151228619/151248619: OBJECT ORIENTED PROGRAMMING I
2022-2023 FALL SEMESTERS
PROJECTS REPORT

Name: Burak

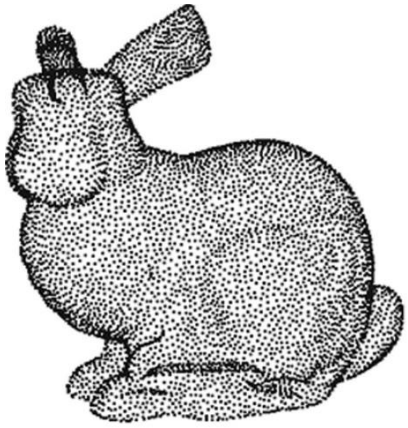
Surname: KAYKAÇ

Student No: 151220174025

In this project, a c++ code was created to process point cloud data with the pcl library.

A point cloud is a discrete set of data points in space. The points may represent a 3D shape or object. Each point position has its set of Cartesian coordinates (X, Y, Z). Point clouds are generally produced by 3D scanners or by photogrammetry software, which measure many points on the external surfaces of objects around them. As the output of 3D scanning processes, point clouds are used for many purposes, including to create 3D CAD models for manufactured parts, for metrology and quality inspection, and for a multitude of visualization, animation, rendering and mass customization applications.

(https://en.wikipedia.org/wiki/Point_cloud)

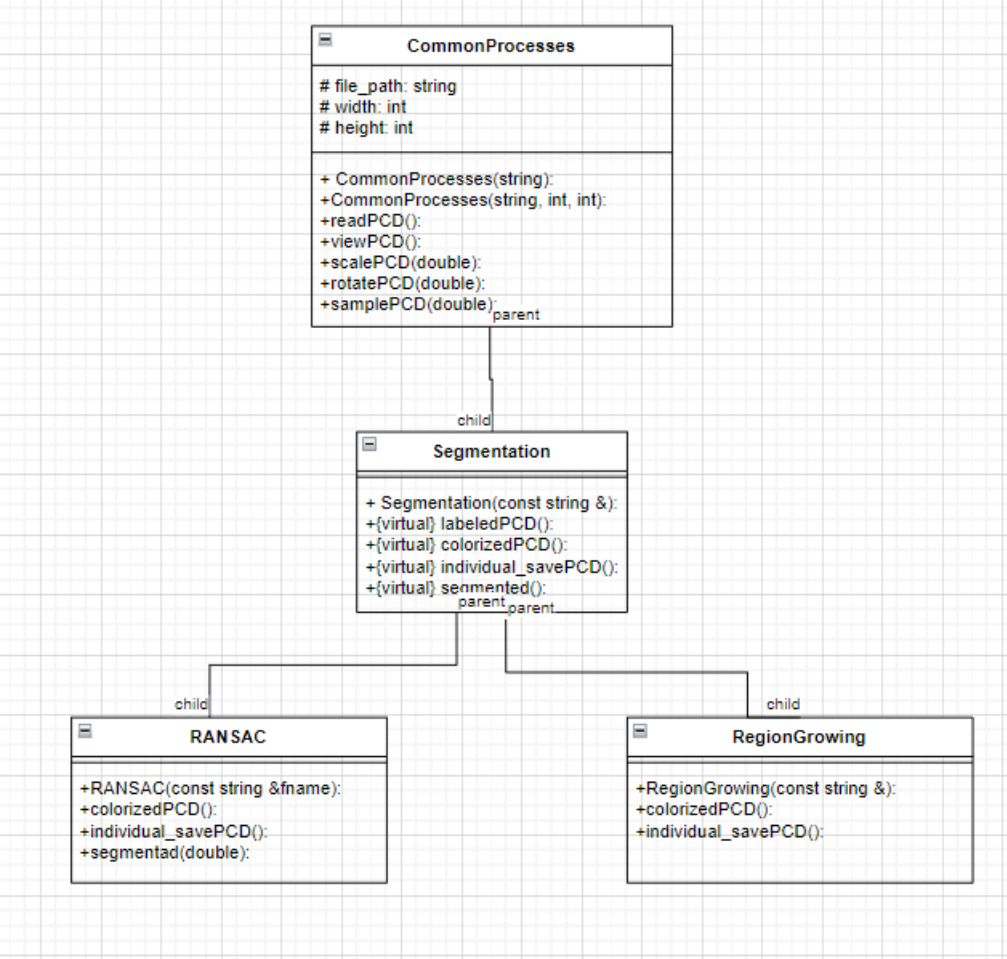


Point Cloud

Bunny 3D Model

WORKING PRINCIPLE

The uml code is as follows:



There are 4 classes. these are 'CommonProcesses', 'Segmentation', 'RegionGrowing' and 'RANSAC'. 'CommonProcesses' class is base class and 'Segmentation' class is inherited from it. 'RegionGrowing' and 'RANSAC' classes are inherited from 'Segmentation' class.

CommonProcesses Class

The 'CommonProcesses' class has 3 protected variables 'file_path', 'width' and 'height'. The 'file_path' variable holds the address of the pcd where the processes will be applied, or just the name if it is in the same folder. The 'width' and 'height' variables hold the values needed to create a random pcd. While the constructor created as CommonProcesses(string) gets file_path, the CommonProcesses(string, int, int) constructor created by overloading takes the width and height values required for the pcd to be created randomly and the

name of the pcd to be saved. The CommonProcesses(string) constructor generates a PointCloud object and says "File could not be read!" gives error. Creates the CommonProcesses(string, int, int) construcur in a random pcd format, saves it to the file and displays the coordinates of the points in the terminal.

The samplePCD(double a) member function samples the pcd read with the object according to the a value it receives.

The rotatePCD (double a) member function rotates the pcd read by the object counterclockwise by the angle a (degree) according to the a value it receives.

The scalePCD(double a) member function scales the pcd read with the object by the a value according to the a value it receives.

These 3 functions overwrites the new pcd that is formed, at the same time it is possible to save the pcd to be saved as a new file without overwriting the old one, with a piece of code that has been commented out.

The viewPCD() member function opens a window to view the pcd in 3D and gives the opportunity to examine the pcd in this window.

The readPCD() member function allows us to read the x, y and z coordinates of pcd in the terminal respectively.

Segmentation Class

Since the Segmentation class is inherited from the CommonProcesses class, it has all the member functions of this class. it also has the segmented(), individual_savePCD(), colorizedPCD() and labeledPCD() member functions that are virtually crafted for use in subclasses.

RANSAC Class

RANSAC class is inherited from Segmentation class. so it also has member functions of CommonProcesses class and Segmentation class.

The colorizedPCD() member function inherited from the segmentation class segments the pcd's surfaces according to the RANSAC method and colors these surfaces randomly. Finally, it opens windows that will allow us to see these surfaces one by one and shows the colored surfaces in each window.

The individual_savePCD() function inherited from the segmentation class separates the surfaces according to the RANSAC method, as in the

colorizedPCD() function, and saves each separated surface to a file with the .pcd extension.

The segmented(double a) function inherited from the segmentation class segments the pcd according to the RANSAC method according to the tolerance value a.

RegionGrowing Class

The RegionGrowing class is inherited from the Segmentation class. therefore it also has member functions of CommonProcesses class and Segmentation class. Functions do not have their real functions because the functions to be processed here failed and these problems could not be solved due to my poor time management. There are terminal outputs that check if functions are called from the correct object. Also, all member functions of CommonProcesses class work perfectly in this class.

Since the constructors of all these classes are inherited from the CommonProcesses class, they work the same as in the base class.