2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

A Clustering Method for Efficient Segmentation of 3D Laser Data

Klaas Klasing Dirk Wollherr Martin Buss
Institute of Automatic Control Engineering
Technische Universität München
80290 Munich, Germany
{kk,dw,mb}@tum.de

Abstract— **In this paper we present a novel method for the efficient segmentation of 3D laser range data. The proposed algorithm is based on a radially bounded nearest neighbor strategy and requires only two parameters. It yields deterministic, repeatable results and does not depend on any initialization procedure. The efficiency of the method is verified with synthetic and real 3D data.**

I. Introduction

Over the last decade the advent of affordable laser range finders has enabled mobile robotics to shift towards a new level of environment perception. Localization, map building and obstacle avoidance are nowadays almost exclusively performed based on 2D range data. While there exists an overwhelming body of work on building meaningful 2D representations (*occupancy grids*, *line-based maps*) and 2.5D representations (*traversability maps*) for navigation, comparatively few researchers have addressed the problem of dealing with 3D laser data. There are several reasons for this. First of all, acquiring meaningful 3D range data is a problem of its own. The number, position and actuatability of laser scanners mounted on a mobile robot greatly influences the quality of the acquired data. Secondly, the amount of data obtained from a 3D sweep is generally several orders of magnitude larger than that of a simple 2D scan. This requires efficient algorithms and data structures for processing. Finally, the segmentation, extraction and interpretation of geometry of 3D laser data is not as straight-forward as e.g. in a 2D occupancy grid.

In this paper a method for the efficient segmentation of 3D laser range data is proposed. We use an agglomerative nearest neighbor clustering algorithm to segment the raw data into meaningful portions and filter noise. The obtained clusters may then be processed by further supervised or unsupervised classification algorithms or be augmented with information obtained from other sensors, e.g. cameras or other laser scanners. The main advantage of our method lies in its speed and simplicity. In this context it can be seen as a preprocessing step to more sophisticated clustering algorithms. The remainder of this paper is structured as follows: In Section II the motivation for this work within the *Autonomous City Explorer* Project is briefly explained. Section III briefly touches on 3D laser scanning setups and data structures and then provides an overview of clustering algorithms. Section IV explains the proposed framework in detail. In Section V the performance of the method is evaluated with synthetic and real data. Section VI concludes the paper and gives an outlook on future work.

II. Motivation

The goal of the *ACE* Project [1] is to create a mobile robotic platform that autonomously finds its way to a designated goal location in a crowded urban setting without the use of GPS or map information. *ACE* is required to only rely on directions obtained through interaction with pedestrians. Ideally, a semantic understanding of the environment would allow for interpreting these directions in the correct context and derive low-level navigation actions. However, apart from the obvious challenges of outdoor localization and map-building, one of the crucial prerequisites for reaching a semantic level is the robust recognition of objects[1] in the vicinity of the robot. *ACE* is equipped with several laser scanners and a powerful vision system, see Figure 1 and [1] for details. Even before the challenge of classifying the information from the laser scanners, we are faced with the more fundamental problem of segmenting meaningful portions from a set of 3D points. Thus, in this paper the focus lies on the issue of robustly segmenting a 3D laser point cloud acquired by a single laser scanner.

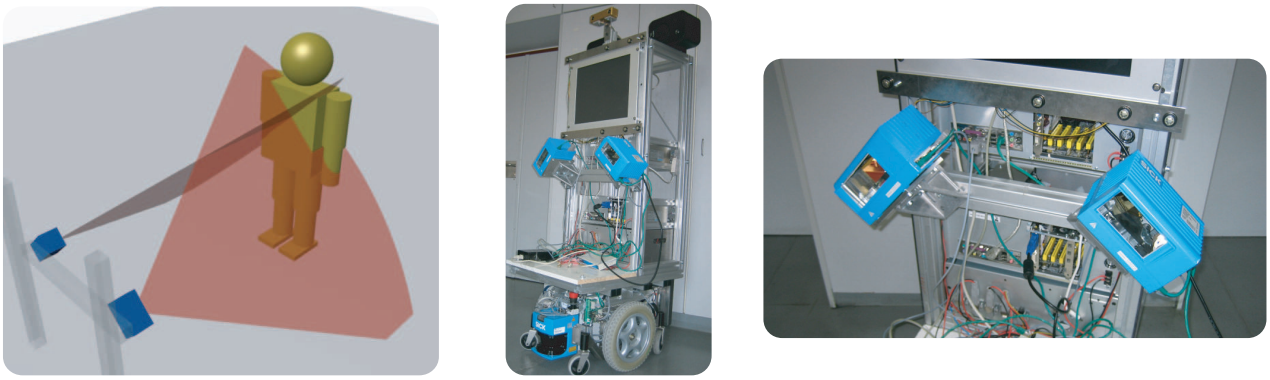
III. State of the Art

This section gives some brief remarks on 3D laser range scanning hardware setups and suitable data structures as well as a non-exhaustive overview of clustering methods.

A. 3D Laser Range Data Acquisition

The acquisition of 3D range data is common practice in the field of civil engineering, where powerful LIDAR devices are employed to capture high-resolution data of buildings and structures. While delivering the highest accuracy available, these devices are not well suited for mobile robotics because they have to be in a fixed position and are not designed for time critical applications. IHigh-definition LIDARs for autonomous vehicles, e.g. the Velodyne HDL-64E[2], are generally not affordable for the average roboticist. The most popular solution to the problem of acquiring 3D laser data is to mount a 2D laser scanner, e.g. a SICK LMS, in some vertical orientation at a fixed position on the robot [1], [2]. Actuated setups with pan, tilt or pan/tilt platforms allow for

[1] humans being a specific kind of object
[2] www.velodyne.com/lidar/whatis.html



(a) (b) (c)

Fig. 1. (a) Schematic configuration of two SICK LMS400 for object detection in the vicinity of *ACE* (b) The *ACE* robot (c) A close-up of the actual sensors mounted on *ACE*

more refined scanning, see e.g. Kurt3D [3] or the Stanford SegBot[4]. An interesting approach is presented in [3], where a SICK LMS200 laser scanner has been modified so as to perform a continuous rolling scan without cable wind-up.

B. Suitable 3D Data Structures

Existing data structures for processing 3D range data can be divided into two types: those that incorporate beam information and those that do not. Occupancy grids as an example of a beam-based method are by far the most prominent data structure for 2D navigation because they can among other things robustly deal with occlusions. Unfortunately, the use of 3D occupancy grids is computationally prohibitive due to the drastic increase in the number of cells that need to be updated. An efficient beam-based method using triangle meshes is desribed in [4]. Neglecting beam information, point clouds merely store the actual spatial coordinates of each measurement, while mesh-based methods try to retain some amount of surface information.

C. 3D Clustering

Data clustering is a vast research discipline that spans – among others – the areas of data mining, image segmentation, object recognition and information retrieval. A comprehensive overview of existing techniques can be found in [5] and an even more exhaustive one in [6]. On a top level, clustering methods can be divided into *hierarchical* and *partitional* techniques. As the name suggests, the former try to capture a hierarchy of levels of similarity in the form of nested groupings represented by *dendrograms*. For the segmentation of laser data into discrete groups of 3D points such a hierarchy is largely irrelevant, which is why hierarchical methods are not considered in this paper[5]. Partitional techniques yield a single partition of the data set and encompass *square error* algorithms (e.g. *k-means*, *ISODATA*), *mixture resolving* algorithms (e.g. *expectation maximization*), *mode seeking* algorithms as well as *graph theoretic* algorithms. Square error algorithms such as k-means and ISODATA are highly popular in the image segmentation community, e.g. for LANDSAT satellite image segmentation [7] These algorithms are unsuitable because they are restricted to finding hyperellipsoidal clusters in the feature space; the laser data may, however, be arbitrarily shaped. Mixture resolving approaches such as *expectation maximization* are deemed similarly inapplicable to the segmentation because they require a model of the underlying distributions and an initial guess of the number of clusters. Iterating over different initial guesses is computationally costly (see [8]) and renders the method unsuitable for online processing. Mode seeking algorithms such as MeanShift [9] and the recently developed MedoidShift [10] are powerful nonparametric methods for robust feature space analysis of (multispectral) images. However, here segmentation is performed on a 2D lattice with different thresholds in the *spatial* and the *range* domain. The application of mode seeking algorithms to a 2D depth map projection of the 3D laser data is not investigated in this paper. Graph-theoretic approaches are the most promising class of algorithms for robust segmentation of 3D laser data. This is because they can capture arbitrarliy shaped clusters using only the concept of local neighborhood. In this paper the focus is on k-nearest neighbor graphs and more refined clustering methods based on these (DBScan [11], *Chameleon* [12]). Section IV explains our proposed method which is based on a simple nearest neighbor strategy. At this point, the application of algorithms such as *Chameleon* has not been investigated. The reader is referred to the discussion in Section VI for an outlook on future work.

In spite of the richness of clustering literature, we are unaware of significant applications in the field of 3D laser data segmentation. Many approaches try to segment points by matching them to geometrical templates such as planes, cylinders, tori etc. [13], [14]. However, such model-based

[3] `www.ais.fraunhofer.de/ARC/kurt3D/`

[4] `robots.stanford.edu/segbot/`

[5] This is with the exception of graph-theoretic algorithms that operate hierarchically but are intended to yield partitions.

matching algorithms can only succeed if the data is already reasonably segmented. For large 3D point clouds obtained from complicated geometry, template matching is comparable to searching a needle in a haystack. In [15] object detection in 3D data is performed by generating depth and reflectance maps of the 3D data through off-screen rendering methods and then running a cascade of (supervised) feature classification algorithms on the 2D representation. While this seems to yield satisfactory classification results it bypasses the problem of segmenting the 3D representation.

IV. Clustering Method

In this section the proposed clustering framework is explained in detail. A description of the segmentation algorithm is given along with remarks on complexity and applicability of the method.

A. Definitions and Notation

Before we start describing the clustering process, it is useful to formalize the problem description. Given n points or patterns in d-dimensional feature space, $\underline{x}_i = [x_{i_1}, \ldots, x_{i_d}]^T,\ i = 1, \ldots, n$, we seek to find m clusters $C_j,\ j = 1, \ldots, m$ such that every cluster contains at least one point, that is $C_j = \{x_{j_1}, \ldots, x_{j_k}\},\ \forall j : k > 0$, and such that all clusters are disjoint, that is $C_i \cap C_j = \emptyset,\ \forall i \neq j$. Furthermore, we assume that some distance metric $\rho(\underline{x}_a, \underline{x}_b)$ exists that accurately reflects the simliarity between any two points $\underline{x}_a$ and $\underline{x}_b$.

For the scope of this paper, we are concerned with a 3-dimensional space in which the distinguishing features of points are their spatial locations along the three cartesian axes. We use a Euclidean distance metric $\rho(\underline{x}_a, \underline{x}_b) = \|\underline{x}_a - \underline{x}_b\|_2$.

In graph notation every point $\underline{x}_i$ is mapped to a corresponding node u_i in a directed graph $G(U, E)$, where $U = \{u_1, \ldots, u_n\}$ represents the set of nodes and $E = \{e_1, \ldots, e_m\}$ the set of edges. An edge is defined as a triplet $e = \{u_1, u_2, d(u_1, u_2)\}$ containing the two connected nodes and their distance $d(u_1, u_2) = \rho(\underline{x}_1, \underline{x}_2)$. We will abbreviate $d(u_1, u_2)$ as $d_{1,2}$ in the following.

B. Clustering Procedure

The clustering method that we employ to segment the laser data can best be described as a *radially bounded nearest neighbor* graph (RBNN). In this graph every node is connected to all neighbors that lie within a predefined radius r. Formally, the set of edges in the RBNN graph is:

$$E_{\text{RBNN}} = \{\{u_i, u_j, d_{i,j}\} \mid d_{i,j} \leq r\},\ \forall u_i, u_j, \in U, i \neq j \tag{1}$$

This is in contrast to the well-known *k-nearest neighbor* graph (kNN) [16], which connects every node to its k nearest neighbors, regardless of distance:

$$E_{\text{kNN}} = \{\{u_i, u_j, d_{i,j}\} \mid u_j \in \text{NN}_k(u_i)\}, \forall u_i \in U, \tag{2}$$

where $\text{NN}_k(u_i)$ represents the outcome of a k-nearest neighbor query for u_i. To clarify how the two methods differ in terms of segmentation performance, consider Figure 2 which shows a basic 2D data set with two clusters and a noisy outlier. Given the correct r as a parameter the RBNN method nicely separates the two clusters and the outlier from each other[6]. While the 1-NN graph produces too many distinct clusters, the 2-NN and 3-NN graphs (and in fact all k-NN graphs with $k > 3$) produce one continuous cluster. The reader may rightly argue that the k-nearest neighbor graph must not be used naïvely but instead be constructed and then cut at a certain distance threshold to yield a proper partitioning. This, however, is the main point of our method: the construction of a suitable k-NN graph causes considerable computational overhead. The main advantage of using the RBNN method is that we do not actually have to perform a nearest neighbor query for every node and there is no graph cutting and rearranging of graph structures involved. In fact it is not even necessary to build a graph structure. The algorithm can briefly be described by the following steps:

1) Step through the list of all points.
2) If the current point has been assigned to a cluster go to the next point.
3) For the current point
   - Find all neighbors within distance r.
   - If any of these neighbors is in a cluster, assign the current point to the same cluster, then assign all neighbors without a cluster to the same cluster.
   - If the current point has been assigned to a cluster and there exist neighbors assigned to different clusters, merge all these clusters.

Algorithm 1 shows a detailed pseudo-code description of the RBNN method.

C. Complexity

The algorithm proposed in the previous subsection is efficient because it avoids maintaining and updating a graph structure and because it does not actually need to perform a nearest neighbor query for every point in the set. As a spatial data structure we have chosen kd-trees because they offer very competitive look-up times for radially bounded queries. An alternative that has not been investigated would be the use of a Delaunay tesselation graph. The construction time of a kd-tree is $O(n\,log(n))$, the expected complexity of a nearest neighbor lookup is $O(log(n))$. While this is close to optimal, nearest neighbor queries still account for the bulk of the running time of NN algorithms. The objective is therefore to reduce the number of needed queries as much as possible. Our algorithm achieves this speedup by skipping all points that have been assigned to a cluster already (Line 4 in Algorithm 1). This means that the more points can be assigned to a cluster in a single nearest-neighbor lookup, the more subsequent lookups can be avoided. Expected computational complexity can be approximated by $O(\frac{n}{k_{\text{average}}} log(n))$, where k_{average} represents the average number of neighbors

[6]We will consider clusters with less than n_{min} points as noise. In this example $n_{min} = 1$.

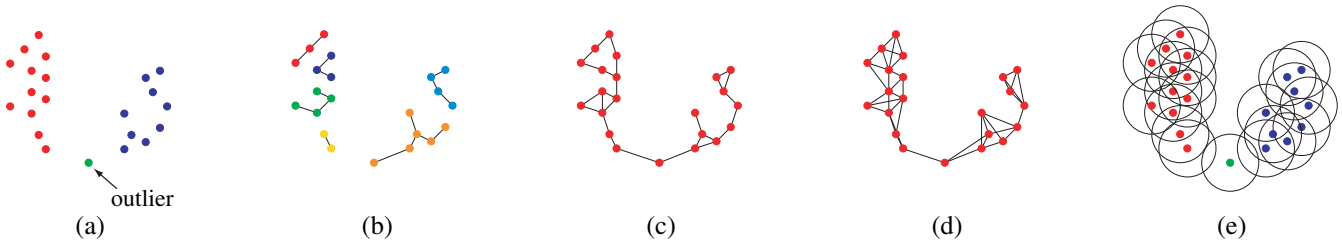


Fig. 2. (a) The data set with 2 clusters and 1 outlier (b) 1-NN graph yielding 6 clusters (c) 2-NN graph yielding 1 cluster (d) 3-NN graph yielding 1 cluster (e) RBNN graph yielding 2 clusters and 1 outlier

Algorithm 1 The RBNN Algorithm

```
1:  RBNN(r, nMin)
2:  for i ← 1, ..., n do
3:      if (hasCluster(u_i)) then
4:          continue;
5:      end if
6:      NN ← findNeighborsInRadius(x_i, r)
7:      for all (u_j ∈ NN) do
8:          if (hasCluster(u_i) ∧ hasCluster(u_j) then
9:              if (clusterOf(u_i) ≠ clusterOf(u_j)) then
10:                 mergeClusters(clusterOf(u_i),clusterOf(u_j));
11:             end if
12:         else
13:             if (hasCluster(u_j)) then
14:                 clusterOf(u_i) ← clusterOf(u_j));
15:             else
16:                 if (hasCluster(u_i)) then
17:                     clusterOf(u_j) ← clusterOf(u_i));
18:                 end if
19:             end if
20:         end if
21:     end for
22:     if (¬hasCluster(u_i)) then
23:         clusterOf(u_i) ← createNewCluster();
24:         for all (u_j ∈ NN) do
25:             clusterOf(u_j) ← clusterOf(u_i);
26:         end for
27:     end if
28: end for
29: for all (C_i ∈ Clusters) do
30:     if (‖C_i‖ < nMin) then
31:         delete(C_i);
32:     end if
33: end for
34: return Clusters;
```

over all queries found within r. It is obvious that r should be chosen as large as possible such that it still yields the same segmentation but maximizes the number of visited neighbors per query.

D. Applicability

A remark must be made concerning the applicability of the method to arbitrary data. For robust segmenation results, the algorithm hinges on the important requirement that the data to be segmented is dense. This means that noisy outliers must be sufficiently spaced in comparison with points that belong to actual objects so that there exists a value for r that separates the two. Fortunately, for 3D data acquired by range finders this is the case. In fact the minimum spacing between scanned points depends only on the angular resolution and the sweeping frequency of the scanner.

V. Results

To verify our analysis and the efficiency of the proposed method the algorithm was benchmarked on synthetic as well as real data from 3D laser scans. This section presents the simulation and experimental scenarios and results.

A. Implementation Details

The RBNN clustering algorithm was implemented in C++, using kd-trees from the ANN library[7] and Coin3D[8] for visualization. All simulation and experiments were run on a 1.8 GHz Intel Pentium with 2GB of RAM.

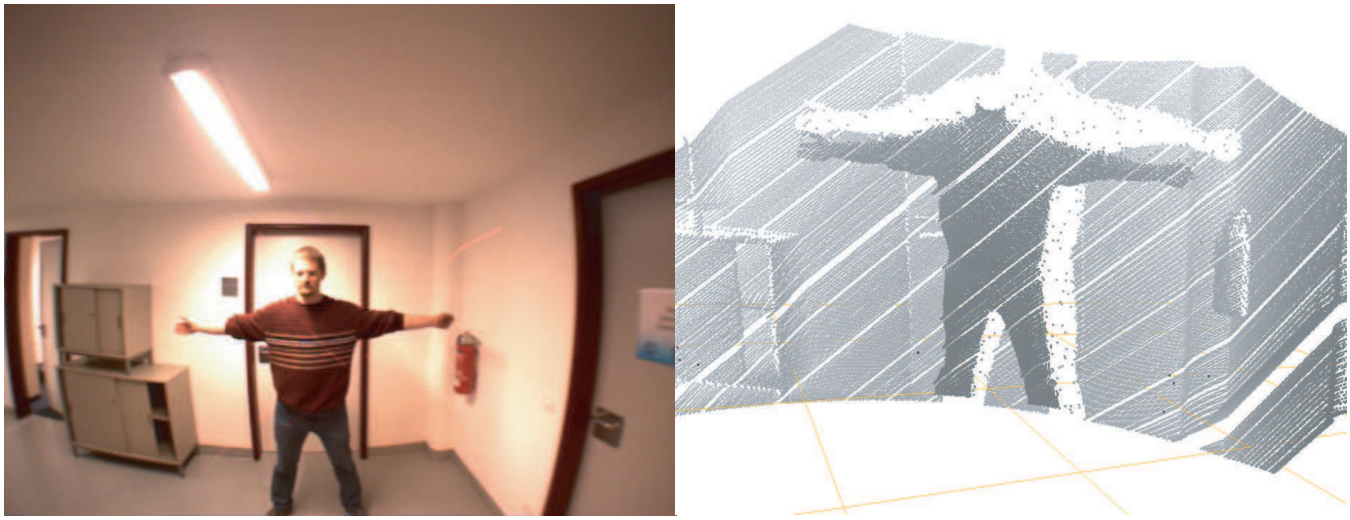
B. Synthetic Data

As a synthetic benchmark scenario we generated point clouds with up to $100,000$ points in a 3D environment of size 8x8x3m. The points were randomly uniformly distributed within flat boxes (similar to object surfaces scanned by a laser) whose number, size, position and orientation was also randomly varied. Figure 3 shows such a data set. For some clustering algorithms, e.g. k-means, retrieving the correct partition in this scenario is a hard problem, because the clusters cannot be captured by ellispoids. Graph-related methods, such as the proposed RBNN approach, however, handle this setup with ease. RBNN correctly identified all clusters in all generated scenarios. To verify our complexity analysis from Section IV-C, we compared a naïve variant of RBNN that performs nearest neighbor queries for every point with the optimized version from Algorithm 1. The number of boxes was fixed at 10, while the total number of points was increased from 10,000 to 50,000 in steps of 10,0000. Figure 4 shows the runtime of both algorithms averaged over 20 runs. Clearly, there is a drastic difference between the exponential complexity of the naïve version and the far better sub-exponential performance of the optimized version. In a second round of simulations we ran the optimized version with different values for the radius r, this time going up to $100,000$ points. Figure 5 confirms our assertion that a larger distance measure yields better performance.

C. Real Data

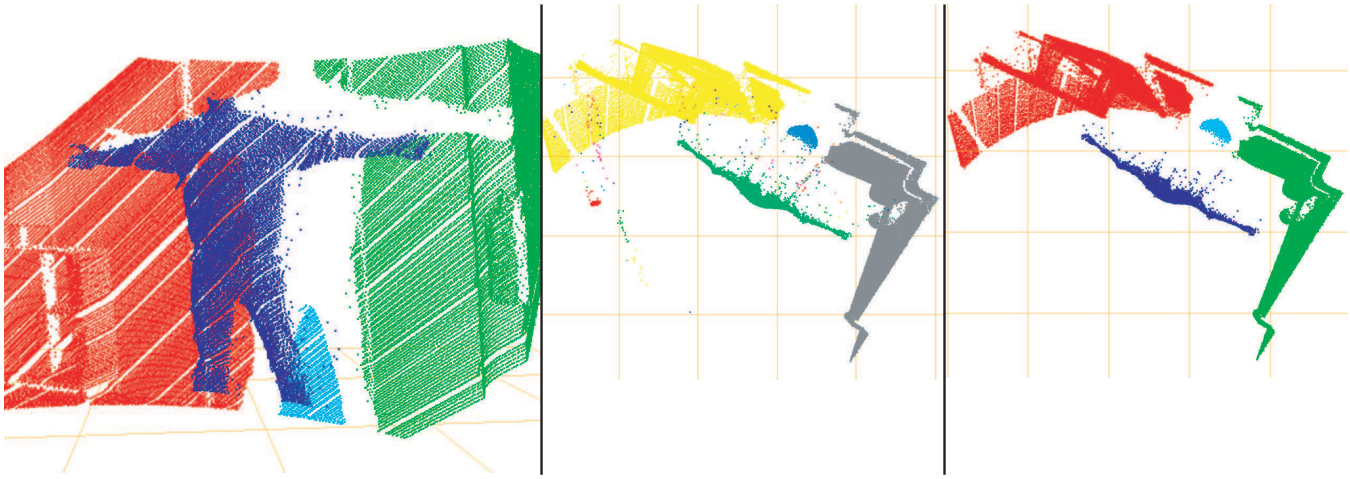
A real 3D laser data set acquired with the *ACE* robot was used to further test the performance of the algorithm. Figure 6 shows the scenario from the view of both the camera and

[7] www.cs.umd.edu/~mount/ANN/
[8] www.coin3d.org



(a) (b)

Fig. 6. (a) The scene as seen by the left robot camera (b) The scene as captured by the left laser in a rotating sweep



(a) (b) (c)

Fig. 7. (a) Front view of the segmented scene (b) Top view before removing noise (c) Top view after removing noise

the left laser scanner. The point cloud of the overall scene contains approximately 150,000 points and was acquired by a 5šecond 150° rotating sweep of the *ACE* robot. With a setting of $r = 1$cm and nMin $= 1000$ the RBNN algorithm yielded the 4 clusters depicted in Figure 7(a) after 468milliseconds. Since the algorithm is deterministic, there is no need to average this result over several runs. The running time of the algorithm on this scenario suggests that it is very well suited for online segmentation of 3D laser data. The segmented portions may then be further processed by more refined algorithms operating at a lower frequency. Notice how a lot of noisy measurements occur around the silhouette of the scanned person. Figures 7 (a) and (b) illustrate nicely how these noisy outliers result in clusters with very few points and can thus easily be filterd.

VI. Discussion

In this paper we have presented a method for online segmentation of 3D laser data. The proposed algorithm is nonparametric in that the only two parameters, r and nMin, depend on the laser scanner and not on the specific data set. Furthermore, the algorithm is deterministic which means that running times and segmentation results are repeatable and do not require initialization. We have analyzed the factors influencing the runtime complexity of the RBNN method and have verified the analysis by a series of benchmark evalutaions with synthetic data. Finally, the efficient performance of the algorithm on a real 3D laser point cloud acquired by a mobile robot has been demonstrated.

There are still several open questions concerning the robustness of the method for sparse laser data. It is expected that the application of more sophisticated graph-theoretic al-

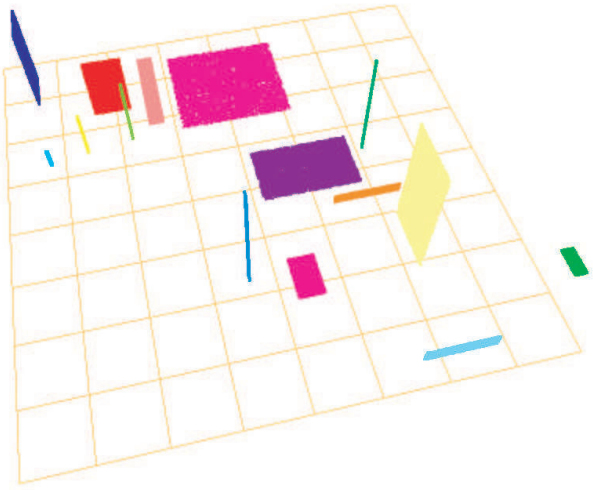


Fig. 3. A synthetic dat set consisting of 100, 000 points distributed over 15 flat boxes.

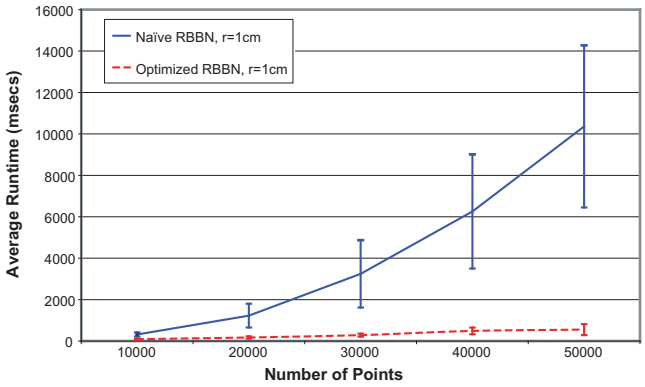


Fig. 4. A runtime comparison of naïve and optimized RBNN segmentation. Clearly, optimized RBNN is a drastic improvement in terms of average runtime and standard deviation.

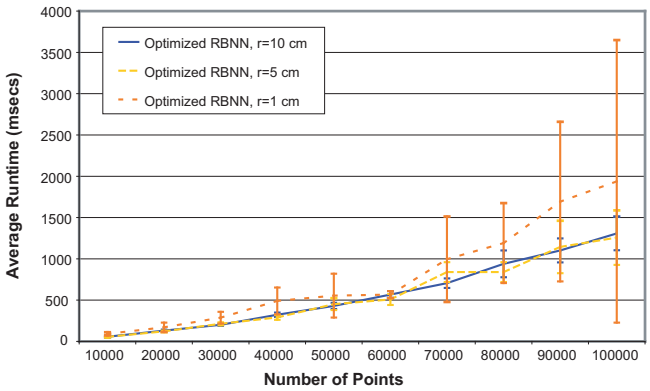


Fig. 5. A runtime comparison of optimized RBNN segmentation with different values of r.

gorithms such as *Chameleon* may increase the segmentation quality in the presence of chains of outliers connecting two distinct clusters. Augmenting the feature space by estimated surface normals will also increase segmentation quality. Both these extensions incur a significant computational overhead, however, since they require the construction and processing of a complete neighborhood graph. Future research will aim to achieve a balancing between more powerful algorithms and realtime capability. Furthermore, the extraction of geometry will be addressed.

Acknowledgments

This work is supported in part within the DFG excellence initiative research cluster *Cognition for Technical Systems – CoTeSys*, see also `www.cotesys.org`.

References

[1] G. Lidoris, K. Klasing, A. Bauer, T. Xu, K. Kühnlenz, D. Wollherr, and M. Buss, "The Autonomous City Explorer project: Aims and system overview," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, (San Diego, CA), October 2007.

[2] D. Hähnel, W. Burgard, and S. Thrun, "Learning compact 3d models of indoor and outdoor environments with a mobile robot," *Robotics and Autonomous Systems*, vol. 44, no. 1, pp. 15–27, 2003.

[3] O. Wulf and B. Wagner, "Fast 3d-scanning methods for laser measurement systems," in *Proceedings of the 14th International Conference on Control Systems and Computer Science (CSCS14)*, (Bucharest, Romania), July 2003.

[4] D. Hähnel, D. Schulz, and W. Burgard, "Mobile robot mapping in populated environments," *Advanced Robotics*, vol. 17, no. 7, pp. 579–598, 2003.

[5] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[6] R. Xu and I. Wunsch, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, no. 3, pp. 645–678, 2005.

[7] N. Memarsadeghi, D. M. Mount, N. S. Netanyahu, and J. L. Moigne, "A fast implementation of the isodata clustering algorithm," *International Journal of Computational Geometry & Applications*, vol. 17, no. 1, pp. 71–103, 2007.

[8] Y. Lui, R. Emery, D. Charabarti, W. Burgard, and S. Thrun, "Using EM to learn 3D models of indoor environments with mobile robots," in *Proc. 18th International Conf. on Machine Learning*, pp. 329–336, Morgan Kaufmann, San Francisco, CA, 2001.

[9] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603–619, 2002.

[10] Y. A. Sheikh, E. A. Khan, and T. Kanade, "Mode-seeking via medoidshifts," in *Proceedings of the IEEE International Conference on Computer Vision*, 2007.

[11] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2 nd Int. Conf. on Knowledge Discovery and Data Mining(KDD ' 96)*, pp. 226–231, 1996.

[12] G. Karypis, E.-H. S. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[13] D. Marshall, G. Lukacs, and R. Martin, "Robust segmentation of primitives from range data in the presence of geometric degeneracy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 3, pp. 304–314, 2001.

[14] A. Bab-Hadiashar and N. Gheissari, "Range image segmentation using surface selection criterion," *IEEE Transactions on Image Processing*, vol. 15, pp. 2006–2018, 2006.

[15] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "Accurate object localization in 3d laser range scans," in *Proceedings of the 12th International Conference on Advanced Robotics (ICAR '05)*, (Seattle), pp. 665 – 672, July 2005.

[16] R. Paredes and E. Chávez, "Using the k-nearest neighbor graph for proximity searching in metric spaces," in *SPIRE*, pp. 127–138, 2005.