BÁO CÁO ĐỀ TÀI NGHIÊN CỨU

Tái tạo đối tượng 3D từ đám mây điểm 3D (Sử dụng thư việc PCL)

Sinh viên thực hiện:

- Dương Thị An - 20120240
- Phan Đình Anh Quân - 20120635
- Phạm Gia Thông - 20120201

CONTENT

- Motivation
- Problem Statement
- Introduce to PCL
- Marching Cube

MOTIVATION

Motivation

Interesting general scientific problem

Creating, recognizing and analyzing objects from the physical world

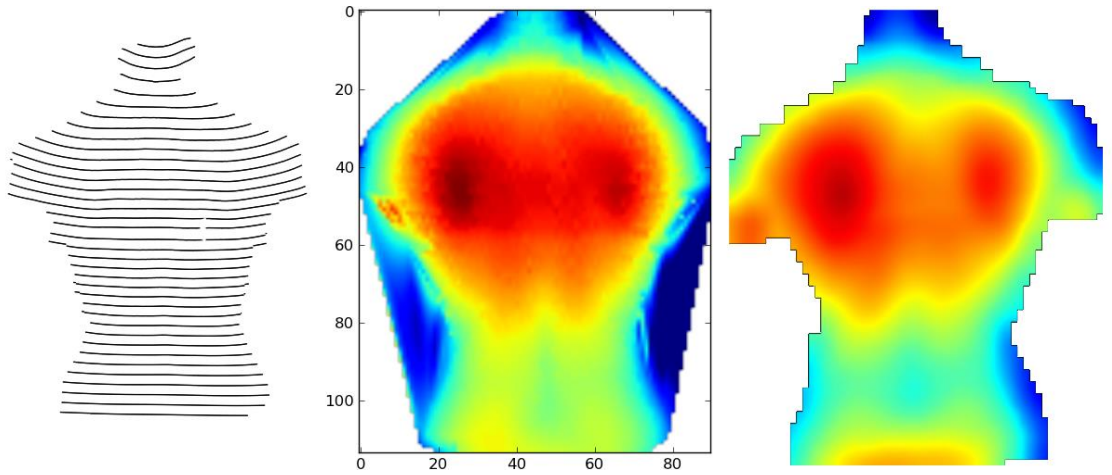
Reconstructing objects from the physical word to the digital representation

Motivation

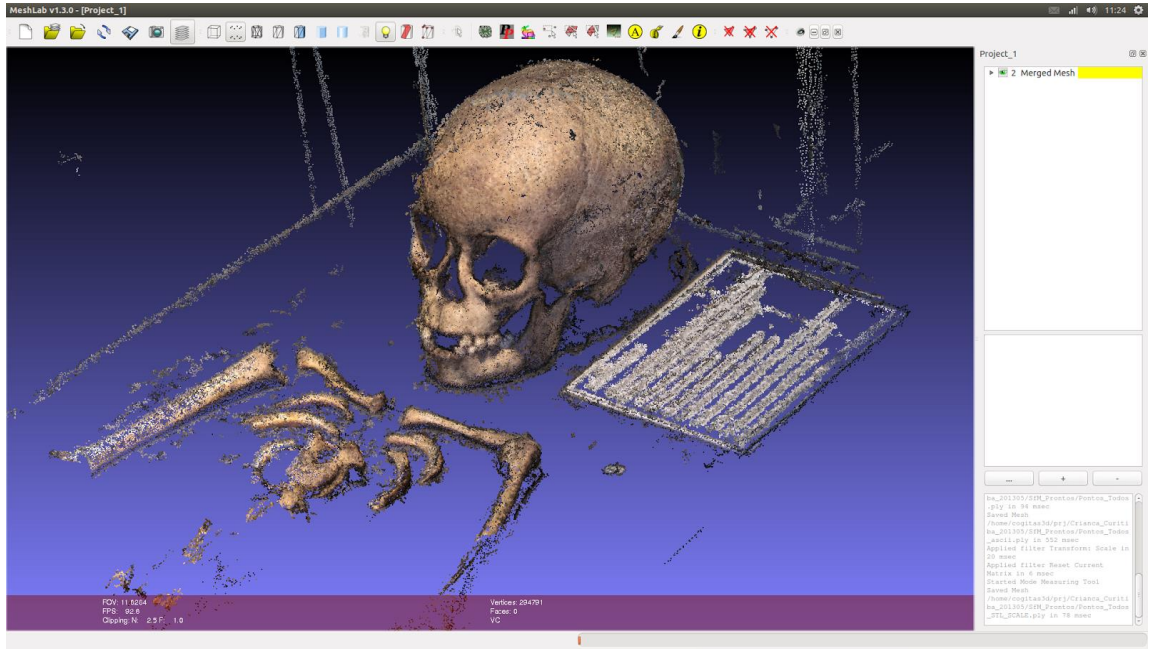
Have a wide variety of applications:

- Virtual reality
- Medical imaging
- Archeological exhibitions
- SLAM
- Urban planning
- …

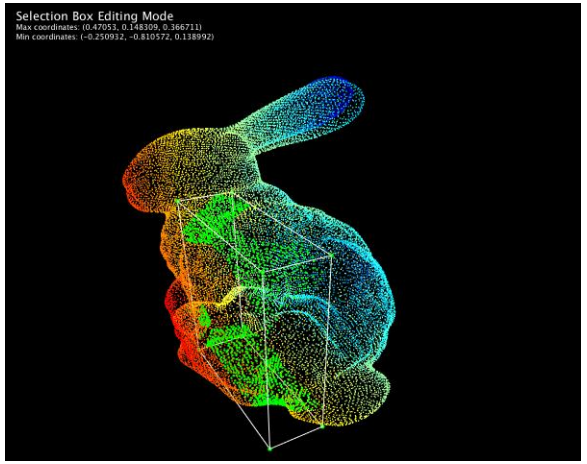
Simulation of human body parts

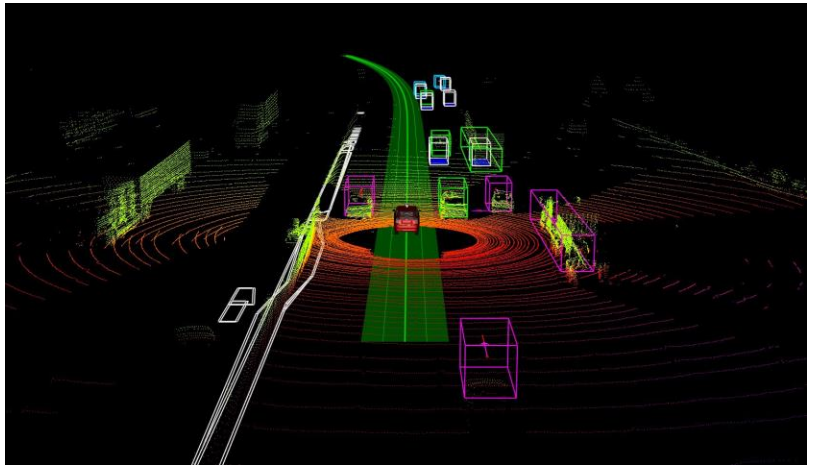


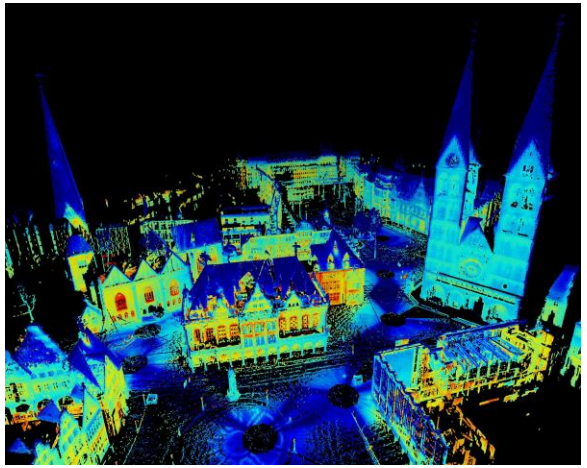
Replica of relics found in archaeological works

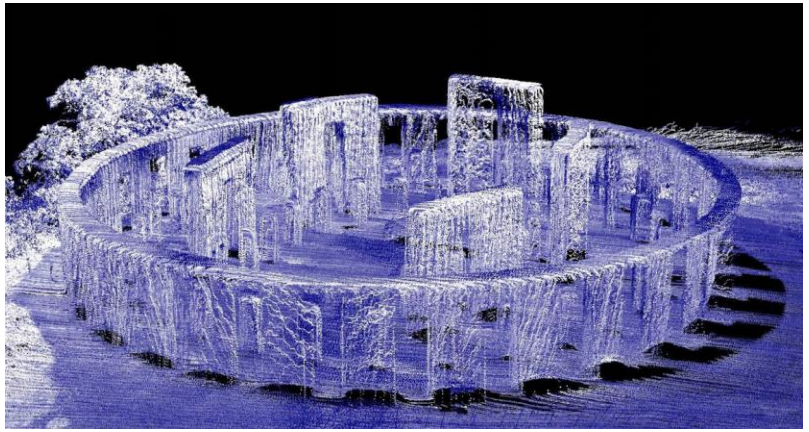


The computer recognizes the point cloud dataset





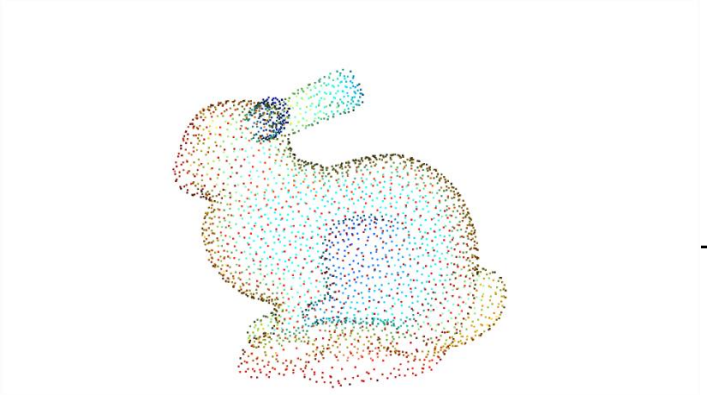




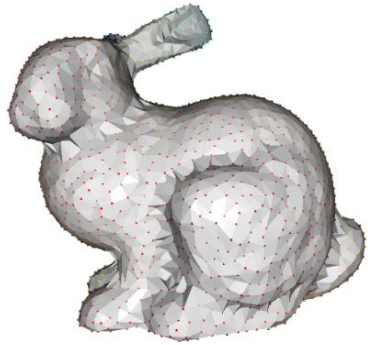
PROBLEM STATEMENT

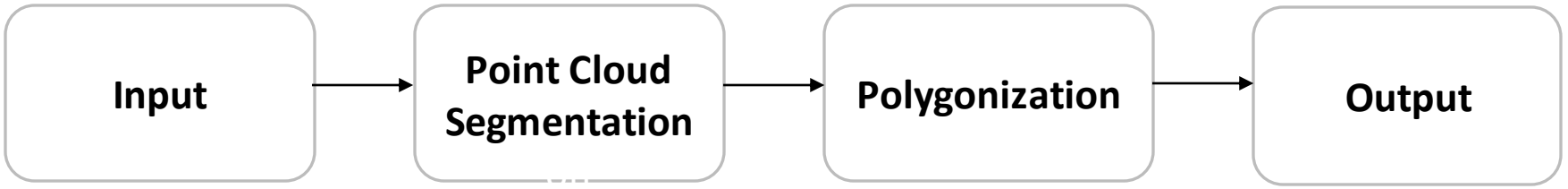
Problem Statement

Input



Output





INTRODUCE TO PCL

- An open-source library of algorithms for point cloud processing tasks and 3D geometry processing.
- Written C++.
- Supports MacOS, Windows, Linux and Android.

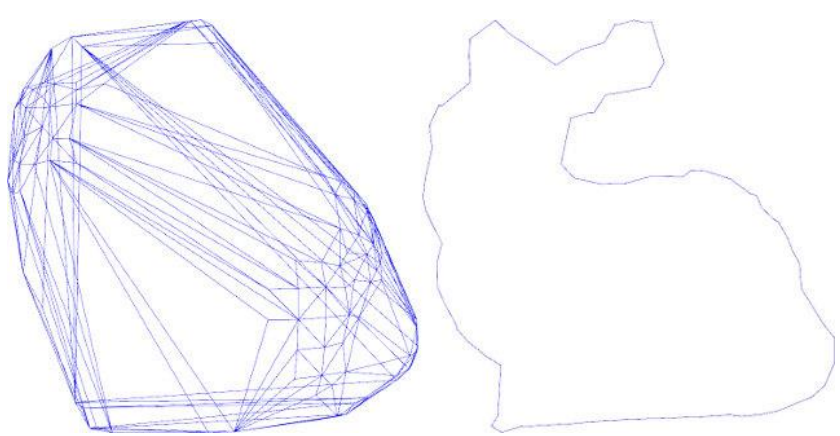
Module Surface

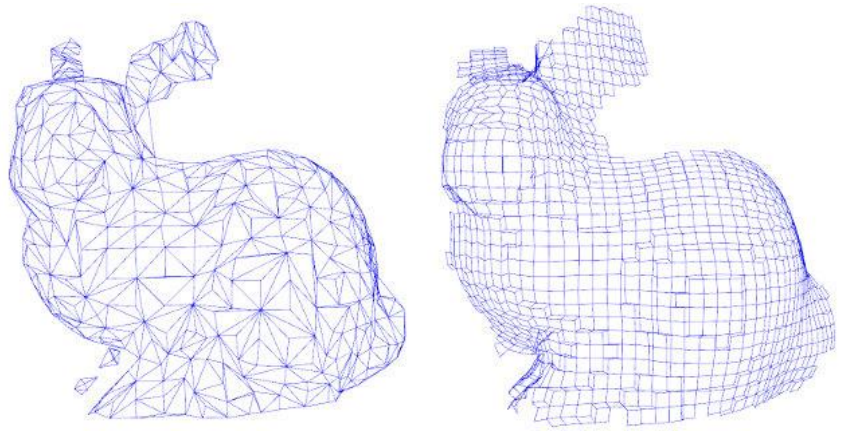
The **pcl_surface** library deals with reconstructing the original surfaces from 3D scans.

Smoothing and resampling can be important if the cloud is noisy, or if it is composed of multiple scans that are not aligned perfectly.

Meshing is a general way to create a surface out of points.

Creating a convex or concave hull is useful.

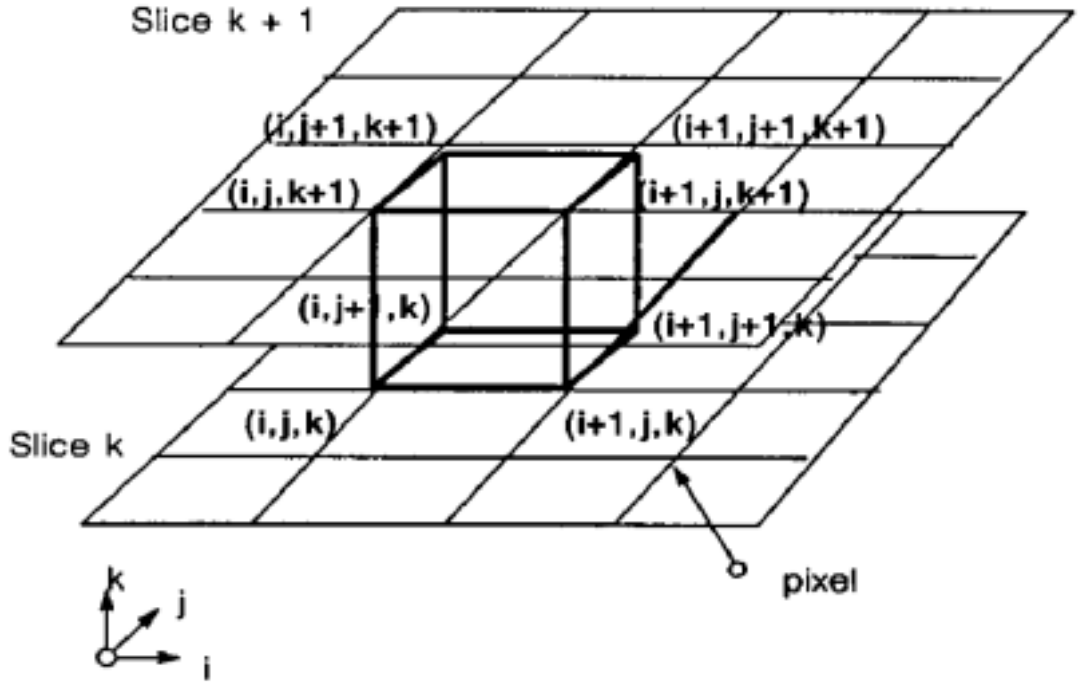




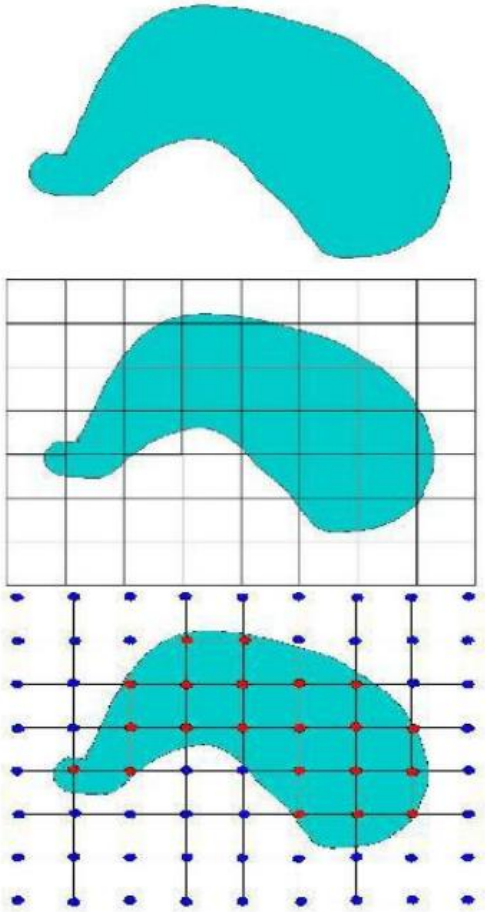
MARCHING CUBE

Marching Cube

Marching cubes uses a divide-and-conquer approach to locate the surface in a logical cube created from eight pixels; four each from two adjacent slices.



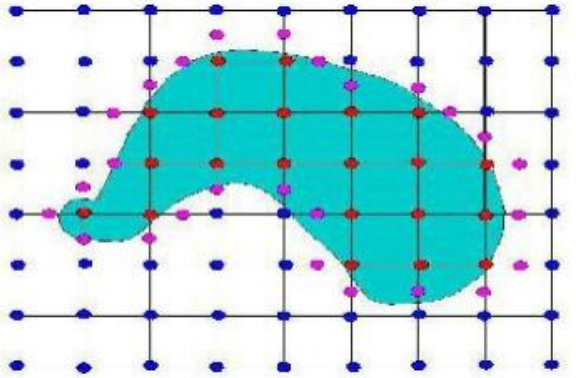
Marching Cube



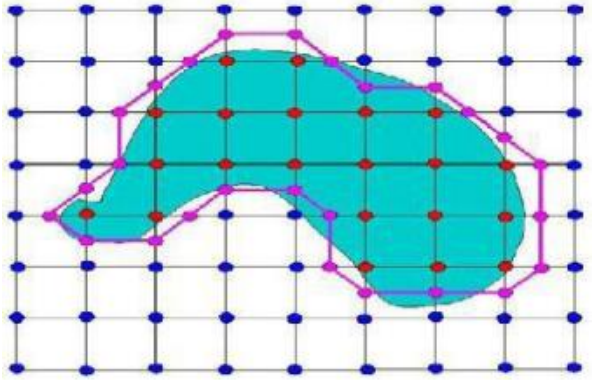
Step 1: Image Object

Step 2: Segmented Image

Step 3: Intersected Point Finding



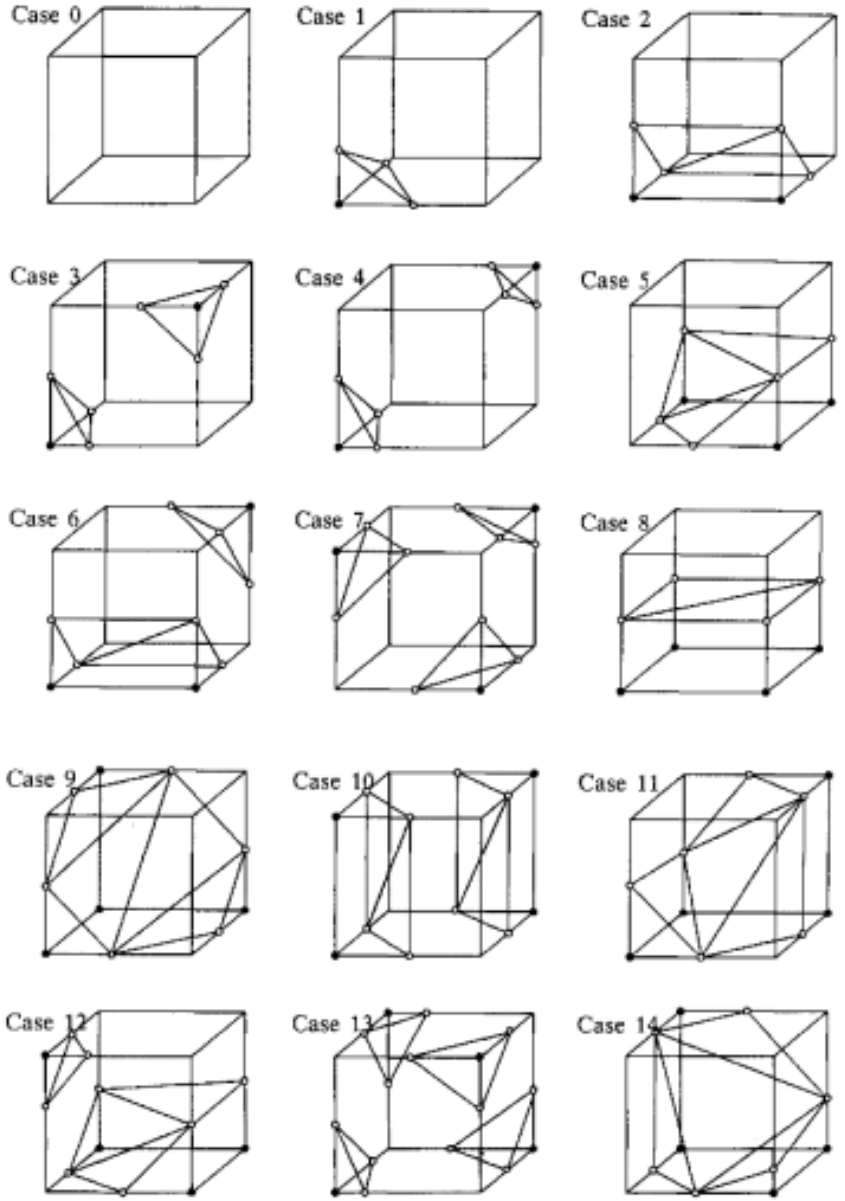
Step 5: Find Offset Points



Step 7: Joint all Offset points

Marching Cube

Each 8 vertex cells has only two possible states, there is total of cases of $2^8 = 256$ intersection between isosurface and edges. For the faster execution they are listed in lookup table. We can reduce the 256 cases into 15 unique marching cube



Marching Cube

The final step in marching cubes calculates a unit normal for each triangle vertex. The rendering algorithms use this normal to produce Gouraud-shaded images. A surface of constant density has a zero gradient component along the surface tangential direction; consequently, the direction of the gradient vector, 7, is normal to the surface. We can use this fact to determine surface normal vector, $\vec{n}$, if the magnitude of the gradient, $|\vec{g}|$ is nonzero. The gradient vector, $\vec{g}$ is the derivative of the density function

$$\vec{g}(x,y,z) = \nabla\vec{f}(x,y,z)$$

To estimate the gradient vector at the surface of interest, we first estimate the gradient vectors at the cube vertices and linearly interpolate the gradient at the point of intersection. The gradient at cube vertex (i,j,k) is estimated using central differences along the three coordinate axes by:

$$G_x(i,j,k) = \frac{D(i+1,j,k) - D(i-1,j,k)}{\Delta x}$$

$$G_y(i,j,k) = \frac{D(i,j+1,k) - D(i,j-1,k)}{\Delta y}$$

$$G_z(i,j,k) = \frac{D(i,j,k+1) - D(i,j,k-1)}{\Delta z}$$

References

[1] Dirk Holz, Alexandru E. Ichim, Federico Tombari, Radu B. Rusu, and Sven Behnke, “Registration with the Point Cloud Library A Modular Framework for Aligning in 3-D”, 2015

[2] Werner Purgathofer, Markus Vincze, “Reconstruction of 3D Models from Images and Point Clouds with Shape Primitives”, 2013

[3] Radu Bogdan Rusu and Steve Cousins Willow Garage, “3D is here: Point Cloud Library (PCL)”

[4] Laurent Caraffa, Yanis Marchand, Mathieu Brédif, Bruno Vallet, “Efficiently Distributed Watertight Surface Reconstruction”, 2021

[5] Marcos Vinicius Mussel Cirne, Hélio Pedrin, “Marching Cubes Technique for Volumetric Visualization. Accelerated with Graphics Processing Unit”, 2013 [6] William E. Lorensen, Harvey E. Cline, “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”, 1987

Thank you