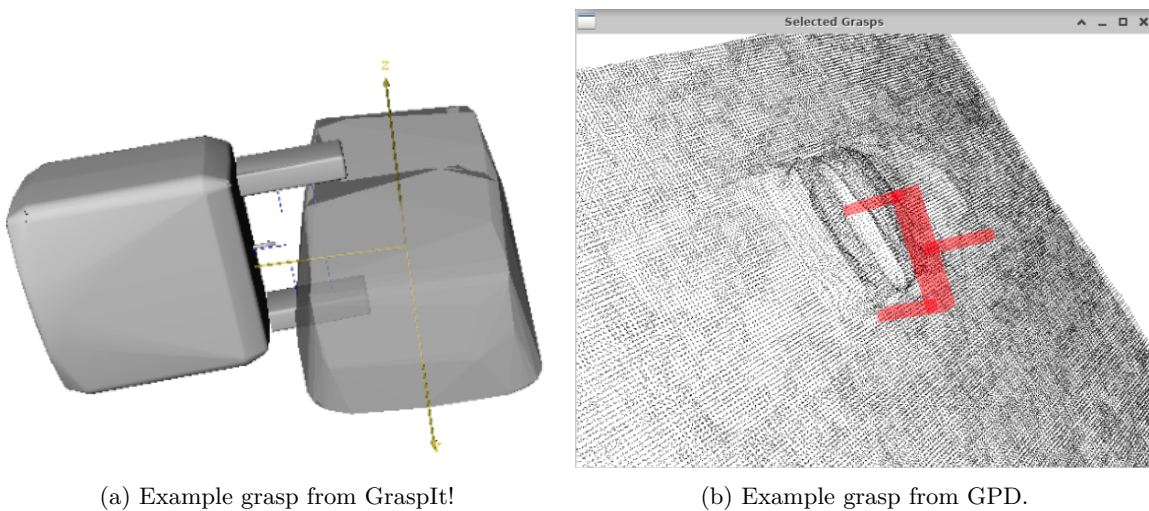


Figure 1: Examples of grasps.

### 2 Assignment

- Download assignment from the course website.
- Open Jupyter notebook and the code template provided.
  - In the docker image, run `jupyter-notebook` in the terminal and open <http://localhost:8888> in browser (in Docker or in your machine).
- Complete the code. Please, **pay attention to code quality and performance**.
- Set limits in *z-axis* for the bounding box of workspace.
  - Try something and update it based on visualizations.
  - Different minimal values may be needed for GraspIt! and for GPD.
- Combine the provided point clouds in one. You can process them as you want and as needed—downsample, outliers removal, bounding box crop.
  - The processed point cloud should have the right number of points and should not contain unnecessary holes
  - Decide whether to use the processing on the final point cloud, or on individual samples.
  - See [Open3D Point Cloud Class](#) and [Open3D Point Cloud Tutorial](#).
- Prepare point cloud for GraspIt!. It must contain only the object, without the table.

- 
- Create a mesh from the point cloud, translate it to position (0,0,0), and save it to file.
    - Select the appropriate method that will work in GraspIt!
    - See [Surface Reconstruction Tutorial](#) and [Open3D Triangle Mesh Class](#).
    - **Note:** The item in the point cloud is an opened box, *i.e.*, it is concave and has a hollow part. However, it is hard to obtain mesh for this point cloud, so it is fine if your mesh looks like a closed box—see the example in Fig. [1a](#).
  - Open the GraspIt interface and:
    - Clear the World;
    - Import *fetch\_gripper* as a robot;
    - Import your mesh as a graspable body;
    - See [GraspIt! commander API](#).
    - **Note:** if you see only black/grey after you load the robot and the body, zoom-out in the GraspIt GUI.
  - Run the Eigengrasp planner and sort the grasp by  $\epsilon$ -quality.
    - $\epsilon$ -quality: the closer to 1, the better. **Note:** if you close the GraspIt interface, you will probably need to restart the kernel in the notebook before you run it again.
  - Check if the grasp looks like you would assume and take a picture of it.
  - Prepare point cloud for GPD.
    - Modify the code you used for GraspIt! point cloud.
    - This point cloud **must have** “a table” under the object.
    - It is better to translate it into (0,0,0) otherwise you will have to zoom out in the GPD output.
  - Run the GPD and take a picture of the output.
    - Make it run as fast as possible.
      - \* The run-time can vary on different computers, but if the GPD runs for more than a 5seconds, it is too much even on a slow computer.
      - \* Right processing of the point cloud can help you to reduce time, or you can play with the values in *eigen\_params.cfg* (in Docker located in */gpd/cfg/eigen\_params.cfg*).

### 3 Points

- Correct GraspIt! output - 3 points
  - Screenshot of the grasp
  - Correct code
- Correct GPD output - 2 points
  - Screenshot of the grasp
  - Correct code
- Code quality and performance - 2 points