FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Neighbor detection in point clouds using the Boundary Complex

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Simon Brenner
Matrikelnummer 0927175

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Dr. Stefan Ohrhallinger

Wien, TT.MM.JJJJ

(Unterschrift Verfasser)

(Unterschrift Betreuer)

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Neighbor detection in point clouds using the Boundary Complex

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Simon Brenner
Registration Number 0927175

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Dr. Stefan Ohrhallinger

Vienna, TT.MM.JJJJ

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Simon Brenner
Hauptstraße 10, 2392 Grub

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Kurzfassung

In vielen Anwendungen (z.B. Robotic Vision) ist es wichtig Punktwolken in logisch zusammenhängende Cluster zu zerteilen, um Objekte oder Oberflächen erkennen zu können. Ein essentieller Teil der meisten existierenden Segmentierungsansätze ist die Definition von Nachbarschaft zwischen Punkten. Diese Arbeit beschreibt und evaluiert die Verwendung des 'Boundary Complex', einer aufspannenden, verbundenen Menge von Dreiecken über die Punktwolke, für die Nachbarschaftsermittlung im Zusammenhang mit der Segmentierung von Punktwolken. Die erzielten Ergebnisse werden mit existierenden Ansätzen zur Nachbarschaftsermittlung verglichen.

Abstract

For many applications (e.g. robotic vision) it is important to separate the Point Clouds produced by a 3D scanner into logically associated clusters in order to recognize objects or surfaces. A crucial part of most of the existing segmentation approaches is the definition of neighborhood between points. This thesis describes and evaluates the use of the 'Boundary Complex', a spanning connected set of triangles on the point cloud, for neighbor detection in the context of cloud segmentation. The results will be compared to existing neighbor picking approaches.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Problem Statement . . . 1
- 1.3 Methodological Approach . . . 2
- 1.4 Structure of the work . . . 2

2 Background and Related Work **3**
- 2.1 The Minimum Boundary Complex . . . 3
- 2.2 Other definitions of neighborhood . . . 5
- 2.3 F1-measure for unsupervised clustering . . . 5

3 Methodology **7**
- 3.1 The Segmentation Algorithm . . . 7
- 3.2 Parameters . . . 7
- 3.3 What to compare . . . 9
- 3.4 Evaluation . . . 10

4 Implementation **11**
- 4.1 External Software . . . 11
- 4.2 Program Flow . . . 11

5 Results **13**
- 5.1 Determining the parameters . . . 13
- 5.2 Analyzing the test results . . . 13
- 5.3 Conclusion . . . 18

6 Summary and future work **23**

A Input clouds **25**

B Numbers **31**

Bibliography **37**

CHAPTER 1

Introduction

1.1 Motivation

Point clouds are a common representation for range images that we receive from 3D scanning devices such as laser range scanners. The use of those devices is steadily increasing, for example in the area of robotic vision. Therefore the task of acquiring semantic information out of those point clouds is a big issue. In many applications, acquiring this information is closely connected to detecting objects or surfaces. This in turn corresponds to a proper segmentation of the cloud.

A recently developed method for boundary reconstruction of point clouds, the "Minimum Boundary Complex" (see Section 2.1), can possibly support and improve existing segmentation techniques. For many segmentation approaches, the detection of "neighboring" points is of great importance: when following a region growing approach for example, neighbors are needed to determine where the region should grow to. Especially in unorganized point clouds, where the points are not given in a two-dimensional array which is produced by the laser scanning device (like assumed in [2]), this is not a trivial task; neighborhood can be defined in different ways [7], some of which will be discussed in Chapter three. This is where the Boundary Complex comes into play: Due to its spanning graph structure and other properties which will be discussed later, it can be used as an indicator for neighborhood.

1.2 Problem Statement

The aim of this work is to evaluate how well the neighborhood function provided by the Boundary Complex performs in the field of point cloud segmentation, compared to other neighborhood functions. More accurately, the application in a region growing algorithm in the Euclidean sense will be examined.

1.3 Methodological Approach

The results of the new approach for neighbor finding will be tested and evaluated empirically. For that, a simple segmentation algorithm, which will be described later, has been implemented using three variants of neighbor selection:

- Boundary Complex
- K nearest neighbors
- Radius search

The computed clustering solutions are then each compared to a solution which has been created manually, in a way that seems reasonable for a human observer. For comparison of the solutions we use an F1-measure for unsupervised clustering, as described in Section 2.3. The F1-measures produced by numerous tests, using various input clouds of different sampling densities and noise levels, will be the basis of the result and conclusion of this work.

1.4 Structure of the work

In Chapter 2 we will provide background for our thesis. First the Minimum Boundary Complex will be introduced, then we describe the other neighborhood definitions that will serve as references for comparison in this work. Also the F1-measure for unsupervised clustering will be explained. Chapter 3 describes our approach to the stated problem. We introduce the clustering algorithm to which the neighbor finding approaches will be applied, explain the parameters that need to be set and describe how we generated the ground truth for evaluating the clustering results. In Chapter 4 a short overview on our implementation will be given. In Chapter 5 the results of our experiments will be discussed. Some statistical analysis will be given, along with visual examples of the different clustering performances. We will then make conclusions, if and under which circumstances the Boundary Complex holds advantages over the other methods.

CHAPTER 2

Background and Related Work

2.1 The Minimum Boundary Complex

The Minimum Boundary Complex, which is described in [9], is a connected set of triangles on a point cloud, which was developed for being a basis to the construction of an aesthetically pleasant boundary representation of the cloud. The term 'aesthetically pleasant' can be described more accurately by the fulfillment of the Gestalt laws of proximity, good continuity and closure. In this context, proximity means that spatially close points should be connected, continuity means that there should be no abrupt changes in curvature and closure refers to a surface that bounds a solid region.

Construction of the Minimum Boundary Complex

A Boundary Complex BC is defined as a spanning complex of connected triangles on the point set P, with the constraint that each edge has at least 2 incident triangles. The basis of the Boundary Complex is a three dimensional Delaunay triangulation DG of the point set, which is, per definition, already a BC. To create the Minimum Boundary Complex BC_{min}, a subset of triangles must be selected from DG, in a way that minimizes the sum of the longest edges of the triangles:

$$BC_{min} = \sum_{t_i}^{T} \lambda(t_i) \rightarrow min$$

where $\lambda(t_i)$ is the longest edge in triangle t_i.

This is an NP-hard problem, but a very good approximation BC_0 of BC_{min} can be found by using a greedy algorithm in a computing time of $O(n\ log\ n)$. The triangles of BC_0 largely overlap with those of the ideal solution BC_{min}, because they are selected by the same minimization criterion.

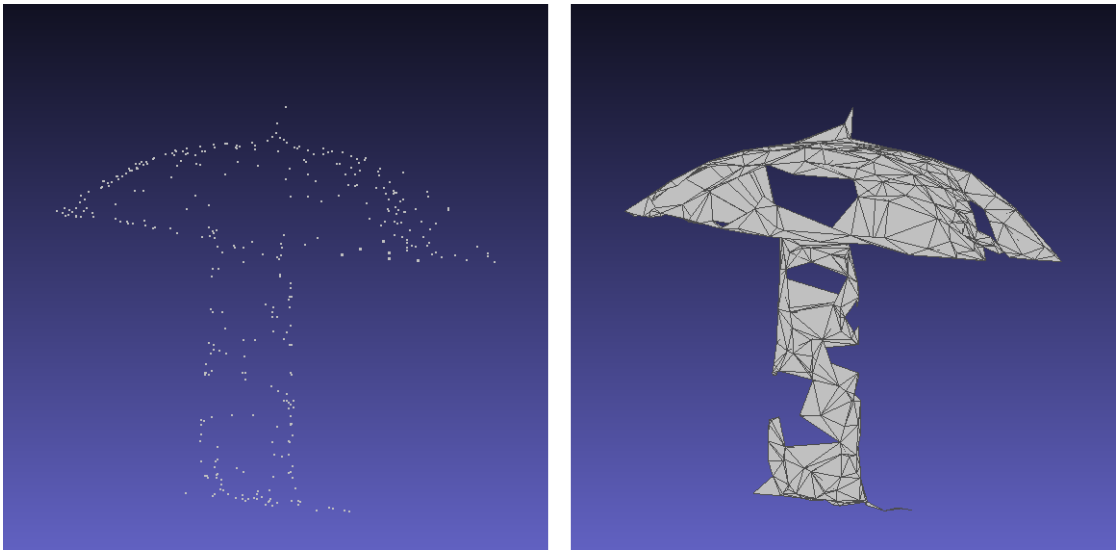

Figure 2.1: Left: 3d Point Cloud. Right: BC_0

Properties of BC_{min}

- Due to the fact that BC_{min} is created on the basis of a Delaunay Triangulation, which minimizes the length of edges and maximizes the angles between edges, the Gestalt laws of proximity and good continuity are implicitly mostly followed. We also acquire a low curvature, which corresponds to a good boundary shape.
- "Since an edge can have two or more incident triangles, BCmin is a single connected set but in general, not manifold." [9]
- Also works well for significant noise
- The resulting shape is hardly affected by downsampling

The use of BC_{min} for neighbor picking

The edges of the triangles of the Minimum Boundary Complex can be viewed as a graph spanning all the points in a point cloud. In that graph, two vertices are adjacent, if they are (at least once) part of the same triangle. So, two points are neighbors, if they are adjacent in the described graph (Figure 2.2). The reason for the assumption that this neighborhood can be superior to other approaches lies in the properties of the Minimum Boundary Complex itself, which has been described above. First, the lengths of edges are minimized. This means that neighbors can be generally expected to show smaller Euclidean distances than non-neighbors, and there is a good chance that spatially close points should belong to the same cluster. Furthermore, in contrast to other approaches such as a radius search, the Boundary Complex approach should be robust to non-uniform point distributions, which naturally occur in laser scanning images.

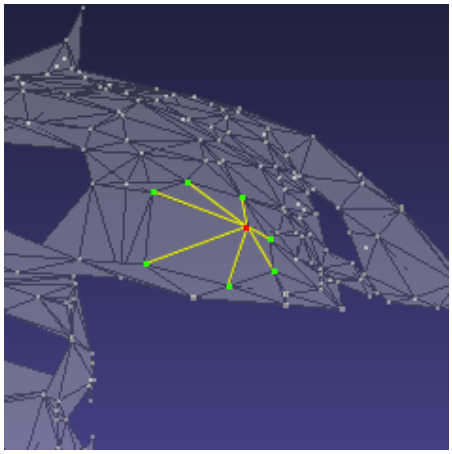

Figure 2.2: Neighborhood in the Boundary Complex. Red: seedpoint. Yellow: relevant edges. Green: neighbors.

2.2 Other definitions of neighborhood

Radius Search

In this rather simple approach, neighboring points are defined as all points that lie within a defined radius from the seed point [10]. The number of selected neighbors varies of course. The problem of this approach is to find a suitable radius. Especially when facing very non-uniform point distributions, it may be impossible to find a radius that works globally: if chosen too big, too many neighbors are selected in dense regions; if chosen too small, too few or no neighbors are found in sparse regions.

K nearest neighbors

The k nearest neighbors method is a widely spread approach, used in diverse fields of application [11]. In contrast to the radius search, a fixed number of the nearest neighbors, defined by parameter k, is selected. This makes the Knn-approach more suitable for non-uniform point clouds, as there is no fixed distance threshold. On the other hand, this method can lead to the selection of distant points, which should not belong to the same cluster.

2.3 F1-measure for unsupervised clustering

In order to achieve statistical significance, we need a numerical value describing the similarity of two clustering solutions. A common measure for describing the accuracy of classification tests is the F1-Measure or F1-Score, which is a harmonic mean of the statistical values precision and recall:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

The F1-measure ranges between 0 and 1, where 1 means total congruence. If objects only have to be classified as positive or negative, precision and recall are defined as:

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

The comparison of clustering solutions however is more complicated:

- We do not know how many clusters (=classes) there will be
- The number of clusters in the computed solution can be different from the number of clusters in the ground truth
- The mapping between clusters of the ground truth and the computed clusters is unknown

The solution for that problem which is used in this work is the "F1-Measure for Unsupervised Clustering with Non-Determined Number of Clusters", as described in [5].

If C is the ground truth and K is the computed solution, the basic idea of this F1-measure is as follows:

For each cluster c_i of C, precision $P(c_i, k_j)$ and recall $R(c_i, k_j)$ are determined for each cluster k_j of K using these formulas ($n_{c,k}$ is the size of the overlapping of clusters c and k):

$$P(c_i, k_j) = \begin{cases} 1 - \frac{\sum_{c \in C, c \neq c_i} n_{c,k_j}}{\sum_{c \in C, c \neq c_i} |c|} & \text{if } |C| > 1 \\ 1 & \text{otherwise} \end{cases}$$

$$R(c_i, k_j) = \begin{cases} 1 - \frac{n_{c_i,k_j} - 1}{|c_i|} & \text{if } |c_i| > 1 \\ 1 & \text{otherwise} \end{cases}$$

The total precision $P(c_i)$ and recall $R(c_i)$ of c_i are then the sums of $P(c_i, k_j)$ or $R(c_i, k_j)$, weighted by the size of the overlappings of the corresponding clusters k_j with c_i. The overall precision P and recall R are then the sums of $P(c_i)$ and $R(c_i)$ respectively of all clusters in C, weighted by the size of the corresponding clusters c_i. From those total values for precision and recall, the F1-measure can be calculated (see formula above).

CHAPTER 3

Methodology

3.1 The Segmentation Algorithm

In order to test and compare the neighbor picking approaches in the field of point cloud segmentation, a simple segmentation algorithm has been implemented: In the first step the cloud runs through a Random Sample Consensus (RANSAC) [3] plane fitting segmentation, which detects the largest planes within the point cloud. This is a reasonable step because as scenes are usually set in a human made environment, which consists of planes for a big part (floors, tables, walls). We exclude those planes because segmenting them is trivial and so they do not contribute significantly to the evaluation of the different neighborhood finding approaches; this way, the other objects of the scene are isolated.

After that, the remaining points are clustered using a flood fill approach, as known from image processing (Algorithm 3.1), which was inspired by [10]. Of course, the crucial part of that algorithm is the function $getNeighbors()$ (line 11); here the different neighbor finding approaches, which were described in Chapter 2, are plugged in.

3.2 Parameters

RANSAC

There are two parameters that influence the first processing step, in which a RANSAC plane fitting is performed. First, the distance threshold d, i.e. the distance from a plane up to which points are considered to support the plane model. Second, the break condition of the RANSAC loop: we start detecting the largest plane and continue finding the next smaller plane up to a certain point. The RANSAC segmentation is just a preprocessing step that is performed before the actual clustering steps, in which the different neighbor picking approaches are compared; so the results of this step do not contribute to the results of our comparisons. Therefore, we chose parameters which led to the detection of the largest planes in our test scenes. Those values were

input : set of unlabeled points U, minimum cluster size m
output : set of clusters $\{C\}$
initialize: $\{C\} \leftarrow \emptyset$

```
1  while U ≠ ∅ do
2      current list of seeds S ← ∅;
3      current cluster C_cur ← ∅;
4      first point in U → u;
5      S ← S ∪ u;
6      C_cur ← C_cur ∪ u;
7      U ← U \ u;
8      while S ≠ ∅ do
9          first point in S → s;
10         S ← S \ s;
11         N ← getNeighbors(s);
12         foreach n ∈ N do
13             if n ∈ U then
14                 S ← S ∪ n;
15                 C_cur ← C_cur ∪ n;
16                 U ← U \ n;
17             end
18         end
19         if size of C_cur ≥ m then
20             {C} ← {C} ∪ C_cur;
21         end
22     end
23 end
```

Algorithm 3.1: Clustering algorithm

found by empirical testing: The distance threshold d is the maximum extent of the plane in x, y and z direction divided by 100; the plane detection stops when the last detected plane contained less than 10% of the total number of points of the cloud. An exception has been made for the input files showing the "outdoor" scene A.5, as there are very few planar parts. For these files the algorithm stops when the last plane contained less than 5% of the points of the cloud.

Knn search

The only parameter we need for the knn neighborhood is the value of the global parameter k, that is, how many neighbors are selected for a seed point.

Radius search

Here we need to find a good radius r, which is a difficult task, since we are observing point clouds of different scale, density and uniformity. The point densities of our example point

clouds however are more or less uniform, so we decided to make the radius a function of the maximum extent in the dimensions of the cloud, similar to the way the distance threshold is calculated for the RANSAC step. Precisely, r is multiplied by the thousandth of the maximum extent to acquire the actual search radius.

Boundary Complex search

In general, neighborhood in the sense of the Boundary Complex is strictly defined, as mentioned above in Section 2.1, and does not require customization. However, there are rare cases in which connections exist between very distant points, which obviously should not belong to the same cluster. In order to detect those outliers, an additional threshold was introduced: If a point is farther away from the seed point than the median distance of all neighbors, multiplied a constant c, that point is not considered a neighbor. So c is the additional parameter.

3.3 What to compare

In order to evaluate the quality of segmentations using the different functions for neighbor picking, a ground truth is needed for comparison. For that purpose, a couple of point cloud datasets (taken from [8] and [1]) have been manually clustered in a way that seems reasonable for a human observer. This manual segmentation was done by coloring the clouds using a point cloud editing software [6]. All manually clustered input clouds in full sampling densities and with zero noise can be found in appendix A. Our implementation then interprets the colors as clusters. After the implemented clustering algorithms have performed the different segmentations, their F1-measures with respect to the ground truth can be calculated.

Varying point density

The original point clouds have been downsampled to different sampling densities, in order to compare the performances of different approaches in relation to the input data resolution. Each data set have been examined in five sampling densities:

- all points
- $\frac{1}{2}$ of all points
- $\frac{1}{4}$ of all points
- $\frac{1}{8}$ of all points
- $\frac{1}{16}$ of all points

Introducing noise

The neighbor detection approaches will also be tested for their robustness to noise. For that purpose we add a uniform random variable to the x, y and z coordinates of each point. The

maximum offset (in positive or negative direction) can be controlled by an additional parameter u. If u is set to zero, no artificial noise is applied to the point cloud.

3.4 Evaluation

The first thing that needs to be done is to find optimal values for the parameters that are specific to the different neighbor picking approaches, as described above. This will be done by running the segmentation for each approach, for each test dataset in each resolution but with zero noise, varying their respective parameters. For each input dataset the parameter value that produced the best F1-measure is picked; the median of all picked parameters is then supposed to be the most suitable one. The parameters that we find this way are then fixed; this is necessary to reduce the complexity of the following comparison. For each parameter, the following values will be considered:

- k: {1,2,3,...30} The determination of this interval was in fact the result of empirical testing, where ks of different magnitudes have been tried to find out in where an optimal k definitely cannot be found. What remained was the interval from 1 to 30. We decided to set the step size to one, because the interval is not very large and k has to be an integer.

- r: {1.0,6.0,11.0,...96.0} Note that r is, as mentioned 3.2, not a total value for a radius but a factor multiplied by the thousandth of the maximum extent of the cloud. A radius has to be non-zero in order to select neighbors, so we started our interval of observation at r=1. Also, an r greater than 100 seems to be absurd; this would correspond to more than a tenth of the clouds maximum extent, which would lead to the blending of all structures. Since we are covering a quite large interval, a step size of 5 seems reasonable.

- c: {1,1.2,1.4,...5} Since outliers cannot be closer than the median neighbor distance, c must be ≥ 1. So we start our experiments at $c = 1$ and go up to 5 in moderate steps of 0.2, since 5 times the medium neighbor distance is already quite far away from the seed point.

Then the different approaches will be tested with different data sets, different resolutions and different noise levels. Then the results will be checked for correlations between resolutions and F1-measures, and between noise levels and F1-measures. This way the three approaches will be compared in order to find out if and under which circumstances the Boundary Complex approach is superior to the other methods in terms of neighbor finding.

CHAPTER 4

Implementation

For the empirical evaluation and comparison of the neighbor finding approaches, a small c++ application has been developed. The following chapter will just roughly describe what the program does, without going too much into detail.

4.1 External Software

We used the Point Cloud Library (PCL) [4] for major point cloud processing tasks, namely:

- Reading and writing point cloud files in .ply (Polygon File Format) and .pcd (PCL's point data format)
- Representation of point clouds in memory
- Rendering of point clouds
- RANSAC plane fitting
- Creating Kd tree structures for efficient radius- and knn search

For the construction of the boundary complex we used the implementation that was used in [9]

4.2 Program Flow

At startup, the application performs the following steps:

1. Read a manually clustered (colored) point cloud as input data and store it in PCL's PointCloud datastructure.
2. Interpret the coloring as clustering and, for the sake of simple handling, store it to an array of integers, where the value at position i is the number of the cluster of point i. This will be the ground truth for future comparison.

3. Apply artificial noise to the input cloud (if the noise intensity parameter is not set to zero), as described in 3.3.

4. Create the Boundary Complex.

After that, the following procedures can be initiated by user input:

Perform the clusterings

The input cloud can be clustered using the neighborhood definition by knn, radius search and Boundary Complex. This is done by an implementation of the algorithm described in 3.1. Like the ground truth, the clustering is stored in an array of integers; but also a copy of the input cloud is created and colored according to the computed clustering, so that the clustering can be represented visually. The clustering functions also deliver various data about the clustering processes, such as total number of clusters, the size of each individual cluster, and other statistically relevant data, such as the average number of neighbors detected by the Boundary Complex approach.

Visualize the clusterings

The clustering solutions and also the ground truth can be visualized as differently colored versions of the input cloud. For this we use a simple cloud viewer built in to PCL.

Compare the clusterings to the ground truth

The different clustering solutions have to be compared to the ground truth. This is done by an implementation of the F1-measure for unsupervised clustering, as described in 2.3.

Save the colored clouds

The colored clouds created in the clustering process can be saved to the disc in .ply format.

CHAPTER 5

Results

5.1 Determining the parameters

As mentioned in 3.4, first we want to find the parameters for the neighbor picking approaches that are most suitable for our set of test data. According to our empirical results (see table B.1), we fixed the parameters to the following values:

- $k = 16$
- $r = 41$
- $c = 1.8$

5.2 Analyzing the test results

With the parameters defined above, we carried out numerous tests. A full record of the test results can be found in tables B.2, B.3, B.4 and B.5. Looking at the mean F1-measures, averaged over all input data, sampling densities and noise levels (table 5.1), we do not see huge differences at first glance. At this point it should be noticed, that most of the F1-measures ranged between 0.7 and 1.0. This is because the RANSAC part of the segmentation, which often takes up a big part of the total points and usually agrees with the ground truth very well, is also included to the calculation of the total F1-measure of a solution. But the results are not biased; the near-perfect ransac results just dilute the overall f1-measure, so even small differences are expressive. For now it seems that over all knn seems to be slightly better than the boundary complex approach, whereas radius search is clearly the worst. But we want to go more into detail now.

Approach	Mean F1-measures
knn	0.87839
radius	0.86041
bc	0.87486

Table 5.1: F1-Measures, averaged for all input combinations B.2B.3B.4B.5

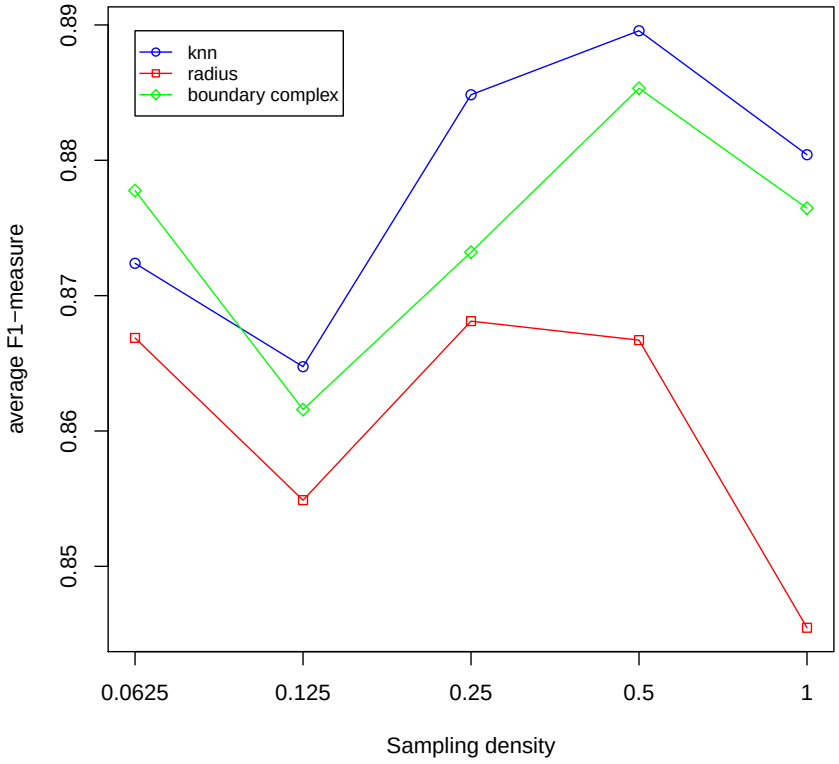

Figure 5.1: Average F1-measures over all point clouds and all noise levels grouped by the different sampling densities.

F1 vs. sampling rate

If we take a look at the performance of the different approaches in relation to the sampling densities (see figure 5.1), we see that all approaches show similar tendencies. Just the radius search drops off very early when it comes to higher sampling densities. The Boundary Complex results more or less copy the trend of the knn results on a slightly lower level; however it should be noted that at the lowest resolutions, the Boundary Complex performs a bit better.

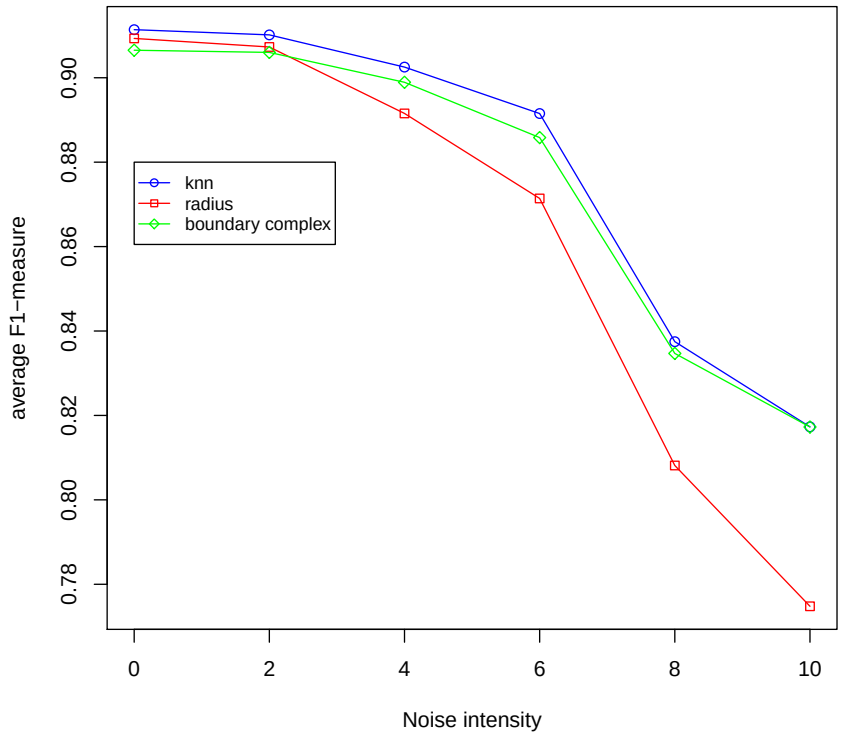

Figure 5.2: Average F1-measures over all point clouds and all sampling densities grouped by the different noise levels.

F1 vs. noise level

Regarding the results for different noise levels (figure 5.2), we get a similar picture: while all the F1-measures are decreasing with increasing noise, Boundary Complex and knn stay level and radius search is dropping more rapidly.

Boundary Complex vs. knn with equal number of neighbors

During all the tests, the rounded average number of neighbors per seed point in the Boundary Complex was 7. So an interesting question would be, how well the knn approach performs if k is set to 7. As the previous results showed, Boundary Complex and knn react to noise almost equally, therefore we restricted the experiments on this subject to zero noise test data. Figure 5.3 again shows the performances in relation to the sampling densities. Now we can see large differences; at an equal number of neighbors, the Boundary Complex is obviously superior to knn.

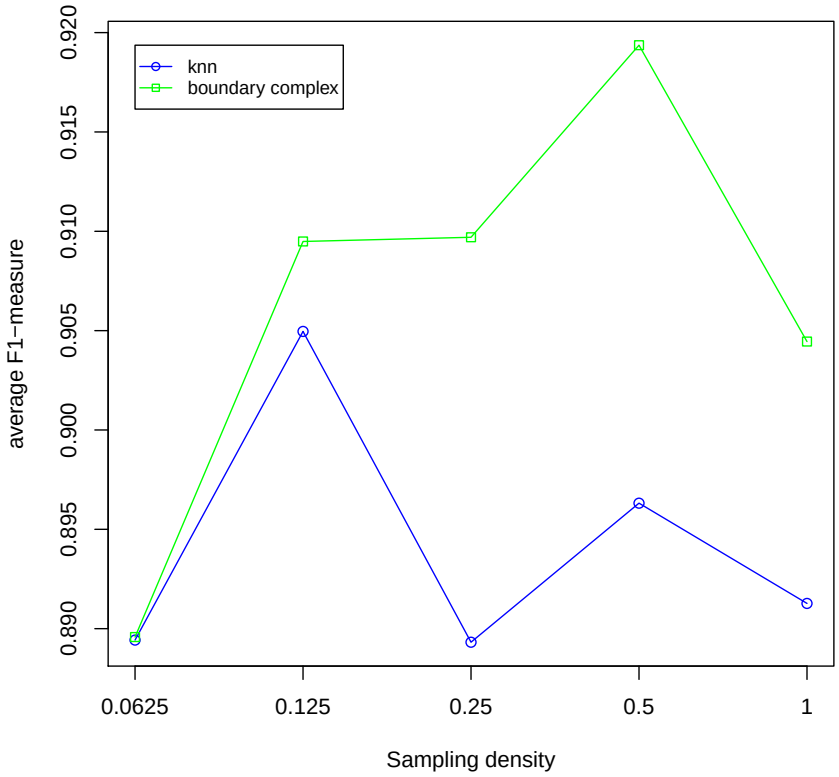

Figure 5.3: Average F1-measures over all point clouds (without noise) grouped by the different sampling densities, for BC and knn with $k = 7$

Visual comparison

Figure 5.4 shows two examples of the neighborhood of seedpoints delivered by the different neighbor finding approaches. Note that these are two dimensional images of a three dimensional cloud. While the first example (first row) shows a planar region, the second example is placed in a more three dimensional structure.

Now we give some examples of clusterings done by our application, using the different neighborhood approaches. Note that if the noise level is greater than zero, the F1 measures stated in the image captions may not agree with those listed in appendix B. This is because those images have been created later, and as the applied noise is truly random (within defined boundaries), the input clouds will vary in different executions of the program.

Figure 5.5 shows the "couch" scene in a resolution of 20 000 points. All three approaches did a similarly good job; according to the F1-measures, in this case actually the radius search

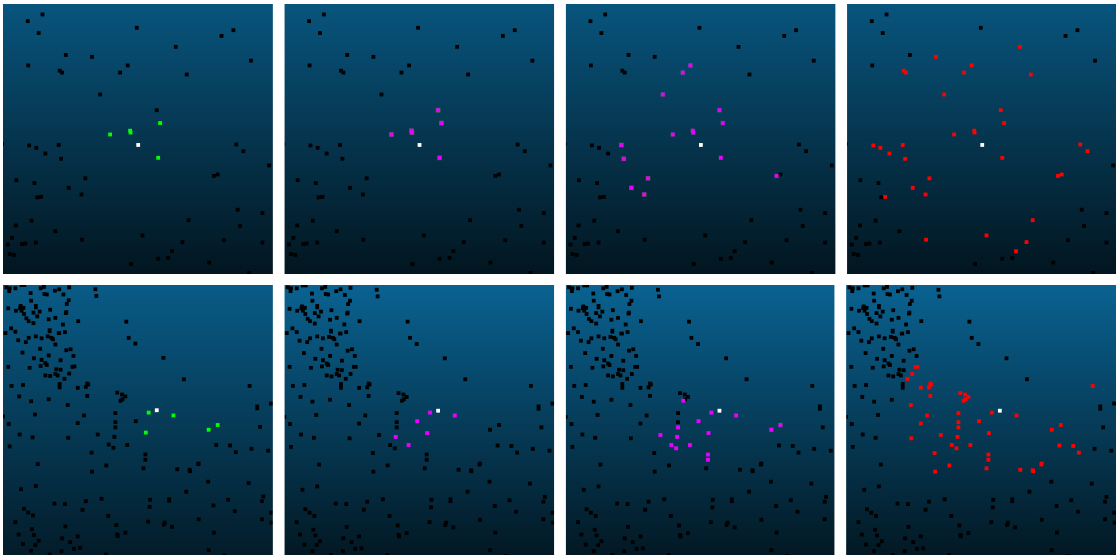

Figure 5.4: Neighborhoods of two seed points (white), using the optimal parameters defined in 5.1 (except second column from left). From left to right: Boundary Complex, knn with k=7, knn with k=16, radius search. In the radius search neighborhood of the second row black points can be seen that look like lying inside the radius, but are in fact too far behind the seed point.

delivers the best results. In figure 5.6 we see the same scene downsampled to 2500 points. For a human observer, the visible point cloud is no longer perceived as the underlying scene; also the F1-measures decreased, but not as dramatically as could be expected. Especially the radius search, again, delivers the best results, which can also be comprehended visually (when taking a close look).

Figure 5.7 shows the "room" scene in its full sampling density of 16841 points. Artificial noise of level 10 (the highest that we used) was applied to the input cloud. If we look at the resulting clouds, the subjective impression would be that BC delivers the best results, closely followed by knn, while radius search delivers the worst result. In the left part of the scene, where we see an office chair and some other objects, BC produces the same amount of clusters as the ground truth, radius search puts it all together in one cluster, and knn is somewhere in the middle. But the F1-measures contradict these observations: First, the radius search does not receive a much lower value than the other solutions. This could be explained by the fact that in this scene there is a very large planar component (the floor) which weights much more than all the rest of the scene. Second, knn is actually rated higher than BC. This might be because of those little black (which means 'does not belong to a cluster') regions at the feet of the office chair. Those unclustered regions occur if detected clusters are smaller than a defined minimum cluster size (see Algorithm 3.1). This example shows the limitations of automated numerical comparison techniques like the F1-measure for unsupervised clustering.

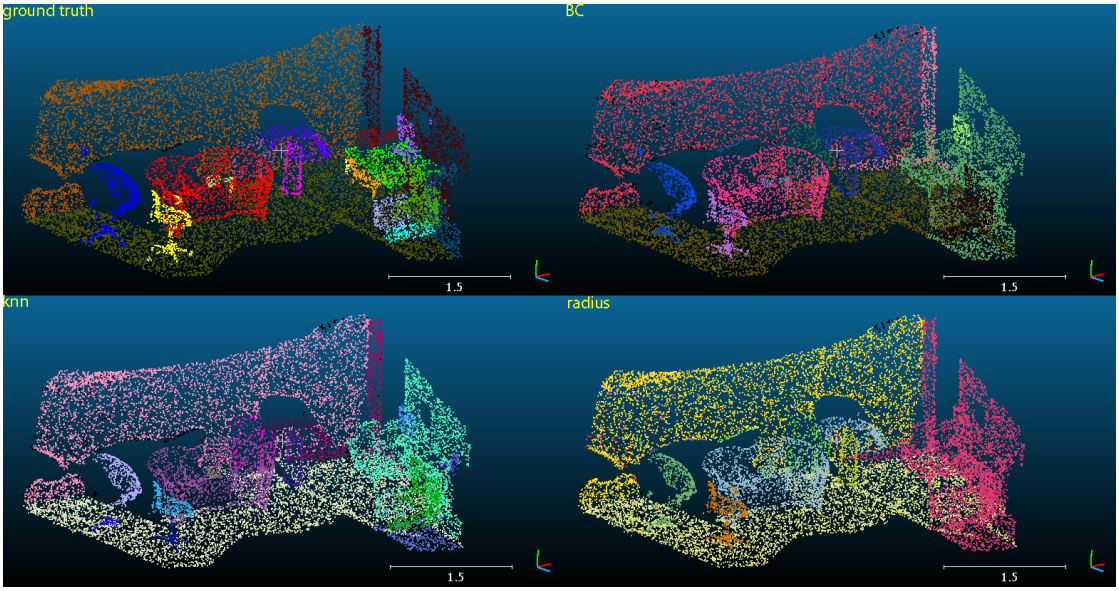

Figure 5.5: "couch", 20 000 points, noise level 0; top left: ground truth, top right: BC, bottom left: knn, bottom right: radius; F1-measures: BC: 0.8510, knn: 0.8498, radius: 0.8718

In figure 5.8 we see a rather trivial scene ("table") which serves as an example of how the choice of k in fact influences the results. In the bottom left picture, where k=7, the right leg of the table is divided into two parts; the result seen in the bottom right picture, where k=16, is more smooth and therefore in this case more similar to the ground truth.

Figure 5.9 shows the "outdoor" scene at a very sparse sampling density (in relation to its actual complexity) of 5000 points with maximum noise, which is a quite extreme setting. As the statistics above showed, radius search is quite weak at high noise levels; this is also visible in this example, where it achieves an F1-measure of only 0.5869. BC and knn perform much better, on a quite equal level.

Figure 5.10 is just another example in which BC performs much better than the other two approaches.

5.3 Conclusion

The overall conclusion that we have to make is that there is no clear "winner" among the examined approaches. The results always depend on the structure and complexity of the input scene, the sampling density and the level of noise. So it can be stated that the Boundary Complex as another good approach for neighbor picking, which delivers equally good results as existing techniques. We want to discuss now some special properties of the Boundary Complex neighborhood in context of segmentation, that have emerged from our studies.

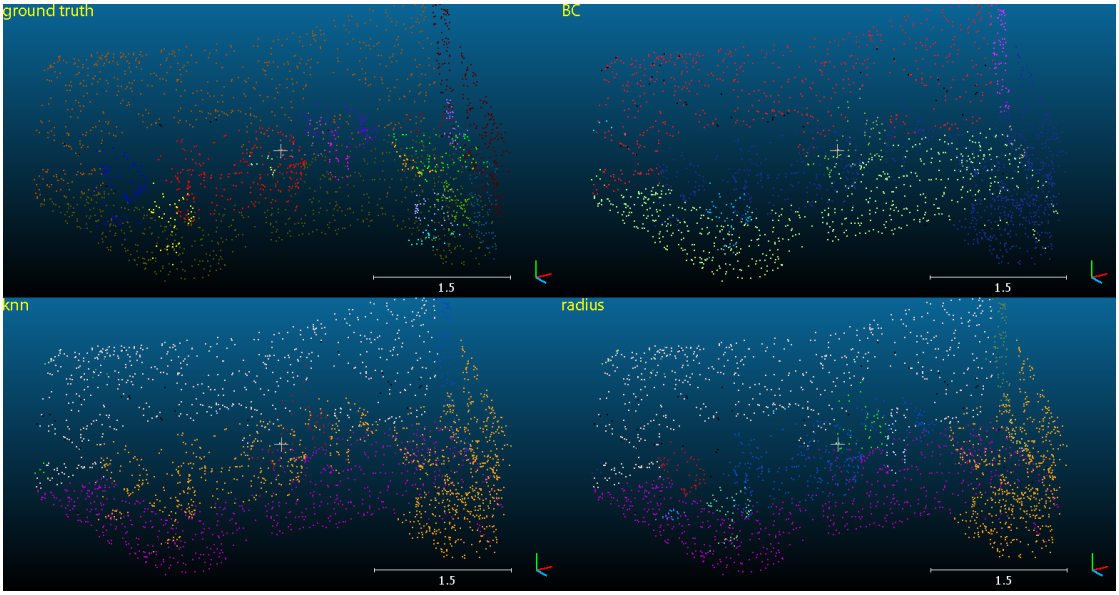

Figure 5.6: "couch", 2500 points, noise level 0; top left: ground truth, top right: BC, bottom left: knn, bottom right: radius; F1-measures: BC: 0.7882, knn: 0.7849, radius: 0.8214; At the bottom left corner of the scene, the radius search detected structures that were just mixed with everything else in the other approaches.

- A big advantage of the Boundary Complex is that it does not require parametrization per se. We have introduced an additional parameter, but it is only used as a threshold to handle extreme situations and does not influence the principal structure of the Boundary Complex. At knn and radius search, the parameters have a much stronger influence on the results and should be adjusted for each input dataset, which means additional manual overhead.

- The Boundary Complex is one connected component. This means that, if we do not set distance thresholds and/or perform a plane segmentation as a first clustering step, we will receive one big cluster as a result. This can be seen as a disadvantage. On the other hand, if we perform a plane segmentation, objects that lie on different sides of the plane can never falsely be regarded as belonging to the same cluster, because the connections between them have been "cut". For radius search and knn, a plane would be no "barrier".

- As for knn, clouds with non-uniform sampling densities are not a problem for the Boundary Complex. If a region is sampled more sparsely than the rest of the cloud, even the threshold for outlier detection scales accordingly, as it is calculated based on the distances of the neighbors of the current seed point.

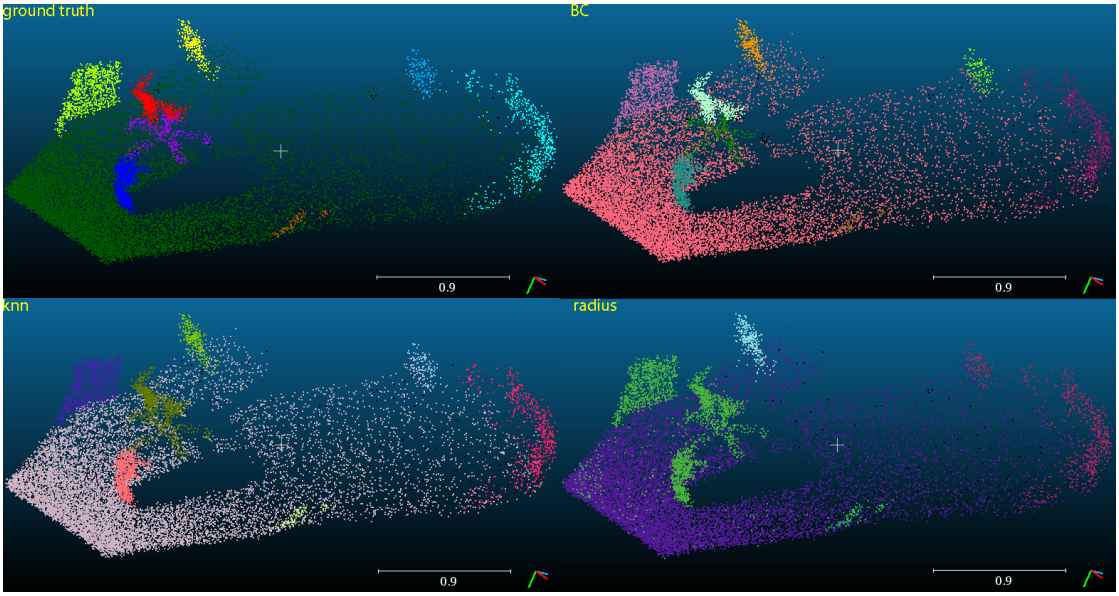

Figure 5.7: "room", 16 841 points, noise level 10; top left: ground truth, top right: BC, bottom left: knn, bottom right: radius; F1-measures: BC: 0.9446, knn: 0.9483, radius: 0.9427

- The Boundary Complex is equally robust to noise and sparse sampling as knn. In very sparse clouds, the Boundary Complex even delivers slightly better results.
- Another advantage of the Boundary Complex over knn is the number of neighbors per seedpoint needed to deliver equally good results. We have shown that for similar results, we have to set k to 16. The Boundary Complex however has an average of 7 neighbors per seed point, that is, less than half of knn.

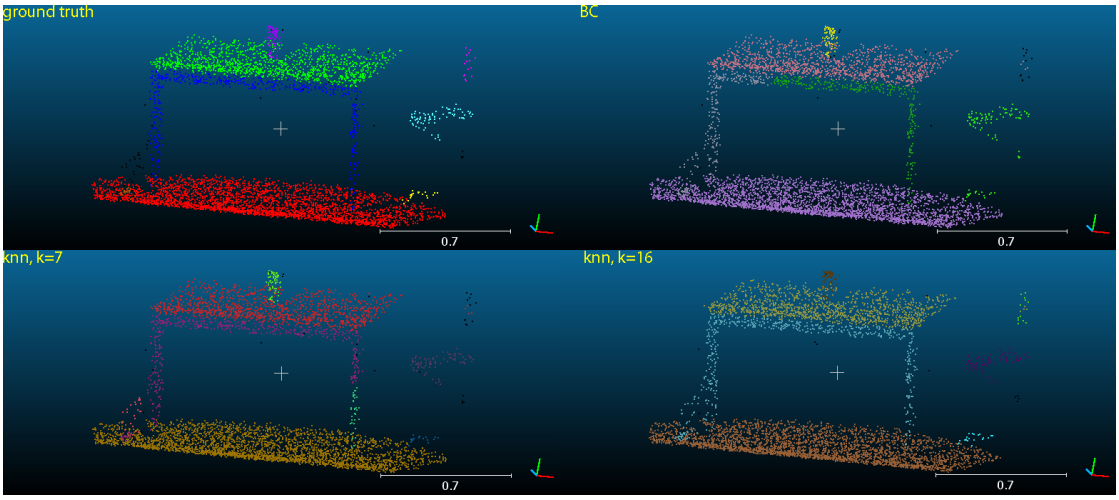

Figure 5.8: "table", 5000 points, noise level 0; top left: ground truth, top right: BC, bottom left: knn with k=7, bottom right: knn with k=16; F1-measures: BC: 0.9580, knn with k=7: 0.9638, knn with k=16: 0.9833

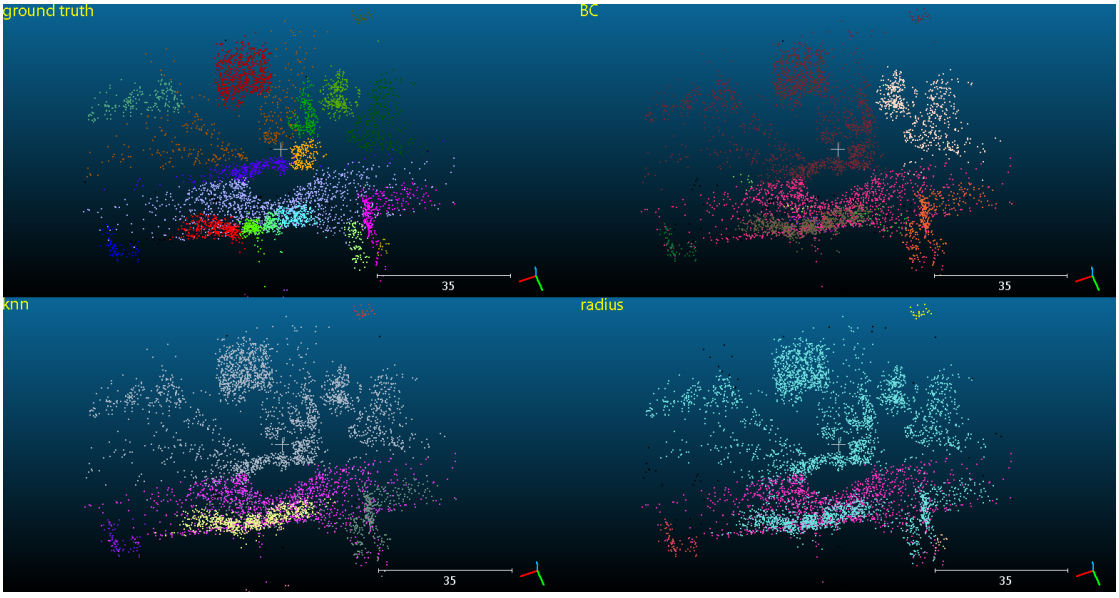

Figure 5.9: "outdoor", 5000 points, noise level 10; top left: ground truth, top right: BC, bottom left: knn, bottom right: radius; F1-measures: BC: 0.7972, knn: 0.7643, radius: 0.5716

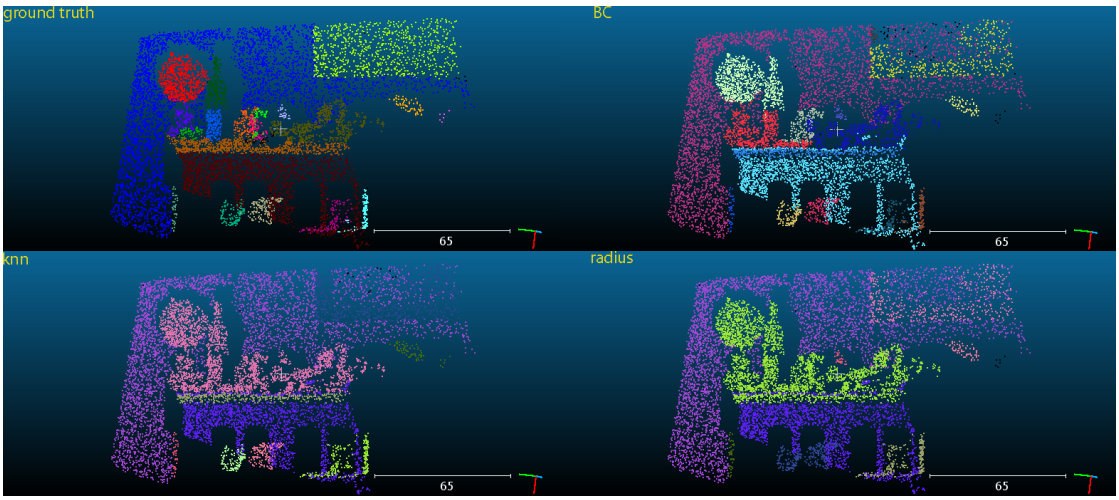

Figure 5.10: “cabinet”, 10 000 points, noise level 4; top left: ground truth, top right: BC, bottom left: knn, bottom right: radius; F1-measures: BC: 0.8918, knn: 0.8798, radius: 0.8859

CHAPTER 6

Summary and future work

We have shown that the Boundary Complex is a good alternative for existing neighbor finding approaches in the are of point cloud segmentation. It delivers similarly good results and has some other advantages, such as not needing parametrization, and not needing a great amount of neighbors per seedpoint to deliver good results.

Future work

Thinking of applications where we want to perform a segmentation and construct a surface, the Boundary Complex could be used for both of these tasks; therefore it has to be constructed only once. So as the Boundary Complex would have to be constructed anyway and we could create the Boundary Complex in a datastructure that explicitly stores the neighbors of each point (some kind of graph structure), the process of finding neighbors could be done in constant time.

Another interesting application of the Boundary Complex in the area of point cloud processing could be the computation of surface normals. In general, the computation of good normals from a point cloud is a difficult and much discussed problem [2]. In contrary, the computation of normals on a mesh is a rather trivial task: If we want to calculate the normal of a vertex (point), we just need to calculate the normals of all adjacent faces, which is simple vector analysis, and then average them. Now in principle, the Boundary Complex provides a (non-manifold) mesh on the underlying point cloud; actually a mesh that represents the boundary of a point cloud pretty well. So theoretically, the Boundary Complex could be a very good tool for computing surface normals. Those normals then could be used for applications like curvature based segmentation techniques. We have not examined that aspect in this work, but it could be a promising basis for future research.

APPENDIX A

Input clouds

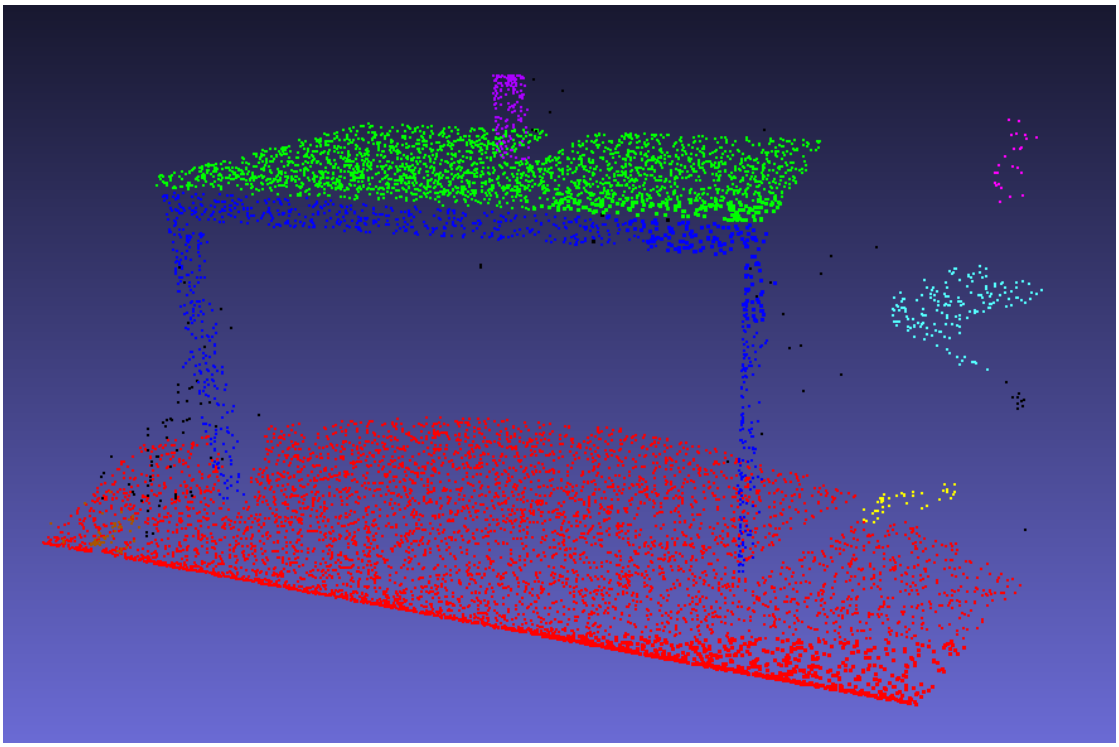

Figure A.1: “table”, 10 000 points

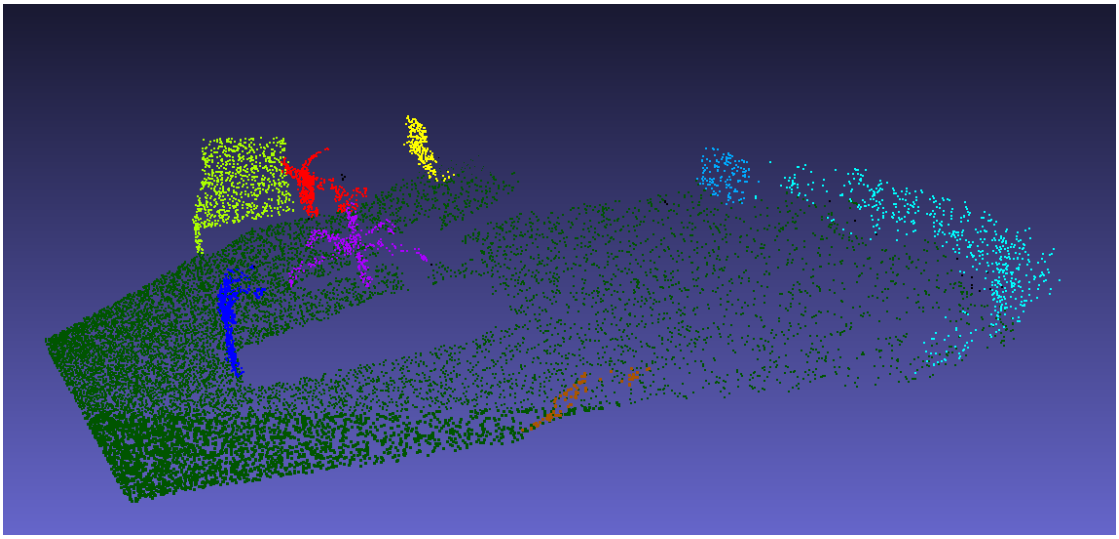

Figure A.2: “room”, 16 841 points

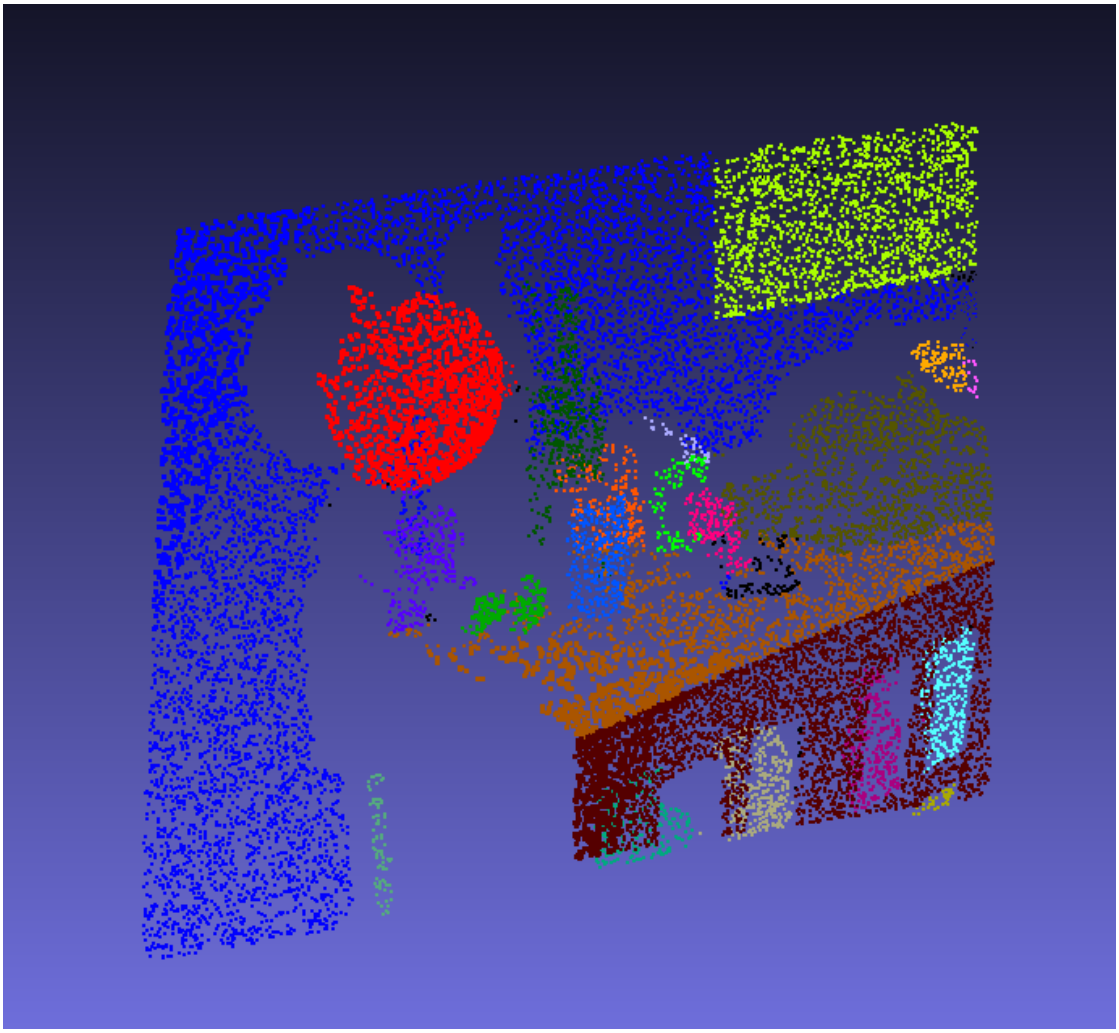

Figure A.3: “cabinet”, 20 000 points

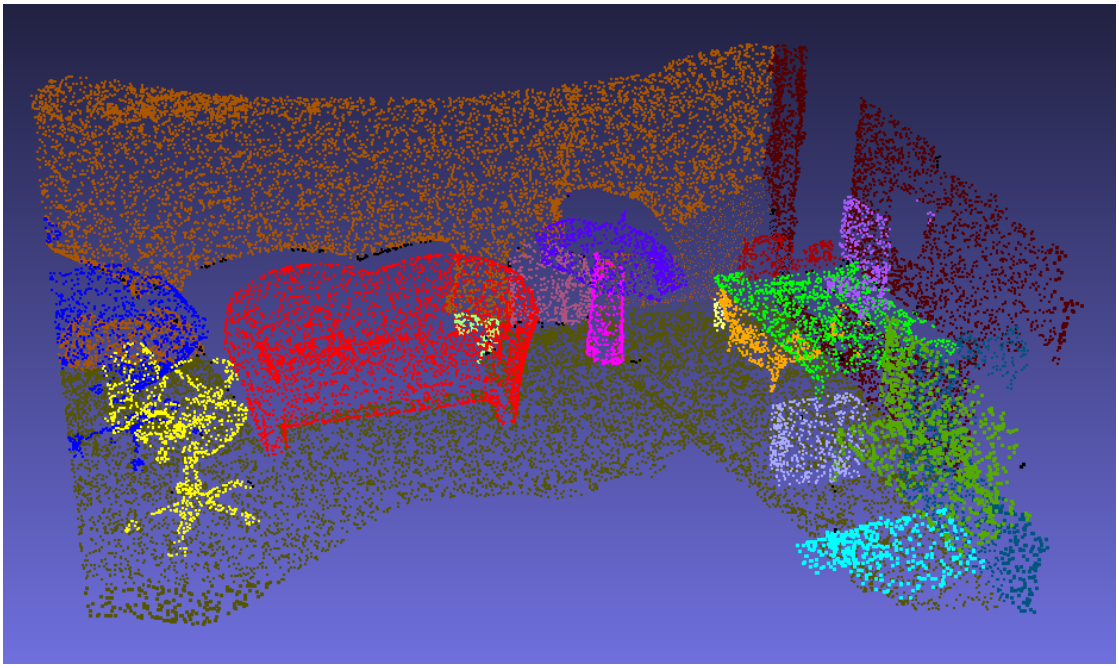

Figure A.4: “couch”, 40 000 points

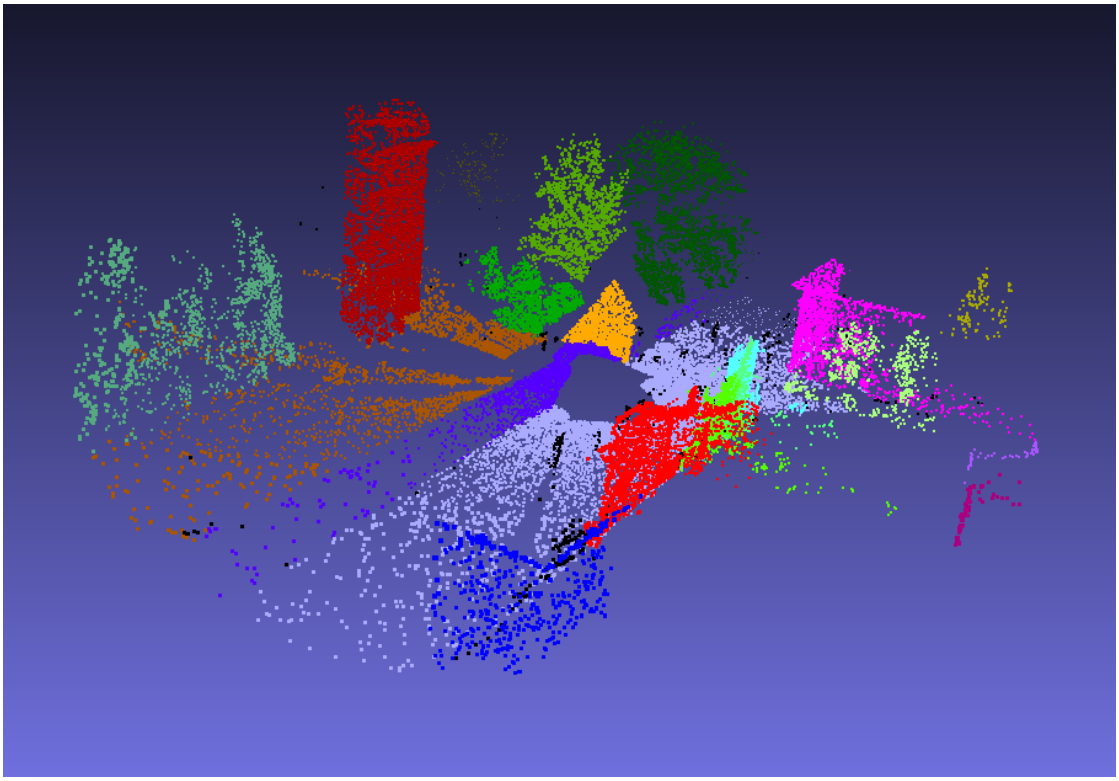

Figure A.5: “outdoor”, 40 000 points

APPENDIX B

Numbers

input file	**best** k	**best** r	**best** c
cabinet_1000pt.pcd	15	76	2.2
cabinet_2500pt.pcd	12	41	1.6
cabinet_5000pt.pcd	11	26	4.4
cabinet_10000pt.pcd	11	21	2.2
cabinet_20000pt.pcd	25	16	3
couch_2500pt.pcd	8	51	1.2
couch_5000pt.pcd	11	46	1.8
couch_10000pt.pcd	16	36	1.6
couch_20000pt.pcd	30	36	1.6
couch_40000pt.pcd	20	36	3
room_1000pt.pcd	21	71	2.2
room_2000pt.pcd	9	71	1.8
room_4000pt.pcd	15	56	2.4
room_8000pt.pcd	23	46	2.8
room_16841pt.pcd	27	36	2.4
table_500pt.pcd	10	91	1.4
table_1000pt.pcd	12	76	1.8
table_2500pt.pcd	21	96	1.8
table_5000pt.pcd	12	41	1.6
table_10000pt.pcd	16	36	1.6
outdoor_2500pt.pcd	9	56	1.2
outdoor_5000pt.pcd	18	36	2.8
outdoor_10000pt.pcd	18	41	1.8
outdoor_20000pt.pcd	27	36	1.8
outdoor_40000pt.pcd	20	21	1.6
median	16	41	1,8

Table B.1: Best parameters for each input file

input file	noise	knn	radius	bc
cabinet_1000pt.pcd	0	0.859717	0.8346	0.852034
cabinet_2500pt.pcd	0	0.880519	0.892892	0.874599
cabinet_5000pt.pcd	0	0.864831	0.864357	0.856511
cabinet_10000pt.pcd	0	0.880037	0.883018	0.893524
cabinet_20000pt.pcd	0	0.880303	0.872286	0.87527
couch_2500pt.pcd	0	0.784861	0.821388	0.788218
couch_5000pt.pcd	0	0.826289	0.834904	0.830071
couch_10000pt.pcd	0	0.863854	0.863056	0.856731
couch_20000pt.pcd	0	0.849753	0.871775	0.851041
couch_40000pt.pcd	0	0.828681	0.857214	0.831954
room_1000pt.pcd	0	0.978412	0.963445	0.975383
room_2000pt.pcd	0	0.986513	0.974061	0.981148
room_4000pt.pcd	0	0.987753	0.984087	0.97801
room_8000pt.pcd	0	0.982744	0.987252	0.984777
room_16841pt.pcd	0	0.981589	0.98935	0.978051
table_500pt.pcd	0	0.979331	0.944098	0.961345
table_1000pt.pcd	0	0.985631	0.977803	0.984689
table_2500pt.pcd	0	0.985406	0.980437	0.984201
table_5000pt.pcd	0	0.984568	0.984847	0.981838
table_10000pt.pcd	0	0.984584	0.984875	0.957217
outdoor_2500pt.pcd	0	0.891442	0.877767	0.870909
outdoor_5000pt.pcd	0	0.895682	0.880326	0.876932
outdoor_10000pt.pcd	0	0.879586	0.87294	0.873043
outdoor_20000pt.pcd	0	0.881921	0.863975	0.885653
outdoor_40000pt.pcd	0	0.880731	0.872144	0.87976
cabinet_1000pt.pcd	2	0.847699	0.837511	0.855475
cabinet_2500pt.pcd	2	0.879031	0.890503	0.869888
cabinet_5000pt.pcd	2	0.852245	0.853375	0.844836
cabinet_10000pt.pcd	2	0.885253	0.886001	0.887773
cabinet_20000pt.pcd	2	0.884385	0.877125	0.875491
couch_2500pt.pcd	2	0.787166	0.823336	0.82504
couch_5000pt.pcd	2	0.852759	0.85227	0.837722
couch_10000pt.pcd	2	0.8584	0.855879	0.832844
couch_20000pt.pcd	2	0.846055	0.862956	0.846034
couch_40000pt.pcd	2	0.809888	0.822093	0.80691
room_1000pt.pcd	2	0.977313	0.961021	0.977165
room_2000pt.pcd	2	0.983931	0.971963	0.979348
room_4000pt.pcd	2	0.987476	0.985329	0.979013
room_8000pt.pcd	2	0.98102	0.985553	0.97728
room_16841pt.pcd	2	0.981094	0.988903	0.976799

Table B.2: F1-measures part 1

input file	noise	knn	radius	bc
table_500pt.pcd	2	0.979387	0.976122	0.979387
table_1000pt.pcd	2	0.984678	0.976549	0.983603
table_2500pt.pcd	2	0.98662	0.984217	0.963964
table_5000pt.pcd	2	0.98524	0.985553	0.959194
table_10000pt.pcd	2	0.984382	0.98467	0.98008
outdoor_2500pt.pcd	2	0.893717	0.884055	0.892762
outdoor_5000pt.pcd	2	0.900009	0.882603	0.897228
outdoor_10000pt.pcd	2	0.868986	0.863019	0.868873
outdoor_20000pt.pcd	2	0.880263	0.862432	0.871378
outdoor_40000pt.pcd	2	0.876863	0.829084	0.881958
cabinet_1000pt.pcd	4	0.860013	0.827806	0.856911
cabinet_2500pt.pcd	4	0.816476	0.819488	0.818289
cabinet_5000pt.pcd	4	0.85923	0.858079	0.850449
cabinet_10000pt.pcd	4	0.864473	0.872172	0.879416
cabinet_20000pt.pcd	4	0.886888	0.88128	0.883632
couch_2500pt.pcd	4	0.78814	0.818229	0.791661
couch_5000pt.pcd	4	0.818781	0.827899	0.79367
couch_10000pt.pcd	4	0.825995	0.828192	0.796154
couch_20000pt.pcd	4	0.834973	0.856198	0.833863
couch_40000pt.pcd	4	0.826091	0.792934	0.825383
room_1000pt.pcd	4	0.979512	0.961404	0.974748
room_2000pt.pcd	4	0.98399	0.976833	0.982858
room_4000pt.pcd	4	0.987224	0.985239	0.983713
room_8000pt.pcd	4	0.97825	0.982822	0.97349
room_16841pt.pcd	4	0.98338	0.987736	0.980104
table_500pt.pcd	4	0.979291	0.976031	0.979291
table_1000pt.pcd	4	0.983131	0.974041	0.980575
table_2500pt.pcd	4	0.986756	0.983418	0.975087
table_5000pt.pcd	4	0.983939	0.984251	0.97594
table_10000pt.pcd	4	0.98249	0.982926	0.978221
outdoor_2500pt.pcd	4	0.893096	0.867879	0.894588
outdoor_5000pt.pcd	4	0.900043	0.881516	0.885381
outdoor_10000pt.pcd	4	0.828259	0.766293	0.838114
outdoor_20000pt.pcd	4	0.877683	0.80495	0.877946
outdoor_40000pt.pcd	4	0.855044	0.790946	0.86359
cabinet_1000pt.pcd	6	0.855811	0.824881	0.844586
cabinet_2500pt.pcd	6	0.867316	0.872932	0.861638
cabinet_5000pt.pcd	6	0.789786	0.793578	0.772992
cabinet_10000pt.pcd	6	0.82858	0.842236	0.834661
cabinet_20000pt.pcd	6	0.746552	0.763381	0.740709

Table B.3: F1-measures part 2

input file	noise	knn	radius	bc
couch_2500pt.pcd	6	0.782879	0.800775	0.804182
couch_5000pt.pcd	6	0.8096	0.843022	0.82018
couch_10000pt.pcd	6	0.825924	0.827966	0.804399
couch_20000pt.pcd	6	0.802988	0.733091	0.788011
couch_40000pt.pcd	6	0.843439	0.786228	0.828615
room_1000pt.pcd	6	0.96843	0.952512	0.97025
room_2000pt.pcd	6	0.983075	0.971158	0.97975
room_4000pt.pcd	6	0.984477	0.981624	0.980035
room_8000pt.pcd	6	0.97777	0.981798	0.97701
room_16841pt.pcd	6	0.980294	0.985542	0.980394
table_500pt.pcd	6	0.974732	0.941322	0.957009
table_1000pt.pcd	6	0.976213	0.967467	0.97414
table_2500pt.pcd	6	0.9825	0.977632	0.944308
table_5000pt.pcd	6	0.98336	0.983743	0.957984
table_10000pt.pcd	6	0.979481	0.981085	0.97583
outdoor_2500pt.pcd	6	0.864764	0.866763	0.868176
outdoor_5000pt.pcd	6	0.897631	0.875718	0.88817
outdoor_10000pt.pcd	6	0.864777	0.798656	0.863048
outdoor_20000pt.pcd	6	0.883241	0.820277	0.883777
outdoor_40000pt.pcd	6	0.83439	0.611647	0.845787
cabinet_1000pt.pcd	8	0.826076	0.781079	0.816335
cabinet_2500pt.pcd	8	0.851576	0.84842	0.845162
cabinet_5000pt.pcd	8	0.733164	0.758871	0.708761
cabinet_10000pt.pcd	8	0.834366	0.835011	0.82155
cabinet_20000pt.pcd	8	0.750514	0.72111	0.739981
couch_2500pt.pcd	8	0.704613	0.770274	0.726504
couch_5000pt.pcd	8	0.00284115	0.00331991	0.00614553
couch_10000pt.pcd	8	0.797368	0.760634	0.775794
couch_20000pt.pcd	8	0.801476	0.755188	0.792295
couch_40000pt.pcd	8	0.795514	0.722312	0.793393
room_1000pt.pcd	8	0.971616	0.950654	0.971047
room_2000pt.pcd	8	0.977048	0.967961	0.971418
room_4000pt.pcd	8	0.971688	0.969047	0.968744
room_8000pt.pcd	8	0.970107	0.97177	0.957257
room_16841pt.pcd	8	0.972772	0.979915	0.975253
table_500pt.pcd	8	0.975553	0.939573	0.942871
table_1000pt.pcd	8	0.980852	0.970944	0.981493
table_2500pt.pcd	8	0.966362	0.963801	0.965833
table_5000pt.pcd	8	0.961231	0.961677	0.958787
table_10000pt.pcd	8	0.964008	0.965817	0.939969

Table B.4: F1-measures part 3

input file	noise	knn	radius	bc
outdoor_2500pt.pcd	8	0.852365	0.860401	0.86025
outdoor_5000pt.pcd	8	0.835527	0.786563	0.833678
outdoor_10000pt.pcd	8	0.854711	0.809433	0.874622
outdoor_20000pt.pcd	8	0.812984	0.58394	0.82332
outdoor_40000pt.pcd	8	0.772597	0.565718	0.81674
cabinet_1000pt.pcd	10	0.722946	0.648914	0.714786
cabinet_2500pt.pcd	10	0.765189	0.7762	0.767009
cabinet_5000pt.pcd	10	0.737482	0.737575	0.718313
cabinet_10000pt.pcd	10	0.756859	0.756066	0.743353
cabinet_20000pt.pcd	10	0.754705	0.715765	0.742193
couch_2500pt.pcd	10	0.694552	0.722147	0.727803
couch_5000pt.pcd	10	0.604891	0.678385	0.640174
couch_10000pt.pcd	10	0.783767	0.706306	0.7595
couch_20000pt.pcd	10	0.72058	0.677789	0.710148
couch_40000pt.pcd	10	0.766813	0.669716	0.728906
room_1000pt.pcd	10	0.92854	0.899856	0.919308
room_2000pt.pcd	10	0.930052	0.918924	0.927598
room_4000pt.pcd	10	0.903463	0.900625	0.896542
room_8000pt.pcd	10	0.946881	0.945058	0.934599
room_16841pt.pcd	10	0.948787	0.951452	0.947472
table_500pt.pcd	10	0.945014	0.90736	0.91674
table_1000pt.pcd	10	0.972462	0.965083	0.971073
table_2500pt.pcd	10	0.895474	0.894952	0.857408
table_5000pt.pcd	10	0.885788	0.895358	0.880091
table_10000pt.pcd	10	0.875915	0.885096	0.868078
outdoor_2500pt.pcd	10	0.624695	0.765028	0.818221
outdoor_5000pt.pcd	10	0.810841	0.586936	0.803848
outdoor_10000pt.pcd	10	0.831709	0.634922	0.824554
outdoor_20000pt.pcd	10	0.824768	0.584626	0.817719
outdoor_40000pt.pcd	10	0.800214	0.545475	0.796123

Table B.5: F1-measures part 4

Bibliography

[1] Ben Bongalon. http://borglabs.com/blog/create-point-clouds-from-kinect. Accessed: 2013-01-31.

[2] Cecilia Chao and Chen Ioannis Stamos. Range image segmentation for modeling and object detection in urban scenes.

[3] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[4] Point Cloud Library. http://www.pointclouds.org/. Accessed: 2013-02-01.

[5] Ricard Marxer and Hendrik Purwins. An f-measure for evaluation of unsupervised clustering with non-determined number of clusters, 2008.

[6] MeshLab. http://meshlab.sourceforge.net/. Accessed: 2013-01-31.

[7] J.F. O'Callaghan. An alternative definition for "neighborhood of a point". *Computers, IEEE Transactions on*, C-24(11):1121 – 1125, nov. 1975.

[8] Point Cloud Repository of PCL. http://svn.pointclouds.org/data/. Accessed: 2013-01-31.

[9] S. Ohrhallinger. *The Intrinsic Shape of Point Clouds*. PhD thesis, Concordia University Montreal, 2012.

[10] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

[11] Horng-Lin Shieh. Semi-supervised clustering based on k-nearest neighbors. In *Digital Manufacturing and Automation (ICDMA), 2012 Third International Conference on*, pages 759 –762, 31 2012-aug. 2 2012.