Report: Optyczne techniki skanowania i analizy danych trójwymiarowych, Project 2

Task: Find keyboard/keyboards

Made by: Bartłomiej Guś

1. Introduction

The aim of this project was to find keyboard in the cloud of points. It turned out that, there is two keyboards and I managed to find them both. In the purpose to achieve the goal of this project I put to use OGX|QtFRAM3d in which we are able to load the cloud of points and with the application of Example.cpp file calculate for example distance between points. The whole program consists of the following parts: GetLessCloud and FindKeyboard.

I add in program parts of code, which protects from errors. Also the whole code is clean code and I have added progress bars, informative massages: which task is being solutioned by code.

Figure 1 - Cloud of points

2. Solution

2.1. GetLessCloud

This part of the project is responsible for simplify the cloud, which allows for faster calculations in the second part of the project: FindKeyboard.

In this part of project, I use below parameters:

```
Data::ResourceID m_node_id;
float points_to_remove_ratio = 0.7;
```

The first variable was applied in purpose to specify the value of the node where the cloud of points is store. The second variable: `points_to_remove_ratio` was used to specify how many of points will not be added to new cloud, 0.7 means that 70% of points will not be added. Points were choosing in random way. It causes that amount of points decrease from around 1 million to 0.3 million of points.

Figure 2 - Cloud of less points

To the new cloud I only add XYZ parameters and COLOR parameters, because based on them I will try to find keyboard/keyboards in the cloud of points.

2.2. FindKeyboard

This part of the project was responsible for marking the points, which are associated with the keyboards.

In this part of project, I use below parameters:

```
Data::ResourceID m_node_id;
float distance_hausdorf = 0.01;
int minimum_points_in_group = 100;
double ratio_of_minimum_black_points_in_group = 0.8;
double minimum_diagonal = 0.42;
double maximum_diagonal = 0.48;

struct hausdorff_struct
{
	int group; // which group belongs point
	bool was_or_not; // was or not calculted
	ogx::Data::Clouds::Point3D xyz;
	ogx::Data::Clouds::Color color;
};
```

The m_node_id was applied to define where is allocated the cloud. The second parameter was used as hausdorff distance. The third parameter was applied due to what is the minimum amount of points in one group to treat them like an object. Ratio of minimum is used in first category of selection object: what is the minimum number of black points in object. Maximum and minimum diagonal is the second category which select object due to the maximum distance between points, which are associated with this object. The struct hausdorff_struct was used in Hausdorff algorithm and in these two categories of selection.

2.2.1. Grouping points using Hausdorff algorithm

In this part I applied the Hausdorff algorithm, which was explained during the lectures.

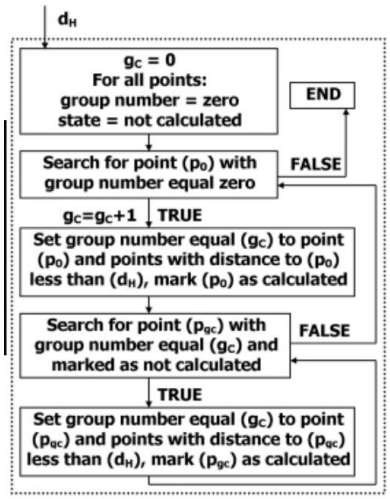

Figure 3 - The pseudocode of Hausdorff algorithm

I have chosen 0.01 as the Hausdorff distance, this value allows me to group the points in object in the right way, what means the two keyboards and most of object were grouped separately.

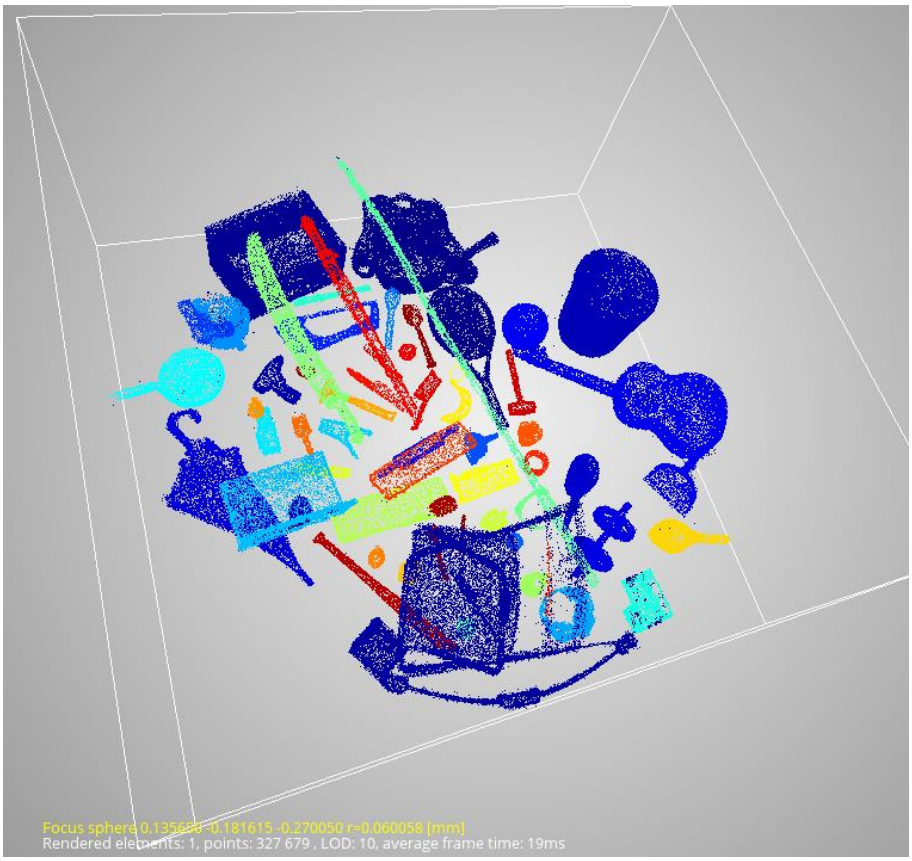

Figure 4 - Grouped object after Hausdorff algorithm

As you could notice not all objects were grouped to separate division e.g. the guitar and the scoop, the bow and the chair. But as I mentioned before the keyboards and most of objects were grouped in separate divisions, so this goal has been achieved.

If the group has less points than it is predicted in parameter: `minimum_points_in_group` than the points was qualified in 0 group which is the group of rejected points. The navy blue points belong to the 0 group.

The result is added as layer: *Group*.

This part is implemented in functions:

- `auto hausdorff(ogx::Data::Clouds::ICloud* cloud, Context& context)`
- `void color_layer(std::vector<hausdorff_struct> hausdorff_vector, Context& context, ogx::Data::Clouds::ICloud* cloud)`

2.2.2. Grouping points by the color

In this part I am starting to find the points which are associated with keyboards. I have done this in this way: If the object has fewer black points than in `ratio_of_minimum_black_points_in_group` was qualified in 0 group which is the group of rejected points. The parameter `ratio_of_minimum_black_points_in_group` equals to 0.8 allows me to narrow down the option to only 17 objects from 57. The black points were searched by the Value in the HSV represantion, when the Value of point was between 0 – 0.3 it was classified as black point. Also, in the project was calculated the hue and saturation value from RGB representation which is not used.

The result is added as layer: *Group after sorting color*.

This part is implemented in functions:

- `auto sort_due_to_color(std::vector<hausdorff_struct> hausdorff_vector, Context& context, int *max_group, std::vector<hausdorff_struct> *temporary)`
- `void color_layer_after_sorted_color(std::vector<hausdorff_struct> hausdorff_vector, Context& context, ogx::Data::Clouds::ICloud* cloud)`

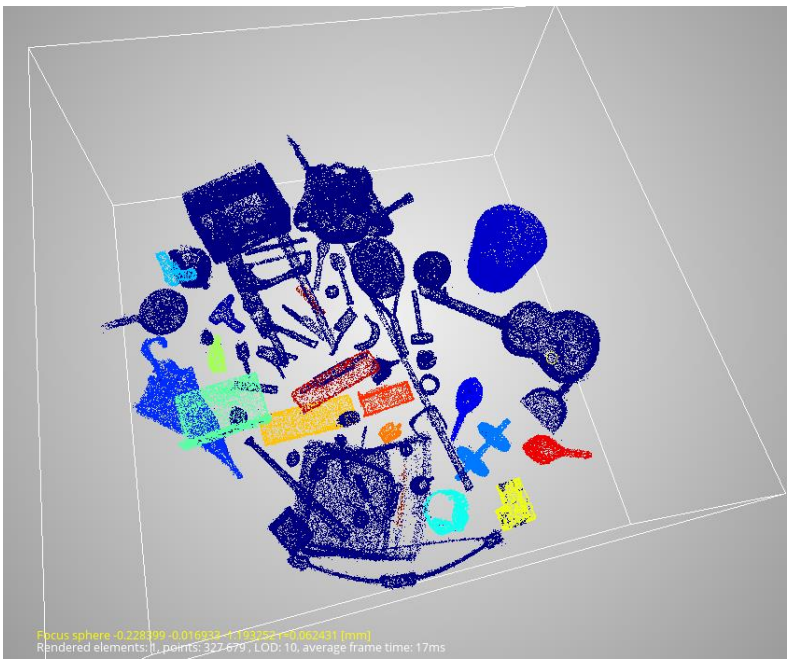

Figure 5 – Grouped object after color selection

2.2.3. Grouping points by the diagonal

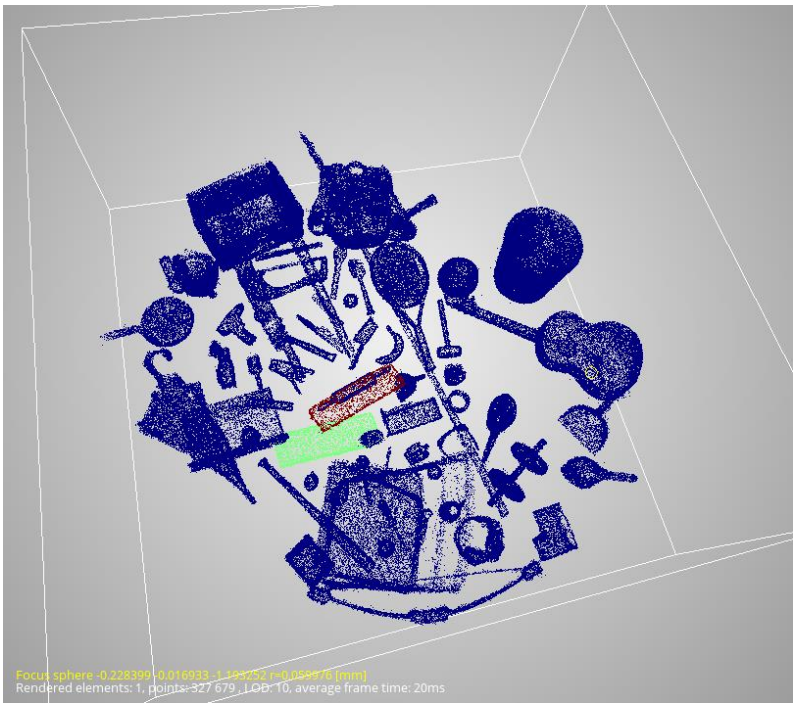

Figure 6 - Grouped object after color selection and diagonal

In this part I grouped the object, which have been already grouped by color, using the diagonal value. This quantity was calculated as the distance between the most extreme points. In the purpose to find the keyboards I set the minimum value of diagonal equals to 0.42 and maximum value of diagonal equals to 0.48.

This process allows to find only two objects, which are the keyboards. The rest of points are marked to 0 group, which are rejected points.

The result is added as layer: *Group after sorting color and diagonal*.

This part is implemented in functions:

- `auto sort_due_to_diagonal(std::vector<hausdorff_struct> hausdorff_vector, Context& context, int* max_group, std::vector<hausdorff_struct> temporary, std::vector<ogx::Math::Point3D> *points_to_primitive)`
- `void color_layer_after_sorted_color_and_diagonal(std::vector<hausdorff_struct> hausdorff_vector, Context& context, ogx::Data::Clouds::ICloud* cloud)`

3. Summary

Fortunately, my processing process based on grouping by colors and diagonal allowed me to find in the cloud of points, only these points which are associated with the keyboards. Also, this way of thinking allowed me to find not only one object but two of them. Maybe the grouping by color category could be omitted and the grouping by the diagonal could be enough, what would shorten the computation time. But this way of processing I did seem to be right.

Not all objects were grouped to separate division. To achieve that we should decrease the value of Hausdorff distance and also decrease the `points_to_remove_ratio`, this should cause: all items would be in different groups, but also objects with fewer points should not be divided into several areas.

An interesting approach to carry out the segmentation of points would be with used of artificial neural network e.g. PointNet, because sample dataset also includes these items, which we are ought to segment. As well to this task we can use DBSCAN (Denisty-Based Spatial Clustering of Applications with Noise): it creates the group of points reachable by density category, it analyzes the neighborhood of this point, and if this neighborhood has more than definite amount of points, it is included in the region. What means that, DBSCAN is going to have a problem to find clusters of different densities.

Grouping to find keyboards could be done using geometric features as rectangularity ratio.