Hochschule
Ravensburg-Weingarten
Technik | Wirtschaft | Sozialwesen

Department of
Electrical Engineering
and Computer Science

Master's Thesis

Recurring Adaptive Segmentation and Object Recognition

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree
of
Master of Science
in
Computer Science

By: **B. Sc. Martin Bertsche**
Mat. No.: **21734**
First Examiner: **Prof. Dr. rer. nat. Wolfgang Ertel**
Second Examiner: **Prof. Dr. Ing. Konrad Wöllhaf**
Submission Date: **31. August 2012**

Affidavit

I assure that I have written this thesis myself. No further sources or aids were used than those explicitly stated. All citations in this work are labeled as such.

_____, _____ _____

(Place, Date) (B. Sc. Martin Bertsche)

Abstract

Object recognition approaches which are currently used, mostly rely on segmentation for two reasons. They either are based directly on pre-segmented data or they benefit largely from segmentation in terms of required computation time. In the field of Service Robotics the applied segmentation approaches and therefore, the work environment are often limited to tabletop environments. This thesis investigates methods of extending the current segmentation capability by selecting between different segmentation approaches and algorithm parameter sets. In order to know which choice to make an object recognition based quality metric for point cloud segmentation is developed. A method is proposed that allows the robot to autonomously decide which algorithm and parameter set to use. It is shown that using multiple segmentation approaches and parameter sets can widely increase the the number of scenarios suitable for object recognition

Contents

Chapter 1

Introduction

The ability to perform image segmentation comes naturally to humans and many animals. We use it in almost every part of our lives since eyesight is the main way how we perceive the world around us. It is our ability to semantically partition the images passed from our eyes to our brains in an incredibly short period of time. Unfortunately, the high-level processes taking place inside our brains to perform this task are not very well understood. Even if they were, it is highly improbable that hardware or software can be created in the near future to replicate this behaviour on a similar performance level. Segmentation is therefore a major research topic as it has been since the beginning of digital image processing in the 1960s.

Within digital image processing segmentation comes in many different forms that depend on the particular use case. In industrial image processing one of the main tasks is to perform quality checks on work pieces such as a photogrammetric survey or checks for completeness after assembly. Almost all of these tasks are performed applying some sort of segmentation. Lighting and camera optics are optimized to a level that the detection of a work piece is often reduced to finding lighter or darker areas in an image. The results are so highly accurate that they can even be used for measurement.

In medicine a very well known application of digital image processing on 3D data is magnetic resonance imaging which enables Physicians to virtually view a patients body on the inside. 3D segmentation assists doctors in performing their diagnosis. Physicians are able to virtually highlight or remove individual organs, vessels, bones and other sorts of tissue. However, in contrast to most other 3D segmentation disciplines the types of media like liver tissue or muscle tissue can already be identified by the imaging process itself, making the job of segmentation considerably easier.

Today's cartography also relies on 2D and 3D imaging from satellites and airborne camera systems. Segmentation is used to extract elements such as roads, fields, houses or even whole cities. Even today these processes cannot be fully automated. Automati-

cally generated results need to be manually checked and corrected as needed. However, as in industrial or medical image segmentation the goal is always to identify parts of consistent semantics inside the data.

Semantics is the key term which makes segmentation such a difficult task for software developers and scientists in the field of computer vision. Just consider the term "road" for a moment and envision how many images can be imagined. These are your image based semantic associations with the term. A computer has no such associations and is therefore not capable of extracting roads from an image without further help. The software developer has to define a model for "road" in terms of the image data. An example of a simple and also very bad model could be: "Roads are gray." Assuming a good definition of "gray" exists it is certainly possible to make up other gray entities that can occur in areal images that are not necessarily roads. The computer vision developer must be skilled at defining such models and encoding them in software. The task becomes even harder as levels of abstraction are introduced that widen the semantic range and make borders blurry between parts that belong to one segment and parts that belong to another.

1.1 Motivation

This Master's Thesis is set in the field of mobile service robotics which is dedicated to developing robots that help people with their household tasks. A major requirement for a service robot is therefore, its ability to perform manipulation tasks like making coffee or setting the table. These tasks involve objects that are to be handled such as a cup for example. Before any action takes place, the objects required for the task must be identified and located. Segmentation is performed in order to isolate object candidate clusters within the acquired data which are subsequently processed for identification. Today's robots mostly use 3D sensors to perceive their environment. Segmenting 2D or 3D data for objects is one of the most demanding tasks because the semantics of "object" is so generic. A model for the term "object" has to be devised that covers very different entities such as a book and a coffee maker. However, before going deeper into the matter a few terms need to be defined:

Segmentation labels the pixels in an image or the points inside a point cloud such that ideally all points having the same label belong to the same real world object in the scene. The definition of a real world object however can vary depending on the application.

Object Recognition is a two-step process consisting of training and recognition. *Training* describes the process of gathering sensor data belonging to a real world object which is subsequently transformed into a model. *Recognition* is the process of matching new sensor data to models in a database of known objects. The output is the matched model's identifier or name.

Pose Estimation is an optional capability of an object recognition approach. It means that in addition to the corresponding model name the object's position and orientation can be determined as well.

All object recognition approaches which are currently used by our work group rely on segmentation for at least one of two reasons. The first reason is that the object recognition approach is based on a global description of object candidate clusters. Such descriptions can be bounding boxes that encode the object's dimensions but also color histograms encoding information about the objects texture[1]. This means that the candidate cluster is transformed into a representation that is compatible with the model data stored in the database of known objects. It is required that the candidate cluster contains as few falsely segmented points as possible. The second reason is the processing time. There are also object recognition approaches that do not rely on segmentation for recognition whereas they do for training purposes. Instead of describing all the data belonging to the object, distinctive features are extracted that are likely to be recognized when the object is scanned again. The features are usually computed for each pixel or point which can result in overwhelming computational overhead. By using data segmentation not all data points need to be inspected, which clearly reduces computation time.

Currently there are four different object recognition approaches implemented on our demonstration system including a simple bounding box algorithm, color histograms [7], a SIFT [13] based approach and Viewpoint Feature Histograms [19]. However, there is only one rudimentary segmentation algorithm which is optimized for an IKEA kitchen table. The robot's work environment is therefore limited to the table and the top of a sideboard for which segmentation results are already deteriorating. This situation is not acceptable for a robot that is to demonstrate suitability for daily use. Thus the motivation for this thesis is the need to extend the robot's work environment.

Three major challenges have been identified: The first challenge is the problem of over- and under-segmentation of objects. Under-segmentation means that segments are spread across multiple real world objects which leads to two or more objects being seen as one by the robot. Over-segmentation being the opposite case relates to segmentation where one real world object is represented by multiple segments. The main source for light cases of these undesirable effects is improper algorithm parametrization. Choosing a different parameter set can quickly solve the problem.

The second challenge are scenes that cannot be segmented at all by the current approach because major assumptions encoded in the algorithm are not fulfilled. Such cases can only be mastered by using alternative segmentation approaches.

[1] Texture in this context relates to surface color and print. It does *not* relate to the surface's character.

The last challenge is the segmentation of objects containing other objects such as shelves or other furniture. The problem here is the desired level of detail which has a strong influence on what is to be counted as real world object. A general observation for segmentation algorithms is that they perform better when the desired segments are approximately of equal size. The question when segmenting a shelf is whether its contents should be part of the shelf segment or have their own segments. This of course depends on the use case. If the task is just to find the shelf no segmentation of its contents is necessary.

The goal of this thesis is to implement a segmentation and object recognition approach that extends beyond the demonstration system's current capability and addresses at least two of the three challenges stated above.

1.2 Approach

This section briefly describes the approach for tackling the challenges as stated in Section 1.1. It adheres to the definition of the thesis proposal which is printed in Appendix A for further reference. In addition to an alternative description of the approach it also contains a definition of six work packages along with the questions that need to be answered in this document. Please note that this thesis only covers a subset of the work packages since the proposal exceeds the scope of a Master's Thesis by far. For clarification the work packages covered by this thesis are are discussed after the approach along with an outlook over the chapters of this document.

1.2.1 The Proposed Approach

In order to extend the current segmentation and object recognition capability, flexibility must be increased. Since over- and under-segmentation primarily occur due to improper parametrization of the segmentation algorithms, static parameter sets are no longer an option. Using a single parameter set for an algorithm limits its segmentation capability considering that slight parameter adjustments can largely improve segmentation performance for otherwise difficult scenes. Therefore multiple parameter sets must be available for each algorithm. However, it would be unwise to have highly specialized parameters for every single scene. Instead a small set of parametrizations should be found which are applicable for larger numbers of scenes. This increases the probability for the parameter sets to also apply to scenes for which they have not been optimized. A method for finding such parameter sets needs to be developed.

Flexibility is further increased by adding additional approaches to the segmentation repertoire. These algorithms must be carefully chosen in a way that the set of algorithms achieves a high degree of versatility. A rule of thumb for the algorithm selection should be the use of heterogeneous approaches. Strongly related approaches are very

likely to fail under similar conditions. Heterogeneous segmentation algorithms are expected to be complementary such that the failure of one approach at segmenting a given scene is compensated by the success of another approach.

Given a set of algorithms and a set of parametrizations per algorithm it should be possible to considerably improve segmentation and recognition beyond the capabilities of the current system. However, the choice of algorithm and parameter set must be manually configured for each scene. This is possible since information about which type of scene to expect can be obtained from the robot's self localization system.

In order to fully automate the selection process by applying a trained classifier is a straight forward approach. The classifier must learn to map a given scene to a suitably parametrized algorithm. A method for scene description, similar to the global description methods used for object recognition, must be developed since raw point cloud data is not a suitable input vector. The input labels needed for training represent the best parametrized algorithm for a given scene and must be automatically generated as well. Manual assessment of segmentation results for many scenes is tedious and may also be biased in the case of similar results. Therefore a segmentation quality benchmark is needed. A sufficiently trained algorithm is expected to make correct choices also for unknown scenes.

For the segmentation of large scale objects like furniture that may support other objects, a hierarchical approach is proposed. Segmenting furniture requires a segmentation approach that is able to detect the coarse structure of a scene without being disturbed by inhomogeneities introduced by the furniture's contents. Such an approach is very likely to lead towards under-segmentation. If necessary the coarse segmentation can be applied recursively to its sub-segments until the segments are identified as known objects or become too small. The segmentation of furniture contents is most probably an ill-conditioned task for such a coarse segmentation approach. As the sub-segments become smaller it becomes likelier that the segments are not further subdivided by the next recursion step. Therefore, recursion must also stop if coarse segmentation does not progress. In this case segments have to be passed on to the regular segmentation algorithm.

1.2.2 Processed Work Packages

The first work package demands the implementation of at least three different segmentation approaches which are necessary to generate a versatile set of algorithms. In Chapter 2 the implementations of Color-based Segmentation, Scene Interpretation and Spectral Clustering are discussed in depth. Furthermore reasons for choosing this set of algorithms are given in Section 2.4 along with the reasons as to why the algorithms are expected to be complementary.

Chapter 3 is dedicated to the definition of a segmentation quality metric. An existing

approach for scoring 2D segmentation, which can be used for 3D segmentation as well, is evaluated. However, it requires an amount of work force that is not available for this thesis. Therefore an alternative segmentation quality metric is devised which is based on object recognition performance. The work done here strongly relates to the fourth work package in the thesis proposal. It demands the implementation of a segmentation quality metric in order to automatize data set labelling for classifier training. However, in this work the segmentation quality metric is used differently.

The second and third work package are covered by Chapter 4. In Section 4.1 justification for the scenes that were chosen to test the approach is provided. Their characteristics are compared to the strengths and weaknesses of the segmentation algorithms. In Section 4.2 manual parametrization is discusses along with an initial performance evaluation of the single algorithms. Fine tuning of the parameters is performed in this section as well. The segmentation quality metric derived in Chapter 3 is used to implement a brute force parameter optimization approach. A voting based approach is proposed to extract generic parameter sets from the results of the brute force optimization. Section 5 evaluates the parameter sets regarding their segmentation performance and applicability for automated selection.

1.3 Tools

The most important tool used on the hardware side is the Microsoft™Kinect™. It is equipped with a 2.5D depth sensor capable of VGA resolution and an SXGA RGB camera. The depth sensor developed by PrimeSense is using an infrared pattern, also called *structured light*, which is projected on the environment and recorded by an infrared camera. Geometric deformations of the pattern are used to calculate a distance for each sensor pixel to the next surface in its line of sight. The pattern consists of three different layers. Many small patterns of low intensity are designed for low distance measurement. Further layers are designed for medium and long distance measurement. They have increasing pattern sizes and increasing intensity whereas their resolution decreases. This results in reduced resolution along the z-axis with increasing distance.

An important tool on the software side is the Point Cloud Library (PCL) [20]. It is an extensive tool set for point cloud processing offering implementations of many basic algorithms. In addition to the algorithmic side it also provides many utilities such as an easy-to-use interface for 3D sensors and data formats for filesystem persistence.

All software written for the purpose of this thesis is implemented using the BOR3D [5] Framework. The framework is designed to facilitate the development of new signal processing algorithms by offering ways to quickly create an environment which already performs data acquisition and preprocessing if needed. It is also designed such that single processing steps can easily be replaced by alternative approaches without the need to rewrite large portions of code. Its main advantage is however that it is freely

available to anyone at SourceForge.

Chapter 2

Segmentation Algorithms

2.1 Color-based Segmentation

A straight forward approach to the 3-D point cloud segmentation problem is the Color-based Segmentation [1] approach as proposed in [24]. The basic agglomerative method extends standard euclidean clustering with the ability to consider color information available for each given point. It is further extended by a refinement step. This discussion will cover the differences between standard euclidean clustering and CbS as well as some necessary differences between the proposed approach and the actual implementation used for the purpose of this thesis.

2.1.1 Comparison of CbS and Euclidean Clustering

In order to better understand color based segmentation a quick discussion of the underlying euclidean clustering operation is needed. Let

$$P = \{p_1 \ldots, p_n\} \text{ with } P \subset \mathbb{R}^3$$

be a set of points or alternatively a *point cloud*. Further let

$$knn : \{P, \mathbb{R}^3, \mathbb{N}\} \rightarrow Q, Q = \{q_1, \ldots, q_m\} \subset P$$

be a function returning the m nearest neighbors in P for an arbitrary point in three-dimensional euclidean space. The desired result is a set of clusters

$$S = \{C_1, \ldots, C_t\} \text{ where } \forall j, k : j, k \in \{1, \ldots, t\}, j \neq k, C_j \subset P, C_j \cap C_k = \emptyset.$$

[1] Throughout the rest of this thesis Color-based Segmentation will be abbreviated by CbS.

This is the same result that is asked for with any segmentation algorithm. The pseudo code in Algorithm 2.1 explains how the set S is obtained. The runtime complexity of

Algorithm 2.1 Euclidean Clustering(P, l, m)

$S \leftarrow \emptyset, R \leftarrow \emptyset$
while $\bigcup_{j=1}^{t} C_j \neq P$ **do**
  $R \leftarrow R \cup \{p\}$ with $p \notin \bigcup_{j=1}^{t} C_j$
  $C \leftarrow \emptyset$, $C \leftarrow C \cup \{p\}$
  while $R \neq \emptyset$ **do**
    $R \leftarrow R \setminus \{q\}$
    for all $r \in knn(P, q, m)$ **do**
      if $\left(r \in \bigcup_{j=1}^{t} C_j\right) \vee (||q - r||_2 < l)$ **then**
        continue
      else
        $R \leftarrow R \cup \{r\}$, $C \leftarrow C \cup \{r\}$
      end if
    end for
  end while
  $S \leftarrow S \cup \{C\}$
end while
return S

Algorithm 2.1 cannot be seen directly because of the coupling of the two innermost loops. The outer loop's complexity is clearly $O(n)$. Its sole purpose is to provide an initial cluster point if no further points fulfil the euclidean distance criterion for the current cluster. Using appropriate data structures it can be guaranteed to iterate over the points in P only once. The innermost loop's runtime is also easy to see since it will always iterate over the m nearest neighbours and thus is $O(m)$. The second loop's runtime seems to depend on the amount of new points added by the innermost loop. However when looking closer at the collaboration of the innermost and outermost loops it becomes clear that in any case it will run exactly once per point in P. If a k-d tree [4] data structure is used to perform knn which can be created in $O(n)$ and queried in $O(\log(n))$ time, an upper bound is given for the whole algorithm by $O\left(n^2 m \log(n)\right)$.

Obviously the termination condition for the two innermost loops constructing the clusters C_j is only triggered by the distances between the points in P. So the algorithm is looking for contiguous regions C_j in P that are separated by gaps of minimum width l. Cases where this approach fails to satisfy the demand of a bijective mapping among the clusters found and the real world objects are easily constructed.

Consider an artificial scene where a cube is sitting on a plane. Although the cube and the plane are different objects there is no such gap between the two which could be used to separate them. Extending this consideration it can be said that whenever there

exists an undirected, weighted Graph $G(V, E)$, where $V = P$ with weights defined as the euclidean distance between connected vertices and all weights being smaller than l, S will have only one element $C_1 = P$. Unfortunately, this is very prominent in 2.5-D sensor data making Euclidean Clustering ineffective when used on its own.

Zhan et. al. overcome this problem by using a sensor which is able to provide color information at each sampled point. The basic algorithm which they call Region Growing is extended by adding a color-based point rejection criterion. They define a colorimetric dissimilarity measure to be the euclidean distance between two vectors $\lambda = [r, g, b]$ containing the point's color encoded as RGB. In addition to the distance threshold l, a colorimetric dissimilarity threshold α is introduced. They also divided the operation into two processing passes. Region Growing being the first pass has to do preparatory work for the second pass by collecting statistical data about the color distribution inside a cluster C_j such as the mean color and the standard deviation.

The second pass performed by Zhan et. al. is called Merging and Refinement. In essence this step is a repetition of Region Growing performed on the clusters $C_j \in S$ instead of the points $p \in P$. In order to handle the clusters in almost the same way as it is done with the points, the statistical data collected in the previous step is used. A cluster's position is defined by its center of gravity, its color is the mean color of the points it contains. With these definitions a conversion of Region Growing to merge adjacent regions with similar colors is trivial. All that is needed is another set of thresholds l_{reg} and α_{reg} to perform region rejection analogously to point rejection. This step is extended by a search for regions that contain less than a desired number of points d. These regions are merged with their respective nearest neighbours in euclidean space.

Since the second processing pass of CbS is done on t clusters - a number that is significantly smaller than n - we can neglect its runtime complexity because the first pass will dominate. The changes made in Euclidean Clustering to perform Region Growing are a second threshold operation with complexity $O(1)$ and the statistics update for the clusters which is done whenever a new point is added. Updating the statistics requires to look at all points inside the cluster. So the worst case is that all points belong to the same cluster resulting in $O(n)$ complexity for such an update. Therefore Region Growing as proposed by Zhan et. al. has a runtime of $O(n^3 m \log(n))$. However looking back at the box-on-a-plane example the algorithm is now able to cluster the scene correctly if the plane and the cube are colored differently.

2.1.2 Implementing Color-based Segmentation

Transforming the pseudo code from Algorithm 2.1 to any programming language is straight forward as long as the k-d tree and some basic data structures for sets and lists are available. However two changes were made during implementation. The first

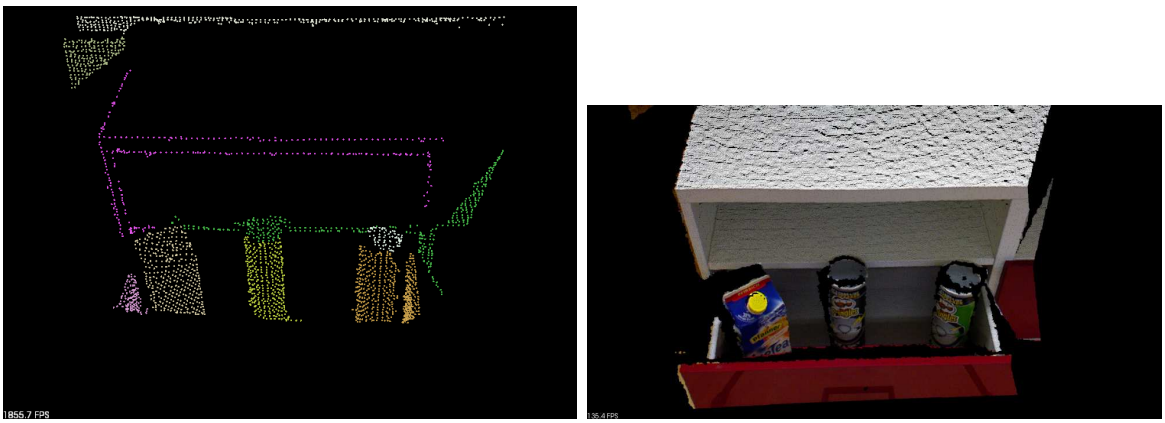


Figure 2.1: Example of Color-based Segmentation segmenting the contents of a drawer.

reduces the complexity of Region Growing from $O(n^3 \log(n)m)$ to $O\left(n^2 \log(n)m\right)$. This is achieved by moving the computation of a region's center of gravity and its mean color out of the innermost loop to the outermost loop. Since these values are used nowhere else inside the Region Growing process, this does not affect the algorithm's final result. The second alteration concerns the final merging step in the second processing pass. The proposed algorithm joins clusters C_j with $|C_j| < d$ to their nearest neighbouring cluster. The implementation used for this thesis however does not follow the proposal since stray points can occur due to noise of the sensor used. Instead these small clusters are dropped and will not be considered in the algorithm's output.

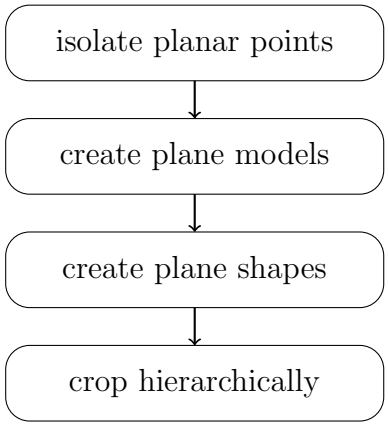


Figure 2.2: Simplified processing pipeline of the scene interpretation segmentation approach.

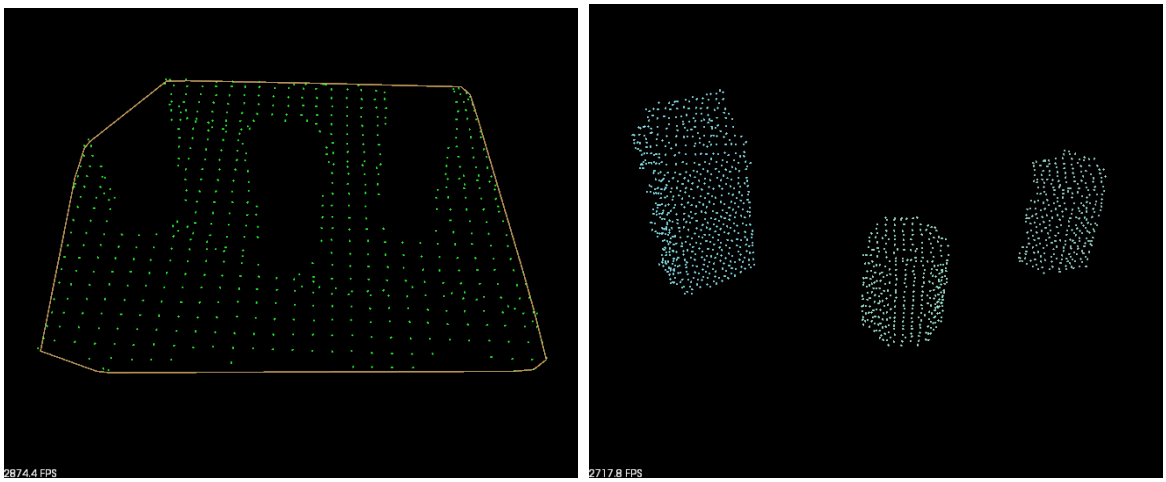


Figure 2.3: ScI Segmentation. On the left the result of planar surface detection can be seen including the convex hull. The object candidate clusters are shown on the right.

2.2 Scene Interpretation

Throughout this thesis the segmentation approach suggested by Rusu et. al. in [20] will be called Scene Interpretation (ScI). The goal of this approach is to use segmentation and surface reconstruction methods in order to create 3-D models of objects in the scene in real time. The models can then be used to plan manipulation tasks such as grasping.

Matching the scope of this thesis the surface reconstruction part of the proposed algorithm is omitted and only the segmentation approach – which is highly specialized for household environments – is considered. In contrast to CbS from Section 2.1 this approach does not use any color information and it does not rely on a single processing step to perform scene segmentation.

The assumption of ScI's segmentation approach is that objects of interest to a service robot are always located upon planar surfaces such as the floor, a table, a cupboard or a kitchen counter as shown in figure 2.3. The idea is to identify and isolate these planar surfaces inside a scene. Knowing the shape and position of the surfaces makes it possible to isolate those points which are on top of them and therefore belong to the objects of interest. These points are then clustered in a euclidean sense to associate them with the individual objects.

A large set of specialized processing steps is used. Since so many steps are involved the discussion of the process is divided into the four parts which can be seen in Figure 2.2.

2.2.1 Isolating Planar Points

The goal of the first two steps is to find all planar surfaces inside the scene which are parallel to the ground. At first only those points are of interest who have a normal pointing approximately upward. For that reason it is necessary to know the sensor's orientation relative to the ground so that an up-vector $\mathbf{e}_u$ can be defined.

In the given environment there exist at least two ways to obtain this information. (1) The sensor itself is equipped with a built-in accelerometer. However being mounted on a mobile platform a lot of noise in the data read from the accelerometer is to be expected. This uncertainty disqualifies the method for ScI. (2) Alternatively the last command of the pan-tilt unit (PTU) – a device to adjust the sensor's pitch and yaw angles – can be used to get the sensor's orientation. Currently the segmentation algorithms are running as standalone software. Therefore the current orientation is a configuration parameter that stays fixed for all scenes.

Next the orientations of the points $p \in P$ need to be determined. This is done using the knn function already known from Section 2.1. With knn the k nearest neighbours of a given point are found. These neighbours are used to form an overdetermined system by inserting them into the plane equation $ax + by + cz - d = 0$. Solving the system using the method of pseudo inverses or Singular Value Decomposition yields an estimate for the normal vector $\mathbf{n} = [a, b, c]$ at point p. $\mathbf{n}$ has to be normalized in order to get the orientation's unit vector $\mathbf{e}_n = \frac{\mathbf{n}}{|\mathbf{n}|}$.

Filtering the point cloud for points having a normal which is similar to $\mathbf{e}_u$ is done using the dot product's definition: $\mathbf{ab} = \cos(\angle(\mathbf{a}, \mathbf{b}))\,|\mathbf{a}|\,|\mathbf{b}|$. Since the $\mathbf{e}_u$ and $\mathbf{e}_n$ are both unit vectors one can simply solve for $\angle(\mathbf{a}, \mathbf{b})$ by evaluating $\arccos(\mathbf{ab})$. An angular threshold parameter γ for $\angle(\mathbf{a}, \mathbf{b})$ is needed to tell which normals are still counted as pointing up and which are not.

2.2.2 Creating Plane Models

The second step to finding planar surfaces begins with clustering the remaining points using the basic algorithm from Section 2.1. It is now useful because it is assumed that the planar points are surrounded by points not belonging to planes in the scene. These non-planar points have been removed in the last step leaving gaps large enough to perform Euclidean Clustering. The result are m clusters of plane candidate clouds that from now on will be examined individually.

For each planar point cluster the corresponding plane model needs to be determined. An effective way for finding models of geometric primitives like planes inside point clouds is the Random Sample Consensus (RanSAC) [8] method. The general idea behind RanSAC is to form a mathematical model of the desired primitive from a minimal set of random samples in the input data. The consensus step then checks the

quality of the model by computing the sum of distances for all the remaining data to the model. This process is repeated for a user defined number of times. The models are then stored together with their respective quality measures. In the end the model with the lowest sum of distances is selected as the result.

Clearly there is no guarantee for the RanSAC approach to yield an optimal result. Actually, the chances for an optimal result are close to zero for two reasons. (1) There has to be a subset of points in the input data that defines the optimal model. (2) The whole subset has to be selected within one random sample draw. So the user of the RanSAC method must be satisfied with approximate results or even bogus results, depending on the level of noise within the data and the number of iterations. However Fischler et. al. have shown that for sufficiently large data sets and a sufficiently large number of iterations RanSAC yields close to optimal results.

RanSAC is not only effective but also efficient. The sample step can be considered to have a runtime complexity of $O(1)$ since the number of sample points is fixed for a given model. The following consensus step runs in $O(n)$ time, where n is the number of data points in the set. All points need to be compared with the model. The always optimal alternative of doing a least squares fit with all sample points has cubic complexity in n. So in cases where no optimal solution is needed using RanSAC is a good choice for fitting primitives.

2.2.3 Plane Shapes

Having determined the parameters a, b, c and d of the plane model for each planar point cluster the next step is to remove some outliers in the clusters. That means points that are too far away from the model to be counted as part of the plane. For each cluster an ϵ-margin had to be defined around the corresponding plane model. All points outside the margin are rejected.

Next, the clusters' two-dimensional shapes have to be determined. It cannot be guaranteed that the shape of the planes will always be convex. An L-shaped table arrangement for example would result in a concave shape. The PCL's concave hull class was used to perform the task of approximating concave polygons for the cluster hulls.

Numerical Issues

By solely applying the concave hull algorithm to the clusters it does not produce a desirable behaviour. The algorithm has a built-in switching mechanism to choose between 2-D and 3-D concave hulls. Since the data points are scattered around the actual plane model the concave hull will always be a three-dimensional one.

In a next step the inliers that are scattered around the plane need to be projected onto the model. The PCL's Sample Consensus tool set contains a function to do the

projection automatically. However, this does not prove to be useful for this task. Due to numerical issues, not all projected points are coplanar after the projection. Therefore, the concave hull output is still three-dimensional in most cases. The projection has to be reimplemented in a numerically stable fashion.

Hardening the projection numerically is done by transforming the plane model along with the point cluster onto the x-z-plane. In order to do this the plane model has to be aligned with the x-z-plane by rotating it in such a fashion that its normal vector points in y-direction. The rotation is determined as quaternion. A rotation axis is needed as well as a rotation angle. The angle is computed by evaluating $\alpha = \arccos(\mathbf{e}_n \mathbf{e}_y)$. The rotation axis had to be perpendicular to both $\mathbf{e}_n$ and $\mathbf{e}_y$ and was therefore determined by $\mathbf{e}_r = \mathbf{e}_n \times \mathbf{e}_y$. The second step of the transformation is to translate the plane to the $y = 0$ level. The distance d_y of the plane to the origin needs to be known. Inserting $x_0 = [0, d_y, 0]$ into the plane equation and solving for d_y yields $d_y = \frac{d}{b}$.

Zeroing the y-coordinates of each point results in the proper projection of coplanar points that can be used for stable computation of the concave hull.

2.2.4 Cropping

The last steps of ScI consist of extracting the spaces that are on top of the planar surfaces and clustering the respective point clouds to find individual objects.

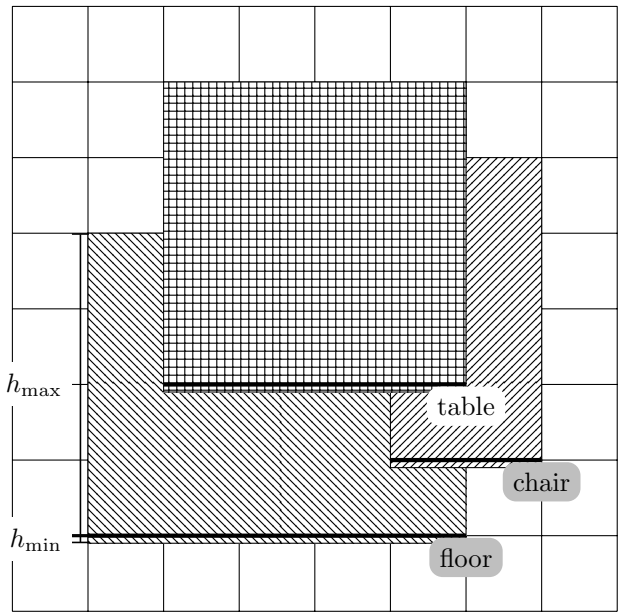


Figure 2.4: Hierarchical Cropping of Spaces. The planes for floor, chair and table are ordered by their heights. The highest plane keeps all its points above it while lower planes lose their points to intersecting spaces above them.

For cropping the PCL offers a useful function which is able to extract the content of polygonal prisms from point clouds. The function uses a 2-D polygon which can be arbitrarily placed and oriented in three-dimensional space. It automatically determines the direction of the polygon's surface normal and extrudes the two-dimensional shape along this axis. The user must define two values $h_{\min}$ and $h_{\max}$ which are the extents in negative and positive direction along the normal to which the prism is extruded. The result is a polygonal prism which is used to filter the points in the cloud. All points lying outside the prism are rejected whereas all points inside the prism are kept. The extraction needs to be performed on the original point cloud data that has not undergone any previous filtering steps.

Unfortunately applying the polygonal prism extraction for all polygons does not work correctly in the case of overlapping surfaces. That means if for example the surface of the floor is found beneath the surface of a table. Depending on the choice of $h_{\max}$ there may be spaces where the two polygonal prisms of the floor and the table intersect. All points lying inside these spaces would therefore be inspected twice in the following clustering step, increasing computational cost or even duplicating object clusters.

Figure 2.4 shows the solution to that issue in a simplified two-dimensional case. There the polygons are represented by thick horizontal lines. Their respective spaces are the shaded rectangles above them. On the left side $h_{\min}$ and $h_{\max}$ are drawn in for the floor to clarify their meaning. In order to obtain this solution the points being inliers of the polygonal prisms are treated as sets T_i where $i \in \{1, \ldots, n\}$ and n is the number of surfaces. Given this definition Algorithm 2.2 creates the hierarchical spatial subdivision for an arbitrary set of planar surfaces.

Algorithm 2.2 Hierarchical Cropping

sort T_i (ascending) by height of corresponding surface
for $i = 1 \rightarrow n$ **do**
  $T_i \leftarrow T_i \setminus \bigcup_{j=i+1}^{n} T_j$
end for

As a last step Euclidean Clustering is used in order to get the segments of the individual objects.

2.3 Spectral Clustering

Spectral Clustering or SC is a graph theoretic approach to the clustering problem which has been used successfully for both 2-D [9], [1] and 3-D [15] scene segmentation purposes. Being also a method used in many other domains for exploratory statistical analysis such as social sciences, psychology and biology its main virtue is that it does not impose implicit or explicit assumptions on the data that were heavily used for ScI

in Section 2.2. The following Section 2.3.1 will give some mathematical background needed to understand how and why Spectral Clustering works. Additionally a fast method for approximating eigenvectors is described which is needed for the algorithm presented in Section 2.3.2.

2.3.1 Mathematical Basis

The basic goal of a clustering approach is to divide points in a data set into clusters in which all points have a high degree of similarity and – at the same time – have a high degree of dissimilarity to points belonging to other clusters. The 3-D scene segmentation problem has a very similar demand given the assumption that the individual objects in a scene are represented by points inside the sensor data which have similar properties.

The standard method of representing similarities inside a data set $P = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ is an adjacency matrix W. This symmetric, positive semi-definite matrix is of size $n \times n$ and connects each data point with all other vertices by an adjacency metric $w : \{P, P\} \to \mathbb{R}^+$ such that $W_{ij} = w(x_i, x_j)$. The matrix can also be interpreted as fully connected, weighted graph $G(V, E)$, where $V = P$ and the weight of $E_{ij} = w(x_i, x_j)$. In graph theory clustering can be viewed as cutting G in several pieces in such a way that $V = \bigcup_{i=1}^{p} C_i$.

A formalization of cutting graphs is the normalized cut (Ncut) developed by Shi et. al. [21]. In the following the notation $i \in A$ will be used as short hand for $\{i | v_i \in A\}$. Let $d_i = \Sigma_{j=1}^{n} w_{ij}$ be the degree of a vertex, defined as the sum of weights of all connected edges. Furthermore let $\mathrm{vol}(A) = \Sigma_{i \in A} d_i$ be the volume of A and $\mathrm{cut}(A, B) = \Sigma_{i \in A, j \in B} w_{ij}$ be the sum of weights of the edges connecting A and B, with $A \cup B = V$ and $A \cap B = \emptyset$.

With $\mathrm{cut}(A, B)$ there already exists a formal definition for finding a bipartition of G by solving $\mathrm{argmin}_{A,B}\{\mathrm{cut}(A, B)\}$. However it is obvious that $\mathrm{cut}(A, B)$ monotonically increases with the number of elements in A thus favouring the isolation of single points in the dataset. According to Shi et. al. segmentation approaches using this standard cut iteratively have not been very successful. Instead they proposed the normalization $\mathrm{Ncut}(A, B) = \frac{\mathrm{cut}(A,B)}{\mathrm{vol}(A)} + \frac{\mathrm{cut}(A,B)}{\mathrm{vol}(B)}$. The formulation removes the bias that is seen for the standard cut. Unfortunately Shi et. al. also proved the problem of finding $\mathrm{argmin}_{A,B}\{\mathrm{Ncut}(A, B)\}$ to be NP complete.

Similarity Graphs

There are many ways to define a similarity graph and the corresponding adjacency matrix. The way which is easiest to implement is a fully connected graph. Obviously such a graph has n^2 edges. Even for not very large $n \sim 10^5$ the corresponding adjacency

matrices become intractable for personal computers. Very often Spectral Clustering methods are referred to as globalized methods because they take into account all the points in the dataset whereas methods like k-means or Euclidean Clustering make the cut decision locally by saying: "I cannot find any point that is close enough to the point I'm looking at." Since in most cases it is impossible to use a fully connected graph, sparse matrices are created instead.

A sparse matrix can be created by finding the k nearest neighbours of each point inserting only the similarities between the current point and its neighbours, zeroing all other entries in the adjacency matrix. This method however will lead to non-symmetric matrices due to the fact that if point a has a nearest neighbour b, b does not necessarily have a nearest neighbour a. The matrix must artificially be made symmetric since no fully connected graph is generated. This can be done by copying the obtained values into the transposed matrix position. The upside is that the process does not need to inspect all n^2 pairs of points. Knowing that nearest neighbour search can be done in $O(\log(n))$ this reduces the runtime complexity of finding the adjacencies to $O(n \log(n)k)$.

Another way is to use Gaussian similarity kernels such as $s(a, b) = \exp(\frac{||b-a||^2}{2\sigma^2})$. The properties of this function are very well suited for creating a sparse matrix because the values it takes for $||b - a|| > 3\sigma$ can be approximated by zero. σ is used to define the size of the neighbourhood that is taken into account. This method however still entails calculating the adjacencies for all n^2 pairs.

Sparse Matrix Datastructures

In addition to the runtime problem there is also the problem of limited space. It is even more apparent than runtime because sofware requiring more memory than available will crash instead of running for a very long time. For $n = 10^5$ the squared amount of $n^2 = 10^{10}$ memory locations are needed to store a complete adjacency matrix. However current personal computers are only capable of storing $\frac{1}{10}$ of this amount. Luckily using sparse matrices most of the entries are zero. This knowledge is used to define data structures optimized for memory consumption. By only storing the non-zero entries along with their positions the memory needed is reduced to $O(nm)$ where m is the number of non-zero values per row.

In most cases memory optimization is taken even further such that contiguous patches of non-zero values are summarized into dense submatrices occupying certain index ranges of the actual matrix. This way there is no need to store the indices of each individual entry.

While there are many software packages offering these kinds of data structures there are only few that also offer functions that perform efficient calculations with them. [11] gives a comprehensive summary of recent and deprecated software packages that

actually perform efficiently on sparse matrices with a focus on eigenvalue decomposition. In the course of this work SLEPc [12] which is based on PETSc [3] was used to perform the calculations. Although these libraries are written for handling large matrices in parallel using the Message Passing Interface (MPI), they also offer very efficient sequential algorithms.

Approximating Ncut

Despite Ncut being proven to be NP complete there is a method also developed in [21] which is known to give good approximations of an optimal bipartition:

Using the definitions of Section 2.3.1 let D be a diagonal $n \times n$ matrix with $D_{ii} = d_i$. The normalized graph Laplacian is defined as $L = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$. Solving for the λ_i of the eigen system $Lx = \lambda x$ results in n eigenvalues and eigenvectors which have to be sorted in ascending order of λ_i. Thresholding the eigenvector of the second smallest egenvalue λ_2 in the form $l_i = \Theta(x_i)$ whith Θ being the Heaveside function yields the desired bipartitioned labelling of the points in point cloud P.

Shi and Malik propose two ways in order to make Ncut usable for data clustering in general and image or point cloud segmentation in particular. These application domains demand the method to create a k-partition of the data.The most obvious way of obtaining a k-partition from a function performing bipartition is to recursively reapply the function on the resulting clusters until some termination criterion is met.

An alternative way is based on the assumption that additional partitioning information is encoded in the eigenvectors x_i following the one of λ_2. Thus for each point $p_i \in P$ it is possible to define a q-dimensional descriptor $\delta_i = [x_{i2}, \ldots, x_{iq+1}]$ that can be applied for k-means clustering.

2.3.2 The Nyström Method

As described in Section 2.3.1 the computation of spectral clusters entails solving a matrix eigenvalue problem of the Laplacian L which is of same size as W. It has been shown by Pan et. al. [17] that regarding runtime complexity the matrix eigenvalue problem can be reduced to matrix multiplication. Although considerably fast algorithms exist for large matrices e.g. $O(n^{\log_2 7})$ [22] which outperform naïve approaches solving for eigenvalues must be considered time intensive operation.

Looking at the way how eigenvectors are used by spectral clustering reveals however that exact solutions are not needed. For example for the first recursive approach only the correct sings are of interest making it possible to use approximation methods for determining the eigenvalues instead of solving the actual problem.

The following derivations are taken from [9]. Some comments have been altered or

added for better understanding. An example for an eigenvalue eigenvector approximation method is the Nyström extension. Its original purpose is according to Fowlkes et. al. to find "numerical approximations to eigenfunction problems"[9, 3. The Nyström Extension]. What eigen functions are can be seen in the following equation:

$$\int_a^b W(x,y)\phi(y)dy = \lambda\phi(x).$$

Using the quadrature rule the integral is transformed into a sum. The quadrature rule however only approximates the integral between a and b.

$$\frac{b-a}{n}\sum_{j=1}^{n} W(x,\xi_j)\hat{\phi}(\xi_j) = \lambda\hat{\phi}(x) \tag{2.1}$$

Therefore, $\hat{\phi}(x)$ is the approximation of $\phi(x)$ – the eigenfunction of W. In order to solve the equation x also needs to be discretized with the $\xi_1, \ldots, \xi_n$ which yields a system of equations.

$$(b-a)A\hat{\Phi} = n\hat{\Phi}\Lambda \tag{2.2}$$

Please note that $A_{ij} = W(\xi_i, \xi_j)$ and $\hat{\Phi} = \left[\hat{\phi}_1, \ldots, \hat{\phi}_n\right]$ are the n eigenvectors of A with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$. Solving the eigen problem allows to reinsert the λ_i and ϕ_j into Equation 2.1 and solve it for $\hat{\phi}(x)$. This yields the Nyström Extension:

$$\hat{\phi}_i(x) = \frac{b-a}{n\lambda_i}\sum_{j=1}^{n} W(x,\xi_j)\hat{\phi}_i(\xi_j) \tag{2.3}$$

What happened so far? Two discretization steps are used to transform the integral eigenvalue problem into a system of linear equations. By sampling the function W approximate samples of the eigenvectors and eigenvalues are created. The reinsertion of these results into Equation 2.1 yields Equation 2.3 which can be used to extend the eigenfunction samples found by solving Equation 2.2 to any x.

In order to transfer the Nyström Extension to the Ncut problem the affinity matrix W is interpreted as discretized version of the function $W(x,y)$. The process of sampling the integral is imitated by taking a random sample of the input data $S \subset P$ to generate a dense sample affinity matrix A containing the affinities of all $p \in S$. The equivalent of $W(x,\xi_j)$ is obtained by creating a matrix B which contains all affinities between the points p and $q \in \bar{S} = P \setminus S$. As a consequence $W(\xi_i, x)$ can be seen as $B^\top$.

As it is done in Equation 2.2 A is decomposed into $A = U\Lambda U^\top$. Using the decomposition the left hand side of Equation 2.3 can also be recreated by $B^\top U\Lambda^{-1}$ which allows to determine approximate eigenvectors for all unsampled parts of W. For better un-

derstanding W can be rewritten such that the $p \in S$ occupy the lowest indices. Please note that C in this matrix relates to all unknown affinities.

$$W = \begin{bmatrix} A & B \\ B^\top & C \end{bmatrix} \tag{2.4}$$

Extending all Eigenvectors it is possible to define an appoximate eigenvector matrix of W.

$$\bar{U} = \begin{bmatrix} U \\ B^\top U \Lambda^{-1} \end{bmatrix} \tag{2.5}$$

Using $\bar{U}$ one can again approximate W with $\hat{W} = \bar{U}\Lambda\bar{U}^\top$. Inserting the definition of $\bar{U}$ from Equation 2.5 results after some expansion steps in

$$\hat{W} = \begin{bmatrix} A & B \\ B^\top & B^\top A^{-1} B \end{bmatrix}. \tag{2.6}$$

So it can be said that the Nyström Extension approximates the unknown C with $B^\top A^{-1} B$.

The eigenvectors in $\bar{U}$ however are not orthogonal. Fowlkes et. al. propose a method on how to obtain othogonal eigenvectors of $\hat{W}$ if A is positive definite. They define a matrix $S = A + A^{-\frac{1}{2}} B B^\top A^{-\frac{1}{2}}$ with its eigen decomposition being $S = U_S \Lambda_S U_S^\top$. For S they prove in [9, Appendix A] that

$$V = \begin{bmatrix} A \\ B^\top \end{bmatrix} A^{-\frac{1}{2}} U_S \Lambda_S^{-\frac{1}{2}} \tag{2.7}$$

contains the orthogonal eigenvectors of $\hat{W} = V \Lambda_s V^\top$.

For the Normalized Cut there is still work to do. In order to create the graph Laplacian the row sums $\hat{W}\mathbb{1}$ of $\hat{W}$ are needed. The sums can be calculated without evaluating the very large matrix C, because $C\mathbb{1} = B^\top A^{-1} B \mathbb{1} = B^\top A^{-1} \mathbf{b}_c$, with $\mathbf{b}_c$ being the column

Algorithm 2.3 Nyström Spectral Clustering

Construct A from a random sample $S \subset P$
$D \leftarrow \mathrm{diag}(\hat{W}\mathbb{1})$ ▷ remember $C\mathbb{1} = B^\top A^{-1}\mathbf{b}_c$
$\bar{A} \leftarrow D^{-\frac{1}{2}} A D^{-\frac{1}{2}}, \bar{B} \leftarrow D^{-\frac{1}{2}} B D^{-\frac{1}{2}}$ ▷ normalize A and B
$S \leftarrow \bar{A} + \bar{A}^{-\frac{1}{2}} \bar{B} \bar{B}^\top \bar{A}^{-\frac{1}{2}}$ ▷ assure orthogonal eigenvectors
$[U_S, \Lambda_S] \leftarrow \mathrm{EVD}(S)$ ▷ eigenvalue eigenvector decomposition
$V \leftarrow \begin{bmatrix} \bar{A} \\ \bar{B}^\top \end{bmatrix} \bar{A}^{-\frac{1}{2}} U_S \Lambda_S^{-\frac{1}{2}}$ ▷ approximate k eigenvectors of $\hat{W}$
Normalize row vectors of V to magnitude 1 ▷ improve k-means performance
Perform k-means using the row vectors of V

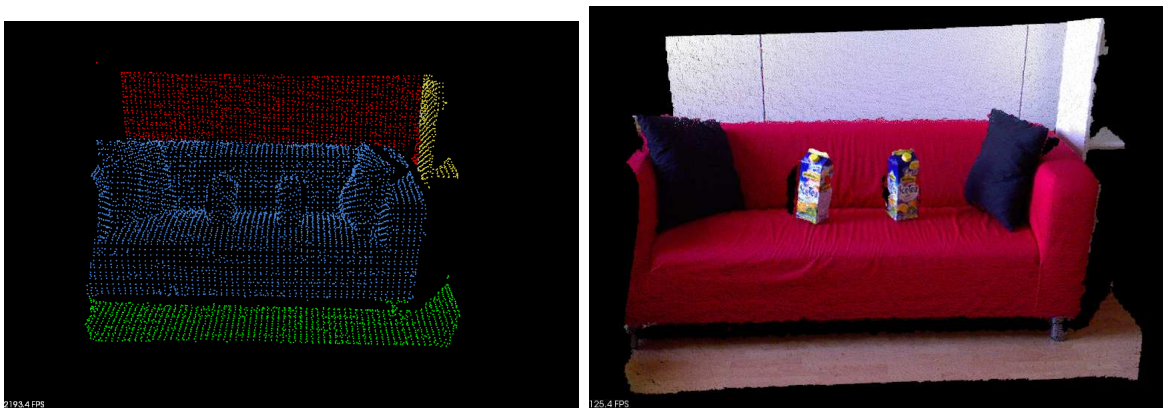


Figure 2.5: Example of spectral clustering segmenting a scene.

sum of B. In order to create $L = D^{-\frac{1}{2}}\hat{W}D^{-\frac{1}{2}}$ the Blocks A and B must change to

$$\bar{A}_{ij} = \frac{A_{ij}}{\sqrt{\hat{d}_i \hat{d}_j}}, \qquad i,j \in \{1,\dots,n\}$$

$$\bar{B}_{ij} = \frac{B_{ij}}{\sqrt{\hat{d}_i \hat{d}_{j+m}}}, \qquad i \in \{1,\dots,n\}, j \in \{1,\dots,m\}$$

The algorithm stated in [6, p. 10] briefly summarizes all the necessary steps to perform Spectral Clustering using the Nyström method. It is rewritten here in Algorithm 2.3 for consistency of variable names. See Figure **??** for an example of a spectral clustering result.

2.4 Expectations

In this section the selection of algorithms is explained along with the criteria used in the process. Applying the theory from Sections 2.1, 2.2 and 2.3 expectations are formulated for the algorithms with a focus on conditions upon which two or more algorithms complement each other.

In light of the goal to prepare a highly versatile set of segmentation algorithms this selection of methods is a disputable choice of course. However, there exists no applicable taxonomy of segmentation approaches which would allow to select the correct number and types of algorithms to guarantee a desired degree of versatility. Therefore, the main guideline for selecting the algorithms is a high degree of heterogeneity of the methods. This particular course of action is based on the assumption that segmentation approaches processing the input data in a similar fashion will show strongly related performance profiles and thus do not increase the versatility of the algorithm set.

Despite the lack of a taxonomy, paper research shows that all segmentation algorithms are grouped around central more general processing approaches. Many of today's 3D segmentation methods are based on existing processing algorithms such as clustering approaches that have been developed long before they became interesting for this particular task. Specialization is achieved by extending the general approach or by refining its result in subsequent processing steps. Therefore, the first criterion for selecting a segmentation approach according to the heterogeneity guideline is to only choose algorithms originating from different general approaches.

In addition to the general method further criteria are applied to differentiate the algorithms. The next important criterion is the type of data being processed by the algorithm. Since the sensor used for this work is capable of providing color information for each point in addition to the plain Cartesian coordinates it is natural to prefer algorithms bringing this information to use. The expected run time of the algorithm is also a major influence on the selection process as well as the estimated time needed for implementation.

2.4.1 Scene Interpretation

ScI will serve as a reference algorithm for comparison with the other algorithms presented in this chapter. It is the latest evolution of a basic approach which is the one most commonly used in Service Robotics. See [23], [20] or [10] for a selection applications. Object manipulation and thus object recognition are very important tasks in this field of research. Therefore, a wide variety of segmentation approaches are now available. They are based on the assumption that manipulable objects are located on planar surfaces such as tables, desks and counters. The assumption is true for many environments service robots are expected to work in.

In contrast to the other segmentation methods discussed in this chapter ScI is a local segmentation approach. Therefore, only a selection of points which is deemed worthwhile is labeled in the process while the rest is dismissed as being the background. The selection is based on the detection of planar surfaces in the scene. Thus every algorithm performs a planar surface detection using Sample Consensus techniques.

While such algorithms perform very well in the cases where planar surfaces are detectable they are very certain to fail when this is not the case. A major weak point of planar segmentation is data reduction achieved by down-sampling of the data. In order for planar segmentation to run with high frame rates the down-sampling is a necessary step as long as using GPU implementations is not an option. Reducing the data however becomes an increasingly worse problem as more and more objects populate the planar surfaces. The objects take away necessary information from the Sample Consensus step by occluding the surface points behind them. Given the normal based filtering done in ScI no pollution of the data by points belonging to the objects is to be

expected. However, the decreasing number of surface points seen by the sensor forces the user to lower the threshold on the number of points needed to accept a planar surface. This in turn leads to the acceptance of even very small surface patches which are likely to be the result of sensor noise.

A strongly related effect is that planar surfaces are cut into two or more parts by the occlusion of the objects on top. The resulting planar surfaces are correctly detected however they do not support the objects anymore. Only the gaps between the objects are passed to the hierarchical clustering step which is therefore unable to find the object clusters.

2.4.2 Color-based Segmentation

Color-based Segmentation is the most obvious example for an extension of a pre-existing clustering approach. It introduces an additional colorimetric similarity criterion to Euclidean Clustering. Of course there are further extensions to the Euclidean Clustering approach such as the one proposed by Rabbani et. al. [18] which adds a surface smoothness criterion.

The main reason for choosing Euclidean Clustering based algorithms is their simplicity. They are quickly understood and therefore easy to implement compared to the other algorithms presented in this Chapter. As a result the effects of the algorithm's parameters can be well anticipated. Color-based Segmentation is chosen over the approach of Rabbani et al. due to the processing of color information. The results presented by Zhana et al. are also more convincing since they show that for colored objects the problem of adjacent or connected objects being merged was largely reduced by their approach whereas the segmentation examples of Rabbani et al. could not prove that the smoothness constraint is able to do the same.

The most obvious failure condition for CbS is the case where adjacent objects have similar colors near the object borders. For such cases the gap assumed between objects due to the use of XYZRGB space does not exist. This inevitably leads to under-segmentation.

Since all data is being sampled down to a resolution of 1 cm^3 – which is also done for the color information by replacing the points inside the 1 cm^3 voxel with a point having the mean color – for objects with textures consisting of many small batches of different colors a homogenizing effect can be expected. This leads to a low level of color based over-segmentation of objects. The same is true for objects having a predominant color in their textures or only soft color transitions. Objects which are difficult to segment with CbS will have large batches of different colors that are separated by sharp color transitions. These textures generate unwanted gaps in XYZRGB space leading to over-segmentation.

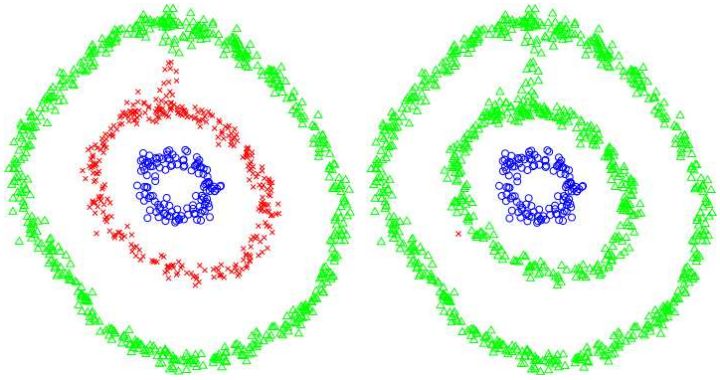


Figure 2.6: On the left:Example of Spectral Clustering dealing with proximity. On the right: Result of Euclidean Clustering on the same dataset. Source [16]

The way in which Color-based Segmentation and Scene Interpretation are complementary is now evident. The cases for which planar surface detection works insufficiently due to a scene cluttered with objects, CbS is expected to outperform ScI as long as the objects in the scene expose favorable textures. Conversely for scenes with objects having ill-conditioned textures ScI is anticipated to outperform CbS as long as the planar surfaces beneath the objects are detected well enough.

2.4.3 Spectral Clustering

Spectral Clustering is a generalized approach in itself. Multiple applications of the basic approach are already mentioned in Section 2.3. Two methods of application are explained in order to perform multi-cluster segmentation. The Nyström Extension being an efficient method for approximating eigenvectors shows that specialization is not always done in order to improve segmentation results on given scenes, but also to decrease runtime.

Spectral Clustering and Euclidean Clustering are related approaches. The implementation of Spectral Clustering in this work uses the same input data as Color-based Segmentation and therefore also makes use of color information. Their names imply that they both perform a type of clustering of the input data. However, the fashion in which the clusters are obtained are not related at all. Therefore the heterogeneity criterion is fulfilled. The paper research brought to light a variety of interesting examples exposing the capabilities of SpC as can be seen in Figure 2.6. These examples suggest that the shortcomings of CbS can be compensated while preserving the advantages that CbS has over Scene Interpretation.

SpC's benefits of course come at a cost. Despite the use of the Nyström Extension it is expected to be slow compared to Scene Interpretation or Color-based Segmentation. This is owed to the fact that although the Matrix A is relatively small matrix B

still is very large. Many matrix multiplications and multiple eigen-solving steps are performed.

However, setting the speed criterion aside it, seems as if Spectral Clustering is capable of solving a superset of the problems being solvable using CbS. While it is essentially true that Euclidean Clustering based algorithms are outperformed by Spectral Clustering there remain the problems of approximate solutions and non-determinism. As discussed in Section 2.3 the method proposed by Shi et al. only gives approximate results. The same is true for the Nyström Extension which additionally relies on random sampling. So when trying to solve the Ncuts problem what is actually done is the non-deterministic approximation of an approximation of the real solution. Therefore, in the sense of repeatability, it is not expected to be as reliable as ScI or CbS and thus only applicable if neither ScI nor CbS are expected to successfully segment the scene.

Chapter 3

Segmentation Quality

Measuring segmentation quality is a cornerstone of this work. First it is necessary to understand how segmentation quality is measured in the 2D world and it will be shown further that such methods are also applicable for 3D segmentation. However, these methods are very time consuming and require a large human workforce since basic information is collected from human subjects. Therefore an alternative method for segmentation quality measurement is presented which matches the scope of this work.

Given the goal to train a classifier that is supposed to choose the best segmentation algorithm for a given scene, correctly labelled data is needed. The labels identify the best parameter set for a given scene.

3.1 The 2D World

The most famous effort to measure 2D segmentation algorithm performance is the Berkeley Segmentation Dataset and Benchmark [14]. It is a $1,000$ image subset of the Corel image database which is widely used by the computer vision community. In total the dataset contains eight man made segmentations per image. Four of them are segmented from grayscale and the other half is segmented from color images. Special hardware, special software and a simple set of rules is used to simplify and accelerate the process for the subjects.

As anticipated by Martin et. al. human segmentations differ between the subjects. The test persons are given plenty of scope to choose the granularity of their segmentation which is only bounded by the rule that the segments should have approximately equal *importance*. Proof of a high degree of consistency between the subjects is needed in order to verify that these segmentations are valid. A metric capable of indicating consistency is therefore defined. The Berkeley Segmentation Benchmark uses a local

non-symmetric error measure.

When comparing two segmentations S_1 and S_2 for each pixel p_i the containing region (or segment) $R(S, p)$ is determined. The error is then defined as

$$E(S_1, S_2, p_i) = \frac{|R(S_1, p_i) \setminus R(S_2, p_i)|}{|R(S_1, p_i)|}.$$

Two metrics are derived from the error measure that consider all points in an image. The Global Consistency Error (GCE) and the Local Consistency Error (LCE). Both form a mean of E. The only difference between them is that GCE minimizes between the Error directions $E(S_1, S_2, p_i)$ and $E(S_2, S_1, p_i)$ gobally and LCE does so locally on a per pixel basis.

Martin et. al. are able to show that for human segmentations of the same image these metrics are very low whereas they become close to 1 whenever the segmentations of two different images are compared. This naturally proves the validity of LCE and GCE and shows the consistency of human segmentations at the same time.

Unfortunately manual segmentation of 3D data is nearly impossible due to the lack of visualization and selection methods which would allow the efficient definition of volumes in point clouds. However manual annotation methods for 2D images can easily be used in the 3D world as well.

Today's sensors like the Kinect use a two-dimensional depth map in order to compute the point cloud with a set of calibration parameters. For such sensors it is trivial to transform back and forth between 3D and 2D data representations. Being able to reduce 3D segmentation to 2D segmentation, the annotation techniques used for the Berkeley Segmentation Dataset are also applicable for this work. Unfortunately the hardware and the software used in the process of creating the dataset are custom made. Their specifications have not been published.

3.2 Recognition Quality

Since basic information cannot be collected from human experts it is not possible to directly measure the segmentation performance. The idea for an alternative approach is sparked by Martin et al. who state in [14]:

> *"It is considerably easier to quantify the performance of recognition algorithms than segmentation algorithms".*

The statement is verified by the fact that recognition techniques rely on the minimization of some quality measure. It is safe to say that the definition of a performance measure quantifying recognition performance is inherent to object recognition. So the

question is: How can the measurement of object recognition quality help in determining the performance of a segmentation algorithm and what are the necessary conditions?

Obviously, a strong coupling between recognition quality and segmentation performance is vital to such an approach. Global description techniques – i.e. techniques that describe complete object clusters in contrast to describing single points – as for example the Viewpoint Feature Historam (VFH) [19] or Bounding Box expose such a strong coupling.

Taking Bounding Box as example, the direct relation between recognition quality and segmentation quality can be seen immediately: In the case of oversegmentation bounding boxes are smaller than the model bounding box whereas in the case of undersegmentation the bounding boxes are too big. Both cases lead to increasing error measures in the recognition process or even to false recognitions when segmentation keeps becoming worse.

There are also object recognition methods that do not explicitly rely on well segmented input data or which do not need segmentation at all. Such methods are for example based on registration in combination with a keypoint extraction approach. For these methods recognition runtime *could* be a performance indicator. A major effect of segmentation is data reduction resulting in faster execution of registration based algorithms. However the relation between the runtime of such algorithms and segmentation performance was not further investigated because registration runtime also heavily depends on the dataset itself. A low signal-to-noise-ratio is expected making it nearly impossible to seperate ill-conditioned input data and bad segmentation.

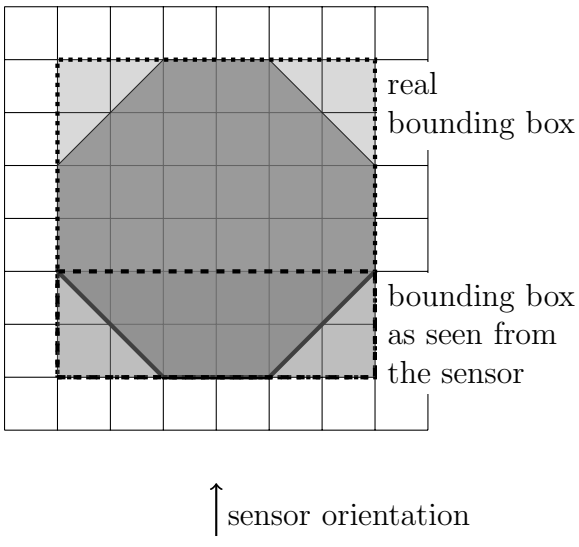


Figure 3.1: Bounding Box yielding bad recognition results despite perfect segmentation.

This reduces the list of recognition approaches to the ones using global description of object clusters. Bounding Box and Viewpoint Feature Histogram are the only global description methods available without further implementation overhead. Since no work package has been allotted for the implementation of recognition algorithms one of the two has to be chosen.

Reason for disqualifying Bounding Box is given by the fact that an object's orientation relative to the sensor is not considered during training. As shown in Figure 3.1 there are view points from which it is impossible to estimate the correct bounding box of an object. This happens when the front facing the sensor covers parts of the object from the sensor's view that are needed to estimate it's bounding box. VFH in contrast does not show this viewpoint specific behaviour because it does not need information about the objects real size. The VFH description technique allows to arbitrarily choose object and sensor positions when testing segmentation performance. Therefore, VFH is used to determine segmentation performance in this work.

3.2.1 A probabilistic quality measure for VFH

In this section it is discussed how the VFH quality measure is defined and why it cannot be used as is to quantify segmentation performance. Bayesian reasoning is used to transform VFH's quality measure into a form that will allow for comparison between different view angles. It is shown how this method can be applied to compute an object recognition confidence value using a normalization criterion. The object confidence measure however does not prove to be a as useful for measuring segmentation performance as an intermediate result obtained in the process.

The Viewpoint Feature Histogram is a vector containing 308 floating point values. See Figure 3.2 for an example. Given an input point cluster $C = \{c_1, \ldots, c_m\}$, for each point c_i a surface normal is estimated using a k-neighbourhood of points. For each normal n_i the deviation angles α_{ij}, ϕ_{ij} and θ_{ij} are computed which rotate the normal to a k-neighbourhood of normals n_j. The results are then combined with the viewpoint direction $v_p - c_i$, v_p being the sensor position. The values of α, ϕ and θ are then discretized into equally sized value ranges. Binning the rotation angles of each point given the value ranges results in the Viewpoint Feature Histogram. Rusu et. al. prove empirically that VFH has a high discriminative power making it possible to effectively discern up to 60 objects with $90 \times 6 = 540$ view angles each.

For recognition an object model containing a sufficiently large number of descriptors of different view angles is needed. Recognition is performed by generating a descriptor of a new point cloud and searching for its nearest neighbour within all models that have been trained. Having found the nearest neighbouring descriptor not only the object itself is known but also its orientation relative to the sensor. The recognition performance measure inherent to VFH recognition is a distance measure for two Viewpoint

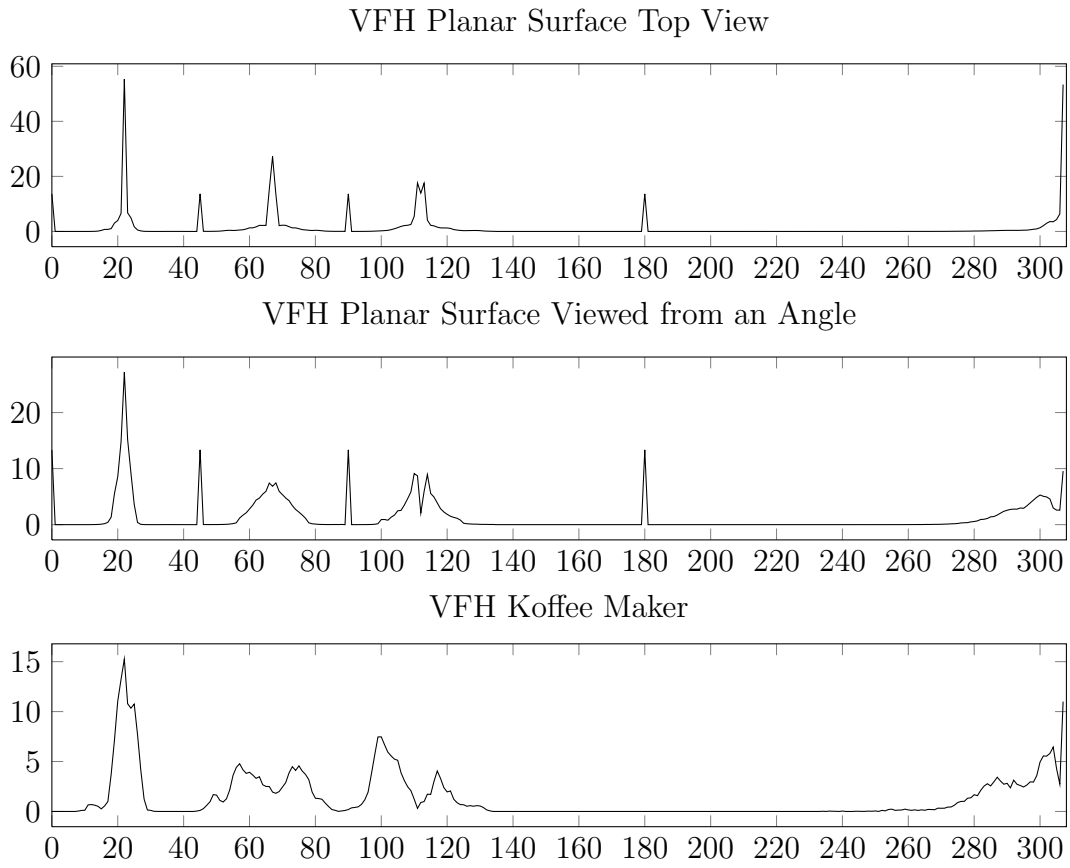


Figure 3.2: Viewpoint Feature Histograms of multiple objects and view points.

Feature Histograms. Exposing the highest performance in finding similarities between histograms the Chi Squared Distance is chosen as proposed by Rusu et. al..

Unfortunately, the distance does not result in comparable values for different view angles. The value range is different of each descriptor. Distances also vary slightly from scan to scan due to sensor noise. So the benefit of VFH described in Section 3.2 – which is that objects can be arbitrarily placed in a scene – seems to vanish. However due to VFH's discriminative power a high certainty of finding the correct model descriptor within the k nearest matches[1] remains. For Bounding Box this is not the case.

The task is to transform the distance measures in a fashion that makes them comparable although they result from scans of different view points. A probabilistic approach is chosen because probabilities are perfectly comparable. The value of interest is $p(D = \delta | E = \epsilon)$ – the probability of the descriptor match δ actually representing the scanned object and view point under the condition of getting a Chi Squared Distance ϵ. ϵ is a

[1] Where $k \ll n$ is much smaller than the total number of descriptors in all models.

continuous random variable. Therefore, it is impossible or at least tedious to gather the basic information needed to directly compute $p(D = \delta|E = \epsilon)$. Bayes' rule is used instead:

$$p(D = \delta|E = \epsilon) = \frac{p_{\text{pri}}(E = \epsilon|D = \delta)p_{\text{pri}}(D = \delta)}{p_{\text{pri}}(E = \epsilon)}. \tag{3.1}$$

The prior probabilities $p_{\text{pri}}(E = \epsilon|D = \delta)$, $p_{\text{pri}}(D = \delta)$ and $p_{\text{pri}}(E = \epsilon)$ are needed to compute the desired value. The method used in order to obtain these values is discussed in Section 3.2.2. By applying the same reasoning step as in 3.1 to each of the k best matches in the database, allows to determine the most probable match given its error. So instead of using the best match, considering distance ϵ, as result the most probable one is taken.

Being a result of the matching process, ϵ is known during recognition. Therefore, $p(E = \epsilon) = 1$ is valid permitting use marginalization in order to solve for $p(D = \delta)$.

$$p(D = \delta) = \underbrace{p(D = \delta|E = \epsilon)p(E = \epsilon)}_{=p(D=\delta|E=\epsilon)} + \underbrace{p(D = \delta|E \neq \epsilon)p(E \neq \epsilon)}_{=0} \tag{3.2}$$

Further, assuming that k has been chosen large enough such that the probability of the correct descriptor being in the set of best matches is approximately 1, one can normalize the probabilities with the criterion in Equation 3.3. Please note that this summation neglects all undoubtedly existing dependencies of the form $p(D = \delta_i|D = \delta_j)$, where $i \neq j$.

$$\sum_{i=1}^{k} p(D = \delta_i) = 1 \overset{normalization}{\Rightarrow} p_n(D = \delta_i) = \frac{p(D = \delta_i)}{\sum_{i=1}^{k} p(D = \delta_i)} \tag{3.3}$$

For the normalization in Equation 3.3 the assumption is made that the set of k best matches is certain to contain the correct match. In order to validate the assumption the desired level of certainty has to be specified. However, a trade-off has to made between the desired certainty and the time needed for validation. For this work a certainty of 0.999 is chosen since it is possible to be verified in about 16 hours. Validation entails generating 1000 test descriptors from sensor data which takes about 24 seconds per descriptor. This has to be repeated for all descriptors in the database. Applying the process to several k values shows that a k of 20 robustly delivers the desired certainty.

In order to obtain the object confidence value $p(O = \omega)$ the marginalization in Equation 3.4 is used. Herein δ represents a descriptor of the model for object ω.

$$p(O = \omega) = p(O = \omega|D = \delta)p(D = \delta) + p(O = \omega|D \neq \delta)p(D \neq \delta) \tag{3.4}$$

The posterior probability $p_n(D = \delta)$ is inserted for $p(D = \delta)$. $p(O = \omega|D = \delta)$

is approximately 1. No two descriptors in the database are equal. VFH does not qualify as a highly discriminative descriptor otherwise. With n_d being the number of all descriptors in all models and $n_{d\omega}$ being the number of descriptors in the model for object ω, Equation 3.4 can be rewritten to:

$$p(O = \omega) = p_n(D = \delta) + \frac{n_{d\omega} - 1}{n_d}(1 - p_n(D = \delta)) \tag{3.5}$$

Equation 3.3 is needed to conclusively determine a confidence value for recognizing an object. However, there are also disadvantages related to the normalization step which will be discussed in Section 3.2.3.

3.2.2 Gathering Prior Knowledge

As seen in Equation 3.1 prior knowledge is needed to perform probabilistic reasoning with the VFH error measure. This knowledge has to be obtained during training by examining the error behaviour for each descriptor in the database. The current training process needs to be altered slightly in order to generate recognition performance data on-the-fly after the model descriptor is generated. Every model descriptor is supposed to represent a small range of object orientations relative to the sensor. Therefore, the object is rotated within that range. Further descriptors are generated while ro-

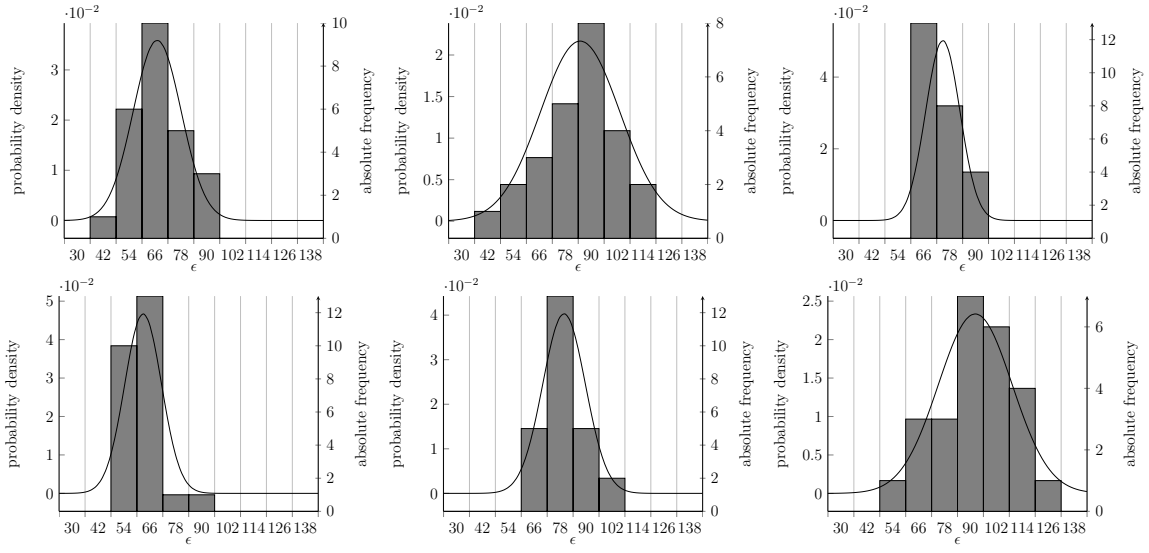


Figure 3.3: Error histograms indicating a normally distributed error. Each histogram represents the error behaviour of a single descriptor. The abscissa denotes the error ϵ in a range between 30 and 150. Assuming a normal distribution, the ordinate on the left represents the probability density value for an error, whereas the ordinate on the right stands for the absolute frequency of errors within sub-ranges of size 12.

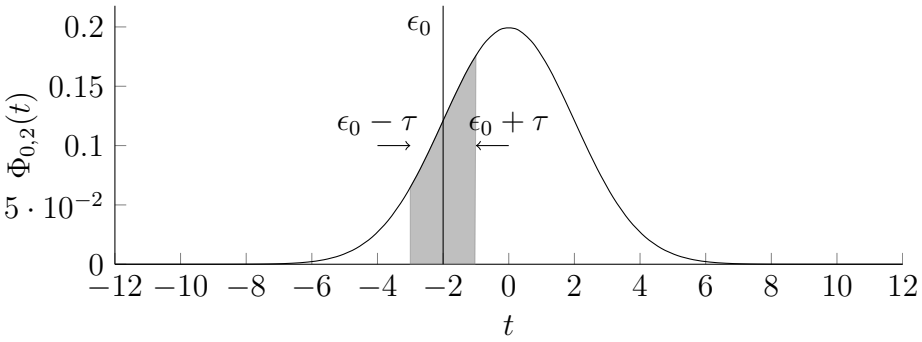


Figure 3.4: Calculating the probability of ϵ_0 given that descriptor δ_0 has been matched.

tating and the Chi Squared Distance to the model descriptor is computed for these orientations.

The histograms shown in Figure 3.3 are indicative for a normally distributed error. For that reason the errors' means and standard deviations are used to describe recognition performance on a per descriptor basis. Given a matched descriptor δ_0 and the error of the match ϵ_0 it is now possible to also extract the error mean μ_0 and standard deviation σ_0 for recognizing the descriptor. The probability of having the match error given the descriptor δ_0 is then given by:

$$p(E = \epsilon_0 | D = \delta_0) = \int_{\epsilon_0 - \tau}^{\epsilon_0 + \tau} \Phi_{\mu_0, \sigma_0}(t) dt, \tag{3.6}$$

as plotted in Figure 3.5, where $\tau > 0$ denotes a small range of values around ϵ. The τ range is needed here since the probability of getting an instance of a continuous random variable is zero. $p(E = \epsilon)$ can be determined in the same fashion. For μ and σ the combined mean and standard deviation of the recognition error is determined over all descriptors in all models.

However it is possible to remove τ from further considerations since it is not $p(E = \epsilon_0 | D = \delta_0)$ that we actually need. For clarification Equation 3.1 is rewritten:

$$p(D = \delta | E = \epsilon) = \frac{\int_{\epsilon_0 - \tau}^{\epsilon_0 + \tau} \Phi_{\mu_0, \sigma_0}(t) dt}{\int_{\epsilon_0 - \tau}^{\epsilon_0 + \tau} \Phi_{\mu, \sigma}(t) dt} p_{\text{pri}}(D = \delta) \tag{3.7}$$

The $\pm\tau$ range is an approximation of the actual probability. While the limit of $\tau \to 0$ vanishes for $\int_{\epsilon_0 - \tau}^{\epsilon_0 + \tau} \Phi_{\mu_0, \sigma_0}(t) dt$ and $\int_{\epsilon_0 - \tau}^{\epsilon_0 + \tau} \Phi_{\mu, \sigma}(t) dt$ it does not for the Quotient in Equation 3.7. It resolves to $\frac{\Phi_{\mu_0, \sigma_0}(\epsilon_0)}{\Phi_{\mu, \sigma}(\epsilon_0)}$. With the prior probability of matching descriptor δ_0, which is $p_{\text{pri}}(D = \delta) = \frac{1}{n_d}$, where n_d is the total number of descriptors in the

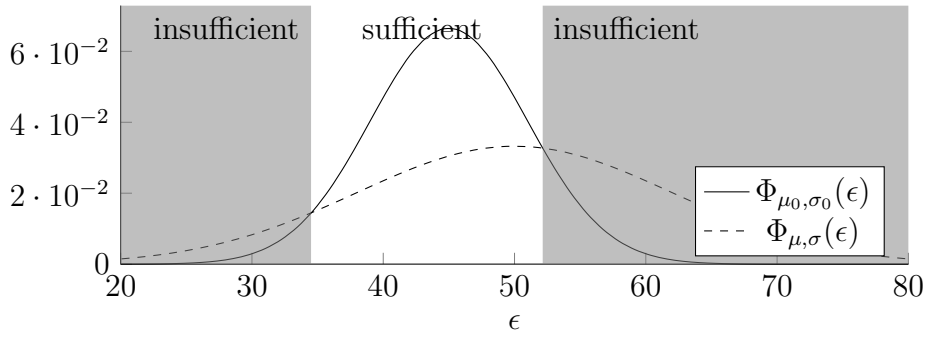


Figure 3.5: Deciding between sufficient and insufficient matches. The points where the two density functions intersect delimit the error range for sufficient matches.

database Equation 3.1 is simplified to:

$$p(D = \delta | E = \epsilon) = \frac{\Phi_{\mu_0,\sigma_0}(\epsilon_0)}{\Phi_{\mu,\sigma}(\epsilon_0)} \frac{1}{n_d}. \tag{3.8}$$

3.2.3 Normalization Issues

The normalization which has been formulated in Equation 3.3 unfortunately has undesired effects. The intuition leading to the normalization is that in the set of k best matches there is only one descriptor δ_0 which has a high probability of being the correct match. All other descriptors are assumed to have considerably lower probabilities such that $p(D = \delta_0)$ is the main contribution to the sum. The contributions of all descriptors which are not in the set of k best matches are considered negligibly low.

In the case of over-segmentation parts of the information needed to create a descriptor similar to the one in the database are unavailable. For under-segmentation additional information is provided also alienating the generated descriptor from its counter part in the database. Due to VFH's discriminative power the assumptions made above still hold true. The most probable neighbouring descriptor still contributes the major share to the sum of probabilities. The complete sum however is very small, while the normalized probability of the most probable match is heavily amplified. In fact the normalized probability only contains information about how good the best match is compared to the other $k - 1$ matches. Information about the absolute quality of the match which is encoded in $p(D = \delta | E = \epsilon)$ is lost. Therefore $p(D = \delta | E = \epsilon)$ is used to quantify segmentation quality instead of $p_(D = \delta | E = \epsilon)$.

There is also a case for which these assumptions do not hold true. In the case of viewing an unknown object the assumption $\sum_{i=1}^{k} p(D = \delta_i) = 1$ is falsified. The correct descriptor cannot be found in the database because it has not yet been trained. So there must be a lower limit to the match probability in order to reject bad matches.

In order to solve the issue it must be possible to discern true positive recognitions of varying quality from false positive recognitions. For this work an intuitive criterion was chosen. In Equation 3.8 the prior probability of matching descriptor δ_0 is set to $\frac{1}{n_d}$. This value is multiplied by a Quotient $Q = \frac{\Phi_{\mu_0,\sigma_0}(\epsilon_0)}{\Phi_{\mu,\sigma}(\epsilon_0)}$. Therefore, the probability of a correct match is lower than or equal to the prior probability as long as $Q \in [0..1)$. Such matches are interpreted as insufficient since only negative confidence can be gained from them. Insufficient matches, where $\Phi_{\mu_0,\sigma_0}(\epsilon_0) < \Phi_{\mu,\sigma}(\epsilon_0)$ are rejected during recognition which effectively avoids false positive recognitions for unknown and also known objects.

3.3 A Segmentation Quality Metric Based on Object Recognition

In this section the process of obtaining segmentation quality measurements from object recognition results is described and a mathematical definition for a segmentation quality metric is proposed.

This work requires segmentation quality measurement to be feasible for a single person. Therefore, man-made annotations of test scenes must be kept as scarce as possible. Since object recognition is the basis for this approach it the simplest method to only annotate the objects contained within the scene. For this work this method is slightly extended by defining that the order in which the objects are annotated is also the order how the objects are positioned in the scene from left to right from the sensor's point of view. This extension allows minimizing the effects of false positive identifications of known objects while introducing almost no further annotation overhead. On the implementation side a small overhead is to be expected, since the recognition output also has to be ordered from left to right.

A few limitations have to be made with respect to the position of the objects. Of course all objects must be inside the field of view of the sensor. Additionally there is a further limitation that is imposed by the training. During training for this work the objects are scanned from six different elevations resulting in view angles in the range between 20 and 45 degrees. It has to be expected that recognition performance will severely drop when positions are chosen such that they are not within that view angle range.

Deteriorating recognition results are to be expected as well for increasing distances to the object. During training the sample resolution is artificially reduced by a down-sampling step to one point per cubic centimeter. For areas in the scan where this resolution is surpassed all points within a 1cm^3 cube are replaced by their center of gravity. Therefore, the down-sampling process smooths the data and removes noise. Due to the perspective projection inherent to pattern projection sensors, sample resolution decreases with increasing distance to the sampled object. Subsequently there

is a distance beyond which sample resolution is lower than the training resolution. Beginning with this distance there is less information available to compute the descriptor than there was during training which leads to increasing differences in object description.

It is tempting to use the distance of equal resolution between training and recognition as maximum distance for recognition. It is also easily computed from the optics's focal length, sensor size and sensor resolution. However, the true maximum distance permitting reliable recognition is lower: For increasing distances the down-sampling, which is also done for recognition, uses less points to calculate the center of gravity. The result is therefore more susceptible to noise. At some point only the raw input data is used because the point distance is larger than the down-sampling grid. As the smoothing effect of down-sampling diminishes with increasing distance, the noise amplitude of the sensor increases. These effects however have not been thoroughly investigated. Instead the absolute maximum distance for recognition of 3.2 meters has been determined experimentally This is the maximum distance for which all objects trained are recognized in more than 50% of the cases while in the rest of the cases there were no false positives.

Given these restrictions scenes are constructed which contain the trained objects. These scenes are used to compare the performances of two or more segmentation algorithms as well as for multiple parametrizations of the same segmentation algorithm. The score S of an algorithm for a scene is computed by summing up the quotients $Q = \frac{\Phi_{\mu_j,\sigma_j}(\epsilon_j)}{\Phi_{\mu,\sigma}(\epsilon_0)}$, with $j = 1, \ldots, m$ where m is the number of correctly recognized objects. The reason why $p(D = \delta_j | E = \epsilon_j)$ is not used is that the difference between the two values is only a constant factor $\frac{1}{n_d}$. Given the rejection criterion from Section 3.2.3 each Q is normalized to the Interval $[0 \ldots 1]$. Therefore, the maximum score of an algorithm is bounded by m.

$$S = \sum_{j=1}^{m} Q_j \tag{3.9}$$

Chapter 4

Optimization and Analysis

4.1 Scene Selection

In preparation of parameter optimization and algorithm analysis scenes are chosen according to the anticipated strengths and weaknesses of the available algorithms discussed in Section 2.4. The scenes are mostly taken from an artificial household environment that has been constructed in the lab. In addition the scenes are annotated applying the scheme proposed in Section 3.3. For this work 36 scenes have been recorded using 100 scans each. The scenes are subdivided into ten groups. Within each group the actual setting stays the same while the objects in the scene are changing locations or are replaced by other objects. These variations are introduced to avoid over-fitting for special objects or object locations when fine tuning the parameters using a brute force method. For the following discussion the ten groups are subdivided into four classes which relate to the algorithms' anticipated performance profiles.

The preparation also entailed object training as described in detail in Appendix B. The object database contains five objects. Please note that in the following the objects are referred to by their database names as shown in Table 4.1. For this work multiple instances of the same object can occur in a scene. Since the object recognition approach

Object	**Database Name**
coffee maker	KAFFEEMASCHINE
coffee pot	KAFFEEKANNE
chips can	PRINGLES
ice tea carton	PFANNER
box of chocolates	CHOCLAIT-CHIPS

Table 4.1: Mapping between database names and obejects.

is shape based there is no way of differentiating different objects of the same shape. For example an ice tea carton will be referred to as PFANNER without discrimination although one carton may be peach flavoured and the other lemon flavoured.

4.1.1 Tabletop Settings

Since planar segmentation techniques like Scene Interpretation are the most widely used approaches in Service Robotics ScI is the reference algorithm and represents the Status Quo of what is currently the standard segmentation capability. Tabletop settings are the type of scene for which these algorithms are optimized. Therefore, they are the reference scenes which expose the strengths of the standard approach.

Considering the scenes depicted in Figure 4.1: Please note, that these images have been taken taken from a 3D point cloud viewer where the points have been assigned colors as seen by the RGB-Camera. The four scenes on the left, which are arranged on a kitchen table, are the best case scenario for ScI. Large portions of the table surface are seen by the sensor. All visible parts are connected. This ensures that the planar segmentation is able to detect the complete table surface and therefore, all objects it supports. Another benefit of the scene is the high elevation angle of the sensor which results in less occlusion behind the objects and also more sample points per area unit.

The scenes on the right hand side are designed to be slightly more demanding. By lowering the sensor elevation less sample points of the sideboard's top surface are sampled. Furthermore, objects have been also placed a little closer to each other. This is especially the case for the two scenes at the top and bottom on the left, where `PFANNER` and `KAFFEEKANNE` and `PFANNER` and `KAFFEEMASCHINE` respectively are placed in close proximity. Nevertheless, it still is possible to find and connect all parts of the sideboard's surface. Therefore, good segmentation results are expected for Scene Interpretation.

Figure 4.1: Left: Four tabletop settings using a standard kitchen table. Sensor elevation is 35 degrees. Right: Five tabletop setting using the top of a sideboard. Sensor elevation is 25 degrees. Each scene contains three known objects that are more or less arranged in a straight line at the center of the planar surface they stand on.

Another interesting property of the scenes on the right is that the top shelf of the sideboard is visible and will be segmented as another planar surface as well. For these cases, where planar surfaces occur in a stacked fashion, the hierarchical clustering is expected to be proven useful. No parts of the objects placed on the top surface should be segmented redundantly.

4.1.2 Tabletop Settings with Disturbance

The next class of scenes as shown in figure 4.2 is designed to be challenging for Scene Interpretation. The scene with the red couch at the top does not contain a planar surface due to the warped seating surface of the couch. Therefore, it is to be expected that the normal based filtering, which is done before planar segmentation, is going to remove many points of the surface supporting the objects.

Since the couch is behaving like a plain red background, there is reason to assume for Colour Based Segmentation to outperform ScI. The colors of the objects and the couch are sufficiently dissimilar. This is especially true for the coffee maker in the two right most scenes of the top image. However, it is unclear how the colourful textures of the other objects like `PFANNER` or `PRINGLES` are going to behave. The question is whether the texture pattern has a sufficiently fine granularity such that the mean colors computed by the down-sampling process are similar enough for no over-segmentation to occur.

The scenes at the bottom are taken from a desk. They contain unknown objects such as key boards and a mouse that cover large portions of the desk's surface. While the mouse

Figure 4.2: Top: Couch scenes. The surface is warped. Bottom: A desk scene where the keyboards and the mouse are unknown objects occluding parts of the desk's surface.

is most certainly removed by the normal based filtering step of Scene Interpretation, this is not necessarily the case for the keyboards. The normals are computed from a surface fit using a k-neighbourhood of each point. The keys on a key board are sufficiently close to form planar surfaces. Due to the Euclidean Clustering step, which is applied after filtering the point cloud by normal direction, points of the table will be merged with points of the keyboards. This produces noise in the data passed to RANSAC reducing the probability of finding the correct plane model.

Although there is no homogeneous background in the desk scene, as there is for the red couch, CbS can be expected to outperform Scene Interpretation in those cases where – due to the unknown objects – the detection of planar surfaces fails. Of course the texture based restrictions apply.

4.1.3 Cluttered and Occluded Scenes

Cluttered Scenes as they are shown in Figure 4.3 are expected to be nearly impossible to segment using planar surface detection methods. Certainly the supporting surface planes can always be detected when viewing the scene from above. Robots however, are often incapable of positioning the sensor directly above the scene. From the scenes

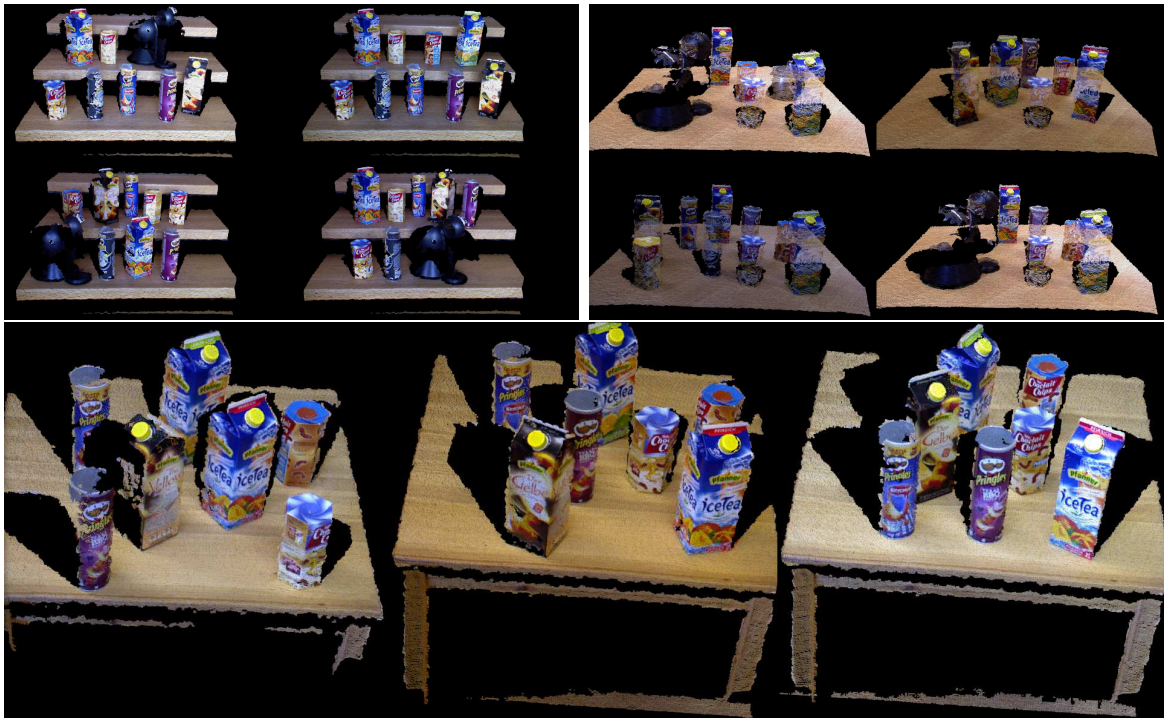


Figure 4.3: Top left: Objects on a flight of stairs. Top right: Objects on the floor. Bottom: Tabletop scene. All scenes are cluttered and contain multiple instances of the trained objects.

depicted in Figure 4.4 it can also be seen that even manipulator mounted sensors can not be positioned on top due to obstructions. Furthermore, since the object training done for this work only permits recognition of objects from a maximum elevation of approximately 45 degrees, top views are not suitable for recognition. Therefore, the Color Based Segmentation approach is expected to perform significantly better on these scenes than Scene Interpretation.

Clutter can also be achieved by placing only few objects in a very confined space such as a drawer or a box. While clutter and mutual occlusion certainly are challenging factors for Scene Interpretation in the scenes on the top right and bottom of Figure 4.4, the major difficulty is the occlusion of the planar surface by the front wall and front panel of box and drawer. Thus Scene Interpretation lacks the major capability of segmenting objects inside containers. Another unfavourable property of containers which is also present in the top left scene is the presence of side walls. Although Sample Consensus model fitting is very accurate it is never optimal. The plane model will always be slightly tilted. Therefore, one of the side walls is often recognized as object on the surface because the surface is tilted in its direction. As pointed out for the disturbed tabletop settings it is also difficult for the cluttered scenes to estimate whether the textures of the known objects are favourable for the approach. Spectral Clustering is assumed to be capable of outperforming CbS at the cost of bad repeatability. These

Figure 4.4: Top left: Objects on a shelf. Top right: Objects in a box. Bottom: Objects in a drawer.

scenes are expected to be a well chosen basis to test and compare the capabilities of the two approaches.

4.2 Generic Parameter Sets

Choosing between different approaches is one particular way to increase the segmentation capability, another method is to choose from multiple parameter sets per algorithm, thereby increasing the segmentation capability of each approach. As for the set of algorithms selected in Chapter 2 the goal is to achieve a high degree of versatility when choosing the generic parameter sets. However, a major difference between the processes is that the decisions for parametrization are not as final as those for the segmentation approaches. While the implementation of an algorithm is a major investment in terms of work load, leaving no margin for testing a wide range of methods, the parametrization only takes a few minutes and can even be automated. This permits the determination of the generic parameter sets by means of experimentation.

Finding the generic parameter sets involves the following three steps: First a rough tuning of the parameters, which is done manually. Applying the knowledge from Chapter 2 many parameters can be well estimated. The initial parametrization is also used to perform an initial evaluation of the algorithms. For the main part this evaluation consist of testing whether the different approaches are behaving as anticipated. For some parameters their influence will be unclear especially when they are interacting; i.e. when changes of one parameter strongly influences the behaviour of another parameter. The second step is a brute force optimization. It is used to find value combinations that show good segmentation performances. Based on the brute force optimization the best parameter sets are chosen using a voting based approach.

4.2.1 Manual Parametrization

There are by far too many parameters for the three segmentation algorithms to discuss them all. The parameters that are used by all algorithms and their values are be explained in this section. Furthermore, the most important parameters of CbS, ScI and SpC will be discussed.

One of the limitations of the VFH object recognition approach covered in Section 3.3, is the small field of view in which objects can be reliably recognized. Therefore all algorithms share a common preprocessing step which removes all points lying outside this field of view. For simplicity reasons the field of view is modelled as an axis-aligned box which can be defined by minimum and maximum values along the x, y and z axes. The virtual box used for this work is one meter in width, height and 1.1 meters in depth. It is centered in x and y direction. In z direction it is shifted by 0.4 meters. This is necessary since the sensor cannot reliably measure distances below 0.4 meters

leading to erratic behaviour in this range.

Down-sampling is a second processing step that all algorithms share is. All data passed to the segmentation algorithms first undergoes voxel grid filtering. The leaf size is set to 1 cm^3, which means that all points inside a 1 cm^3 volume will be replaced by their center of gravity. For points that contain further information like color the same is done for the additional components.

Parametrizing Color-based Segmentation

Manual parametrization of CbS shows that no parameter set can be found that will perform well for very colorful objects like `PRINGLES`, `PFANNER` and `CHOCLAIT-CHIPS`. This lack of the parameter set has been anticipated. Their unfavourable texture leads to over-segmentation whereas `KAFFEEKANNE` and `KAFFEEMASCHINE` are perfectly segmented.

However, some parameter sets show a very interesting behaviour. When setting the colorimetric threshold for both the points and the regions to considerably low levels the segmentation looks like a negative of the desired result. While the background is still segmented correctly, there are holes where the objects should be. This happens because the over-segmentation effect is amplified to a level that even after merging and refinement the segments are very small where the objects are located. The original paper proposes to merge very small segments to a greater neighbouring region whereas the implementation for this work discards them.

Both ways of handling the small regions yield unwanted results. Therefore, the algorithm is changed to collect all of the small regions and put their points into a single point cloud. This cloud subsequently undergoes a euclidean clustering step. The result is small segments that are close to each other are used to form larger segments regardless of their color. Thereby, the missing objects are reconstructed. The regions generated this way are added to the segments found by ordinary Color-based Segmentation.

The two most important parameters of CbS are the colorimetric distance thresholds. They are an upper bounds for the euclidean distance of the colors of two points or regions. CbS does not transform the 24-bit RGB values to the $[0 \dots 1]$ range. The maximum difference between two colors is therefore $255 \times \sqrt{3} \approx 442$, which is the difference between black and white. In order to obtain the over-segmentation necessary for the extended Color-based Segmentation approach to work properly the point-to-point colorimetric distance threshold has to be as low as 10 - 30. For merging and refinement the region-to-region threshold must be chosen even lower (4 - 7) since these are represented by their mean color.

Another key element to the over-segmentation based extension of CbS is the definition of what a small segment is. CbS defines segment size by the number of points inside

the segment. A cluster representing a correctly segmented object roughly contains 400 points depending on the object's size. Therefore the minimum segment size has to be considerably smaller, approx. 10% of the nominal segment size. Multiple tests show that indeed best results are achieved when choosing the minimum segment size between 30 and 50 points.

Also important is the euclidean distance threshold for points since CbS is euclidean clustering based. The point-to-point distance threshold encodes information about the relative locations of objects within the scene. Increasing the threshold leads to under-segmentation because adjacent objects are merged. Decreasing the threshold leads to over-segmentation of self-occluding objects like `KAFFEEMASCHINE`. However, from previous work with euclidean clustering good parameters are already known to lie within the range between 1 centimeter and 5 centimeters.

Parametrizing Scene Interpretation

In contrast to Color-based Segmentation Scene Interpretation worked as expected. ScI is designed to be a parameterless segmentation algorithm. Despite its many processing steps, which of course need various parameters, there should be no need to tune them for a specific application. A parameter set applicable for one scene should ideally be applicable for all scenes. This is basically true given the fact that Scene Interpretation very restrictively defines the type of scene for which it applies. However, the goal of this work package is to soften the restrictions by showing that using multiple parameter sets permits good segmentation results even for scenes that would otherwise be questionable cases.

Some leverage points toward that goal are already identified in Section 2.4.1. They are all related to the planar surface detection step, since this is the primary point of failure of the approach. The first important parameter in this context is the down-sampling resolution. In addition to the down-sampling already performed on the data during preprocessing, ScI performs another voxel grid filtering step after normal based filtering. Lowering the resolution decreases the processing time considerably but it also leads to less planar surfaces being detected. The down-sampling resolution encodes the user's expectations about how clearly the surfaces are visible in the scene. For very distinct surfaces ScI will succeed even for very low resolutions while scenes where the planar surfaces are occluded may also fail for considerably high resolutions. Since planar segmentation proves to be already fast enough even for high resolutions, down-sampling is set to values between 2 cm^3 and 4 cm^3.

After down-sampling, the planar point candidates which have normals pointing upward are clustered applying a euclidean clustering step. The most important parameter in this step is the distance termination criterion discussed in Section 2.1. Obviously the distance threshold must be chosen larger than the down-sampling resolution. So due to the upper bound for down-sampling being 4 cm^3 the lower bound for the threshold

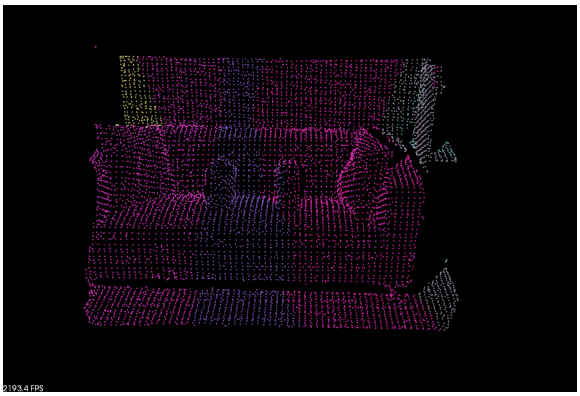


Figure 4.5: Example of a usual spectral clustering result. The clusters bear no meaning.

is 5 centimeters. The upper bound encodes the minimum distance between two planar surfaces to be recognized as separate. Manual measurements in the recorded scenes showed that this distance should never exceed 15 centimeters.

As for CbS there also exists a minimum cluster size threshold for euclidean clustering. The process discards all segments having a point count below the threshold. In terms of planar surface detection this is the number of points visible per surface. When filtering the points of the input cloud by their normal direction, upward pointing normals are found in various places that are not expected. This is due to the normals being estimated from a neighbourhood of eight points. Even very irregular shapes can therefore generate upward pointing normals which happens particularly often in combination with sensor noise. Very good segmentation performance can be achieved when choosing the minimum cluster size between 100 and 200 points.

Parametrizing Spectral Clustering

Unfortunately Spectral Clustering using the Nyström Extension is a great disappointment. Although the segmentation results are expected to have a lower reliability than the other algorithms it was not expected that the rate of good segmentation results would be as low as approximately every 200 scans. This low rate makes it impossible to do any manual parameter tuning because there is a very long waiting period between changing the parameter set and seeing the result. A usual result of spectral clustering cab be seen in Figure 4.5. It is a more or less regular pattern that is in no way connected to the real world objects inside the scene.

Nevertheless Spectral Clustering segmentation results are astonishing in case they work well. The results displayed in Figure 4.6 sparked the idea that Spectral Clustering could be an ideal candidate for the hierarchical segmentation approach. As can be seen the segmentation algorithm is not influenced by the objects that are placed in and on the shelves. The pieces of furniture are segmented together with their contents or the

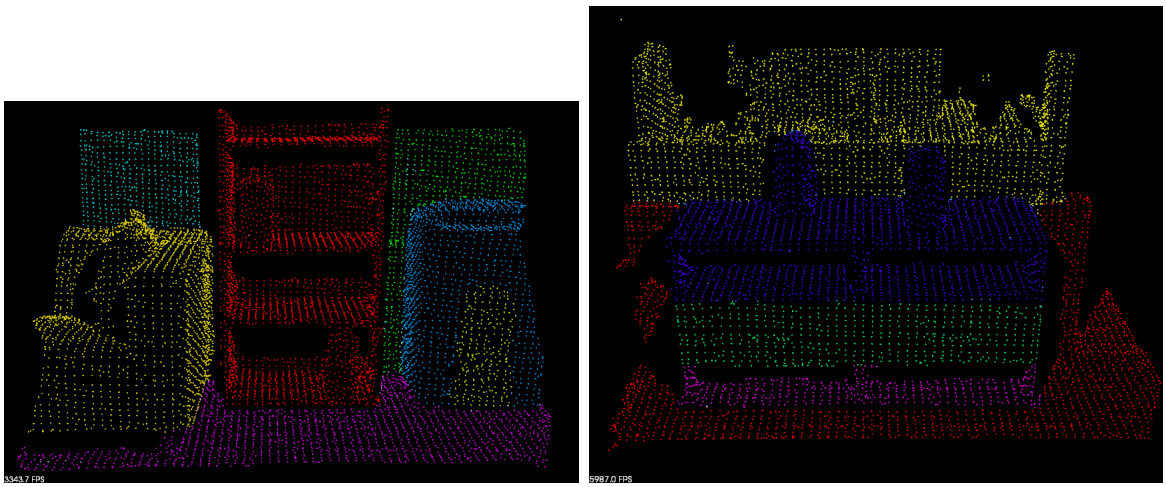


Figure 4.6: Examples of Spectral Clustering working well.

objects on top of them. However with such results being only generated every 200 runs it is not applicable for any task.

The reason for the bad performance most probably lies with the low sample size of just 50 sample points that has to be chosen for Spectral Clustering to perform segmentation in little under a second. Test runs using twice as many samples do not improve the reliability by a measurable amount. However they come at the cost of seven seconds processing time per point cloud. Tests using 200 and more samples were aborted prematurely because processing takes already almost a minute per frame.

The only remaining method to improve the number of samples that can be processed in a reasonable amount of time is to choose a parallel implementation for the GPU. A very effective CUDA-based eigenvalue/eigenvector-solver is presented in [2]. However the published source code turned out to be too old for the current hardware. Runtime errors were constantly encountered due to incompatibilities of the software with the relatively new graphics card.

4.2.2 Brute Force Optimization

For most of the parameters that were roughly tuned during manual parametrization, applicable value ranges were found instead of single values. Many of the parameters are highly interdependent. Thus they cannot be optimized one by one, which is the only manually feasible optimization method. An automatic way of finding the best parameter set for a scene has to be found.

Finding the best parameter set is essentially a search task. The search space is determined by the number n_p of parameters that need to be optimized and by the number of values each parameter is allowed to assume. Given that there is a fixed number of

values n_v each parameter can assume, the size of the search space is $n_v^{n_p}$. Manually identifying the most influential parameters and identifying their best value ranges, as it is done in Section 4.2.1, is a necessary precondition, since the search space is growing exponentially with the number of parameters. It is not feasible to search the whole space of averaged 15 parameters per algorithm for each scene. The number n_v of allowed values per parameter also has a strong influence on the search space. Particularly problematic are real valued parameters as well as integer valued parameters with large value ranges. For these parameters the value ranges must be discretized to keep the search space within a workable size.

For Scene Interpretation three important parameters are identified. Each of the parameters is assigned three allowed values within the ranges determined during manual tuning. This totals 27 parameter sets being generated for each scene. Four parameters are used for experimentation with Color-based Segmentation. The lists of allowed values are ranging between two and four entries. A total of 180 experiments are generated per scene. Given the 36 scenes that have been recorded, for ScI a total of 972 experiments are performed whereas for CbS there are 6480 experiments.

In order to automate the search for the best parameter set per scene, a brute force approach is applied. The segmentation processes are implemented using the BOR3D framework. BOR3D offers a file configuration mechanism using the JavaScript Object Notation (JSON) file format. Each segmentation process is equipped with a data acquisition component that loads previously stored point clouds from a source directory which is given by the configuration file. The test scenes from Section 4.1 are stored in a directory tree where each of the 36 scenes has its own folder. Every folder is occupied by 100 files containing point clouds of the respective scene. An instance of a BOR3D process is only able to use one parameter configuration at a time.

For this work a process instance is called an experiment. An experiment is defined by a configuration file that contains a combination of allowed values for the parameters and a path to the source directory of a scene. The experiments are automatically generated from a configuration file template. Additionally, an experiment definition file contains the parameters that must be changed for each experiment along with a set of permitted values for each parameter. A script reads the information in an experiment definition file and produces as many versions of the template configuration as there are permitted value combinations. Each experiment is placed in a separate working directory. A shell script is used to sequentially execute the BOR3D process for each configuration.

For manual parameter tuning the segmentation result is inspected by looking at a visualization of the segmentation result. This is not feasible any more since there are simply too many results that have to be inspected. Instead, the scoring approach proposed in Chapter 3 is implemented. The required annotation files are placed inside the folders which contain the point clouds of the scenes. After segmentation, recognition and score calculation for each recognized object the process compares the recognition

#	annotation	recognition
1	3 2 4 1	3 2 5 1 4 1
2		3 2 3 2 4 1
3		1 2 5 4 1 3
4		4 1 5 4 3 2

Table 4.2: Examples of an annotation of a scene and some recognition results that need to be mapped to the annotation. The numbers represent database names of known objects. Please note, that particularly bad results were artificially constructed for demonstration purposes.

result to the contents of the annotation file. The total score of all correctly recognized objects represents the segmentation score which is written to a score file inside the experiment's working directory. For each scan of the scene there is one entry in the score file. Using the scenes recorded for the purpose of this work there are 100 entries per score file.

Mapping Annotations and Recognition Results

As discussed in Section 3.3 the scenes are only annotated with the database names of the objects in the scene. The sequence of the names is that of the objects' occurrences in the scene from left to right. The recognition process is extended to output the objects' positions in the scene so they can be sorted according to the annotation by the scoring step. Subsequently the sequence of object names in the annotation are compared to the sequence of object names within the sorted recognition output.

The process of comparing the sequences is explained here because it is not as trivial as expected. For shorter notation integer numbers are used to define annotations and recognition results. In order to understand how the current comparison algorithm was derived the evolutionary steps will be described briefly along with the rationale why they are not suitable for segmentation scoring.

The simplest approach to map an annotation to a recognition result is to compare the annotated sequence one by one with the recognized sequence and check whether they are equal. Considering the first row of Table 4.2 this method determines that objects 3 and 2 are correctly recognized whereas the rest of the sequence does not match the annotation. Therefore, only the scores of these two recognitions are counted although objects 4 and 1 are also recognized in correct sequence. Multiple tests showed that recognitions such as those of objects 5 and 1 mostly happen due to unknown objects in the background being segmented and falsely recognized as known objects. For such cases this trivial approach generates segmentation scores that are too low.

In order to deal with these cases the simple approach needs to be extended such that

it does not stop as soon as a mismatch between the sequences occurs. The next development iteration starts by searching the first annotation element in the recognition sequence. When it is found it searches for the next annotated object in the rest of the recognition sequence. This is continued until either the end of the annotation or the end of the recognition is reached. Such an approach finds the sequence **3 2** 5 1 **4 1** and correctly scores the first row of table 4.2.

The disadvantage of this greedy matching method is shown by the second row of Table 4.2. The current version is bound to find **3 2** 3 2 **4 1**. However looking at the scenes, it is seen that the objects occur in bulk. Thus it is much more probable that 3 2 **3 2 4 1** is the correct result. Therefore, a criterion is introduced that prefers the shortest matching sequence in recognition. This shortest sequence is found by deleting the first element from recognition as shown in Table 4.3 and rerunning the search for the annotation sequence. This is done for as long as the annotation sequence can be found.

The method however still fails in some cases: Consider the third row for which it would only find 1 2 5 4 1 **3** whereas the correct solution most probably is 1 **2** 5 **4 1** 3. Obviously the approach is unable to deal with annotated objects that have not been recognized. In the unfortunate case of the second row this already happened for the first annotated object, thereby dismissing the correct matches for objects 2, 4 and 1.

To counter the effects of not recognized objects a recursive implementation is chosen. Keeping the current practice of ignoring false positives within the recognition it is now demanded that the algorithm finds the complete annotation sequence. If the annotated sequence cannot be found, a set of sub-sequences is generated, each of which accounts for one of the missing annotated objects. To these sub-sequences the algorithm is applied recursively. As soon as the recursion reaches the second level – which means that two objects cannot be recognized – redundant checks of sub-sequences are performed. A global object maps annotation sub-sequences to matching sequences in the recognition. So, if a recursion finds a sub-sequence in recognition it adds the result to the global object and terminates. If the same subsequence has already been found by another recursion step, the result is simply overwritten. An additional termination criterion is given by the demand to find matches that contain as many annotated

Call #	**Input**	**Result**
1	**3 2** 3 2 **4 1**	Found it! Next...
2	2 **3 2 4 1**	Found it! Next...
3	**3 2 4 1**	Found it! Next...
4	2 4 1	Not found! 3 Wins!

Table 4.3: Finding the shortest match. The first element is removed from the input each time the sub-sequence is found.

objects as possible. Therefore, a recursion also terminates if the sub-sequence it is processing, is smaller than a sub-sequence that has already been found in recognition. If a recursion finds a larger subsequence in recognition than those that have already been found it deletes all the entries in the global object and inserts its own result.

All results in the global object by design relate to equally sized sub-sequences in the annotation. In most cases there will actually be only one result. The condition that can lead to multiple results is that recognition containing multiple equally sized sub-sequences of the annotation in a wrong order. The fourth row of Table 4.2 is an example of such a case. There is no way of telling whether **4 1** 5 4 3 2 or 4 1 5 4 **3 2** is the correct result. Therefore, such a result is scored with the mean score of the alternatives.

The current algorithm uses all the extensions described in this Section and therefore finds the shortest sub-sequence in recognition containing the most annotated objects in correct sequence.

4.2.3 Voting Based Set Selection

Due to the large number of experiments being performed, evaluation also needs to be automatized. A script traverses the experiment folder structure and collects the data from the score files. Since segmentation and recognition performance varies between scans, the mean score is computed from each file. This reliably scores the performance of a parameter set on a given scene.

A matrix is created from the data. It contains the score that each parameter set achieves for each recorded scene. This is done for each algorithm. For consistency it is defined that the columns of the matrices relate to the parameter sets while the rows relate to the scenes. Due to their size the matrices can be confusing, making it difficult to extract information. They are therefore not printed in this document. Instead charts are used to visualize the results in a more readable format.

The data format needed to determine a versatile set of generic parametrizations is a performance profile which shows how well a parameter set copes with the scenes compared to other parametrizations. Such a profile, as shown in Figure 4.7, has peaks and valleys. When inspecting the profiles of multiple parameter sets, it is expected that the peaks and valleys do not always occur for the same scenes. The goal is to find a combination of parameter sets that has as few and as little valleys as possible. The combination should also be kept small since it is not the goal to add parameter sets which add just one scene to the algorithm's segmentation capability.

Profiles can be quickly determined using the score matrix by extracting the columns. However, the scores of different scenes are not comparable because the scenes contain different numbers and kinds of objects. It would therefore be hard to say whether a peak is a peak or a valley is a valley. Normalization is needed for the rows to have

all column entries in the same value range. The maximum row value marks the best performing parameter set for the scene. A simple normalization can therefore be done by transforming the row values into percentages of the maximum row value.

Yet, there are 27 performance profiles for ScI and 180 performance profiles for CbS. Selecting complementary parameter sets from these many candidates is still a demanding task. Further help is needed to narrow down the search space to a level that can be handled manually.

As shown in Figure 4.7 the parameter set performs well on multiple scenes. The same observation can be made for almost all of the parameter sets of both Color-based Segmentation and Scene Interpretation. From another viewpoint a similar observation can be made. When determining which parametrization performs best on a given scene there is usually a set of parametrizations having the same or similar scores. Parameter sets achieving high scores for many scenes are very likely to be generic. The observation therefore suggests the implementation of a voting mechanism. A vote counter is initialized to zero for each parameter set. Furthermore, for each scene the highest scoring parameter set is determined along with additional parametrizations that perform in the range of over 99% of the high score. The vote counter of each parameter set determined in this way is incremented, resulting in a chart as it is shown in Figure 4.8. For Scene Interpretation only parametrizations achieving a vote count greater that ten are considered in the search for generic parameter sets.

The same voting approach is also used to narrow down the search space for Color-based Segmentation. However, for CbS there are much more parameter sets to be tested than there are scenes. As a result a total of 28 parametrizations achieve the maximum vote count of four. A run-off vote is performed for these 28 parameter sets which reduces the selection to four parameter sets. Three of them are chosen as the generic parameter

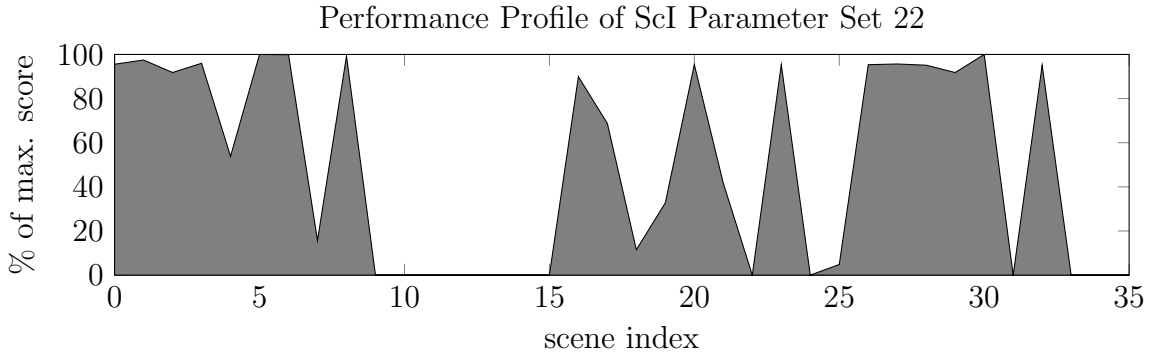


Figure 4.7: Example of a performance profile taken from one of the ScI parameter sets. Please note that all values are relative only to the maximum score achieved by Scene Interpretation. They do not allow any conclusions for the absolute segmentation performance.

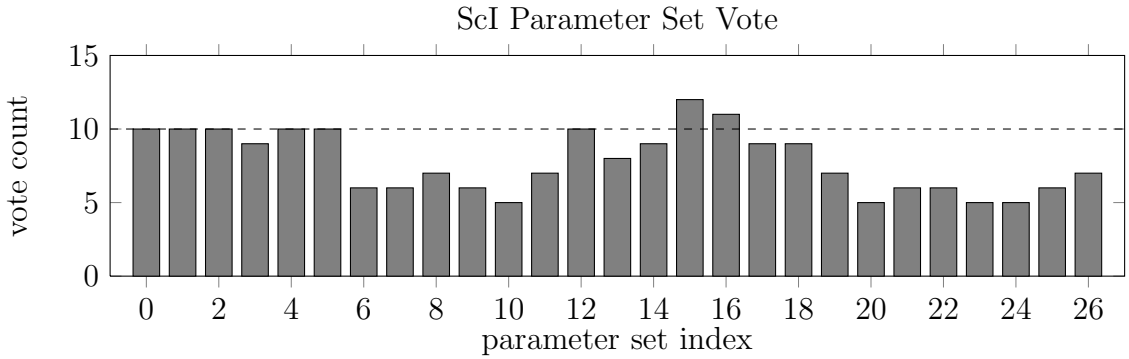


Figure 4.8: The ScI parameter set voting result. Only parameter sets receiving ten or more votes are considered. These are the sets 0, 1, 2, 4 and 5 as well as 12, 15 and 16.

set for Color-based Segmentation.

Please see Figure 4.9 for the resulting performance profiles of Color-based Segmentation and Scene Interpretation respectively.

4.3 Segmentation Results

In this section the segmentation results generated by the experiments are discussed. First it is determined which parameter set of which algorithm is to be used for which scene. This means the combined performance profile composed of the generic parameter sets for Color-based Segmentation and Scene Interpretation needs to be generated. This profile is a perfect basis for comparing the expected algorithm performances formulated in Sections 2.4 and 4.1 to the experimental results. A high consistency between expectations and the performance profile is a strong indication for the validity of the proposed segmentation approach, the choice of implemented algorithms and of the recognition based quality metric. However, particularly interesting are the inconsistencies. They need to be traced back either to flaws in the approach or to false expectations. Since all performance profiles depicted in this document are only relative to maximum scores achieved for the scenes, a last step of validation is taken by manually inspecting the recognition performance using visualization.

4.3.1 Experiment-Expectation-Consistency

In Figure 4.10 the performance profile of Scene Interpretation is printed in red whereas the one of Color-based Segmentation is printed in green. Wherever the red color is visible a parameter set for ScI has performed better than any parameter set for CbS. The diagram is therefore ideal to confront expectations and experimental results. For better understanding a Table 4.4 is provided which maps the scene indices to the

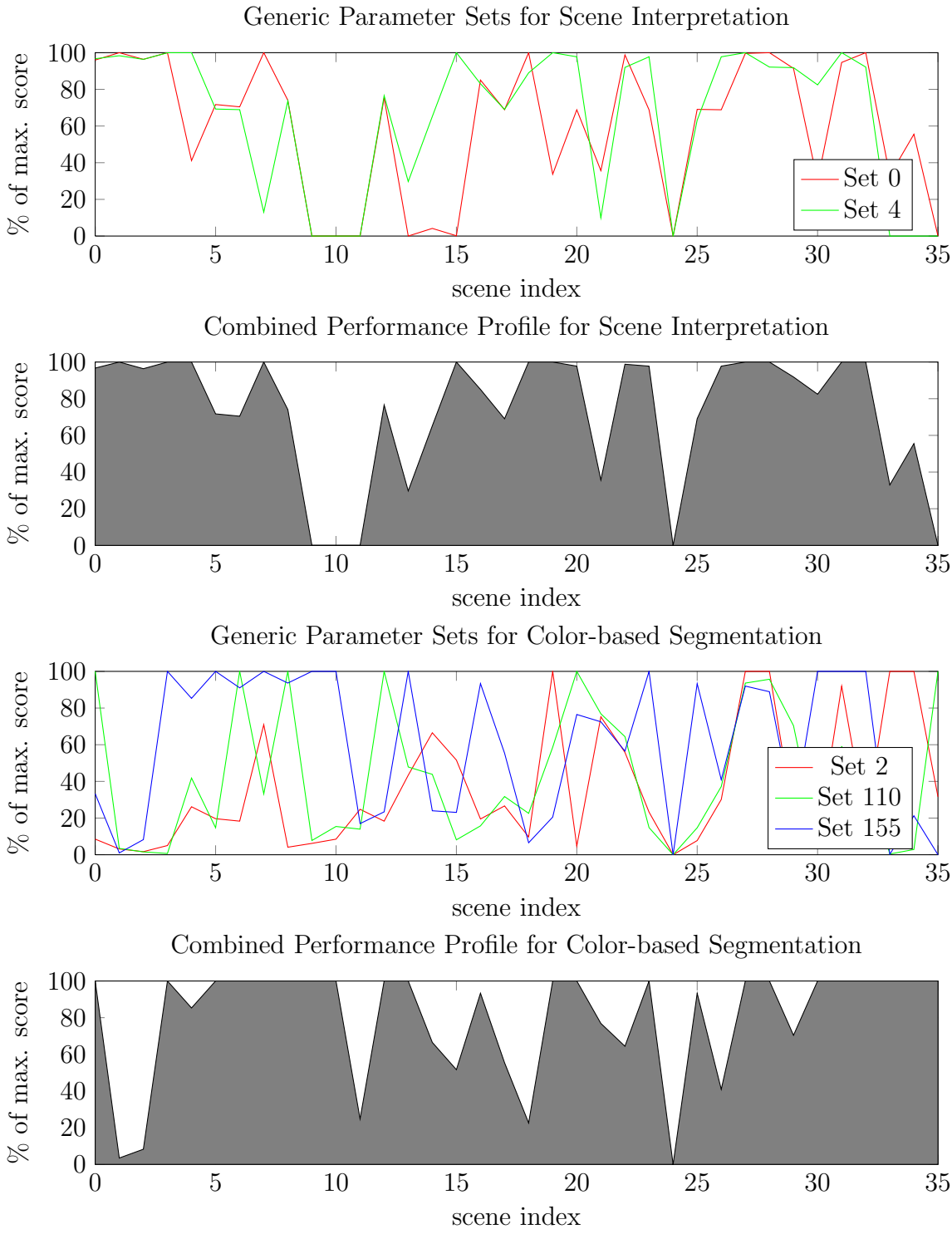


Figure 4.9: Performance profiles for the generic parameter sets of Scene Interpretation and Color-based Segmentation. The combined performance profiles show how optimal the algorithm can operate if the correct parameter set is chosen for each scene.

scenes. The scenes are the discussed in the same order as it is done in Section 4.1. Scene boundaries are printed in Figure 4.10 as black vertical lines for better orientation.

The scans 0 - 4 relate to the tabletop setting. As anticipated this scene belongs to the realm of Scene Interpretation. For most of the object combinations it scores far better than Color-based Segmentation. However, for the scenes arranged on top of the sideboard the situation is not as clear. Obviously there are object arrangements which are better segmented applying CbS instead of ScI. This is unexpected because the planar surface of the sideboard is assumed to be clearly visible. The assumption proves to be wrong when inspecting intermediate steps of the segmentation process. It shows that for the scenes where CbS outperforms ScI according to the quality metric, the planar surface is not reliably detected when the recommended parameter set from Section 4.2.3 is applied. There is also indication that this behaviour is related to the choice of the parameter sets. Considering the combined performance profile of the two parameter sets for ScI, a deviation from optimal performance coincides with the transition from ScI to CbS being the better algorithm.

The red couch scene spans indices 16 - 18. As for the sideboard the results are ambiguous. For two scans CbS is the preferred segmentation algorithm whereas one scan should be segmented using ScI, according to the proposed quality metric. The result is not unexpected since the scene has been rated a questionable case but not intractable for Scene Interpretation. Despite its warped surface the supporting plane is detected quite often depending on the objects' positions. For the desk scene Figure 4.10 gives a clear vote to Color-based Segmentation. This supports the assumption that unknown objects occluding the table surface impair ScI's segmentation capability.

Three scenes belong to the class of cluttered scenes. In contrast to the expected results only one of the scenes is unambiguously attributed to CbS. For the cluttered tabletop scene spanning indices 30 - 32 there even is a tendency towards Scene Interpretation

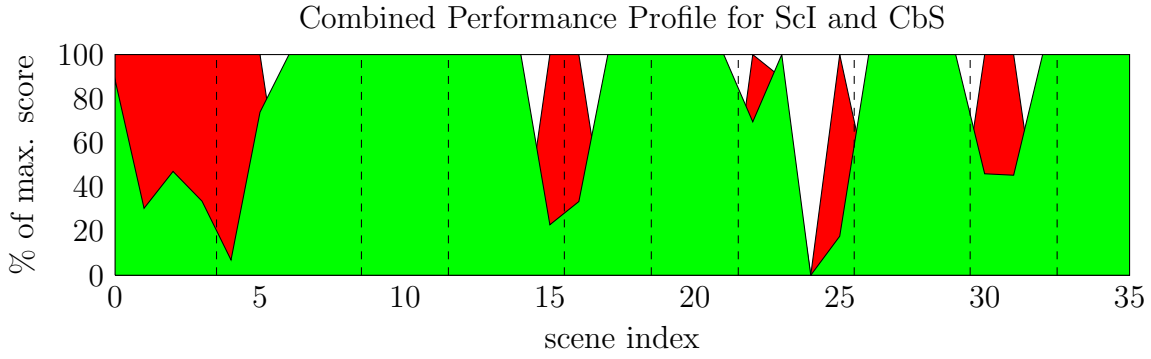


Figure 4.10: Combined generic parameter sets of ScI and CbS. It can be clearly seen at which points the two algorithms are complementary.

Index Range	Scene
0 - 3	tabletop
4 - 8	top of sideboard
9 - 11	drawer
12 - 15	shelf
16 - 18	red couch
19 - 21	desk
22 - 25	stairs
26 - 29	cluttered floor
30 - 32	cluttered tabletop
33 - 35	box

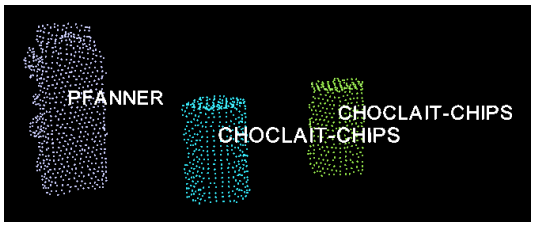
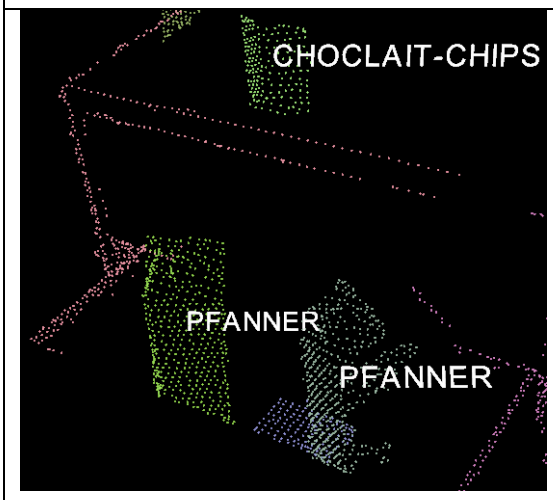
Table 4.4: Mapping of scene indices to the scenes discussed in Section 4.1.

which scores higher than CbS on two of the three scans. Looking at the charts in Figure 4.9 it can be seen that the generic parametrization enables both algorithms to work at maximum performance for scans 30 and 31 however ScI's performance drops severely for index 32. Further investigation showed that the goal of occluding large portions of the tabletop surface with known objects such that it cannot be detected was only achieved for the last scan. For indices 30 and 31 planar surface detection was working properly by applying the generalized parameter sets.

The stair scene is a very special case. It proves to be the hardest segmentation problem in the repertoire. Although for three of the four object arrangements the generalized parameter sets allow their respective algorithms to run at maximum performance, the absolute scores are a factor ten below the scores achieved for the other scenes in this class. Index 24 cannot be segmented at all by either of the algorithms. This situation suggests that the stair scene is intractable for both Scene Interpretation and Color-based Segmentation. It is therefore not further investigated.

Indices 9 - 11 are scans of a drawer containing objects. They belong to the highly difficult scenes exposing occlusion of the planar surface beneath the objects and the objects themselves. The same is true for the box scene which occupies indices 33 - 35. For both scenes the quality metric unambiguously recommends applying Color-based segmentation as it is expected. However, the scene showing objects in a shelf has an outlier. Its scans span the indices 12 - 15. While three of the four scans show the expected preference of CbS, this is not the case for scan 15. The reason for this is the coffee pot. Large areas of its metal surface reflect the surrounding white walls. These areas are therefore segmented as parts of the shelf. Considering Figure 4.9 it can be seen that CbS is working below optimal levels for this scene. Hence there is a parameter set for CbS exposing better performance. However, this particular parameter set is not part of the selected parametrizations.

Only relative scores were considered until now because for the selection process it is only important to know whether an algorithm and parameter set is outperforming another algorithm/parameter set combination. Using the relative scores it is also possible to quickly determine whether an algorithm is running at top performance. However, when doing segmentation for object recognition it is not sufficient to have segmentation running at top performance. Top performance can still yield bad results. It must be proven that the segmentation performed with the algorithms and parametrizations recommended by the quality metric generate useful segmentation results. The best way of showing the segmentation performance is therefore to show on how many scenes object recognition is working well. Table 4.5 depicts a selection of the scenes discussed in Section 4.1. For the full table of recognition results please see Table C.1 in the appendix. The images show the segmented scenes. Points belonging to the same segment are printed in the same color. The names of the objects, if recognized, are printed in white.

Recognition Result	**Original Scene**	i
		0
		9

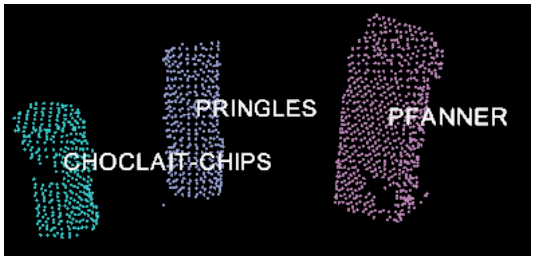
Recognition Result	Original Scene	i
		16
		30

Table 4.5: A subset of the recognition results achieved using the parameter sets and algorithm recommended by the segmentation quality metric. For a full set please refer to Table C.1.

Chapter 5

Conclusion

A major contribution of this thesis is the implementation of three segmentation approaches. The algorithms are implemented using the BOR3D Framework. While Color-based Segmentation and Scene Interpretation both achieve very good segmentation results Spectral Clustering does not live up to the expectations. This is due to the very small amount of sample points given to the Nyström Extension. Probably the only method to use Spectral Clustering on a personal computer in a reasonable amount of time is to use a GPU implementation. Considering the nearly flawless results when Spectral Clustering is working well, it is certainly worth the while to try and port the code by Catanzaro et al. for newer CUDA devices.

The newly developed segmentation quality metric is considered a major contribution as well. Using its recommendations, algorithms and parameter sets can be chosen that generate convincing segmentation and recognition results. Only for one of the 36 scenes no segmentation can be achieved at all. This result is achieved despite one missing algorithm. The quality metric can be used to create performance profiles. They are a precise diagnostic instruments able to pinpoint weak spots of a segmentation approach. The only thing needed is a series of different scenes. The experimentally determined performance profiles do not confirm some of the previously made assumptions. However, further analyses of the discrepancies between the anticipated and the measured performance profiles show that the measurement is far more accurate. All differences could be traced back to misconceptions in the scene classification done in Section 4.1. This strongly supports the validity of the quality measure.

Currently the selection of algorithm and parametrization is done manually. Automation of the selection process can be done using a classifier such as a Support Vector Machine or a decision tree. Since raw point clouds cannot be used as input data a scene description method is still needed. The development of such a description could not be considered within the allotted time. Although the performance profiles are a big step towards an automatic benchmark for segmentation algorithms, there is reason to

believe that using them for training a classifier would still not result in a well trained agent. The performance profiles of single parameters and even whole algorithms do not show consistent results for scenes of the same type. While for some scenes it could be shown that the initial classification was incorrect for other scenes the inconsistencies remain unexplained. There are two possible reasons for such inconsistent behaviour: The first reason could be that the inconsistently scored scenes are simply outliers that are generated by the specific set of objects used for annotation. Simply by increasing the number of scenes per scene type should appreciably reduce the effect of such outliers. Another source could be the fashion in which the parameter sets are chosen as of now it has not been investigated whether using different parameter sets results in more consistent performance profiles.

For the primary goal it is shown that an object recognition approach which has formerly been restricted to tabletop settings, is now capable of performing well on a variety of other scenes. It is proven that the choice of the correct algorithm and parameter set for a given problem improves the capabilities of segmentation and therefore object recognition.

Bibliography

[1] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. (UCB/EECS-2010-17), Feb 2010.

[2] Catanzaro B., Su B.-Y., Sundaram N., Lee Y., Murphy M., and Keutzer K. Efficient, high-quality image contour detection. In *International Conference on Computer Vision (ICCV)*, pages 2381 – 2388, Kyoto, Japan, 09 2006.

[3] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[4] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.

[5] M. Bertsche, T. Fromm, and W. Ertel. Bor3d: A use-case-oriented software framework for 3-d object recognition. In *IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, 2012.

[6] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and E. Y. Chang. Parallel Spectral Clustering in Distributed Systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):568–586, March 2011.

[7] Xiaopeng Chen, Qiang Huang, Peng Hu, Min Li, Ye Tian, and Chen Li. Rapid and precise object detection based on color histograms and adaptive bandwidth mean shift. In *Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, IROS'09, pages 4281–4286, Piscataway, NJ, USA, 2009. IEEE Press.

[8] Martin A. Fischler and Robert C. Bolles. Readings in computer vision: issues, problems, principles, and paradigms. chapter Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, pages 726–740. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[9] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the nyström method. *IEEE PAMI*, 26(2):214–225, February 2004.

[10] Karol Hausman, Christian Bersch, Dejan Pangercic, Sarah Osentoski, Zoltan-Csaba Marton, and Michael Beetz. Segmentation of cluttered scenes through interactive perception. In *ICRA 2012 Workshop on Semantic Perception and Mapping for Knowledge-enabled Service Robotics*, St. Paul, MN, USA, May 14–18 2012.

[11] V. Hernandez, J. E. Roman, A. Tomas, and V. Vidal. A survey of software for sparse eigenvalue problems. Technical Report STR-6, Universitat Politècnica de València, 2009. Available at http://www.grycap.upv.es/slepc.

[12] Vicente Hernandez, Jose E. Roman, and Vicente Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software*, 31(3):351–362, 2005.

[13] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[14] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[15] Carlo Dal Mutto, Pietro Zanuttigh, Guido M. Cortelazzo, and Stefano Mattoccia. Scene segmentation assisted by stereo vision. In *Proceedings of the 2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, 3DIMPVT '11, pages 57–64, Washington, DC, USA, 2011. IEEE Computer Society.

[16] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 849–856. MIT Press, 2001.

[17] Victor Y. Pan and Zhao Q. Chen. The complexity of the matrix eigenproblem. In Jeffrey Scott Vitter, Lawrence L. Larmore, and Frank Thomson Leighton, editors, *STOC*, pages 507–516. ACM, 1999.

[18] T. Rabbani, F. van den Heuvel, and G. Vosselman. Segmentation of point clouds using smoothness constraint. In H.-G. Maas and D. Schneider, editors, *Proceedings of the ISPRS Commission V Symposium 'Image Engineering an Vision Metrology'*, pages 248 – 253, Dresden, 2006. ISPRS.

[19] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3D

Recognition and Pose Using the Viewpoint Feature Histogram. In *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 18-22 2010.

[20] R.B. Rusu, Nico Blodow, Z.C. Marton, and Michael Beetz. Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in domestic environments. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–6. IEEE, October 2009.

[21] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 1997.

[22] Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, August 1969.

[23] Manuel Wopfner, Jonas Brich, Siegfried Hochdorfer, and Christian Schlegel. Mobile manipulation in service robotics: Scene and object recognition with manipulator-mounted laser ranger. In *Robotics(ISR) 41st International Symposium and 6th German Conference on Robotics (ROBOTIK)*. IEEE, 2010.

[24] Q. Zhana, Y. Liangb, and Y. Xiaoa. Color-based Segmentation of Point Clouds. In G. Vosselman F. Bretar, M. Pierrot-Deseiligny, editor, *ISPRS Proceedings: Laserscanning*, number XXXVIII-3/W8, 2009, pages 248–252, Paris - France, 2009. ISPRS.

Appendix A

Original Thesis Specification

A.1 Introduction

Segmentation in 3D is the task of assigning a label to each point inside a point cloud such that all points in possession of the same label belong to the same object in the real world. It is a very important step for many object recognition pipelines. There are several approaches to isolating segments which rely on geometric properties or color properties. Each of them has its advantages and disadvantages.

The disadvantages all result in the two major challenges of point cloud segmentation: (1) over-segmentation which means that an object is represented by multiple segments. (2) under-segmentation which relates to the case where segments contain more than one object. Both describe the failure of an approach to derive a desired bijective mapping of labels to real world objects.

A.1.1 Motivation

Whenever a segmentation method fails there is very likely a complementary approach which is able to succeed. The same observation can be made when looking at different parameter sets for the same algorithm. Human experts are often able to qualitatively predict whether a segmentation method will perform well or poor for a given scene based on their knowledge about assumptions encoded int the algorithm or the parameter set. Such predictions can also be made intuitively after some time when using algorithms as black boxes. It is necessary to investigate to which extent machines are also able to obtains this kind of experience in order to make adequate decisions about which method to use.

Conventional approaches to 3D object recognition perform single-staged segmentation and object recognition. A practice which makes segmentation a demanding task. Al-

gorithms used for large scenes must be able to cope with many different levels of detail simultaneously. Finding parameters for such approaches is commonly a trade-off between over- and under-segmentation of certain real world objects. Recursive – or multi-staged – segmentation on scene clusters of decreasing scales can be expected to be a fertile approach to escape this predicament. Such divide-and-conquer approaches can be further optimized by introducing a recognition step after each segmentation run. The recursion has to step when segment sizes drop below a predefined threshold or all segments have been recognized.

A.1.2 Objective

The goal of this thesis is to propose an approach for 3D segmentation and object recognition. It is assumed that a method for object recognition and a set of heterogeneous segmentation methods are available. The approach will recursively subdivide and analyse segments based on the segmentation and recognition approaches at hand. The segmentation algorithm used for each recursion is automatically selected.

A.2 Work Packages

Six work packages have been identified in order to achieve this objective.

A.2.1 Choice and Implementation of Algorithms

The algorithms considered for analysis are pipelines of several processing steps. Implementations of the single processing steps already exist in parts. Missing steps need to be implemented and combined with existing ones in the way that is specified in the corresponding papers. A minimum of three algorithms need to be implemented but no mere than five are necessary due to the amount of time allotted. The algorithms will be developed using the Point Cloud Library and the BOR3D framework. The implementation must contain visualization.

- Which segmentation algorithms were chosen? Why were they chosen?
- In what way are the segmentation algorithms expected to be complementary?
- Have there been challenges reimplementing the algorithms from the information given in the corresponding papers?
- How were the segmentation algorithms implemented?
- Are there deviations from the original proposal? Why were these alternatives chosen?

- Which recognition algorithm is being used? Why is it being used?

A.2.2 Algorithm Analyses

The performance of each algorithm must be thoroughly evaluated by hand. In order to do that the scenes used for evaluation mus be carefully chosen and documented. A comparison of the algorithms with different parameter sets should provide a reasonable amount of insight to determine generic parameter sets which are not specialized on single scenes.

- Which scenes were chosen for evaluation?
- Which characteristics make them interesting for evaluation?
- Do the segmentation algorithms perform as expected?
- Are there cases where no segmentation algorithm / parameter set succeeds? Why does none of them succeed?

A.2.3 Determining Generic Parameter Sets

Each segmentation approach needs parameters to adapt its performance to a given set of scenes. The performance can vary widely depending on the scene and parameter set. Therefore the approach does not solely rely on choosing between different algorithms but also on choosing between different parameter sets for each algorithm.

- Fine-tuning of the parameter sets to cover all the cases identified during the previous analyses.
- Documentation of the resulting parameter sets. They are to remain constant.

A.2.4 Training Design

Automatic algorithm selection will be driven by a classifier such as a Support Vector Machine, a Decision Tree or a Neural Network. Training such a classifier requires a list of labelled feature vectors. For each algorithm or parameter set there will be one label. The feature vectors will be determined by a global description algorithm which is computing using the scene's entire point cloud. There are many scenes and also many view points from which to look at a scene. It is tedious to label training data manually. This needs to be automatized using a score which is determined using recognition results. A scene description receives the label of the algorithm or parameter set receiving the best score.

- Elaborate a training method which fulfils the requirements.

- The classification method needs to be exchangeable.
- Which classification method was chosen? Why was it chosen?
- Is the method applicable to unknown scenes?

A.2.5 Training Implementation

In order to accelerate the training process a few extra pieces of software are needed. The recognition process must compute a quality metric which enables comparison of recognition results of different objects. A tool is needed to create the training data. It uses recognition quality to score the different segmentation algorithms and assigns the correct labels to the feature vectors. It is to be expected that a few additional features have to be annotated by hand. The actual training will be performed using third party libraries and tools compatible with the existing implementation. Therefore they must expose a C/C++ API.

- Implement Training.
- Perform Training.
- Find classification method performing the best algorithm choice.
- Review segmentation algorithms and parameter sets. Do they work as expected?
- Document statistical results.

A.2.6 Finalization

- Evaluate statistics and draw conclusion.
- Finish writing thesis.

Appendix B

Viewpoint Feature Histogram - Training Manual

The main requirement for training VFH feature models are 3D point clouds of the objects to be trained. Multiple point clouds are needed per object – each made from a different view angle. Putting a spherical frame in the center of the object the recommended resolution for the rotation ϕ is three degrees in the case of 4-degrees-of-freedom (4-DOF) recognition and additionally five degrees for the elevation angle θ for 6-DOF recognition respectively. VFH is not able to estimate the object's position. Such information must be obtained by a preceding step.

B.1 Data Acquisition

The different ϕ angles can be obtained by rotating the object along its y-axis while the sensor position remains fixed. For variation in θ the sensor must be mounted on an arc above the object. The sensor has to be moved along the arc after every ϕ iteration. Alternatively move the sensor parallel to the object's y-axis. Of course by this way not all view angles of the object can be trained. Please take care that the object does not leave the sensor's field of view when manipulating the sensor position.

Rotating the sensor 360 degrees or even more times when training for 6-DOF recognition is tedious. It is therefore recommended to use a system that automatizes the process. Please note that 4-DOF recognition does have further implications. Due to the fact that VFH object recognition is strongly dependent on the view angle differences in θ between training and recognition will lead to degraded recognition results.

The segmentation algorithm performed as preprocessing step relies on the precondition that the object is placed on a planar surface. Please make sure taht at least half of the points in the point cloud belong to that surface.

B.1.1 Software Setup

It is assumed that you are using a turning table driven by a Nanotec SMCI47 S-2 stepper controller to rotate the object. A description of integrating custom training automatization systems will be given later in this section.

In order to perform data acquisition the `dump` application is needed. Please perform the following steps:

1. Install libnanocontrol
   (a) `$> git clone http://git.code.sf.net/p/libnanocontrol/code <NCTRL_ROOT>`
   (b) `$> mkdir <NCTRL_ROOT>/build`
   (c) `$> cd <NCTRL_ROOT>/build`
   (d) `$> cmake ..`
   (e) `$> make`
   (f) `$> sudo make install`
2. Clone BOR3D and build `dump`
   (a) `$> git clone http://git.code.sf.net/p/bor3d/code <BOR3D_ROOT>`
   (b) `$> mkdir <BOR3D_ROOT>/build`
   (c) `$> cd <BOR3D_ROOT>/build`
   (d) `$> cmake ..`
      i. install all missing dependencies that are reported by CMAKE they should all be available in the package management system of your Linux distribution.
   (e) `$> make dump`
      i. result: `<BOR3D_ROOT>/build/src/examples/kinect2pcd`$_d$*umper*/*dump*

B.1.2 Using `dump`

In order to invoke `dump` you need the mandatory command line option `--config-file`. The argument should point to an existing JSON formatted configuration file for this application. If you do not have such a file you can create a new one by adding `--configure`. An example of such a configuration file for 4-DOF VFH training data

acquisition can be found in <BOR3D_ROOT>/configs/dump. In order to use the logging mechanism present in any BOR3D application please have the LogConfig configuration file entry point to <BORED_ROOT>/log/log.config or any other log4j-compatible configuration file that suits your needs.

After invoking dump you should get to the BOR3D prompt BOR3D>. Currently there exist four commands: help will give you a short description of the commands. start will start the dumping process. stop will interrupt the dumping process. quit will bring you back to the shell.

dump uses the OpenNI grabber wrapper provided by the Point Cloud Library so you can use any compatible sensor.

dump does not stop running by itself. stop has to be called after all necessary data has been written to <DUMP_WORKING_DIR>/raw.

B.1.3 Customizing dump

It is very unlikely that you have an automatized object training system compatible with version of dump provided in the repository. Nevertheless it is possible to make BOR3D support your own system by writing some custom code. For that task it is best you take the implementation for our system as a template that you adjust to your needs. Create a copy of <BORED_ROOT>/include/bor3d/3d/grabbers/grabber_addons/grabber_table_addon.h.

1. Prepare your template:
   (a) change the file name and adjust the include guards accordingly
   (b) remove all BORED_SLOT macro calls
   (c) remove all BORED_SLOT_INIT initializers from the constructor
   (d) remove all private member declarations
      i. *keep* m_grab_counter
      ii. *keep* m_turn_interval
   (e) clean the protected method definitions
      i. *keep* the BaseType::xxx() calls.
      ii. *keep* all m_grab_counter and m_turn_interval related parts
   (f) rename the Table class to a <SYSNAME> of your choice
   (g) change the specialization parameter Table in Grabber to <SYSNAME>

2. Suit the template to your needs
   (a) if you need to store some data or a communication object create a member for it
      i. declare it `private`
      ii. declare it as a pointer
   (b) if you need configuration data for bus communication or system parametrization create `BORED_SLOT(i, ii, iii, iv, v)`
      i. the <NAME> of the slot. By convention slot names should begin with a capital letter.
      ii. the <DATATYPE> (must support copy-construction)
      iii. the communication class (you should always choose `STATIC`)
      iv. a class tag (you should always choose 0)
      v. define handler methods
         A. `void handle_data(<DATATYPE> d, IMachine *machine)`: assign d to a private member of `Grabber` using the following way: `static_cast<Grabber *>(machine)->member = d;`
         B. `<DATATYPE> get_default() {return <A DEFAULT VALUE>;}`
      vi. for each `BORED_SLOT` you need a `BORED_SLOT_INIT(<NAME>)` initializer in the class constructor
   (c) use the `initialize_grabber()` method to allocate data and communication objects. All configuration data will be available before this method is called.
   (d) `start_grabber()` is called right before image acquisition is started (you probably do not need any code here)
   (e) `stop_grabber()` is called right after image acquisition is stopped (you probably do not need any code here)
   (f) use the `deinitialize_grabber()` method to de-allocate everything allocated by `initialize_grabber()`
   (g) use the `between_grabs()` method to tell your training system to move to the next position. Insert code between `if` and `else`. The move must be finished when the method returns.
3. incorporate your new grabber addon into BOR3D and `dump`

(a) `#include` your new file at the bottom of <BOR3D_ROOT>`/include/bor3d/-3d/grabbers/grabber.h`

(b) copy and rename `DumpMethod` in <BOR3D_ROOT>`/include/bor3d/3d/use-case/dump_method.h` to <METHOD_NAME>

(c) change the last template parameter `bored::_3d::grabbers::Table` of `Grabber` in the `GrabberType-typedef` to `bor3d::_3d::grabbers::`<SYSNAME>

(d) in <BOR3D_ROOT>`/src/examples/pcd2cv_dumper/dump.cpp` change the `MachineType-typedef` to stand for <METHOD_NAME> instead of `DumpMethod`.

(e) change linkage as needed in <BOR3D_ROOT>`/src/examples/pcd2cv_dumper/CMakeLists.txt`, compile and run.

B.2 Training

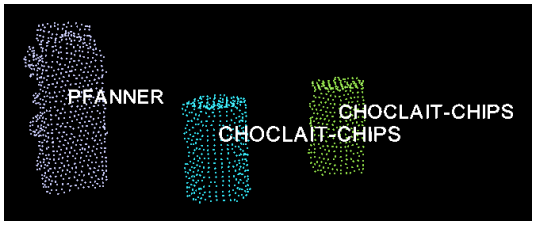
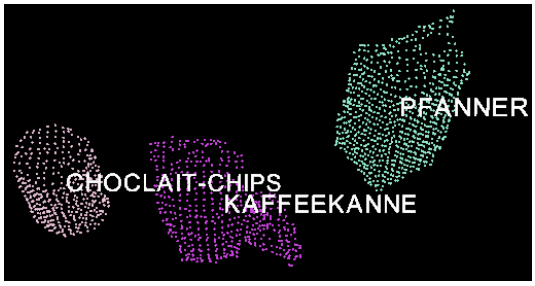
Although training can easily be changed to run for the sensor directly the current version uses `*.pcd` files generated by `dump`. Please build training with the following steps:

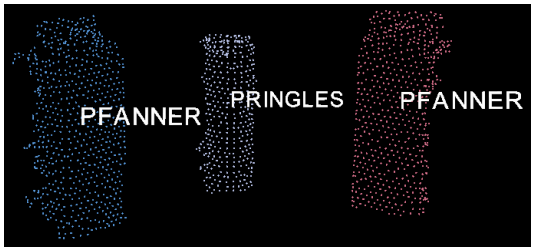
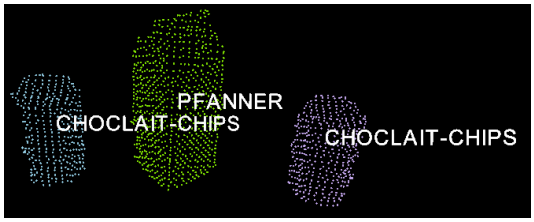
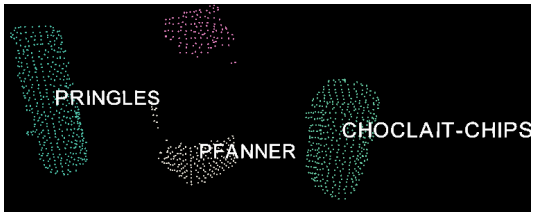
1. `$> cd` <BOR3D_ROOT>`/build`
   (a) install all missing dependencies that are reported by CMAKE they should all be available in the package management system of your Linux distribution.
2. `$> make training`
3. result: <BOR3D_ROOT>`/build/src/examples/vfh_training/training`

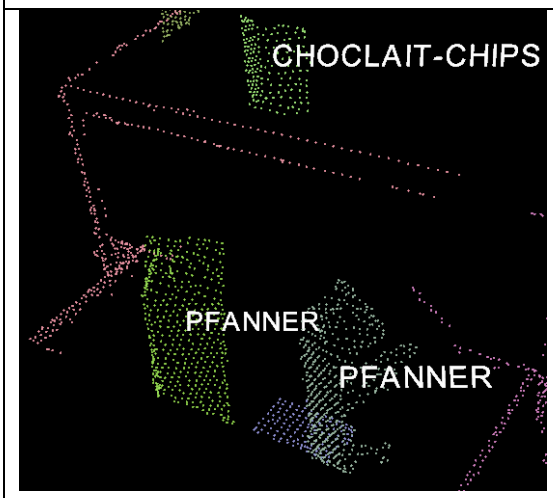
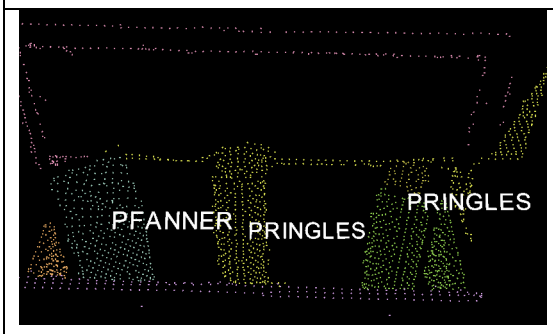
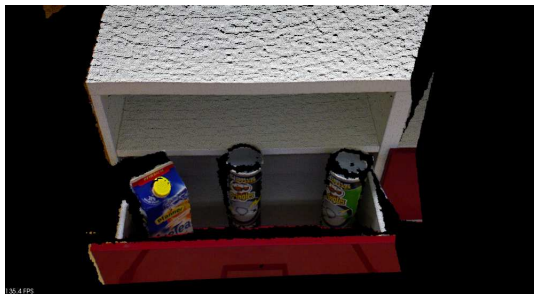
Invocation is exactly the same as for `dump`. Also prompt commands are the same. The process does not stop on its own. Wait for the `*.mdl` file to be created. Depending on the training resolution this can take several hours. Copy the `*.mdl` file to the desired location.

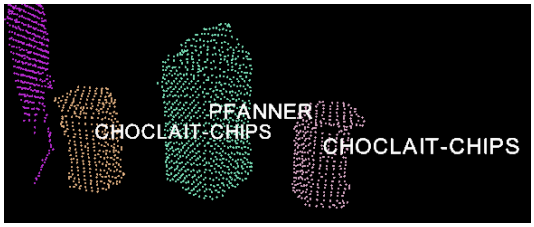
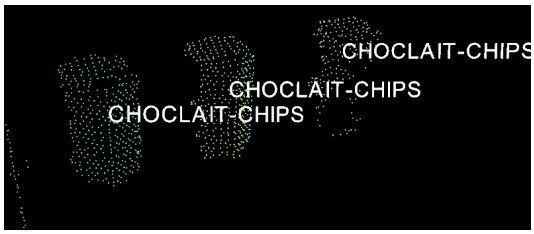
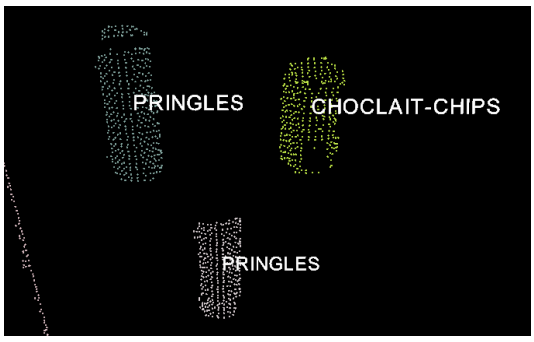
Appendix C

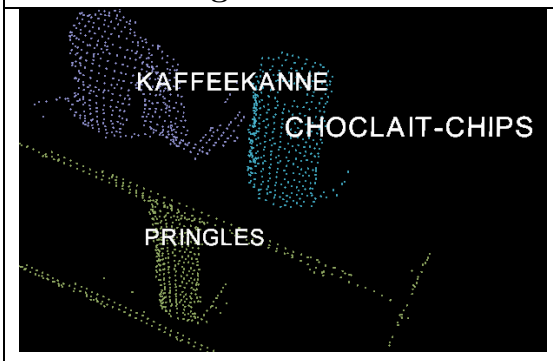
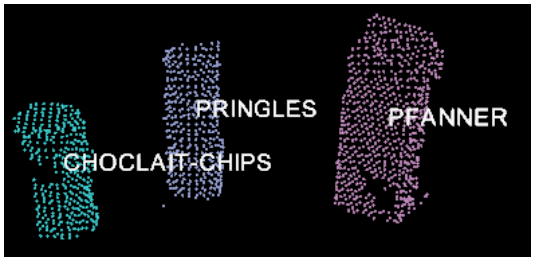
Recognition Table

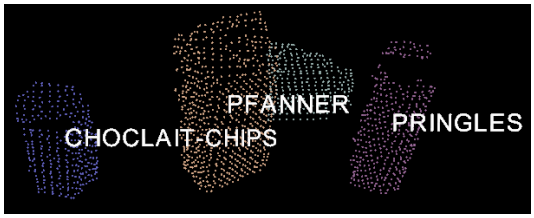
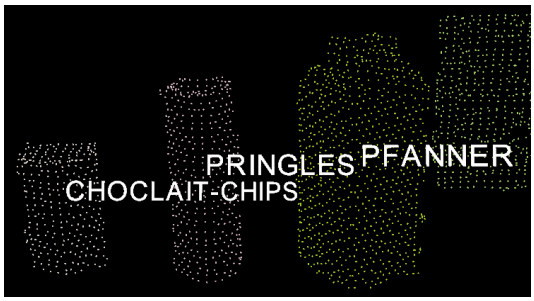
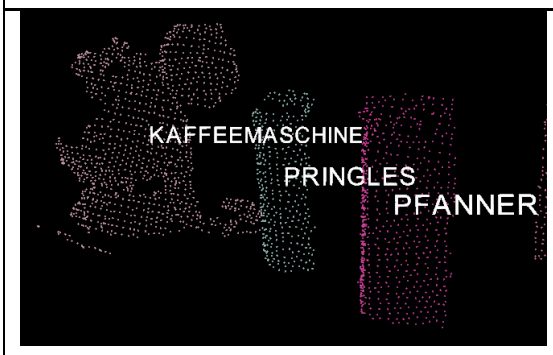
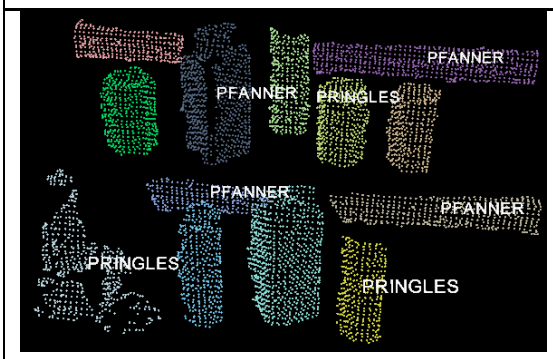
Recognition Result	Original Scene	i
		0
		1
		2

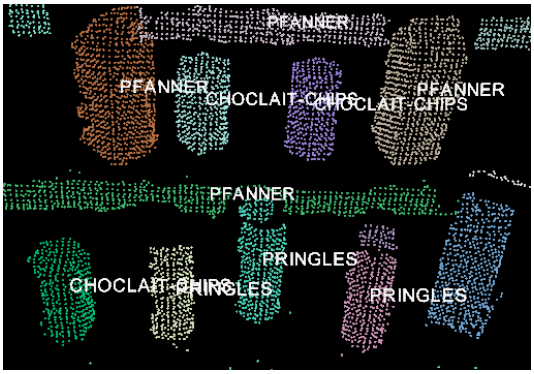
Recognition Result	Original Scene	i
		3
		4
		5
		6

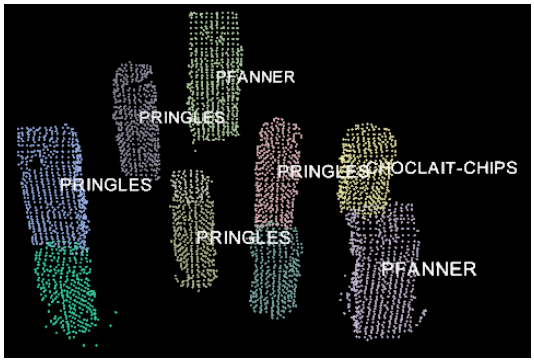
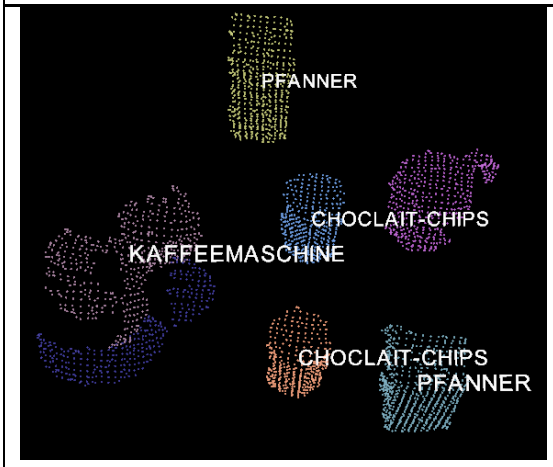
Recognition Result	Original Scene	i
		7
		8
		9
		10

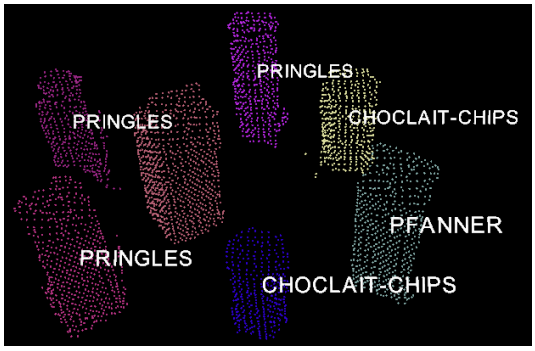
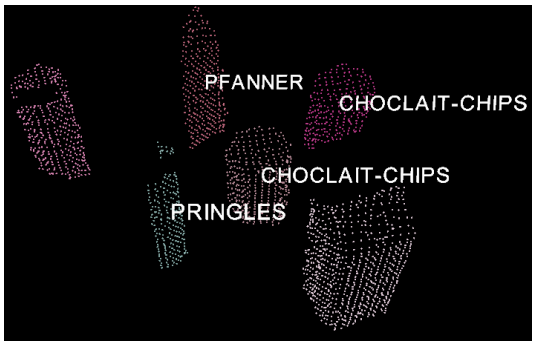
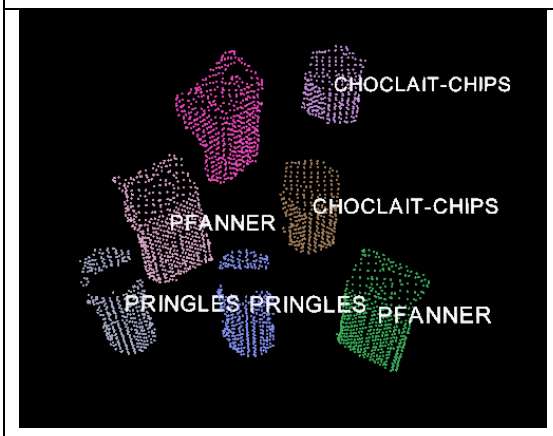
Recognition Result	Original Scene	i
		11
		12
		13
		14

Recognition Result	Original Scene	i
		15
		16
		17
		18

Recognition Result	Original Scene	i
		19
		20
		21
		22

Recognition Result	Original Scene	i
		23
		24
		25

Recognition Result	Original Scene	i
		26
		27
		28

Recognition Result	Original Scene	i
		29
		30
		31
		32

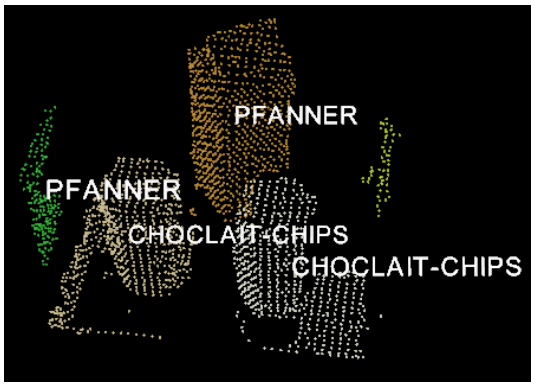
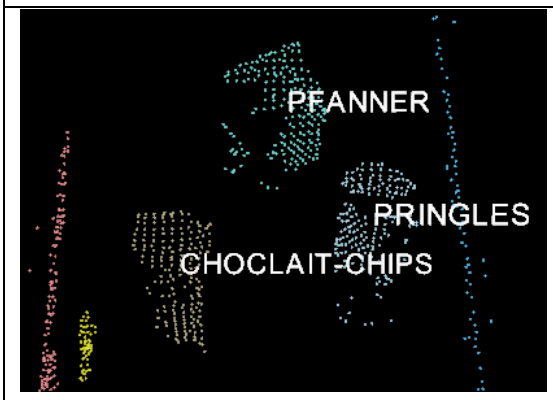
Recognition Result	Original Scene	i
		33
		34
		35

Table C.1: The full recognition table.