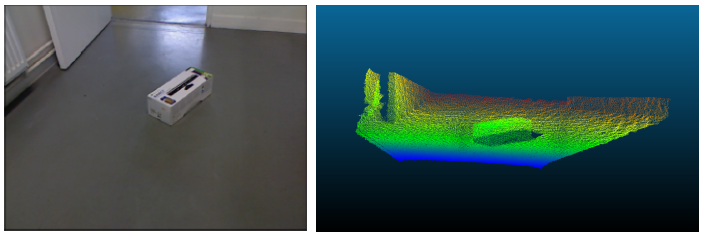
Practical session in 3D Vision

-

3D segmentation and object shape ckecking based on RGB-D sensor



(a) (b)

Figure 1: RGB-D sensor data: (a) RGB image, and (b) associated 3D point cloud of the scene.

1 Overview

This practice aims at performing 3D segmentation and shape checking on polyhedral objects, i.e. verify if such object is correct with respect to a reference geometrical model and/or presents defaults (holes, residues, etc.).

To achieve this, we need beforehand to build this reference model from a RGB-D image of a faultless object (figure 1). Then, for any view of an unknown/test object, we have to segment it from the background and compare to the reference model. This shape checking process must be independent of the view point and, therefore, requires the registration of each associated point cloud with respect to the reference one.

We propose to break down this task as follows:

1. **Extract 3D models** of both reference and test objects (to be controlled) by removing all the unwanted background points from the original scene point clouds. To avoid redundant processing, this step must only be done on the reference scene, `data01.xyz` ; it was already performed on objects to control/check and stored in `data02_object.xyz` and `data03_object.xyz`.
2. **Register** each test object to the reference model in order to compare them *i.e.* align their respective 3D point clouds into the reference coordinate system.
3. **Compare** control and reference models to conclude about potential defects.

By the end of the session, each student pair has to deliver a `.zip` archive containing:

1. A `.pdf` report on the work done during the session. Each numbered item of the following state of work should be explained in one single relevant paragraph and illustrated by a picture.
2. A `.py` file corresponding to the `qualitycheck.py` file completed by you.
3. A `.bin` file corresponding to your CloudCompare project containing the results of registration and shape checking of the provided data.

2 3D reference model processing

In order to process the quality control of the acquired data, we have to build a reference 3D model *i.e.* without any geometrical defaults. For this purpose, we have recorded the point cloud `data01.xyz` thanks to the RGB-D sensor.

Open `data01.xyz` with CloudCompare. On a terminal, launch: `$ cloudcompare.CloudCompare`.

1 How many points does a Kinect-modeled scene include?

Use CloudCompare in order to segment in 3D the polyhedral object from the rest of the scene:

2 Roughly segment the object and walls from the rest of the scene in order to get two point clouds: the rough object and the ground plane.

3 Align a plane on the previously segmented floor. From the rough object points, extract the upper part of the box thanks to the relative distance between them and the plane. Be sure to keep no more than 10000 points. Figure 2 shows the plane fitted into the ground, and the roughly segmented object point cloud with colored distances of each point, relatively to that plane.

4 Use `Point Picking` tool to infer the main dimensions of the object.

5 Export the object as `data01_segmented.xyz` file.

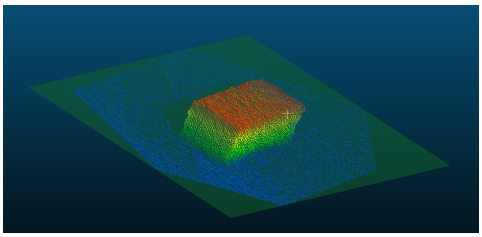


Figure 2: Distances between model point cloud and the ground plane.

3 Online 3D point cloud registration

The 3D reference model is now available. Similar 3D acquisitions of two unknown test scenes were recorded and segmented for further comparison with the reference, namely `data02_object.xyz` and `data03_object.xyz` files. The two associated test objects have to be compared to the reference one in order to check their conformity/shape. Beforehand, we propose to run the Iterative Closest Point (ICP) algorithm.

The `icp.py` file contains the algorithm implementation.

1 Describe the inputs and outputs of the algorithm and how it works.

The ICP algorithm has to be launched from the `qualitycheck.py` file.

2 Complete `qualitycheck.py` file to run ICP on both `data02_object.xyz` and `data03_object.xyz`.

3 Given the obtained results, apply the inferred rigid transformation on each test point cloud for overlapping the reference one.

4 Discuss and justify the impact of the algorithm parameters on the whole performances.

5 List the strengths and weaknesses of the ICP algorithm.

4 3D shape checking

Given the alignment between the test point cloud and the reference one, propose a method to identify faultless parts from defective ones.

1. Export the obtained data (object with transformation) to `.xyz` files.
2. Use CloudCompare to check whether each test point cloud dataset is defective or not.