Rapid Histogram-based Scene Interpretation in Three-dimensional Surface Point Clouds

Eric Wahl


Vorsitzender: Univ.-Prof. Dr.-Ing. G. Färber
Prüfer der Dissertation:
1. Univ.-Prof. Dr.-Ing. K. Diepold
2. Hon.-Prof. Dr.-Ing. G. Hirzinger

Die Dissertation wurde am 04.10.2006 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 14.05.2007 angenommen.
Acknowledgments

This thesis summarizes the investigations of three and a half years. Although, it was my part to guide this process and to draw the final conclusions, it would have never been possible without a lot of help and fortunate conditions.

First of all I would like to thank Prof. Gerd Hirzinger for giving me the chance to work in a highly inspiring environment, namely the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR). At this institute I had the opportunity to collaborate with many wonderful colleagues. Furthermore, he gave me the latitude to develop and realize my ideas.

Second, I want to thank my doctoral advisor Prof. Klaus Diepold. He was very helpful in guiding and motivating me when I was unsure of the direction of my research. Also, he took the time for lengthy discussions.

Further acknowledgments go to all my colleagues who supported me in many different ways. Especially Dr. Ulrich Hillenbrand helped me many hours with contemplations. He asked me the right questions to evaluate my work. Tim Bodenmüller, Klaus Strobl and Michael Suppa kept the hardware running and calibrated whenever I needed it. Moreover, I would like to thank Bernhard Kühler, Christian Ott, and Ulrich and Susan Seibold for giving me feedback on my papers and this work. I am sure it was a hard job. Last but not least I thank Wolfgang Sepp for an endless number of interrupting phone calls and quick answers.

At the end of my thesis I wanted to test my algorithms with many different 3D-sensors. Here, the enterprise ABW GmbH of Mr. Wolf senior and Mr. Wolf junior acquired sets of excellent 3D-data without any questions in return. I thank both gentlemen for this generous help.

Last but not least, I received great support from my family. Especially my wife and my son had to tolerate that I was often short of time. Nevertheless, both helped me to relax at the weekends and to collect new power for long days starting in the early morning. Thank you Nina, thank you Lovis!

Note, the order of this list does not refer to the importance of people’s contribute. Every listed person (and surely some I have forgotten) were necessary to make me reach the end.
Abstract

In this work a new framework for generic scene interpretation is introduced. It is based on several statistical approaches ranging from a novel description of object shapes to their localization and classification in complex scenes. In the present context a scene is a static arrangement of several three-dimensional (3D) rigid free-form objects represented by a large number of 3D-surface points and surface normals. The interpretation of a scene is the segmentation of the scene data into object-specific parts and classification of the identity of each part.

The first part of this work addresses the efficient description of 3D free-form objects using a histogram-based model. It relies on the statistical distribution of four-dimensional surface point-pair relations. This shape representation is derived in a training phase by bin counting a large amount of feature samples. It is demonstrated that once a model histogram is built, a relatively small number of random samples from an object surface is sufficient for recognizing an object from a database containing 20 models. Several metrics for comparing model histograms to sensed feature samples are investigated, and the likelihood criterion is found to be most suitable. The compactness of models in addition to small sample sets enables a high efficiency in terms of processing time and memory in the recognition phase.

Nevertheless, points of an object first need to be drawn. Here, the access to a 3D-shape embedded in a scene strongly depends on the organization of 3D-space. To manage this task, a region search algorithm based on an octal subdivision of space is discussed.

By combining the object description and the region search algorithm it is possible to realize a framework for 3D-scene interpretation. The strategy is determined by the nature of models employed. In detail it is the search for characteristic distributions hidden in a scene’s point cloud. The task is achieved by a cluster-based approach, where iteratively applied filtering conditions clear point clouds of a cluster from irrelevant and disturbing background points. Models are used to control the focus of interest, while the algorithm segments and classifies an object simultaneously.

To demonstrate the potential of the scene interpretation framework, the rate of correct classification and the correctness of the segmented points are discussed in relation to noise, occlusion, point cloud density, and the separation of objects with respect to their distance from each other. For this study, databases of synthetic and real objects are used. The algorithm is capable of handling both in comparably high quality, without the need for changing any parameters of the algorithm. The results obtained for classification rate and speed demonstrate that the system is well suited for tasks that demand high flexibility and low processing costs.
Contents

1 Introduction ................................................................. 9
   1.1 Motivation .............................................................. 9
   1.2 State of the Art ....................................................... 10
      1.2.1 Approaches Based on 2D-Images .............................. 10
      1.2.2 Approaches Based on 3D Spatial Information ............. 11
   1.3 Target Applications .................................................. 13
   1.4 Preview .................................................................... 13
   1.5 Context ................................................................... 14
      1.5.1 Concept of the 3D-Modeler ................................... 14
      1.5.2 Definition of the 3D-Modeler in a Global Coordinate System 15
      1.5.3 Data Format ....................................................... 18

2 Object Representation ..................................................... 19
   2.1 Histogram-Based Object Representation .......................... 19
   2.2 Four-dimensional Geometric Feature ............................... 20
   2.3 Model Generation in Training Phase ............................... 23
      2.3.1 Using Small and Medium Surface Meshes for Training ... 25
      2.3.2 Using Point Clouds for Training ................................ 26
   2.4 Histogram Resolution and Descriptive Capacity ............... 26

3 Histogram-based Classification ......................................... 29
   3.1 Related Work ............................................................ 29
      3.1.1 3D-Search Engines .............................................. 29
      3.1.2 Alternative Classification Approaches ...................... 30
   3.2 Structure of a Query ................................................... 30
   3.3 Histogram-similarity Criteria ....................................... 31
      3.3.1 Squared Euclidian Distance ................................ 31
      3.3.2 Intersection .................................................... 32
      3.3.3 $\chi^2$-Test ..................................................... 32
      3.3.4 Kullback-Leibler Divergence ................................ 34
   3.4 Likelihood Criterion .................................................... 34
   3.5 Penalty Term ............................................................. 35
   3.6 Experiments: Descriptiveness and Classification Rates ....... 37
      3.6.1 Procedure of Training Phase ................................. 37
      3.6.2 Procedure of Recognition Phase ............................. 39
      3.6.3 Surface Meshing ............................................... 39
      3.6.4 Ideal conditions ............................................... 39
      3.6.5 Noisy data .................................................... 40
      3.6.6 Partial visibility ............................................... 42
CONTENTS

3.6.7 Generalization Across Mesh Resolution ..................... 43
3.6.8 Generalization and Similarity ............................. 43
3.6.9 Conclusion to Descriptiveness and Classification ............. 46

4 Space Representation ........................................... 55
  4.1 The Neighborhood Problem .................................. 55
    4.1.1 Related Space Representation Approaches ............... 56
    4.1.2 Requirements in the Context of the Object Model ....... 57
  4.2 Indexing Restricted 3D-Space ................................ 57
    4.2.1 Key Generation ........................................ 58
  4.3 Tree Implementation ......................................... 60
    4.3.1 Data Preparation ....................................... 60
    4.3.2 The Octree ........................................... 61
    4.3.3 The Balanced Binary Tree ............................... 61
  4.4 The Region Search Algorithm ................................ 62
    4.4.1 Performance of the Algorithm .......................... 65
    4.4.2 Examples for both Tree Structures ..................... 66
  4.5 Indexing Unrestricted 3D-Space ............................... 68
    4.5.1 Space Partitioning ..................................... 68
    4.5.2 Super-Key Generation .................................. 69
    4.5.3 Extension to the Region Search Algorithm .............. 69
  4.6 Experiments: Performance of the Region Search Algorithm ... 70
    4.6.1 Test Design ........................................... 70
    4.6.2 Region Search Results ................................ 72

5 Scene Interpretation .............................................. 75
  5.1 Working Environment and Intended Results .................... 75
  5.2 The Idea .................................................. 76
  5.3 Crosstalk .................................................. 77
  5.4 The Segmentation Algorithm .................................. 77
  5.5 Experiments: Scene Interpretation in Practice ............... 83
    5.5.1 Scenario ............................................. 83
    5.5.2 Synthetic Data ....................................... 84
    5.5.3 Experiments on Real Data ............................... 91

6 Outlook - Extensions And Refinements .......................... 99
  6.1 Positioning of the Clusters .................................. 99
    6.1.1 The Floating Cluster Seeds Algorithm .................. 100
    6.1.2 Results .............................................. 101
    6.1.3 Repetitive Clustering .................................. 102
  6.2 Increasing the Number of Cluster Points ...................... 103
  6.3 Parallelism and Online-Recognition .......................... 104
    6.3.1 Parallelized Model Evaluation ........................ 104
    6.3.2 Online-Recognition .................................... 104
  6.4 Pose Estimation ............................................ 105

7 Conclusion ....................................................... 107
CONTENTS

A Notation 109
1 Introduction

1.1 Motivation

Robust scene interpretation by means of machine vision is a key factor in a manifold of different areas. We find it whenever a surveillance system is applied to support or replace a human operator. It is essential for autonomy, e.g. a service robot that needs to perform actions in an unknown and dynamic environment on the basis of its visual sensor inputs. In industrial production, integrated scene interpretation systems address to shorten downtime of robots, since a more generic formulation of a problem reduces the number of changes that have to be made to adapt the controlling software.

The difficulties start with the acquisition of data, which depends on the range and the correlated accuracy of a sensor and on the (often changing) illumination conditions. Once the data is acquired, problems arise in determining how to segment an object, or separate relevant and irrelevant data. Last but not least, a segmented object neither automatically provides its pose which is the relation between an object and the observer, nor its identity that is the subscribing class or label.

Early vision approaches suffered from the limited processing capabilities of their times. Accordingly, they used such high restrictions that most approaches did not leave the laboratories to gain access to industry.

With the availability of ever faster computers and 3D-sensing technology (real-time stereo processing, laser range-scanner, etc.), more general approaches became feasible. They allow weaker scene restrictions and hence facilitate new scenarios. Fundamental to visual object recognition in the sense of facilitating a broad range of applications are descriptions of general free-form shapes. A good overview of the currently prevalent approaches in representation and classification is given in surveys of Besl and Jain [BJ85], Chin and Dyer [CD86], Faugeras and Hebert [FH86], Stockman [Sto87], Belleire [Bel94], Sinha and Jain [SJ94], and Campbell and Flynn [CF01].

In computer graphics, surface meshes are a popular and established description of free-forms. They are also useful for recognition purposes and the Internet makes them
accessible to everybody for testing and comparing algorithms. A major drawback, however, is their large memory requirement. Furthermore, surface meshes are defined with respect to a global coordinate system. Thus time consuming registration is necessary to align the object of interest to the frame of the referred object model before matching is possible. The same problems apply to voxel-based descriptions of shape.

Other volumetric representations based on superquadrics, generalized cylinders, and splines all suffer from a great sensibility to noise and outliers in the sensed data. A significant effort is required to obtain a robust fitting procedure and to select the model order so as to avoid over-fitting.

It is, therefore, most desirable to develop a shape representation that (i) is compact, (ii) is robust, (iii) does not depend on a global coordinate frame, and (iv) has the descriptive capacity to distinguish arbitrary shapes.

A promising approach is to analyze the statistical occurrence of features on a surface in 3D space. This has been pursued by extracting local features such as surface curvatures or geometric relations such as distances. Their distributions are represented as discrete histograms or piecewise-linear approximations thereof. The classification step may be realized by matching a measured distribution against distributions stored in a reference database of prototypes or by the search for characteristic patterns in a distribution.

The following section outlines the state of research and summarizes the most related work of the researchers that influenced this thesis. It further discusses the differences in design and capabilities.

1.2 State of the Art

Scene interpretation algorithms can be divided into two general groups. The first group is the one that is based on 2D-images acquired by a single camera. The second one relies upon sophisticated sensor systems that additionally provide depth information.

1.2.1 Approaches Based on 2D-Images

The first and older group of scene interpretation families used image-based approaches. Compared to other devices, a camera is a low-cost sensor that quickly provides two-dimensional projections of the focused work space. Unfortunately, any information about depth is very weak and has to be acquired indirectly, e.g. reconstruction by shape from shading or interpretation of edge constellations. Thus, this group tends to realizations that are mostly described as appearance-based or view-based. The strategy is the recognition of a typical conglomerate of single or multiple feature-types expected under given illumination from a specific point of view. As a by-product the process of recognition and pose estimation is very correlated.

Seibert and Waxman [SW92] proposed a biological inspired network for object recognition. In their design nodes represent single views, while links between nodes represent the transitions to get from one view to another. The approach simplifies the problem of object segmentation by using blackened objects in front of a bright grey structured background. Nonetheless, reliable segmentation is still a key problem and is not trivial at all. Additionally, the approach calculates the object center of a shape for comparison. This fact denies the occurrence of occlusion, since it leads to shifts of the center, and then a following comparison would suffer.
Another network-based method is proposed by Wunsch [Wun98]. Again, nodes are used to represent different views of an object. The positions of these nodes are optimized with a Kohonen-network. After feeding the network with an image, the node with the most resembling view answers and delivers an approximation of the pose. In the following, the approach uses edge detection for fitting a CAD\(^1\) model to the observed scene. The treatment of occlusions is not designed. Moreover, the specialization on CAD data and edges seems not suited for arbitrary free-forms.

This thesis deals with a histogram-based description of free-form objects. It profits most from the idea to formulate object representation as a characteristic statistical distribution of features. Swain and Ballard [SB91] described this idea for one-dimensional color histograms. The revolutionary new aspect in their approach was to classify objects by the distribution of color values. Thus, instead of using relations of features and their positions in the image, they applied position invariant information.

Schiele [Sch97] and Schiele and Crowley [SC96b, SC96a, SC00] continued this line of research. They demonstrated rapid classification in monochrome images. Their approach worked even in presence of occlusion. These researchers build one essential source of inspiration. Details on the parallelism between their work and the approach discussed here are given in the next chapter (see Chap. 2, p. 19).

Plagemann et al. [PMB05] represented a view-based object localization using probabilistic models of appearance. The set of applied features contains edges, joints, corners and complex groupings thereof. Their work concentrates on situations where the object covers only a small part of an image. However, the work is able to handle difficult lighting condition. Occlusion is not a discussed topic.

1.2.2 Approaches Based on 3D Spatial Information

More sophisticated sensors provide measurements of spatial information. This category includes passive and active systems. Passive systems are observers like stereo vision approaches, while active systems comprehend an emitter and a receiver such as structured light approaches and laser scanners. The chosen kind of sensor determines the speed, the accuracy of measurements, and the final realization of data acquisition.

Hornegger and Niemann [HN95] presented a generic framework for statistical learning, localization and identification of two- and three-dimensional objects. They implied an expectation maximization (EM) approach to determine the pose and type of an object. Unfortunately, the approach suffers the EM-typical long processing times. Moreover, the algorithm needs a large number of views for training phase and seems to expect the presence of exactly one sensed object in recognition phase. This assumption is unrealistic for many applications, e. g. it can not be assumed that a desired object is omnipresent in the sensed area of an exploring service robot. The expectation of present known objects (table or things upon) is violated whenever the service robot enters an empty room and the task to clear a table. Then pre-processing is necessary to prevent the algorithm getting into undefined states.

Osada et al. [OFCD02] sample the statistics of point-pair distances across the whole surface of 3D objects. They demonstrate similarity search based on the distance distribution. However, a lot of information on shape is discarded by reduction to this one-dimensional feature. Vandeborre et al. [VCD02] use three distributions of one-dimensional geometric features, based on curvature, distance, and volume. In

\(^1\)Computer Aided Design
both works, recognition performance is moderate and only suitable for a preliminary selection as performed, e. g., by an Internet search engine.

Hameiri and Shimshoni [HS02] look for symmetric form primitives like cylinders, cones, etc., in depth images. As the basic local feature, they use the two principle surface curvatures, accumulated in a two-dimensional histogram. The surface-curvature histogram is characteristic for each ideal form primitive and known a-priori from geometrical considerations. For real measured data, however, reliance upon curvatures is very sensitive to noise and artifacts. Moreover, for general shapes the distribution of curvatures will not be as crisp as for highly symmetric shapes. Hence, it may be less informative, and many histograms may be required to cover all object views.

Multiple view-based histograms have been used by Hetzel et al. [HLLS01, LHL01] who adapted the already mentioned probabilistic 2D-approach from Schiele and Crowley to depth images. According to Bayes’ rule, the best match is calculated as the one with the highest a posteriori probability, given a set of random feature samples. As a feature they have employed a collection of local surface measures, namely, pixel depth, surface normal, and curvature. Generally, however, a high number of histograms per object model increases processing time.

Gottfried et al. [GLW99] used range images to solve the task of autonomous grasping of an object from a filled box. Their work included the segmentation of an object in the presence of background and occlusion. However, they used two algorithms designed with respect to the very special object geometries of a sphere and a cylinder. An adaptation to general tasks seems to be questionable.

Boyer et al. [BSF02] and Mokhtarian et al. [MKY01] both describe 3D free-forms by multi-scaler representations of surface curvatures. The idea is to identify characteristic regions that remain stable for multiple different scales of shape. These approaches typically suffer from extensive processing.

Stein and Medioni [SM92] proposed to represent objects via splashes and 3D-curves encoded as connected linear segments. Their idea is to store a set of signatures that describe local surface properties. In recognition phase they build hypothesis about the most reasonable object. Accordingly, processing dramatically increases by the number of isolated localities and hypotheses derived therefrom. Moreover, a dependency to local information correlates with a sensitivity to noise.

An alternative line of research has sought for describing single, possibly characteristic points on an object by their local surface shape. This includes the spin images of Johnson and Hebert [JH96, JH99], the spherical spin images of Ruiz-Correra et al. [RCSM00], and the surface signatures of Yamany and Farag [YF02]. For creating their histograms, surface points are picked and a plane is rotated about their local surface normal. The surrounding points are accumulated in that plane. Due to restricted domain, the planes can be interpreted as a 2D histogram or image. To identify an object, both approaches need to find at least three significant points and their histograms. Then the information is used to determine the object pose as well. Nevertheless, the approaches suffer from the requirement of dense surface meshes.

Hillenbrand and Hirzinger [HH02] have characterized singular surface shape by four-point-relation densities that are directly constructed from a 3D-point set. Their approach generates a large number of hypotheses with respect to the object-type and the pose. To keep processing costs low, their algorithm tests every hypothesis in turn according to a probabilistic ordering.
1.3 Target Applications

If the problem of free-form object recognition and scene interpretation is discussed from an economic point of view, the following three subjects will be of major interest:

1. The object description has to be generic to enable modeling of a maximum number of different objects.

2. The classification rate has to be high, while processing costs need to be low. This enables the usage of standard computers and online applications.

3. In general, algorithms are designed for one specific purpose. Applying the algorithms to a similar application usually requires changes. This expense is only justifiable for large quantities of product or performed actions. Whenever production cycles are short, the adaptation of an algorithm has to be minimal. Automated production then becomes interesting as software has a longer life-cycle to pay for its costs.

A system that is able to fulfill all conditions and can be wed in a wide range of applications will be of economic interest. Currently, the European Union recognized the importance of these questions and is financing an EU-project called SMErobots\(^2\) [sme05], which is embedded in the 6th Framework Project (FP6).

This thesis concentrates on typical demands of robotics applications. The presented approach addresses rigid free-form objects being embedded in a static 3D-scene, where input data is given as a cloud of surface points and surface normals. Applications with piecewise rigid shapes, like a robot arm built of several joints, would need an expansion of this work. Deformable shapes are beyond the focus.

Further, the task of fast object segmentation and classification is addressed. Thus a result is desired to be a set of object specific point clouds extracted from the observed scene. The pose of objects, i.e. their spatial orientation, is explicitly excluded. If an application would need pose information, it could be easily calculated from the object’s point clouds via specialized algorithms.

1.4 Preview

The presented research leads the reader to a generic scene interpretation framework, fulfilling the previously summarized pre-conditions. It is generic in many different concerns. First it is an object description capable of representing three-dimensional free-forms. The used information is the distribution of pairwise parameterized surface points. Such a description does not demand for special restrictions to an object and can be obtained with every 3D-sensing device. A system will be discussed which is able to handle different degrees of surface density. Furthermore, collecting surface samples, calculating the distribution, and storing the result in a model is a very simple and not an object specific process. It is possible to generate models from synthetic sources like CAD models or real data without changing any method. As a consequence it is well suited for applications where a user has only limited knowledge about applied technical methods.

It will be shown that the presented framework keeps the simplicity in usage for the scene interpretation as well. This is possible because of a statistical approach,

\(^2\)Small and Medium Enterprises Robots
where the scene is divided into sets of point-pairs which are most likely for a model. The underlying idea is that these sets show evident cumulations at the locations of the supposed object types. Therefore, a cluster refinement strategy is followed that finally delivers point clouds bearing the label of an object type.

Experiments will show the efficiency and effectiveness of the framework with respect to achieved classification and processing time. In detail, it will be shown that not only the type of a local point set is precisely determined. Moreover, the percentage of true to false object points per result is convincing. Again, the scene interpretation satisfies the label "generic", because of strictly relying on the distribution of surface point-pairs. This condition entitles no specific shape or structural requirements to a scene. Furthermore, the parameterization of the framework shows no need for adaptation when changing the sensing device from synthetic data to laser-scanner, laser stripe-profiler, or structured light.

1.5 Context

Most scene interpretation algorithms are designed as an embedded part of a more comprehensive system. Here, the complete system is a multi-sensory device working in a global coordinate frame. It is already applied in two demonstrators with different purpose and source of control. One is a freehand moved sensor for 3D-surface modeling, the other is an autonomous environment exploration of an industrial robot’s working cell. We address the second application to imply recognition of a robot’s environment for further decision making and planning of manipulation tasks.

1.5.1 Concept of the 3D-Modeler

The device used in this context was developed in the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR). It is a multi-sensory device (3D-Modeler) [SH04] that includes three different sensors for the acquisition of range data:

1. The central unit is a laser range-scanner also developed in the Institute of Robotics and Mechatronics [HDH97, las05]. It contains a laser emitter and a receiver assembled in one cylindrical unit, which rotates around a vertical axis. The reflections of the laser beam are measured with a photo-chip. The distance is calculated via triangulation. The sensor provides sampling in a range from 50mm to 150mm at an accuracy better than 0.1mm. The sensor provides sufficiently accurate measurements, except for transluminent or mirroring surfaces.

2. The second sensor is a combination of an observing camera and a transmitting laser-stripe profiler. The distances are processed by calculating the vertical positions of the laser beam in the camera image. Basis for this is the knowledge about the orientation of the emitted laser plane with respect to the camera that is determined in a calibration phase [SSW+04]. The range of the sensor is about 300mm to 1500mm. The accuracy of the measurements is comparable to the laser-scanner module.

3. Alternatively, the camera of the laser stripe system can be coupled with a second camera. The pair of cameras builds a stereo system, which can be used to acquire distances above 500mm. Stereo provides the advantages of fast acquiring
depth maps, where every measurement has easily obtainable neighborhood relations. On the other hand stereo is typically very noisy and suffers from a high dependency to surface texture. Taking into account motion helps to eliminate the drawbacks of a stereo device.

Figure 1.1 shows a sketch of the first freehand prototype of the 3D-Modeler, while Fig. 1.2(a) shows the second generation. The device depicted in Fig. 1.2(b) is designed for the use in robotics. It is more compact and provides no triggers on the outside of the hull.

1.5.2 Definition of the 3D-Modeler in a Global Coordinate System

The acquired data are in relation to the 3D-Modeler. Hence, a second system has to be involved to register the movements of the multi-sensor device and to translate sensor relative measurements into a global coordinate system. Three alternative tools for getting the position and pose of the 3D-Modeler are in use:

1. The first tool refers the position of the modeler by the use of optical methods.

Figure 1.1: The picture shows a sketch of the first freehand version of the DLR 3D-Modeler. From the top downwards the picture shows the camera used for sensing a laser-stripe, the laser range-scanner with the laser emitter above and the receiver beneath, the second camera for stereo-vision, and the laser-stripe projector. The 3D-Modeler is controlled with three triggers and one LCD-display in the grip.
Figure 1.2: Both pictures show sketches of the second generation DLR 3D-Modeler. (a) From the top downwards the picture shows the new freehand 3D-Modeler version shows a pair of cameras used for stereo vision and laser-stripe profiling, then the laser range-scanner unit, and at the bottom the laser-stripe projectors. (b) The picture shows a sketch of the DLR 3D-Modeler for robots. Instead of a grip, the sensor device can be mounted on a robot.
1.5. CONTEXT

(a) (b)

Figure 1.3: The pictures show two possibilities of referring the 3D-Modeler in a global coordinate frame. In Fig. 1.3(a) the 3D-Modeler is equipped with a set of reflecting fiducial markers that are posed in a non-ambiguous configuration. Then a set of multiple cameras send infrared flashes at constant rate, so that the markers can be used to locate the 3D-Modeler. Another possibility is depicted in Fig. 1.3(b). There, referring is done with a passive arm and the 3D-Modeler is fixed at the end. The arm measures all six degrees of freedom and thus is able to provide the 3D-Modeler’s position.

Therefore, at least two cameras are used to survey a working space. Then, a periodically transmitted infrared-flash illuminates a set of fiducial markers that are placed in a definite constellation on the surface of the 3D-Modeler’s hull. By the help of band-width filters the cameras only sense the reflections in the chosen wave lengths. Then, the measured data of the modeler are translated into a pre-defined global coordinate system.

2. The second one is a passive arm realized as a 7-degrees-of-freedom (7-DoF) arm, like it is shown in Fig. 1.3. These systems are mostly used for inspection tasks in industry or reverse engineering. Due to incremental sensors in every joint, the trajectory of the basis to the tool center point is processible.

3. The third method of building a reference for the pose and position of the 3D-Modeler is similar to the second one. It is a robot with proprioception capabilities. Hence, the robot has the transformation matrix at any time and is able to provide three.

The term proprioception describes a systems feasibility to get information about its actual configuration. A robot with proprioception capabilities is able to measure the actual angles in every joint.
transform the sensor outputs into its own coordinate frame.

1.5.3 Data Format

The 3D-Modeler provides multiple data formats for the sampled surfaces. In the case of spatial information four formats are of interest:

**point & direction:** The raw format consists of pairs of 3D-vectors. One vector is a surface point, the other is the view direction of the sensor at acquisition time.

**depth map:** The stereo cameras provide depth maps. Here, the view direction is available for all measurements in the image. Surface normals have to be calculated separately.

**mesh:** The third format displays a surface as a triangulated mesh. Surface normals are easily obtained by calculations over neighboring triangles to a point.

**oriented surface point:** The surfaces can also be directly obtained as pairs of surface points and corresponding surface normals.

The first and the second format demand for preparation of the data since the surface normals are not directly available. Their calculation needs some kind of surface assumption with respect to questions concerning whether a local area is a local planar patch of low density or a hole. The mesh needs a separated calculating step for normals, too, but it is trivial and fast. The data given in oriented surface points contain all necessary information for the discussed scene interpretation approach at acquisition time.
Object Representation

The potential of an object recognition approach with respect to versatility, reliability and performance strongly depends on the used object representation. Each description is based on one or multiple features that are supposed to be almost invariant for changes of the environment. The less specific such a feature is the more objects it can represent.

On the other hand, generic features tend to occur in the objects of interest as well as in the rest of the environment, which is further referred to as background. To avoid ambiguities the relative occurrences of a feature and their constellations are measured and analyzed.

This chapter introduces a novel object representation for three-dimensional free-form objects. It includes the derivation of a generic novel surface feature and explains how it can be used to describe an object. The chapter concludes with two different approaches to generate an object representation.

2.1 Histogram-Based Object Representation

Objects often show characteristics in the constellation as well as in the appearance of features. Statistics have proven to be a powerful tool to describe these characteristics.

Swain and Ballard [SB91] applied a statistical approach to computer vision and represented objects by one-dimensional color histograms. Even though this is not an accurate description of shape nor any other geometric aspect, it shows a simple and fast method that inspired many researchers.

One significant contribution in this line of research is the approach of Schiele and Crowley [SC96b, SC96a, Sch97, SC00]. They ported the histogram-based method from color images to gray-scale images. Therefore, they interpreted grey-scale values as heights in an intensity map. The resulting maps were used for calculating local Gaussian derivatives. The distribution of two-dimensional Gaussian derivatives builds one characteristic histogram per image. Finally, the object representation consists of
several histograms generated from different points of view. Schiele et al. demonstrated an approach, which is able to deal with occlusion and cluttered scenes.

In the following, Hetzel et al. [HLLS01] presented a strongly related approach, where the Gaussian derivatives are calculated from a depth-map (2.5D image). As in a gray-scale version they described an object with a collection of maps, taken from different positions. Nevertheless, the reduction to local aspects is a loss of information that cannot represent global aspects even if using different points of view. Moreover, the multitude of views and corresponding histograms leads to high processing costs for recognition.

Some researchers, e.g. Ankerst et al. [AKKS99], proposed to involve the center of mass in the object description, since this leads to a translation invariance. Considering typical problems of scene interpretation like incomplete surfaces, clutter, and the presence of an unknown background, one can see that the determination of the center of mass suffers. Thus, the circumstances demand other methods.

Other common concepts like the Extended Gaussian Image [Hor84, Bel94, CF01] offer translational invariance and remain rotationally variant. Although other researchers did not cover local and global aspects of shape in one single histogram, they have to be mentioned since their studies gave hints as to important aspects for object representation.

For instance, Osada et al. [OFCD02] sampled the statistics of one-dimensional point-pair distances across the whole surface of 3D objects. They demonstrated similarity search based on the distance distribution. Vandeborre et al. [VCD02] used three distributions of one-dimensional geometric features, based on curvature, distance, and volume. Both works suggest that one-dimensional feature distributions are not capable of distinguishing many objects, since different objects can produce equal or very similar distributions. Distance distributions for example cannot differ mirror image-like objects. Nevertheless, moderate recognition results are still suitable for applications that satisfy with a preliminary or coarse selection. An example of an application is a new type of Internet search engines, where a query consisting of an object sketch results in a object list ordered by a certain similarity criterion.

Hameiri and Shimshoni [HS02] looked for symmetric form primitives, like cylinders, cones, etc., in depth images. As the basic local feature, they used the two principle surface curvatures, accumulated in a two-dimensional histogram. The surface-curvature histogram is characteristic for each ideal form primitive and known a-priori from geometrical considerations. For real measured data, however, reliance upon curvatures is very sensitive to noise and artifacts. Moreover, for general shapes the distribution of curvatures will not be as crisp as for highly symmetric shapes and may hence be less informative, and many histograms may be required to cover all object views.

2.2 Four-dimensional Geometric Feature

In the context of three-dimensional surface representations extrinsic parameters describe the pose of an object in a given coordinate frame, while the shape is defined by intrinsic parameters like curvature. A surface can be decomposed in oriented points. This context, allows for:

**DEFINITION 2.1:** A surflet defines an oriented point. It consists of a 3D point location and a normalized 3D point orientation. In the context of shape description, a
surflet is a surface point and its corresponding surface normal. Surface normals always point outside a shape.

In the context of object classification, the most interesting information concerns the identity of an object. Unfortunately, the appearance of objects can vary a lot depending to the point of observation. As already mentioned in the beginning of this chapter, representations that are not translation and rotation independent need to compensate these degrees of freedom in the model or during recognition. For the first case, a strategy is to record the variant appearances in the model. A common solution is a collection of multiple poses per object. Accordingly, memory cost is high. The second case follows a complementary strategy. A detailed model is compared to the sensed data. If there is enough evidence for a fit, recognition will be successful. Unfortunately, this typically involves a large number of hypotheses that need to be evaluated.

Figure 2.1: (a) Two surface points \( p_1, p_2 \) and their orientations \( n_1, n_2 \). (b) Illustration of the four-dimensional feature parameterization. The vector \( n'_2 \) is the projection of \( n_2 \) in the \( uw \)-plane. The parameters \( \alpha, \arccos \beta, \) and \( \arccos \gamma \) are angles, while \( \delta \) is the length of the vector \( p_2 - p_1 \).
On the other hand, limiting a representation to the intrinsic parameters trades avoiding cost intensive processing for the loss of pose information. This is a minor drawback since it is much cheaper to recover after recognition when it is ever needed. This thesis realizes this concept by the use of features that describe relations between surflets. The distribution of such a feature builds the representation of an object. A feature distribution can be interpreted as an expansion of a single feature. The expansion implies that all degrees of freedom not covered by the feature also cannot be covered by the distribution and lead to ambiguities. Accordingly, this thesis introduces a novel four-dimensional feature that definitely parameterizes the intrinsic surflet-pair relations. Surflet-pair relations can be viewed as a generalization of curvatures. While curvatures measure geometric relations between neighboring surflets, surflet-pair relations encode the same for any two surflets.

Each surflet is described by a pair \((p, n)\), consisting of the position vector \(p\) and the surface normal \(n\). Let the operator \(\cdot\) denote the scalar product of two vectors, \(\times\) the cross product of two vectors, \(\|\cdot\|\) the Euclidean norm of a vector, and \(|\cdot|\) the modulus of a real number. For each surflet-pair

\[
s = ((p_1, n_1), (p_2, n_2)),
\]

an unambiguous coordinate system has to be defined. Therefore, the origin is chosen to be \(p_1\), if

\[
|n_1 \circ (p_2 - p_1)| \leq |n_2 \circ (p_2 - p_1)|,
\]

and it is \(p_2\) else.

Now \(p_1\) is assumed to be the origin. The base vectors of the coordinate system are then defined as

\[
u = n_1,
\]

\[
v = \frac{(p_2 - p_1) \times u}{\| (p_2 - p_1) \times u \|},
\]

\[
w = u \times v.
\]

The relation between the surflets \((p_1, n_1)\) and \((p_2, n_2)\) is described by the parameters

\[
\alpha = \arctan(w \circ n_2, u \circ n_2),
\]

\[
\beta = v \circ n_2,
\]

\[
\gamma = u \circ \frac{p_2 - p_1}{\| p_2 - p_1 \|},
\]

\[
\delta = \| p_2 - p_1 \|
\]

which define the feature

\[
f_s = (\alpha, \beta, \gamma, \delta).
\]

The process of feature generation is further symbolized by the function

\[
\sigma : s \mapsto f_s.
\]

Here the shorthand notation

\[
\arctan(x, y) := \left\{ \begin{array}{ll}
\arctan(y/x) & \text{for } x > 0 \land y > 0, \\
\arctan(y/x) + \pi & \text{for } x < 0, \\
\arctan(y/x) + 2\pi & \text{for } x > 0 \land y < 0
\end{array} \right.
\]
is used. Accordingly, the codomain is
\[
\begin{align*}
\alpha & \in [-\pi, \pi], \\
\beta & \in [-1, 1], \\
\gamma & \in [-1, 1], \\
\delta & \in [0, \delta_{M,\text{max}}].
\end{align*}
\] (2.12) (2.13) (2.14) (2.15)

The range of values is object-independent except for \(\delta\), where \(\delta_{M,\text{max}}\) is the maximum distance that occurs between two surflets of an object.

The attributes \(\alpha\) and \(\beta\) represent \(n_2\) as an azimuthal angle and the cosine of a polar angle, respectively. The angle \(\gamma\) and the distance \(\delta\) represent the direction and length of the translation from \(p_1\) to \(p_2\), respectively. Limiting \(\beta\) and \(\gamma\) to scalar products keeps processing costs low.

This parameterization is illustrated in Fig. 2.1. Of course, if condition (2.2) determines \(p_2\) to be the origin, the parameters will be obtained by interchanging the indices 1 and 2 in the equations above.

Equations (2.6) to (2.9) map every configuration of a surflet pair to a unique set of parameters, and every possible set of parameters describes exactly one such configuration. Moreover, condition (2.2) ensures that the base vectors \(u, v, w\) are defined in the most robust manner: by choosing the more orthogonal angle between \(p_2 - p_1\) and the two surface normals \(n_1, n_2\) for defining \(v\) [Eq. (2.3), Eq. (2.4)] the direction of \(v\) is determined with higher accuracy. Given the fulfilled condition (2.2) and choosing \(p_2\) for the origin of the coordinate system endangers the unambiguity of the feature. In this case a surface normal would be allowed to be collinear to the vector \(p_2 - p_1\). This constellation does not allow for defining an unambiguous coordinate frame.

### 2.3 Model Generation in Training Phase

One feature describes the relations of two surface points to each other. The occurrence of a certain feature parameterization depends on the shape of an object. Accordingly, the probability of a feature parameterization to occur, varies with the observed shape. Moreover, it is possible that some parameterizations are invalid in the context of an object. Regarding a sphere for example, it is not possible to get a pair of surface points, where the surface normals are parallel\(^1\) Thus, an object description can use the distribution of features for representation.

Starting with a set
\[
\Phi = \{(p_1, n_1), \ldots, (p_m, n_m)\}
\] (2.16)
of \(m\) surflets \((p_i, n_i)\), we define the function
\[
\Upsilon : \Phi, k \mapsto \mathcal{S},
\] (2.17)
which generates the set
\[
\mathcal{S} = \{s_1, \ldots, s_k\}
\] (2.18)
of \(k\) surflet-pairs \(s_i\). The number \(m\) of surflets results in a maximum number of
\[
nt_\Phi = \frac{m(m-1)}{2}
\] (2.19)

\(^1\)Note: The point lying exactly on the other side of the sphere is collinear but anti-parallel.
surfeit pairings, when every surfeit is combined with every other surfeit of the shape. In this case, we get \( k = n_{fs} \). For real objects, the number of surfeits results in very large numbers for each \( n_{fs} \). Accordingly, \( k \) is chosen to be much smaller than \( n_{fs} \). The distribution of the corresponding four-dimensional features

\[
\Psi = \{ f_s | f_s = \sigma(s), \forall s \in S \}
\]

is approximated by the histogram \( H_M \). Therefore, the mapping-function

\[
h_M : f_s \mapsto b \in \{1, 2, \ldots, d\}
\]

transfers a feature \( f_s \) to the corresponding bin \( H_M(b) \). Although the feature vector consists of four dimensions the histogram is linearized to simplify further calculations. Thus, the histogram size

\[
d = d_{\alpha}d_{\beta}d_{\gamma}d_{\delta}
\]

is the number of bins, where \( d_{\alpha}, d_{\beta}, d_{\gamma} \) and \( d_{\delta} \) are the resolution steps that define the number of intervals per feature component. The choice of resolution steps specify the descriptiveness as well as the compactness of the object representation. Figure 2.2 shows a sketch of a linearized four-dimensional histogram. It shows the relation between indexed histogram bins and the corresponding discretization intervals of the four-dimensional feature.

A histogram entry

\[
H(b) = \frac{\text{card}\{s \in S, f_s \in \Psi | h(f_s) = b\}}{\text{card}\Psi}
\]

is the normalized frequency of features \( f_s = \sigma(s) \) that are mapped onto bin \( b \), where \( \text{card} \) denotes the cardinality of a set.

A model

\[
M = (H_M, \delta_{M, \text{max}}, \bar{r}_M, r_{M, \text{max}})
\]

consists of a normalized histogram of feature probabilities and additional three characteristic parameters. The first parameter is the maximum distance \( \delta_{M, \text{max}} \) between two

\[\text{bins}^2\]

\[\text{A bin refers to a discrete interval of the histogram.}\]
2.3. MODEL GENERATION IN TRAINING PHASE

object surflets. The remaining parameters are the mean distance $\tilde{r}_M$ and the maximum distance $r_{M,\text{max}}$ of surface points to the centroid. Fig. 2.3 exemplarily visualizes the meaning of $\delta_{M,\text{max}}, r_{M,\text{max}},$ and $\tilde{r}_M$ for a cube.

In the following sub-sections two training methods are discussed.

![Figure 2.3](image-url)

Figure 2.3: (a) The parameter $\delta_{M,\text{max}}$ represents the maximum distance between the two surface points that are farthest to each other. For a cube this is the diagonal. (b) The scalar $r_{M,\text{max}}$ is the distance of the farthest surface point to the centroid. In case of a cube, it is half of the diagonal. (c) The scalar $\tilde{r}_M$ is the mean distance between a surflet and the centroid.

### 2.3.1 Using Small and Medium Surface Meshes for Training

A common description of 3D-shapes is the class of surface meshes. Assuming a triangulated surface mesh, each grid node symbolizes a surface point. It is part of $k$ surrounding triangles. This set of triangles builds a region that can be used to approximate the corresponding surface normal

$$\tilde{n} = \frac{1}{\sum_{i=1}^{k} A_i} \sum_{i=1}^{k} A_i n_i,$$

where $A_i$ describes the areas of the triangles and $n_i$ refers to the triangle normals, respectively. Such a local surface region is also known as an umbrella neighborhood.

Although this normal estimate performs well in practice it is possible to apply more precise methods. This is advantageous for the use of coarse meshes. Then additional points can be sampled from the inside of the facets. Smith [Smi99] observes that the mean curvature from such meshes is zero except for the edges. Thus, methods like the normal vector voting of Page et al. [PSK02] provides better results. Further exemplary methods of surface curvature estimation are described in [FJ89, BSF02]. Of course, directly using surflet clouds avoids pre-processing of surflets. However, meshes provide the user the advantage to see whether training data is complete or still includes holes.

To retrieve the feature distribution, all combinations of surflets are calculated. In case of an iso-mesh, i.e. a surface that consists of triangles of nearly homogenous size, the resulting surflets build an appropriate approximation of the real distribution.

---

$^3$Both parameters $r_M$ and $r_{M,\text{max}}$ are used for an approximative description of the object hull around the centroid. They are necessary for scene interpretation that is discussed in Chap. 5, p. 75.
Methods like the approach of Li [Li93] which are optimizing a mesh by changing the local resolution seem to be problematic. Such meshes contain much more points in discontinuities than in plane parts of the surface. As long as this can be guaranteed for training phase and recognition phase it is an acceptable object representation. Unfortunately, noise, clutter, background, and incomplete sensing produces discontinuities, too. Accordingly, the influence of a surflet during the recognition phase differs to the one in training, so that the distribution is distorted. Therefore, using the same meshing conditions for training phase and for recognition phase is compulsory.

Equation (2.19) indicates that the number of surflet combinations increases rapidly. Therefore, this method gets impractical for high mesh resolutions. In such cases it is advisable to interpret the grid nodes as a point cloud and to proceed with the following method.

### 2.3.2 Using Point Clouds for Training

Real surfaces often contain a large number of surflets. Thus, it is unpracticable to build all surflet-pairs due to enormous computational costs. Moreover, keeping in mind that the feature distribution is finally approximated by a histogram, training with sampled surflet-pairs is advisable. The required number of surflet samples does not depend on the histogram size only, but also on the object shape. Thus, it is difficult to determine a certain number of sample drawings.

In practice histograms of size $d = 5^4 = 625$ show enough descriptive capacity for object representation, where objects typically consist of 3,000 to 150,000 points and training applies 10,000,000 features.

A second factor influencing the quality of a histogram, is the way of sampling. Under ideal conditions, the feature distribution is achieved by pairing every surflet to every other surflet. Hereby the proportional contribution of each surflet to the feature distribution is equal. Accordingly, a uniform drawing of features easily approximates the real distribution. A necessary pre-condition is a homogenous surface density over the whole surface. If this is not fulfilled, dense areas are more likely to produce samples. Hence, the histogram is distorted.

### 2.4 Histogram Resolution and Descriptive Capacity

The maximum descriptive ness of a model histogram depends on the number of available bins. The quantity of bins determines the number of subdivisions of a feature component $\alpha, \beta, \gamma, \delta$, respectively. It is further referred to as the resolution of a histogram.

A high resolution can preserve very fine details of an object. On the other hand, it demands a larger number of samples for training as well as in the subsequent recognition phase. Due to four-dimensional feature histograms, a high resolution obviously leads to rapidly increasing costs for memory and processing.

However, a detailed object representation is often unnecessary, since objects more often differ in global aspects than in local ones. As mentioned before, a relatively low resolution of 5 bins in every dimension which totals 625 bins leads to quite satisfying results in practice.

Moreover, it is possible to benefit from a low histogram resolution. Considering a class of very similar but not identical objects the distribution of surflets will show only little variations. The reason is the angular components always covering the same
2.4. HISTOGRAM RESOLUTION AND DESCRIPTIVE CAPACITY

codomain, while a bin represents a wide range of angles. Thus, small changes in the orientation of the surface normals are likely to cause features pointing to the same bin. Accordingly, the representation can be used to describe not only a specific instance of an object but an object class, too. However, this representation does not contain the same degree of categorization as provided by other researches. For example Froimovich et al. [FRS02] realize their representation as a hierarchical graph of functional parts. This is possible only if a supervisor invests much more knowledge for every desired object. As typical for such classification approaches, the algorithm is too specialized and too complex for generic real world scenarios. Another group of classification approaches is parts-based, e.g. the method of Huber et al. [HKDH04]. The central idea is a decomposition in parts that build up an object. Again classification is only achieved by a massive knowledge transfer from the user that manifests in the object design.

Compared to such approaches, this work performs weaker in the ability to categorize. Nonetheless, this representation is discussed here in this context of object classes, where the variation in an instance has to be small, while minor deformations of the shape are possible. It has to be mentioned that changes of the object size and object shape have different impact on classification. Varying the size leads to larger distortions of an object histogram than deformations do. This derives from the distal component which is object specific and scales the feature histogram. These properties allow for using the representation in applications, where training objects differ from objects in query phase. An analysis of the ability to categorize and to classify is summarized in the experiments section [Chap. 3.6.8, p. 43] of the next chapter.
Robust object recognition in the context of vision computing is necessary for increasing the autonomy of a robotic system. It is used in registration of measurements within a common coordinate system, in exploration of 3D-objects-databases, in industrial fabrication as well as in service robotics.

The previous chapter introduced an object representation based on the distribution of a four-dimensional geometric feature, where the distribution is approximated by histograms. This description does not include any information about object pose since it is invariant to translation and rotation. As a consequence, the pose cannot be used to support the recognition.

It will be shown that a missing object pose is no drawback for classification. It is not necessary for a precise and efficient recognition. In this chapter six different methods are discussed, measuring the similarity of a query-distribution to a database of model histograms.

3.1 Related Work

Contrary to most other current object recognition approaches, this work is based on the description of a complete 3D-object. Instead of view-based approaches it is not a collection of local sub-descriptions. Therefore, the discussion of related work is restricted to approaches that are comparable to the presented issues. Approaches that additionally include aspects of background treatment and clutter are compared later-on in the scene interpretation chapter [Chap. 5, p. 75].

3.1.1 3D-Search Engines

The presented histogram-based object recognition could be compared to the work of many other researchers in the field of 3D-search engines, even if their recognition capabilities are far inferior. The idea underlying 3D-search engines is to get fast access
CHAPTER 3. HISTOGRAM-BASED CLASSIFICATION

to a large database of 3D-objects, where the objects are typically stored as triangulated surface meshes. In most cases queries produce a list of ranked solutions, e.g. the query for a chair is supposed to deliver a variation of chairs in the top ranks.

The work of Suzuki et al. [SKT98, Suz01, SKO00] addresses the search of polygonal models. The models are not invariant to pose and size. Thus classification has to be preceded by a normalization. The main part of the classification is based on an equivalence class grouping, where the surface shape is described by super-quadrics. An interesting approach is the idea of including subjective user preferences. This is achieved by mapping an object feature space to a user preference space. This a-priori knowledge has to be trained in advance.

Further examples for 3D model retrieval approaches are works of Osada et al. [OFCD02], Vandeborre et al. [VCD02], Funkhouser et al. [FMK03], and Kazhdan [Kaz04].

3.1.2 Alternative Classification Approaches

Interpreting an object histogram as a vector in a high-dimensional space leads to the field of support vector machines (SVM). In this line of research, a very large number of object examples is used to train the system. It is equivalent to finding linear or nonlinear hyperplanes with the purpose of separating objects from each other.

Chapelle et al. [CHV99] followed this idea in the context of image classification based on color histograms. They have shown that SVMs are well suited for this problem. This line of research seems to be a promising alternative for histogram comparison. However, it did not influence the work presented in this thesis, as SVMs show a characteristic tendency to binary classification. Thus, a classification result contains no information about the degree of assurance. The following chapters will show that such an information can be used to refine expectations iteratively.

3.2 Structure of a Query

Object representations are already calculated in the training phase, where every object $O$ results in a model $M$ [Eq. (2.24)]. A model includes the characteristic feature-distribution histogram $H_M$ and three object-specific thresholds $\delta_{M,\text{max}}$, $r_{M,\text{max}}$, and $\tilde{r}_M$.

Assuming the necessity to distinguish $k$ different objects $O_1, \ldots, O_k$ the database

$$\Omega = \{M_j | O_j \leadsto M_j, j \in \mathbb{N}_{1,k}\}$$  \hspace{1cm} (3.1)

is used for reference. The term $\mathbb{N}_{1,k}$ is the shorthand notation for the discrete interval

$$\mathbb{N}_{1,k} = \{1, 2, \ldots, k\} \in \mathbb{N},$$  \hspace{1cm} (3.2)

of cardinal numbers $\mathbb{N}$. The operator $\leadsto$ symbolizes that the object $O$ leads to the model $M$ with all parameters included.

An unknown object $\hat{O}$ with the purpose of classification relatively to the database is called a query. As mentioned before, the invariance to rotation and translation does not allow to recognize the pose of an object. Instead a query permits to segment the set

$$S = \{s_1, \ldots, s_n\}$$  \hspace{1cm} (3.3)
3.3. Histogram-similarity Criteria

The models contain shape information as a feature-distribution histogram. Thus, it is mandatory to transform the distribution of the sample set \( S \) into a histogram as well and then compare both.

A set \( S \) is an arbitrary selection of object aspects. It cannot be assured that \( S \) contains every detail of an object. Thus, the set \( S \) is deemed to be incomplete, in the sense that the sample set cannot cover the whole range of possible surflet combinations. According to this estimation the maximum distance \( \delta_{\text{max}} \) is incondent. It cannot be assured that the sample set contains the two surface points with the largest distance occurring in an object. Hence, the set \( S \) cannot be used to determine the fourth parameter of the feature vector [Chap. 2.3, p. 23]. According to this, the feature-distribution histograms of surflet-pairs have to be generated with respect to the model \( M \) that is used for comparison. This means, comparison of the sample set according to a model always includes the processing of the model specific histogram determined by the model specific maximum distance \( \delta_{M,\text{max}} \). The function

\[
\phi_{\delta_{M,\text{max}}} : \{ f_i | f_i = \sigma(s_i), s_i \in S \} \mapsto \hat{H}_M
\]

maps the corresponding feature vectors \( f_i \) of the sampled surflet-pairs \( s_i \) to a histogram \( \hat{H}_M \). Herein, \( \hat{H}_M \) adopts the model specific histogram range \( \delta_{M,\text{max}} \). As previously done for the models, the query histogram is normalized.

Comparing two distributions, where one distribution is of unknown range, implies that not all features of one distribution necessarily fulfill the range limits of the other, namely, if \( \delta_{\text{ref}} > \delta_{M,\text{max}} \) is considered. Then features map outside the covered histogram space. To handle these cases, the mapping function, previously introduced in the training phase [Eq. (2.21)] needs a modification in the query phase. Therefore, all outlying features are varied by limiting the fourth parameter \( \delta_{\text{ref}} \) to the upper border \( \delta_{M,\text{max}} \).

Once the query histogram is generated, there are several possibilities to measure the similarity to the model histograms. In the following, five criteria are considered. A detailed experimental evaluation is given in Chap. 3.6, p. 37.

### 3.3.1 Squared Euclidian Distance

The first criterion is the squared Euclidian distance

\[
\mathcal{E}(H_M, \hat{H}_M) = \sum_{b=1}^{d} (H_M(b) - \hat{H}_M(b))^2 .
\]
It is a straightforward method that measures the length\(^1\) of a \(n\)-dimensional vector difference. According to this, the histograms are treated as vectors. The difference is zero for a perfect match or a positive value otherwise. The best database fit is the model with the minimum sum

\[
\hat{E} = \min\{E(H_{M_j}, \hat{H}_{M_j})|M_j \in \Omega, j \in N_{1,k}\} .
\] (3.6)

This geometric interpretation is known for weak generalization capabilities and sensitivity to disturbances. A reason for this is that the squared distance of an outlier contributes significantly to the final sum. This affects similarity often more than desired, especially if considering that the set of samples is the result of a statistical process where the samples vary in their distribution. Accordingly, the differences to the trained models increase fast and will not allow for trustworthy classification, particularly if the number of samples is kept as small as possible for low processing costs. In real scenarios, noise and incomplete data additionally affect the classification. Both further decrease the weak classification properties of the squared Euclidian distance.

### 3.3.2 Intersection

Another criterion for comparison with a model histogram \(H_M\) is the intersection

\[
\bigcap (H_M, \hat{H}_M) = \sum_{b=1}^d \min(H_M(b), \hat{H}_M(b)) ,
\] (3.7)

which is a simple minimum operator. It is often used with fuzzy-set techniques, where it is known as the \textit{fuzzy-AND-operator}.

In the beginning, it summarizes the minimum conformance with regard to each model \(M_j \in \Omega\). Due to normalization of the model histograms, the maximum sum is 1 indicating a perfect match with total conformance. A value of 0 corresponds to the minimum conformance.

After calculation of every model-specific sum \(\bigcap (H_{M_j}, \hat{H}_{M_j})\), the best match is chosen, which is the maximum

\[
\bigcap = \max\{\bigcap (H_{M_j}, \hat{H}_{M_j})|M_j \in \Omega, j \in N_{1,k}\} .
\] (3.8)

Ballard and Swain [SB91] previously applied the intersection to color-histogram classification. It is very fast to compute, because apart from summation no arithmetic operations are needed.

The recognition performance is unsatisfying due to immense mal-classifications. Nevertheless, it is useful if demands on classification are low and similar results are sufficient. Wolf et al. [WBB05], for instance, successfully applied the intersection criterion for an image-retrieval system that is used to localize a robot.

### 3.3.3 \(\chi^2\)-Test

In statistics distributions are compared whenever the test for goodness of fit is applied. A popular method in this domain is to measure the similarity between two histograms

\(^1\)In a mathematical correct manner the length includes a final calculation of the square root. Since that operation does not change the order of the comparison results, it can be disclaimed to avoid additional processing costs.
with the $\chi^2$-test. Similar to the squared Euclidian distance, it summarizes the squared differences of the histogram bins. In addition, each difference is normalized individually.

In literature the $\chi^2$-test appears in two forms that differ in the normalization term. The first formulation

$$
\chi_1^2(H_M, \hat{H}_M) = \sum_{b=1}^{d} \frac{(H_M(b) - \hat{H}_M(b))^2}{H_M(b)}
$$

(3.9)

normalizes the dividend with respect to the model term. Accordingly, the $\chi_1^2$-test shows a non-symmetric behavior. In the case of $H_M(b) \gg \hat{H}_M(b)$ a value near but beneath 1 is added to the sum. It occurs, whenever less features are sampled than expected in the model. The other case $H_M(b) \ll \hat{H}_M(b)$ tends to infinity. Here, the maximum values will be obtained, if a lot of features are sampled, while the the model denies the occurrence. The conclusion is that the classification process strongly depends on the model expectations with respect to feature quantization. Finding less features than expected has less negative effect to similarity measurement than a confrontation with much more features than assumed.

The second formulation is the symmetric variant

$$
\chi_2^2(H_M, \hat{H}_M) = \sum_{b=1}^{d} \frac{(H_M(b) - \hat{H}_M(b))^2}{H_M(b) + \hat{H}_M(b)}
$$

(3.10)

that treats the query histograms and the model histogram as peers. Thus, the divisor consists of the sum of both. There is no longer a difference for the case of $H_M$ being much larger than $\hat{H}_M$ or vice versa. The maximum difference of one bin being 1 and the other bin being 0 limits the contribution of a bin comparison to an upper bound of 1.

Both cases result in $\chi_1^2, \chi_2^2(H_M, \hat{H}_M) = 0$ for a perfect match. Otherwise, the best match is assumed to be the minimum

$$
\hat{\chi}_1^2 = \min \{ \chi_1^2(H_{M_j}, \hat{H}_{M_j}) | M_j \in \Omega, j \in N_{1,k} \}
$$

(3.11)

and

$$
\hat{\chi}_2^2 = \min \{ \chi_2^2(H_{M_j}, \hat{H}_{M_j}) | M_j \in \Omega, j \in N_{1,k} \},
$$

(3.12)

respectively.

Both versions of the $\chi^2$-test require $d \times k$ divisions to find the best fit in a database. Compared to summation and multiplication the division operation increases larger processing. As a result, they exceed the processing time of the squared Euclidian distance and the intersection. Interestingly, the $\chi_2^2$ criterion performs dramatically weaker than the $\chi_1^2$ criterion. Apparently, the weighting of histogram differences by the reciprocal of the model histogram value alone is much more reliable than also taking the estimate from the test sample into account [Chap. 3.6, p. 37].

Finally, the $\chi_1^2$-test still cannot achieve high classification rates and seems suited for coarse searches, at most.
3.3.4 Kullback-Leibler Divergence

Another criterion from statistics is the Kullback-Leibler divergence [KL51] that is also known as relative entropy. The original notation

\[ K(p, q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)} . \]  

(3.13)

introduced by Kullback and Leibler defines the similarity of two distributions \( p \) and \( q \) in the sense of information theory. It describes the number of wasted bits, when applying the coding of \( q \) on \( p \). Correspondingly, a large value points to a large divergence, while a small value indicates a homogeneity.

In this work, the symmetric form of the Kullback-Leibler divergence

\[ K(H_M, \hat{H}_M) = \sum_{b=1}^{d} (\hat{H}_M(b) - H_M(b)) \ln \frac{\hat{H}_M(b)}{H_M(b)} \]  

(3.14)

is tested. The criterion adds a value of 0 to the sum every time the condition \( \hat{H}_M(b) = H_M(b) \) is fulfilled. For those cases in which the histogram bins differ, a value larger than 1 is added. The higher the difference the higher is the increment. As a consequence, the best model match is defined as follows:

\[ \hat{K} = \min \{ K(H_M, \hat{H}_M) | M_j \in \Omega, j \in \mathbb{N}_{1,k} \} . \]  

(3.15)

Taking a look at Eq. (3.14), one can see that it includes two expensive operations in terms of processing namely a division and a logarithm. Both involve \( d \times k \) function calls per query. Hence, it is not surprising that it is much slower than all other tested criteria. Nevertheless, the classification rate shows very good results [Chap. 3.6, p. 37].

3.4 Likelihood Criterion

The comparison methods discussed before assume that the two distributions are both given as histograms. Then a criterion that measures the distance to a reference is used to select the best match. The likelihood criterion interprets a query in the sense of probability theory. The likelihood that a feature \( f_s \) originates from an object \( O \) changes parameterization. The probability can be taken from the normalized model histogram.

Drawing a small, random subset of all features, it can be safely assumed that individual samples are statistically independent of each other. The logarithmic likelihood of object \( O \) which is described by the database model \( M \) and the histogram \( H_M \), is

\[ L(M|S) = \sum_{s \in S} \ln H_M(h_M(f_s)) . \]  

(3.16)

The processing cost of this formulation depends on the size of the surfeit set \( S \). If the query distribution is already available in form of a histogram, the probability could be used:

\[ L(H_M|\hat{H}_M) = \sum_{b=1}^{d} \hat{H}_M(b) \ln H_M(b) . \]  

(3.17)

in this case the number of summations only depends on the fixed number of histogram bins. This notation will get important in the scene interpretation chapter [Chap. 5.4,
3.5. PENALTY TERM

Equation (3.17) is advantageous because it reduces the number of samples to the more compact histogram notation. This reduces processing costs in every iteration. In contrast to the Kullback-Leibler divergence (3.14), all logarithms can be calculated in a preprocessing step in the training phase. In contrast to other criteria, the histogram consists of the logarithmic bins \( \ln H_M(b) \).

Expanding Eq. (3.16) (and Eq. (3.17) analogous) to the maximum likelihood

\[
\mathcal{L}(M|S) = \frac{\sum_{x \in S} \ln h_M(f_x))}{\sum_{b=1}^{d} \ln H_M(b)}
\]

increases processing time without providing better recognition results. Nevertheless, the maximum likelihood can be useful since it normalizes the criterion and allows for the inclusion of thresholds that permit to set a minimum reliability.

In conformance with Schiele and Crowley [SC00] who tested the same criteria in the field of 2D-image processing, this work comes to similar results for 3D-object recognition. The likelihood criterion shows the best recognition results, slightly better than the Kullback-Leibler criterion. Due to simple operations it further provides fastest processing. Since the likelihood criterion outperforms other criteria in all examined aspects it is applied for histogram comparisons in the ongoing work.

3.5 Penalty Term

Applying the \( \chi^2 \)-tests, the Kullback-Leibler divergence, and the likelihood criterion in practice shows that they will fail. This results from empty bins in the model histograms. The whole range of feature parameters \( f_a = (\alpha, \beta, \gamma, \delta) \) is neither necessarily covered by every subsample, nor even by a full sample of features from an object. Often shapes exist which do not provide specific surface-point combinations. Hence, some features will not occur and the corresponding bins of a histogram may remain zero. An ideal plane for example contains only parallel surface normals such that the histogram does not use the whole range for the angular components of the feature space. This leads to numerical problems when computing divisions or logarithms. A similar problem occurs in the presence of noise. One noisy feature that is not covered by the histogram, for example one single point above an ideal plane, contributes with an infinite value. As a result, the total sum drifts to infinity, too. This is correct in theory, but not acceptable in practice. Figure Fig. 3.1 visualizes the impact of unmodified empty histogram bins.

The solution is to reduce the negative effects of undesirable features by comparison. In cases like these, all zero bins of a histogram are set to a common value

\[
H_{\text{min}} = \tau \min\{H_M(b)|M_j \in \Omega, b \in \mathbb{N}_{1,d}, j \in \mathbb{N}_{1,k}\},
\]

lower than the lowest non-zero value occurring in all histograms. This value has the effect of a penalty term. The influence of this term depends on \( \tau \in [0, 1] \). The smaller \( \tau \) is, the larger the influence of the penalty term.
CHAPTER 3. HISTOGRAM-BASED CLASSIFICATION

Figure 3.1: The sketch shows the problem of zero bins in the case of the likelihood criterion, when using a histogram without penalty term. One single zero value in the model histogram leads the final sum towards infinity.

Assuming the number of zero bins to be $\eta_{M_j}$ with respect to a model $M_j$ and taking into account that a normalized model histogram $H_M$ has to follow the condition

$$\sum_{b=1}^{d} H_M(b) = 1$$

(3.21)

leads to

$$H'_M(b) = \begin{cases} 
H_{\min}, & \text{if } H_{M_j} = 0 \text{ and } \\
(1 - \eta_{M_j} H_{\min}) H_{M_j}(i), & \text{elsewhere}
\end{cases}$$

(3.22)

for each bin $b \in \mathbb{N}_{1,d}$ of the modified histogram $H'_M$. Finally, the changes still satisfy condition (3.21) and preserve the normalization. Notice that in contrast to the database histograms, the query does not require an elimination of zero bins. Therefore, processing time is not affected negatively.

The modification of the model histograms can be interpreted as the addition of a uniform “background distribution”. It permits the processing of formerly impossible surflet-pairs but assumes them to be very unlikely. Thus, such features will cause lower similarity but prevent complete dissimilarity due to a few outliers.
3.6 Experiments: Descriptiveness and Classification Rates

The first question to be answered concerns the descriptive capacity the presented object representation possesses. In other words: How powerful is the ability to distinguish objects and how much distortions can occur until recognition fails?

The second subject of this section is to evaluate which similarity criterion [Chap. 3, p. 29] performs best in terms of recognition reliability and processing time.

The evaluation is based on 20 synthetic objects shown in Fig. 3.3. They feed a database containing a broad range of object types that vary from un-symmetric free-forms to symmetric geometrical primitives. The broad variation of shapes shows that there is no restriction or special demand to the available representation of an object type.

All objects are initially surface meshes, which are however unrelated to the meshes used as inputs to the algorithm. To ensure that classification cannot be dominated by object size, all objects are scaled to a common maximum diameter. Otherwise the feature component that is the Euclidian distance between two surface points would be sufficient to produce high recognition rates as Osada et al. has shown in [OFCD02].

3.6.1 Procedure of Training Phase

Models are trained with the procedure described in Chap. 2.3.1, p. 25. It is the surface mesh based variant of training. For each object, five sets of points from 25,000 up to 389,000 points per set are drawn randomly from the surface and passed to a mesh generator. Since the surface meshes satisfy a Delaunney triangulation, the surface is assumed to be homogeneously sampled. Using five meshed versions of an object instead of one fine mesh has several advantages. First, the details of shape are nearly the same, while second, the number of distortions due to noise are reduced. Furthermore, the number of surface points per mesh is significantly smaller. This is important, keeping in mind the combinatoric explosion that takes place in the following step of the training phase.

A training mesh consists of between 3,500 and 5,500 vertices. Features are built from pairs of surflets, which are in turn picked from each vertex. All features obtained from a training mesh (between 30,616,250 and 75,611,250 features), are collected into a histogram. Further-on, the five histograms are bin-wise added and normalized so that finally the mean object histogram is received. Figure 3.2 shows a sketch of the training procedure.

Figure 3.2: A sketch of the model generation based on various surface meshes.
Figure 3.3: The 20 synthetic objects of the database.
3.6. EXPERIMENTS: DESCRIPTIVENESS AND CLASSIFICATION RATES

The alternative training phase of directly drawing surflets from the point cloud [Chap. 2.3.2, p. 26] was avoided since the demand for a homogeneous sampling of the surface would have caused a much larger number of surflets. As mentioned, this would not allow to calculate the total number of possible features in acceptable time.

3.6.2 Procedure of Recognition Phase

In the recognition phase new meshes are generated from each object. The point clouds and the resulting meshes differ to the ones of the training phase. Features are randomly sub-sampled from the vertices of these meshes. The number of features drawn is 0.005% of all available features. This arbitrary, low sampling rate turns out to be high enough for good recognition. A simple heuristic condition for the minimum number of surflet samples is to be larger than the amount of histogram bins.

3.6.3 Surface Meshing

The mesh generator used in this context was an own simple development before working on this thesis. It works fully automatic but it suffers from artefacts caused by very close surfaces of different orientations. In the database models such surfaces occur for thin parts. Examples are parts like wings or objects that consist of geometric primitives that penetrate each other, e.g. the A-10 or the cannon.

Disturbances from the mesher are omni-present even in the case of ideal conditions. The impact to recognition can be assumed to be very low, since it takes place in training phase as well as in recognition phase. Therefore, the disturbances are treated as characteristic parts of every object.

3.6.4 Ideal conditions

Under ideal conditions\(^2\) the test objects’ surfaces are completely exposed to the sensor and sensed data are free of noise. Table 3.1 shows the achieved recognition rates and times for the six criteria [Chap. 3.3, p. 31]. The measured times include all steps from drawing feature samples to the output of the best matching object model. Generating the surface mesh and calculating the surface normals are not included.

Almost perfect classification has been achieved by the Kullback-Leibler divergence \(K\) and the likelihood criterion \(L\). Regarding the corresponding processing times, the likelihood criterion significantly outperforms the Kullback-Leibler divergence (KLD). The reason is that the KLD requires the expensive processing of logarithms and divisions at recognition stage [Eq. (3.14)], while the likelihood criterion is restricted to a cheap summation and allows for the processing of logarithms in the training phase [Eq. (3.16)].

Interestingly, the \(\chi^2\) criterion performs dramatically weaker than the \(\chi^2\) criterion. Apparently, the weighting of histogram differences by the reciprocal of the trained histogram value alone is much more reliable than also taking into account the estimate from the small test sample [Eq. (3.9), Eq. (3.10)].

Correct classification and confusion rates between all pairs of objects are shown in Fig. 3.4. All classifiers work well for simple shapes like cube or sphere. Interestingly, the objects that are difficult to classify differ drastically across the criteria. On the other

\(^2\)Notice that even ideal conditions still include unpredictable variations and failures residing from the sampling process and the mesh generator.
hand, the intersection criterion \( \cap \) and \( \chi^2 \) criterion exhibit a strikingly similar pattern of classification performance. This similarity will also be retained in all the other tests of the classifiers reported below. The same similarity holds for the two best criteria, the Kullback-Leibler divergence \( K \) and the likelihood \( L \).

Table 3.1: In this test, the six classifiers defined in Chap. 3.3, p. 31 are evaluated using randomly drawn feature samples from complete and noise-free surface meshes of the 20 objects shown in Fig. 3.3. Achieved recognition rates are given in percent; processing times are measured in milliseconds. The processing times are measured on a standard PC with an Intel Pentium IV 2.66 GHz processor, 1GB RAM and Linux as operating system.

<table>
<thead>
<tr>
<th>criterion</th>
<th>recognition in %</th>
<th>time in ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \cap )</td>
<td>42.7</td>
<td>5.12</td>
</tr>
<tr>
<td>( E )</td>
<td>40.6</td>
<td>5.01</td>
</tr>
<tr>
<td>( \chi^2 )</td>
<td>75.4</td>
<td>6.16</td>
</tr>
<tr>
<td>( \chi^2 )</td>
<td>45.5</td>
<td>6.25</td>
</tr>
<tr>
<td>( K )</td>
<td>99.6</td>
<td>7.42</td>
</tr>
<tr>
<td>( L )</td>
<td>99.7</td>
<td>4.79</td>
</tr>
</tbody>
</table>

3.6.5 Noisy data

If the point cloud is obtained from real sensors like laser range-scanners, laser-strip profilers, or stereo cameras, etc., the data will be corrupted in various ways. Therefore, in a second set of experiments, the sensitivity of the feature histograms to noise is evaluated.

Uniformly distributed noise is simulated by randomly translating vertices from a surface mesh inward or outward along the local surface normal. The level of noise is defined as the maximum range of translations, measured in percent of the maximum object diameter\(^3\).

In Fig. 3.5(b) plots of recognition rates are presented for the six classifiers as a function of noise level. For the \( E, \chi^2, K, \) and \( L \) criteria, classification performance rapidly degrades with increasing noise. This is explained by the fact that the angular attributes \( \alpha, \beta, \gamma \) are very sensitive to noise such that surface information is largely lost. Interestingly, these criteria reach a rather stable rate of between 10% and 15% correct classification. This recognition rate is still significantly above 5%, which would be the expected result of a randomly answering system. Thus, the representation must contain a descriptive element that is more robust with respect to noise. Some residual performance may be expected as the distance attribute \( \delta \) remains informative up to much higher noise levels.

The \( \chi^2 \) and \( \cap \) criteria on the other hand, are a lot less sensitive to noise, exhibiting significantly lower performance at low noise and higher performance at high noise levels.

As an example Fig. 3.5(a) shows a surface mesh of the x-wing fighter corrupted by a level of 4% noise. Unfortunately, a typical stereo image usually is comparably low in inaccuracy. Since most stereo approaches search for corresponding patterns in two images, the resolution of the disparity map depends on the pattern size. Consequently,

\[^3\]Remember that the diameter was scaled to the same value for all objects.
3.6. EXPERIMENTS: DESCRIPTIVENESS AND CLASSIFICATION RATES

Figure 3.4: The six arrays represent classification results for the 20 objects shown in Fig. 3.3 using the six different criteria defined in Chap. 3.3, p. 31. Surfaces are completely visible and data are noise-free. In each array, columns represent test objects, rows trained objects. Grey values indicate the rate of classification of a test object as a trained object; a brighter shade shows a higher rate. The more distinct the diagonal, the higher the overall performance of the classifier. Evidently, the $K$ and $L$ criteria achieve almost perfect classification within the database of objects.

Stereo images often show structures similar to stairs. Thus, the calculation of the surface normals suffers and leads to an unacceptable recognition rate. Stereo approaches that address the smoothness of surfaces [Wei98] imply complex processing and do not satisfy time limitations for online applications.

Using the 3D-Modeler introduced in Chap. 1.5, p. 14 an absolute accuracy of below
±1mm for the laser range-scanner and the laser-strip profiler can be expected. Scanning objects with a diameter larger than 200mm leads to a maximum noise of 1%. Assuming a structure of noise as tested in this experiment, the $K$ and $L$ criteria yield a reasonable recognition rate above 80%. Recognition performed on real data can yield higher results since relative errors, i.e. the error of one point to its temporal neighbors\(^4\) is below the worst case scenario tested here.

![Image](image.png)

Figure 3.5: Plots of recognition rates for the 20 objects shown in Fig. 3.3 using the six different criteria defined in Chap. 3.3, p. 31. The conditions for the test data are a varying level of noise (in percent of maximum object diameter). The curves for the $K$ and $L$ criteria nearly coincide in the graph.

### 3.6.6 Partial visibility

In real data applications, i.e. systems which work with measured data, objects to be recognized are often partially visible only. Reasons are self-occlusion in single-view data, occlusions by other objects, accidental surface reflectance, or simply unreachable surface parts with respect to the sensor device. Partial objects yield incomplete surface meshes. Therefore, in this set of experiments, each test object is meshed and classified with varying fractions of visible surface.

Visible parts are determined by intersecting surfeit clouds by a random plane. Subsequently, data on one side of the plane are processed by the mesh generator. Visibility is defined as the sum of remaining triangle areas in percent of the complete surface area. Figure 3.6(a) gives an example of the partially visible x-wing (33%). The mesh consists of the fighter’s tail and nearly two wings.

Results on recognition rates for various visibilities are plotted in Fig. 3.6(b). Performance can be seen to drop off more gradually with occlusion than with data corruption by noise [Fig. 3.5(b)]. Correct classification by the $K$ and the $L$ criteria remains above 80% down to roughly 65% visibility. Exposing less than 25% of the object surface to the algorithm leads to a collapse the recognition rate.

Notice that recognition rate with partial visibility depends in fact heavily on the particular section of the object, which remains visible.

\(^4\)Temporal neighbors define very closely acquired points, e.g. points in one scanned line.
3.6. EXPERIMENTS: DESCRIPTIVENESS AND CLASSIFICATION RATES

Figure 3.6: Plots of recognition rates for the 20 objects shown in Fig. 3.3 using the six different criteria defined in Chap. 3.3, p. 31. The conditions for the test data are a varying visibility in percent of the complete surface area. The curves for the $K$ and $L$ criteria nearly coincide in the graph.

3.6.7 Generalization Across Mesh Resolution

Since this thesis relies upon surface meshes as the input representation, it is interesting to ask how recognition performance is affected by changes to the mesh procedure. One of the most demanding scenarios is generalization across mesh procedures. Generalization implies performing the recognition phase with a mesh of a type essentially different from that which training has been based on.

Accordingly, a final set of experiments has investigated the effect of varying the mesh resolution for the test objects. Figure 3.7(a) shows the x-wing in training mesh resolution which is a space of $40 \times 40 \times 40 = 40^3$ buckets. This means that the maximum resolution per dimension is 40 points. Figure 3.7(b) depicts plots of correct-classification rates under such conditions, where the mesh resolution is given in percent of the (constant) resolution in the training phase.

Apparenty, recognition performance does not critically depend on test-mesh resolution. Only below 50% of the training resolution, recognition performance drops off. In part, this can be ascribed to the low absolute number of feature samples drawn. In particular, the $K$ and $L$ criteria exhibit a high degree of generalization across meshes.

The conclusion to this experiment with respect to recognition phase is to use a lower resolution than in training phase that is still above 50% of training mesh resolution. This is advantageous since a lower resolution decreases artefacts resulting from noise and it decreases the number of samples necessary to describe the surface.

3.6.8 Generalization and Similarity

Chapter 2.4, p. 26 stated that the number of histogram bins is responsible for the model’s ability to generalize around similar but not equal instances of an object, e.g. the capability to distinguish apples form a class of coffee cups. One will talk of a high ability to generalize, if the maximum number of dissimilarities that still allows a correct object classification is large and vice versa. Here, this topic is discussed with respect to the four-dimensional feature representation [Chap. 2.2, p. 20]. It is examined which effect variation of the object shape has on recognition rate. Therefore, local and global
Figure 3.7: Plots of recognition rates for the 20 objects shown in Fig. 3.3 using the six different criteria defined in Chap. 3.3, p. 31. The conditions for the test data are a varying level of mesh resolution (in percent of training resolution). The curves for the $K$ and $L$ criteria nearly coincide in the graph.

variations have to be distinguished.

Local variations describe changes which are limited to a region of the object, e.g. this might be the difference between a horse and a unicorn, being both shapes identical except the horn. Of course, the effect strongly depends on the size of the affected region. This type of shape variation is comparable to the already examined case of partial visibility.

The second group addresses global changes of shape. In this case, the variation of an object is treated as distortion. Since the range of possible distortions is too large to evaluate them all, the focus lies on affine transformations, which also include scaling.

One can define

$$\tilde{f}_n = f_n + w$$

as the feature with a certain variation $w$ from the undistorted object feature $f_n$. The hypothesis is that a small variation $w$ leads to nearly the same feature. Then, the function $h_M(\tilde{f}_n)$ [Eq. (2.21)] can be assumed to mostly map to the same bins as it does for features of the original object. Accordingly, the histograms of the undistorted and distorted shape should be very similar with respect to the likelihood criterion. Here, the relation between the degree of affine transformation and achieved classification rate is investigated in two experiments.

Both experiments base on the object models of the synthetic database depicted in Fig. 3.3. As mentioned, every model uses five subdivisions per feature dimension. This leads to $5 \times 5 \times 5 = 5^4 = 625$ histogram bins. Accordingly, one gets ranges of

- $\eta r^o_{\alpha}$ for $\alpha$,
- $\eta r^o_{\beta}$ for $\beta$ and $\gamma$, and
- $\eta r^o_{\max_{\delta}}$ for $\delta$

[Chap. 2.2, p. 20].

In both experiments test objects are presented to the algorithm, which are derived from the point clouds of the trained database objects. To investigate the ability of categorizing into classes of similar objects, they are modified by applying affine transformations to the point clouds. Therefore, it is possible to define

$$D = r I$$
3.6. EXPERIMENTS: DESCRIPTIVENESS AND CLASSIFICATION RATES

as the transformation that maps the database object \( O \) to the distorted test object

\[
O' = DE^{-1}O.
\]

The vector \( r_\xi \) determines the direction of a random positive or negative elongation, where \( \xi \) scales the distortion. The matrix \( E(e_1^T, e_2^T, e_3^T) \) is the eigensystem of the undistorted object.

**Elongation of One Principle Axis**

The first experiment will address the question, whether the direction of shrinking or growing yields significant changes in the classification rate, i.e. if some direction has higher impact on similarity than another.

If such a dependency exists, it will be largest for the directions of the maximum or minimum disorder. Such directions are the eigenvectors of the surfeit distribution in 3D-space that is expressed with the covariance matrix. In the context of 3D-objects the eigenvectors are called the principle axes. To distinguish the axes, an order is established. The first axis refers to the eigenvector with the largest eigenvalue. The second and third axes correspond to the eigenvectors of the second largest and smallest eigenvalue, respectively. Thus, the vector \( r_\xi \) is selected to be \( (1 + \xi, 0, 0)^T \), \( (0, 1 + \xi, 0)^T \), or \( (0, 0, 1 + \xi)^T \).

In the experiment, the degree of elongation is divided into 30 homogenous intervals starting with 0.4 times to 2 times in relation to the undistorted axis length. The experiment is repeated with 20 random values for \( \xi \) per interval.

The plots in Fig. 3.9 show the mean classification rate for the first (a), second (b), and third (c) principle axis. Here, the x-axis displays the value of \( \xi \), while the y-axis represents the percentage of correct classification. Accordingly, a value of 1 belongs to the undistorted object.

The test displays that shrinking an object affects the similarity much more than growing it. A direction dependent compression of 50% leads to a classification rate below 40%, while an enlargement of 50% still allows for results above 70%. Since all axes perform very similar, there is no reason to assume a significant correlation between the classification rate and the direction of a distortion.

**Multi-axial Distortion**

In the second experiment the distortion affects all principle axes. The transformation matrix

\[
D = \begin{bmatrix}
\xi_1 & 0 & 0 \\
0 & \xi_2 & 0 \\
0 & 0 & \xi_3
\end{bmatrix}
\]

(3.26)
is determined by the random distortion coefficients \( \xi_1 \), \( \xi_2 \), and \( \xi_3 \). To allow for one-dimensional plots, a metric is needed to measure the distortion. Using the mean value is not advisable, since an equal positive and negative elongation in different dimensions compensate each other. If one further assumes the last dimension to be unchanged (that is a value of 1), then the mean of all axes will be 1, too, which is the same value as it will be for the undistorted case. However, a metric should be able to distinguish both cases.

Using the Euclidian distance solves the first problem, but makes no difference between shrinking and growing. Since the previous experiment shows different behavior
in cases like these, the Euclidian distance is modified. Therefore, the metric
\[
\Delta = \begin{cases} 
\|D'\| & \text{for } D' \circ (1, 1, 1)^T \geq 0 \text{ and } \\
-\|D'\| & \text{else},
\end{cases}
\] (3.27)
is applied, where
\[
D' = DD(1, 1, 1)^T - (1, 1, 1)^T
\] (3.28)
to quantify the distortion. The translation included in \(D'\) shifts the undistorted case, where \(\xi_1 = \xi_2 = \xi_3 = 1\) to the origin. The negative axis represents mainly shrunk objects, while the positive x-axis represents elongation.

The parameters \(\xi_1\), \(\xi_2\), and \(\xi_3\) are randomly varied from 0.2 times to 2.3 times of the undistorted axis length. It includes the cases where
- only one randomly chosen principle axis,
- two randomly chosen principle axes,
- and all principle axes
suffer distortion. As for the first experiment every combination in every of the 30 distortion intervals was repeated in 20 iterations.

Figure 3.8 shows the mean result for all objects. As one can clearly see, the classification rate for elongation again outperforms the one for shrinking. The reason might be that in the case of a shrunk object all features fulfill the demand to be beneath \(\delta_{M, \text{max}}\). Thus, all features are taken into account, which seems to highly affect similarity. In case of growing a smaller number of features is valid, since many are above the maximum object diameter \(\delta_{M, \text{max}}\). The remaining features are in a closer neighborhood. They show the same relative changes with respect to the total distortion. However, the discretizing mapping function \(h_{M, \ell}\) depends only on absolute changes. Accordingly, the curve of classification rate drops slower for elongated objects. The peaks to left and right side are results of outliers and of too few examples at both ends of the domain.

Taking a separate look at each object shows very different plots as shown in Fig. 3.10 to Fig. 3.14. Obviously, the ability to classify similar objects is very dependent on the context of a database. The result of classification is a product of object characteristics and the differences between database objects. According to this, it will be difficult to state whether an object or a database is suited for categorizing similarity classes. Nevertheless, if object changes are small, e. g. as it will be expected for classes of apples, jogging shoes, plates, etc. the representation will satisfy.

### 3.6.9 Conclusion to Descriptiveness and Classification

This section discussed the descriptiveness of the introduced novel four-dimensional feature that uses the intrinsic geometrical relation between a pair of surfllets, i.e. between oriented surface point-pairs in 3D space. The statistical distribution of this feature-type as sampled from an object’s surface captures both local and global aspects of shape. Empirically trained histograms of the feature distribution have been demonstrated as a compact and efficient representation of arbitrary 3D shapes. This representation allows for rapid classification of shapes based on a single histogram per object model, which is independent of translation and rotation.

Six different criteria for the shape classifier were evaluated. The Kullback-Leibler and likelihood criteria have been found to perform equally well and superior to the
3.6. EXPERIMENTS: DESCRIPTIVENESS AND CLASSIFICATION RATES

Figure 3.8: The plot shows the mean classification rate over all 20 objects of the database with respect to affine distortions.

others. The Kullback-Leibler and likelihood criteria have shown nearly perfect classification under ideal sensing conditions and robust performance in the face of noise and occlusion. They are, moreover, largely independent of the resolution used for meshing surfaces of test objects. Considering its lower computational cost using the much faster likelihood classifier is recommended.

More specifically, the experiments clearly indicate that for best performance, noise should be reduced during recognition by spatial averaging. Fortunately, decreasing mesh resolution from training phase to query phase achieves this, while keeping processing low.

Moreover, the four-dimensional representation has shown its ability to categorize an object in the presence of weak to medium distortions of shape. Apart from the desired scene interpretation task, the object representation seems suited for similarity search. Such applications, e.g. 3D-object search engines running on the Internet, mostly use prototypes to formulate queries. The design of the presented object description and the efficient comparison to stored datasets, perfectly fit the demands. Since these search engines mostly assume complete objects, it makes sense to normalize models with respect to object size. This provides invariance in scale.
Figure 3.9: The plots show the effect of elongating an object along one of the principal axes. The distortion factor describes the relative length of an elongated axis. Thus a value of 0.5 means that an axis is shrunk to half of its original length. In plot (a) the elongation is applied to the longest principal axes, while (b) refers to the second longest and (c) to the shortest axis.
Figure 3.10: Distortion results for the A10, the ape, the buffalo, and the bull.
Figure 3.11: Distortion results for the bunny, the cat, the cannon, and the cube.
3.6. EXPERIMENTS: DESCRIPTIVENESS AND CLASSIFICATION RATES

Figure 3.12: Distortion results for the cylinder, the dragon, the Gumby, and the heart.
CHAPTER 3. HISTOGRAM-BASED CLASSIFICATION

Figure 3.13: Distortion results for the horse, the kangaroo, the missile, and the shark.
3.6. EXPERIMENTS: DESCRIPTIVENESS AND CLASSIFICATION RATES

Figure 3.14: Distortion results for the sphere, the tetrahedron, the triceratops, and the X-wing.
EFFICIENT data access is an omnipresent and often neglected issue. One reason might be that many approaches are based on relatively small data sets. The data analysis dominates the total computation, so that the performance of the organization of data is of minor importance.

In the context of real-time-applications the circumstances are different. Here, applications consist of several cooperating modules, while processing capabilities are limited. Accordingly, it is obvious that efficiently working modules are necessary. This becomes more and more important with increasing amount of data as it for example occurs in 3D-applications. In circumstances like these, the relation of processing costs needed for data management and data interpretation shifts towards the management side.

In this chapter a method will be introduced that allows for efficient organization of 3D-point clouds with respect to read access. It further provides update capabilities dynamically, which are important since additionally measured 3D-points have to be included. The emphasis is placed on a region search algorithm which is presented in the second part of this chapter. The introduced data structure and the corresponding operators will further be referred as *space representation*.

### 4.1 The Neighborhood Problem

Typically, an algorithm applied to a measured 3D-scene does not need the whole data set for decision making. The usual case is the desire for some local information, e.g. only a region which is supposed to be an object on a table. Unfortunately, segmentation of such a focused region requires a-priori knowledge about the structure of the scene. It is not self-evident that this knowledge is given or easy to obtain. Thus, a common approach is to reduce a scene to a local area around a *region of interest* (ROI). In the following, the term *focused area or region* is synonymous to *neighborhood*. Its size is chosen such that the ratio of interesting to uninteresting data it optimized. This is
necessary for many 3D-algorithms which use local information around a spotted center for further decisions.

Extracting such a subset is a nontrivial preliminary step, especially when using unorganized point clouds. It strongly depends on the 3D-data representations and, therein, the realization of spatial neighborhood relations.

A common procedure to organize the data is applying a spatial subdivision tree. Tree representations differ in their strategies when partitioning space, but they agree on relying on a hierarchical structure. Each data point is mapped to a unique index that is synonymous to the term key. It determines the location in the tree. The indexed location is referred to as a (tree-)node. Accordingly, an index could be interpreted as the discrete location on a one-dimensional space-filling curve in 3D. The dilemma is that such a curve does not preserve spatial proximity in all directions [Sam90, GG98].

A most useful capability of tree representations is that they allow for organizing a sequential stream of unordered data dynamically. This accommodates the nature of many 3D-sensors such as a laser range-scanner and a laser-stripe profiler, where data is continuously incoming.

When working with 3D-data it is not realistic to presume either a complete data set free of empty regions or queries, which always point to an existing tree node. As a consequence, queries are often expanded to be of a fuzzy nature, i.e. a successful result is the return of the node with the desired information or the closest alternative in acceptable distance. This kind of query is referred as closest point access. The major problem is a query demanding information near the spatial border of a node, where the node itself is empty. This causes time-consuming ascensions and descensions in the hierarchy of the tree.

Closest point access is very popular in the field of registration tasks. Here, the problem is to find a transformation which provides the best alignments of two or more surfaces. The solution is mostly processed in several steps, where an error function is iteratively minimized. Criteria that measure the quality of the current alignment are typically based on point distances between the surfaces. Accordingly, such algorithms are referred as iterative closest point (ICP) methods. Examples can be found in [BM92, RL01].

4.1.1 Related Space Representation Approaches

A member of this line of research is the approach of Sagawa et al. [SMI03]. They introduced the technique of the Bounds-Overlap-Threshold for ICP-search. The computational cost for registration tasks is reduced by limiting the search to a region within a predefined threshold. Thus some branches which would cause ascending the tree will be ignored if other non-empty branches are available. Their method accepts a decrease in precision in return of faster computation. Branches which would cause further processing but could possibly provide a better match will be pruned if the current candidate is within an accepted distance. The approach is based on a k-d tree. Hence online tree-balancing seems impractical. As a consequence it is not suited for applications with continuous or periodic data updates.

Bodenmüller and Hirzinger use a tree representation capable of online surface-mesh generation [BH04]. Therein, the goal is to reconstruct the surface of a sensed object while data acquisition is going on. In this scenario some surface areas are scanned multiple times from different points of view. This circumstance and the fact that sensing errors lead to a diffuse object hull makes it necessary to locally modify the surface. Therefore, the representation contains linked leaf-nodes facilitating access to the direct
neighborhood. Thus, the neighborhood has a fixed size corresponding to the desired mesh resolution. An adaption to a neighborhood of variable size increases memory and processing costs for building the links. Thus, the representation is not feasible for an arbitrary region access.

4.1.2 Requirements in the Context of the Object Model

All discussed approaches do not fulfill the demands of scene interpretation algorithms of objects significantly varying in their size. Finding an adequate space representation for the previously introduced object description, demands a short recapitulation of the goals:

1. First, the object model is based on a distribution of surflet-pair relations. This implies that a single surflet-pair is not sufficient for recognition. Since the distribution is derived from an object of a certain volume, the space access should be capable to pick all surface points of such an area.

2. The second issue is that the object sizes vary. Accordingly, a space representation must handle this in the sense of providing access to neighborhoods of arbitrary size as well.

None of the approaches presented above provide both requirements, since they are designed for point search or for fixed neighborhoods, respectively. The solution presented in this chapter is a coding of space which can be implemented on different tree realizations. The enhancement to current approaches is the novel combinations of known strategies (code and trees), which allows the formulation of a new region search of variable size.

4.2 Indexing Restricted 3D-Space

For first contemplations the referred space is assumed to be restricted to a cube with an edge size $e_0$. The coordinate frame is placed in one corner and coincides with the edges in the manner that valid positions are restricted to the first quadrant.

To describe the term “subdivision tree” more precisely, the following definitions are used:

**DEFINITION 4.2:** A *spatial subdivision tree* is a hierarchical partitioning of space, where the first layer is defined to be of depth $d = 0$. It consists of exactly one cell $C_0$ that covers the hole volume. In succeeding layers each predecessor cell splits in eight sub-cells $C$ of equal size, one for each octant. The number of layers in a subdivision tree is limited to a maximum depth $d_{\text{max}}$.

**DEFINITION 4.3:** Cells of the lowest level, i.e. a depth $d_{\text{max}}$, are referred to as *atomic cells*.

The maximum storage capacity of a subdivision tree is synonymous to its resolution. It is the maximum number of entries a tree structure can represent. It is determined by the number

$$n_{d_{\text{max}}} = 8^{d_{\text{max}}}$$  \hspace{1cm} (4.1)
of atomic cells. Respectively,
\[ e_{d_{\text{max}}} = \frac{e_0}{2^{d_{\text{max}}}} \]  
(4.2)
is the edge size of an atomic cell.

To guarantee the access to the cell entries, a method for identification is required. Goodchild and Grandfield [GG83] and Samet [Sam90] presented and discussed several alternatives. The morton code, also known as n-order or z-order, is the preferred notation in this thesis. It is originally designed for two-dimensional images. The morton code can be easily adapted to 3D. Then the code uses eight digits to refer the quadrants. Thus, an atomic cell in the \( d_{\text{max}} \)-th layer can be represented by an octal number of \( d_{\text{max}} \) digits. This notation allows very fast binary computing. Figure 4.1 sketches the two- and three-dimensional version of the morton code.

![Diagram](image)

Figure 4.1: The plots show the enumeration of the quadrants for (a) the two-dimensional and the octants for (b) the three-dimensional morton code. The hidden octant in the cube bears the label ”4”.

### 4.2.1 Key Generation

The reference of a spatial position with respect to a subdivision tree and the morton code is further named its key or index \( I \). The function
\[ m : p \mapsto I \]  
(4.3)
maps a surlet \( p \) to the index \( I \), while
\[ c : I \mapsto C \]  
(4.4)
maps the index \( I \) to the cell \( C \). The length \( l \) of an index is one octal digit per depth. As mentioned above, this value defines one of the eight sub-cells. Accordingly, the prefix \( i(I, d) \) determines the \( d \) leading digits of the index \( I \). It is an index as well and refers to a higher located parent-cell \( C_p = c(i(I, d)) \), which includes the atomic cell \( C = c(I) \).

The computer equivalent to a positive natural number is an unsigned integer. Since a computer is a machine that can only display a limited number of states, that integers are bound by a maximum value, which is realized as a certain number of bytes. Therefore, using one single integer as key would restrict the referable tree depth even more.
than desired. As a consequence, a key has to consist of several unsigned integers. On the other hand, the key is a construction of several octal numbers each representing a layer of the spatial sub-division tree. In the context of a region search algorithm, the efficient access to an arbitrary octal number is essential, since it is a frequently performed action. Accordingly, the realization of a key has to be contemplated. Generally, two possibilities exist:

**Code 1:**

First, the desired depth can be used to calculate the optimal number of required bytes. Integers are typically implemented as a 16-bit or 32-bit value. On the other hand a key is a $d_{\text{max}} \times 3$-bit value. Accordingly, if one integer is not sufficient for housing a key, one octal number will cause an overflow to the next integer. Additionally, the octal digits in the next integer are subject of a 1-bit or 2-bit shift, respectively. Thus, an optimal memory consumption comes along with a complex layer identification. Hence, the benefit in the context of a region search algorithm is uncertain due to additional processing. Figure 4.2 shows the fragmentation of a 32-bit integer value and the bit shifting occurring in octal number 11 and 22.

![Figure 4.2: Space efficient fragmentation of an unsigned 32-bit integer.](image)

**Code 2:**

The second possibility avoids overflow and bit shifts at the expense of two wasted bits per integer in worst case. Then, the positions of the octal digits do not change and allow for fast computation. Figure 4.3 shows the fragmentation of an integer value.

Comparing the advantages to the disadvantages, code 2 was favored form implementation because of simplicity and speed.

![Figure 4.3: Fragmentation of an unsigned 32-bit integer.](image)
Using a maximum tree depth of \( d_{\text{max}} \), the number of required integers is

\[
\hat{n}_{\text{INT}} = \begin{cases} 
\tilde{n}_{\text{INT}} & \text{if } \tilde{n}_{\text{INT}} - \text{truncate}(\tilde{n}_{\text{INT}}) = 0 \text{ is fulfilled} \\
\tilde{n}_{\text{INT}} + 1 & \text{else}
\end{cases}
\tag{4.5}
\]

where the truncate function cuts off decimal places. The real number \( \tilde{n}_{\text{INT}} \) is defined as follows:

\[
\tilde{n}_{\text{INT}} = \frac{d_{\text{max}}}{n_{\text{BPI}} - (n_{\text{BPI}} \mod 3)}
\tag{4.6}
\]

Here, \( n_{\text{BPI}} \) describes the number of bits per integer and \( \text{mod} \) is the modulo function.

In cases of multiple integers one of the free bits can be used for an implicit coding of the key length. This is achieved by setting the leading bit to 1 if the succeeding integer is part of the key or 0 elsewhere. Figure 4.4 sketches the fragmentation of a multiple integer key.

Figure 4.4: Fragmentation of a multiple integer key.

4.3 Tree Implementation

When working with 3D-space, where the references are based on a hierarchical concept, a similar structure is useful for the implementation. As mentioned above, the most common approach is a tree. Nevertheless, the term \textit{tree} is still unspecific since it is more a family of different approaches than a concrete solution. In this thesis, two different types of well-known (pointered) trees are implemented. One is the \textit{octree} and the other one is the \textit{balanced binary tree}.

In both cases, the tree nodes correspond to cells in the subdivision space. As already stated, an atomic cell is the minimum volume which can be referred. In computer graphics a \textit{voxel} is the equivalent to an atomic cell of the spatial sub-division tree. A voxel is a binary spatial element, i.e. it exists in two states. If an entry corresponds to the spatial position it will be set. The exact position in the space of a voxel is lost. Demands of the spatial position of a voxel returns the center. As a result, every voxel-based implementation accepts surface distortions due to inaccuracies. In contrast to a voxel, a cell is much more precise because of the \textit{bucket concept} [Sam90]. Here, an atomic cell contains the mean surfeit \( \bar{p} \) with respect to the included surface points and according surface normals. Accordingly, a voxel can be seen as a discrete atomic cell. Due to precision aspects, this thesis is based on the bucket concept.

4.3.1 Data Preparation

Additionally, the cell concept can solve two typical problems which arise with real data and object classification. Both phenomena result from the structure and reflectance of
surfaces, from multiple measured regions, and from the relation of a sampled surface to the sensor.

Firstly, a sensing device can never guarantee to acquire data with the same density from every part of a surface. This affects the feature distribution since it is calculated from a random set of surflts. The reason is that dense surfaces contribute more than sparse ones and therefore, dominate the distribution histograms. The cells concept with its method of space partitioning can reduce this effect. Here, every atomic cell provides only one value, regardless of the number of underlying surflts. Hence, the cells build an upper bound to the maximum surface density and prevent over-representation of local areas.

Secondly, the measured regions differ in their accuracy, where *accuracy* defines the precision of a measurement. Keeping in mind that the used surface feature depends in three out of four parameters on the bivalent relation of surface normals. The recognition benefits from smooth surfaces. The atomic cells achieve this, since they act like integrated low-pass filters. Thus, noise is suppressed as long as the sensor precision is beneath the tree resolution.

The conclusion is that the cell concept includes two pre-processing steps. Both positively support object classification without the need of additional processing.

### 4.3.2 The Octree

Some tree structures allow for a geometrical interpretation. Representatives of this family are the *k-d-tree*, the *quadtree* and the *octree*.

The group of *k-d*-trees relies on a binary structure. The idea is to split a cell every time it contains more than one entry. The concepts how to split a cell vary in literature. Nevertheless, it is common to all approaches that the final structure strongly depends on the order of the input data. As a consequence *k-d*-trees won’t normally be balanced if balancing is not forced in a post-processing step. This circumstance disqualifies *k-d*-trees for applications with an online data update, since data access time would be unpredictable.

The octree uses the same structure as the hierarchical space division introduced above. Thus, the maximum depth is

\[
d_{\text{max, oct}} = d_{\text{max}}.
\]

Higher level cells are symbolized by nodes, while atomic cells are referred to as *leaf-nodes*. Only leaf-nodes contain information in terms of a 3D-point. Other nodes are exclusively used for traversing the tree. Therefore, an index can be interpreted as a path from the top to the bottom of the octree.

The quadtree is the two-dimensional version of the three-dimensional octree. Hence, the quadtree is further used to simplify the given examples.

### 4.3.3 The Balanced Binary Tree

Every tree structure starts with one root node pointing to several child nodes. The maximum number of child nodes depends on the type of the tree. The mean access complexity correlates with the depth of a tree. The lower the depth, the less operations are necessary to reach the desired node. If all prior levels of the bottom tree layer are completely filled, then a tree will perform best. Trees which provide this behavior are called *balanced*. 


In regard to traversing a minimum number of layers before information access, a direct link would be the optimum. Unfortunately, this case cannot be realized in dynamic environments. Moreover, the decision which branch to select for tree descending gets even more complex with the amount of possible child nodes. On the other hand, trees with only two children per node are sufficient for many tasks. This family is called binary trees.

Adel’son-Vel’skii and Landis [AVL62] first introduced a balanced binary tree known as AVL-tree. The AVL-tree allows a perfect balance which is achieved by checking the balance after every write or delete operation. An unbalanced state occurs when a parent node owns two subtrees which differ more than one level in their depths. Then a rearrangement of the subtrees is necessary to return to the balanced state.

In this kind of tree, only atomic cells need to be realized as tree-nodes, thus, it minimizes memory consumption. The index of a cell still encodes the position of the node but in contrast to the octree, the position depends on the input order of data and does not exactly coincide with the spatial order. Of course, the access time differs with the nodes position in the tree, but it is beneath an upper bound of $O(\log(n))$.

The binary tree has the same capacity $n_{d_{\text{max}}}$ as the subdivision tree, if the equation

$$n_{d_{\text{max}}} = \sum_{i=0}^{d_{\text{max}}_{\text{BTT}}} 2^i = 8^{d_{\text{max}}} \Leftrightarrow$$

$$2^{d_{\text{max}}_{\text{BTT}} + 1} - 1 = 2^{3d_{\text{max}}}$$

is fulfilled. This can be approximated by

$$2^{d_{\text{max}}_{\text{BTT}} + 1} \approx 2^{3d_{\text{max}}} \Leftrightarrow$$

$$d_{\text{max}}_{\text{BTT}} \approx 3d_{\text{max}} - 1 \Leftrightarrow$$

$$d_{\text{max}}_{\text{BTT}} \approx 3d_{\text{max}}.$$  \hspace{1cm} (4.12)

The conclusion is that in the worst case the balanced binary tree needs three times more layers to realize the same storage capacity of an octree. If one deals with sparse point-clouds and a high resolution of space, then the benefit of the octree is low. On the other hand, the benefit grows with the percentage of occupied positions in the spatial sub-division tree.

4.4 The Region Search Algorithm

Based on the data representation of an octree or of a balanced binary tree, the following section introduces a novel method for efficient access of local subsets. This operation is called the region search algorithm. It will be shown that the algorithm can be implemented on both tree structures. It consists of three subsequent processing steps, where the goal is to collect the points $p$, which are within a radius $r$ with respect to a region of interest (ROI)

$$\mathcal{F}(r, q) = \{p \mid \|p - q\| \leq r\}$$  \hspace{1cm} (4.13)

around a center $q$. The basic idea is to reduce the search to a subtree which is large enough to contain the whole ROI and to collect the entries from the tree.
4.4. THE REGION SEARCH ALGORITHM

Step 1: Finding the Center of Search

Every search starts with a ROI, which is defined by the center point $q$ and the radius $r$. First, the algorithm needs to determine the corresponding cell in the subdivision space. This cell $C_F$ has to be large enough to include the hole ROI, i.e. a cell of depth $d_F$ is chosen if the cell edge size

$$e_{d_F} = \frac{e_0}{2^d_F}$$

fulfills the condition

$$2r \leq e_{d_F}.$$  

Accordingly, the cell index

$$I_F = \hat{i}(m(q), d_F)$$

can be calculated.

By chance, the cell includes the complete ROI. Otherwise, if the center $q$ is closer to the cell border than the size of the radius $r$, parts of the ROI penetrate neighboring cells. The sections of the ROI which are located in neighboring cells are further referred to be outliers. In 3D-space, each cell has a maximum number of 26 neighbors

$$p_{[1...6]} = q + \frac{u_{[1...6]} - q}{\|u_{[1...6]} - q\|} r,$$

$$p_{[7...18]} = q + \frac{v_{[1...12]} - q}{\|v_{[1...12]} - q\|} r,$$

$$p_{[19...26]} = q + \frac{w_{[1...8]} - q}{\|w_{[1...8]} - q\|} r,$$

where $u_{[1...6]}$ are the projections of $p$ onto the six cell sides, $v_{[1...12]}$ are the projections of $p$ onto the 12 cell edges, and $w_{[1...8]}$ are the eight cell corners. Condition (4.15) reduces the number of outliers to 7 neighbors. This leads to 8 starting points at most in the case that the ROI is located in a corner of the cell. The following steps are individually applied for each starting point.

Step 2: Prefix Descent

All surflets $p_s, p_t \in F$ included in the space of the sub-cell $C_F$ have the common prefix

$$I_F = \hat{i}(m(p_s), d_F), \text{ with } \forall \{p_s, p_t | c(m(p_s)) = c(m(p_t))\}.$$  

Therein, the algorithm descends the tree until a node with consistent prefix entry is found. If such a node $n$ exists the search will continue from this node, otherwise it terminates.

Step 3: Collecting Surflets

The collection of surflets depends on the tree organization. Hence, both trees have to be discussed separately.
CHAPTER 4. SPACE REPRESENTATION

Collecting Surflets in the Octree:

Since the octree possesses a spatial organization, the search algorithm is very straightforward. All leaf-nodes in the subtree of node \( n \) are checked. If the distance of the stored mean point \( \tilde{p}_n \) to the center \( q \) of the ROI is smaller than the radius \( r \), it is accepted and included in the result. The worst case complexity of region segmentation is

\[
O(8^{d_{\text{max}} - d_F} (d_{\text{max}} - d_F)).
\]  

This situation will occur, when the subtrees of all starting points contain the maximum number of entries and if these are of the same size. The depth of every subtree is \( d_{\text{max}} - d_F \). Since eight is the maximum number of starting points, all subtrees contain \( 8^{d_{\text{max}} - d_F} \) leaf-nodes together.

The expression shows that the expected complexity strongly depends as well on the maximum depth of the octree as on the size of the ROI which determines \( d_F \). In the case of a large ROI the search starts near the root which leads to larger processing times than for smaller ROIs.

Collecting Surflets in the Balanced Binary Tree:

The position of a node in a balanced binary tree is less correlated with the spatial position than it is in the octree. As previously mentioned, this has two major reasons:

1. The final state of tree occupation strongly depends on the order of the input values during tree construction.
2. Every node of a balanced binary tree has two branches instead of eight, which would be necessary to represent all octants.

Accordingly, additional operations are required in order to navigate the tree and to separate the most promising subtree.

The navigation through the tree requires comparing pairs of prefixes. First, it is the prefix

\[
I_1 = m(\tilde{p}_n, d_F)
\]  

(4.22)

of the surflet \( p_n \) stored in the node \( n \). Second, it is the prefix

\[
I_2 = m(\tilde{p}_{n'}), d_F)
\]  

(4.23)

received from a child node \( n' \) of \( n \). The result defines the explore-operator

\[
\text{explore}(I_1, I_2) \begin{cases} 
\text{explore the right branch of } n', & \text{if } I_1 > I_2 \\
\text{explore both branches of } n', & \text{if } I_1 = I_2 \\
\text{explore the left branch of } n', & \text{if } I_1 < I_2 
\end{cases}
\]

(4.24)

that allows a decision which branch needs further examination.

To avoid multiple explorations of a subtree, the prefix \( I_1 \) of \( n \) is kept in an explored-list after complete examination of the correlated subtree. The first element of the list is the prefix of the cell that contains the center of the ROI, followed by a maximum of six outlier prefixes. It is not necessary to include the seventh outlier, since the last starting point won’t contain itself as a subtree.

In case of a balanced binary tree Step 2 has to be modified as follows: At the beginning of the second step, the index of the current node is checked. The search on
the subtree will terminate if the index matches any list element, since this indicates a re-exploration of an already processed volume. Then, the distance of the mean point of a node to the center $q$ of the ROI is checked. If it passes, the point will be collected.

Contemplating the computational costs of the balanced binary tree, it follows that in the worst case, all starting points are located in the fourth level ($d = 3$) of the tree, while the searched points are based in the leaf cells. Then the complete tree has to be examined.

To get the eight starting points a constant number of $8 \times 3 = 24$ operations is required. Every subtree is of depth $d_{max,BBT} = 3$. The maximum number of leaf-nodes is $2^{d_{max,BBT} - 3} - 1$. Thus the total number of nodes can be approximated by

$$8(2^{d_{max,BBT} - 3} - 1) \approx 2^{d_{max,BBT}}.$$  

(4.25)

Using the maximum tree depth $d_{max,BBT} \approx 3d_{max}$ as expressed in Eq. (4.12), allows to approximate the limiting complexity

$$O(24 + 2^{3d_{max}} (3d_{max} - 3)) \approx O(8^{d_{max}} (3d_{max} - 3)).$$

(4.26)

The point clouds of object surfaces are a sparse occupation of 3D-space. The representation is restricted to the surface only. The volume is not covered. At the same time assuming the worst case means that the points of the ROI are located in the starting points and in the leaf-nodes. It follows that nodes between these two levels have to be non-ROI points. This is a rather unlikely constellation. Thus, in practice the mean complexity is much smaller. Nevertheless, it is questionable if the mean complexity can be formulated as well, since it depends on the input order of data and the resulting balancing of the tree.

4.4.1 Performance of the Algorithm

The evaluation of the search algorithm [Chap. 4.6, p. 70] leads to two major conclusions:

The first aspect concerns the choice of the underlying tree structure. Here the octree performs better since the mean number of operations per access is lower than for the balanced binary tree. On the other hand, the octree needs additional memory for tree nodes which are used for navigation only. Thus, the question for the better tree structure has to be answered with respect to the needs of an application. Whenever memory demands are not the bottle-neck it is advisable to choose the octree and to gain more processing speed. Otherwise, the balanced binary tree would perform better. In practice the additional demands for memory are mostly less relevant compared to the processing speed. In the experiments section it will be shown that with the octree the application has a mean processing advantage of 25% percent against the balanced binary tree.

The second aspect is common to both tree representations. It addresses the behavior of the search algorithm with regard to processing time. There the results show that a region access depends on the size of a ROI in relation to the size of the referred cubic space. Best performance is achieved every time the ROI is small. A diameter equal to the edge length of the cube is the upper bound for a convenient usage. Keeping in mind that an object is typically much smaller than a measured scene, this threshold is unlikely to negatively affect object recognition.
4.4.2 Examples for both Tree Structures

Finally, an example should recapitulate the function of the region search algorithm. Therefore the dimensionality is reduced to 2D. The essential problems remain, while the complexity is reduced.

Figure 4.5 shows a hierarchically divided planar space. Twelve points \( p_1, \ldots, p_{12} \) build the tree input. The set
\[
\Omega = \{230, 132, 121, 303, 021, 103, 300, 013, 102, 003, 123, 120\}
\]
contains the corresponding indices.

In the following a tuple notation

\[
[\text{continuous point number}] : [\text{index into spatial subdivision tree}]
\]

is used, which symbolizes a 2D-point and the related tree index. The term 1:230 for example, refers to the point \( p_1 \) with the index \( I_{p_1} = m(p_1) = 230 \).

The circle symbolizes the region of interest \( \mathcal{F} \). All points inside the region build the desired search result.

At the beginning of the algorithm the size and position of the ROI determines the index of the starting cell \( C_\mathcal{F} \). For the given example the index and prefix for the further search is
\[
I_\mathcal{F} = \hat{i}(m(q), d_\mathcal{F}) = 2) = 12.
\]

Theoretically, the cell \( C_\mathcal{F} \) is able to include the ROI completely, but the shift causes an overlap to neighboring cells. Therefore, the inflicted neighbors have to be examined, so that finally two of the eight possible neighborhood positions point to cells with different prefixes. As a consequence the second and third step of the algorithm are applied on three cells.

Since these steps depend on the type of tree, it is necessary to separately discuss the impact on the data representation.

![Diagram](image-url)
4.4. THE REGION SEARCH ALGORITHM

The Octree

Figure 4.6 shows the quadtree which is the 2D-version of an octree after initialization with the input. In the octree all levels except for the bottom level are used to model the hierarchy. The length of the index directly corresponds to the tree level. Accordingly, any access to a single datum requires $d_{\text{max}}$ operations which is equal to the length of the index.

The algorithm descends to the first starting cell $C_F$ in the penultimate level. This node is the root of a subtree containing the points $p_3$, $p_{11}$ and $p_{12}$. All points fulfill the distance condition, thus, they are included in the result.

Next, the outlier cell with the index $I = 10$ is examined. In this subtree only two points are located. The point $p_6$ passes the necessary condition and is added to the result, while $p_9$ is too far from $q$ and, therefore, is ignored.

![Figure 4.6: The initialized quadtree replaces the octree in the two-dimensional case. Nodes corresponding to the starting cells are marked with a circle.](image1)

The Balanced Binary Tree

The balanced binary tree does not provide a one-to-one mapping of a spatial position of a point to the corresponding location in the tree. Figure 4.7 shows the structure of an AVL-tree after establishing the input. Another input order would cause another structure.

As for the octree the algorithm searches the starting cell in the balanced binary tree. Since the balanced tree represents only atomic cells, the prefix $I_F$ has to be compared to the stored full-length indices. The search leads to point $p_3$. The complete subtree fulfills the distance condition and is accepted.

![Figure 4.7: The graph shows the balanced binary tree of the example. Nodes corresponding to the starting cells are circled.](image2)
Then, the outlier search starts with the prefix $I = 10$. The first match with the prefix occurs straight in the root node. Additionally, the distance to $p_6$ is in the tolerated range, so that the point is added to the result. Next, the algorithm has to decide where to continue the exploration. Calling the \textit{explore}-operator on the left branch prunes the left side of the child and provides the points $p_5$ and $p_9$. Both points are too far. As a result the left branch of the root node cannot contain more results. Applying the \textit{explore}-operator on the right child prunes the right branch and leads to $p_2$ and an already examined subtree. The point $p_2$ fails the distance check, thereupon the search ends.

### 4.5 Indexing Unrestricted 3D-Space

The model of the space representation so far is restricted to a single cube. Unfortunately, the real world often demands for unrestricted space. Thus, the representation has to be modified.

One attempt is to degrade the cube to an octant of a larger cube every time a new datum outside the current space occurs. This method yields two drawbacks:

1. The first drawback is limited to the balanced binary tree. There, all nodes represent atomic cells and store the corresponding indices. For expanding the space it would be necessary to modify\footnote{The modification is an elongation of the index. Expanding the subdivision space would require a new leading digit in the key.} the index in each node, which is a very unperformant procedure.

2. The second problem concerns both tree versions. Keeping in mind, that the hardware limits the memory resources and that larger memory demands will require expensive swapping, it is questionable if a space expansion will bring any benefit.

Therefore, this thesis favors another approach which is presented in the following.

#### 4.5.1 Space Partitioning

The method introduced above could be described as a process of growing space which leads to the already discussed problems. Here, the idea is to organize space as a union of independent cubes. Therein, cubes are generated on demand, i.e. every time space has to be referred and is not already covered by an existing cube, a new one with a new tree is generated.

Of course, the cubes themselves have to be administrated, too. Therefore, a super-tree is generated, where cubes are the entries of super-nodes with super-keys.

This concept provides two advantages:

1. The space for every cube can be stored separately and then can be called on demand.

2. The structure supports parallelism. A search which would inflict multiple cubes could fork threads or processes for each cube.
4.5. INDEXING UNRESTRICTED 3D-SPACE

4.5.2 Super-Key Generation

Referring a cube cannot occur in the same manner as for the subdivision space. The reason is that the morton code implies a restricted space.

An alternative method is to use a lexicographic key $I^* = (i_x, i_y, i_z)^T$. This key consists of three components of descending priority. The order is established by first referring $i_x$, then $i_y$, and last $i_z$. In 3D-space this means to divide a space in slices, the slices in bars and finally the bars in the cubes. The super-key

$$ I^* = \begin{pmatrix} i_x \\ i_y \\ i_z \end{pmatrix} = \begin{pmatrix} p_x \text{ div } e_0 - t(p_x) \\ p_y \text{ div } e_0 - t(p_y) \\ p_z \text{ div } e_0 - t(p_z) \end{pmatrix} $$

(4.28)

provides this, with the function $t$ being

$$ t : x \mapsto \begin{cases} 1 & \text{if } x < 0 \text{ or } 0 \\ 0 & \text{else.} \end{cases} $$

(4.29)

The operator $\text{div}$ is defined as follows

$$ x \text{ div } y = \frac{x - (x \text{ mod } y)}{y}, $$

(4.30)

where $\text{mod}$ is the modulus function.

The super-key $I^*$ describes a discrete translation of a cube to a unique position in space and builds the index for a super-node in a super-tree. Here, it is advisable to use a balanced binary tree. This minimizes additional memory expenses and grants fast access. An octree is less useful since the structures of the key and that of the octree do not support each other.

4.5.3 Extension to the Region Search Algorithm

The extended space representation consists of two separated keys. One is the index specifying the location of a cubic part of space. If this index is present in the super-tree of cubes, then it will contain at least one surfeit. Therefore, the first index is a kind of coarse reference. The fine reference is given by a morton code with respect to a cubic space. The region search algorithm presented in the beginning of this chapter is based on the second index.

The extension of the search algorithm is analogous to the basic version. It takes several modifications:

1. Like for the inner of the cube, the first step is to check how many cubes include a part of the ROI. It is very likely that only one cube is affected, since an essential criterion for the choice of the cube size is to be much larger than the ROI. Fulfilling this makes the search algorithm perform best.

2. In the second step the old version of the region search algorithm is started for every affected cube. This step is suited for parallel processing.

3. In the last step, the results of the cubes have to be merged to a final result.
4.6 Experiments:
Performance of the Region Search Algorithm

This section evaluates the access performance of the novel region search algorithm and the underlying representations of 3D-space as introduced in Chap. 4, p. 55. The processing speed is discussed for both the octree and the balanced binary tree. Both are compared to the results of a naive approach. A comparison to recent iterative closest point (ICP) approaches is not appropriate since ICP approaches rather address a search of single closest points than a region search.

The naive approach is implemented as a sequential pointered list, where nodes store the information of an entry and a pointer to the successor. This structure is the simplest dynamic data structure. The main drawback of a list is that there is no index to directly access an entry. Accordingly, a search involves a sequential navigation through a list starting with the root node and moving on from successor to successor until the aim is reached. Since no spatial organization exists, a region query demands to calculate the Euclidian distance from the center of the ROI to every list node entry. Subsequently, every distance is compared to a specific threshold, which is equal the radius of the ROI. The access-complexity is \( N \), as \( N \) is the number of nodes.

4.6.1 Test Design

Obviously, the region search task strongly depends on the structure of data space. A mainly planar surface scene neither can be assumed to produce the same results as a volumetric scene, nor is it realistic to believe that two different shapes yield two similar results. Taking a scene for example with a table and two objects of same diameter placed upon should introduce the problem. One object is a plate, the other is a filigran sailing ship. If the region search is focused on the plate the solution will contain less points than for the focus on the ship, since the ship has the larger surface. Accordingly, the resulting processing time will differ, although the algorithm’s parameterization has not changed. The example shows the complexity of the participating factors and relations. For understanding the system it is necessary to find an experiment, capable to explain the principle behavior of the algorithm, while keeping scene specific influences low.

Since the region search problem is not limited to object recognition or scene interpretation, it is appropriate to work with a uniformly distributed point cloud. The data set presented to every combination of space representation and region search algorithm is a cube of unitary edge size containing the point cloud. Notice that this case fits the worst case for the region search problem.

Aim of the region search is to extract all points within the region of interest (ROI) \( \mathcal{F}(r, q) \) [Eq. (4.13)], where \( r \) is the radius of the desired sphere around the search center \( q \). The term saturation

\[
s = N/8^{d_{\text{max}}}
\]  

(4.31)

describes the state of the presented input space. It is the occupancy of the cube, where the divisor \( 8^{d_{\text{max}}} \) is the maximum number of possible indices. Remember that the amount of possible indices is equal to the maximum number of atomic cells in the subdivision space [Chap. 4.2, p. 57]. Accordingly, a saturation of \( s = 1 \) symbolizes the maximum number of 3D-points which can be encoded with an index of \( d_{\text{max}} \) digits. For the tests a subdivision space of depth \( d_{\text{max}} = 8 \) was used. It allows for a maximum number of about \( 16.8 \cdot 10^6 \) entries.
4.6. EXPERIMENTS: PERFORMANCE OF THE REGION SEARCH ALGORITHM

For the experimental evaluation of the processing costs, the radius $r \in [0.01, 0.5]$ of the ROI $F(r, q)$ was modified in 20 steps. The saturation $s \in [0.001, 0.1]$ of the representation was changed in 10 steps. Each combination of $r$ and $s$ was repeated 500 times. That leads to a total of $20 \times 10 \times 500 = 10^5$ tests. The processing performance at an Intel Xeon 1.7GHz with 1GB RAM is depicted in Fig. 4.8. Notice that the algorithm was implemented as a single process. Further improvement will be attained when starting the region search for every subtree in a separate thread.

Figure 4.8: The plots in the left column show absolute processing time, while the right column addresses the normalized processing time. Normalized times are divided by the measurement of the fastest algorithm regarding a pair of radius and saturation. The plots (a) and (b) are the results of the naive approach, (c) and (d) the results of the octree, while (e) and (f) are the results of the balanced binary tree.
4.6.2 Region Search Results

As mentioned above, the absolute processing time of the naive approach is a function of the saturation \( s \) [Fig. 4.8(a)], since it defines the length of the list which is proportional to the number of necessary operations per search. Furthermore, every performed operation is an expansive floating-point calculation, i.e. the determination of the Euclidian distance. In contrast, the tree-based region search approaches rely on a dual-step approach. First, it is a simple integer comparisons of two indices until a coarse restriction of the search space is reached. Not until the second step, expensive floating-node operations are applied to a much smaller point set.

On the other hand, for both tree representations the absolute processing expenses arise for the parameter \( r \) and \( s \). Like for the sequential list, more samples \( s \) result in more operations. Additionally, a larger ROI causes larger subtrees, too, and decreases the amount of cheap integer comparisons in the early phase. The graphs Fig. 4.8(b) and Fig. 4.8(c) depict the increasing number of nodes which have to be examined. In the worst case the size of the ROI is similar to the represented space. Then all nodes have to be processed. Consecutively, additional costs for navigation and calculation of the index occur. In such cases, the naive approach outperforms hierarchical solutions.

The results so far do not clearly demonstrate the potential advantage of the tree-based approaches over the naive approach. A direct comparison of the performances should change this. Therefore, the relative processing time

\[
\text{rel}(r, s) = 100\left(\frac{t(r, s)}{\min\{t_{\text{naive}}(r, s), t_{\text{octree}}(r, s), t_{\text{avl}}(r, s)\}} - 1\right)
\]

(4.32)

is defined. It is the shifted quotient of the time \( t(r, s) \) achieved for one method to the fastest processing time \( \min\{t_{\text{naive}}(r, s), t_{\text{octree}}(r, s), t_{\text{avl}}(r, s)\} \) of all methods in common.

The left column of Fig. 4.8 shows the absolute access time in seconds. Due to better visualization of the comparison results, a normalized notation of access time is defined. Here, each pair \( (r, s) \) depicts the processing time of an approach with respect to the processing time of the fastest representation. The result is represented in percent and shifted to the origin, i.e. a value of 0 characterizes an approach as being faster than the others. The results are presented in the right column of Fig. 4.8.

The graphs Fig. 4.8(b), (d) and (f) show that the performance variation is reduced to a dependency on the radius \( r \). However, since the saturation \( s \) yields equal expenses for all methods, it is no longer of interest.

Furthermore, a sharp line is noted at a radius of \( r = 0.25 \), which terminates the high performance area of the tree-based approaches. The region above \( r \geq 0.25 \) is of minor importance for the region search algorithm, as the algorithm is designed for the purpose of scene interpretation. Hence, the goal is to decompose some kind of complex scene into objects of interest and background. Here, the object size is typically much smaller than the complete scene. The plots depict that the mean object size is best to be below one half of the scene diameter.

Focusing on the tree-based approaches it has to be taken into account which application is desired. Due to lower tree-depth and less operations during navigation, the octree outperforms the balanced binary tree with around 10% to 20% faster access. On the other hand, the octree needs \( \sum_{i=0}^{d_{\text{max}}-1} 8^i \) additional nodes for the purpose of navigation. This provokes memory costs that are not necessary if using a balanced binary tree. It has to be considered whether processing time or memory costs matter the most.
In times of low cost memory chips and real-time applications, the decision will nearly almost tend towards the octree.
In literature the term scene interpretation is used whenever the task is to analyze the contents of an unknown environment with respect to some kind of a-priori knowledge. It is often a combination of two mechanisms. The first is segmentation describing the process of separating the relevant information from the context. The second mechanism is classification which is equivalent to a mapping of knowledge to observed and segmented data.

In the following, the conditions are described under which a scene interpretation should work. Further, the intended results are discussed. Then the principles of a novel scene interpretation algorithm are sketched. The last part describes the details and explains how the intrinsic lack of shape information is compensated for.

5.1 Working Environment and Intended Results

As mentioned, scene interpretation summarizes a family of strategies. To prevent confusion due to the different popular meanings, the term is specified in the context of this work. Here, a scene consists of a large set of oriented surface points. In repetition of Chap. 2.2, p. 20 those points are referred to as surflets. A set of surflets is synonymously called a (oriented) point cloud. The source of a scene, i.e. the total input point cloud, is a sampled arrangement of rigid objects on a rigid background. Neither the objects, nor the background are allowed to move. Hence, a scene can be described as a three-dimensional snapshot of a working environment.

According to the two parts of scene interpretation, the following terms are used:

**Definition 5.4:** Segmentation is the collection of separated points referring to one instance of an object. The result is a subset of the original point cloud.

**Definition 5.5:** Classification focuses on the recognition of a segmented point cloud. The result is a label with respect to a database of known objects and the corresponding object names.
A scene typically consists of a very large dataset. Since it is desired to support robots with a visual analysis of their environment, the clouds are reduced to a much smaller size in a first pre-processing step. The down-sized data allows for much faster processing. On the other hand, the reduction comes along with a loss of information, which is willingly tolerated for the sake of processing speed. Accordingly, a segmentation is not assumed to contain all object points. A final result resembles the object point cloud with a lower surface density.

Related works of other researchers often use depth maps for scene interpretation. Depth maps are also referred as 2.5-D images. An often proposed method for interpretation is to tessellate the whole image into regions of similar curvature. Edges are used to separate parts. The next step of these approaches tries to group segmented parts so that they are reasonable for an object. In this context, objects are often described as attributed graphs, where nodes represent the surface patches and links describe the connectivities. One representative of this line of research is the work of Fan et al. [FMN89]. As an example for this algorithm group, it shows the drawbacks of these solutions:

1. The objects are supposed to have a structure that allows for tessellation. Edges and significant changes of curvature are essential.
2. Connections invisible due to (self-) occlusion compromises recognition.
3. Since noise leads to inaccurate curvatures, it is necessary to preprocess data for improvement. This typically includes several “magic numbers” in the parameterization.

The environment used in this thesis is not limited to depth maps, but consists of real 3D-data. Thus, scenes are much more complex, so that a transfer to depth map approaches would dramatically increase processing time. Moreover, the approach should segment and classify objects simultaneously.

5.2 The Idea

In this thesis an approach with an object representation relying on the distribution of a four-dimensional surface feature is followed. Analogously, the recognition process embedded in the scene interpretation has to identify and separate such regions in a given scenario. As mentioned above, the object description does not provide direct information about the shape except of the maximum object diameter. Thus, it is very difficult to set expected object boundaries for the search algorithm.

The basic idea is to use a multi-step refinement for separation of a sub-point cloud dependent on its point distribution and to perform the classification consecutively. The method is divided into different phases. The first phase concentrates on scene interpretation with respect to a model. In other words, the search for an object runs on a sampled point cloud, where the points result from surflet-pairs that are of higher significance for the desired object than for every other object. Restriction to these surflet-pairs and to the corresponding features should lead to a model-specific ”view” of space, since much higher densities are supposed at the positions of the corresponding object instances.

The following phase addresses the analysis of compact areas. A new clustering method is introduced. The aim is to keep points inside a cluster set which supports the model distribution and banishes others.
5.3. CROSSTALK

The result of this separation is still not trustable enough. Therefore, the third phase validates every model-specific selection of points. In case of a correct selection, the corresponding feature distribution should show a large similarity when being compared to the stored database model.

Segmented point clouds passing all phases build subsets that are an interpretation of a scene with respect to the object database.

5.3 Crosstalk

One of the most challenging circumstances the algorithm has to face is correctly categorizing surflet-pairs on basis of the corresponding features. Obviously, surflet-pairs do not necessarily derive from one single object. Figure 5.1 sketches some examples of valid and invalid surflet-pairs. In the following invalid surflet pairs are defined.

**DEFINITION 5.6:** Crosstalk is the term for all surflet-pairs, where the origin is not the same instance of an object or where the origin is not an object of the database. Crosstalk occurs when

- surflets are drawn from two different objects,
- surflets are drawn from an object and background, or
- both surflets are drawn from background.

A feature resulting from crosstalk is invalid.

The contamination of the feature distribution histograms with crosstalk is a problem. Therefore, the ratio of crosstalk to real object surflet-pairs has to be kept as small as possible.

5.4 The Segmentation Algorithm

The structure of acquired data is often a point cloud with varying local densities, while objects are trained with randomly and uniformly drawn samples of a surface. The different conditions of surface density affect the probable location of a surflet and therefore, lead to distorted feature histograms. To overcome this problem an initial reduction of data is necessary.

A subdivision space, like it is discussed in [WH05a] and the previous chapter [Chap. 4, p. 55] is applied, which organizes space into homogeneously-sized cells. Each cell contains the mean surflet value in its space. This preliminary step reduces the noise and size of data at once. Moreover, the reduction to one reference value per cell guarantees an upper bound of surface density.

In the following an algorithm is proposed which is divided into seven steps. The algorithm works on the reduced point cloud obtained from the spatial subdivision tree. With every step, the set of initial surflets is refined by sorting out less confident surflet-pair combinations. The aim is to filter a drawn surflet set until it is free of crosstalk and ready to be classified, which is done in the last two steps.

Surflets are represented with the variable $p$. Remember that this includes a position and the local surface normal, which are both necessary to calculate a feature as introduced in Chap. 2.2, p. 20. Every time the focus is on the relative position of a surflet to some 3D-point, the surface normal is not of interest. Thus, spatial difference
operations address only the position part of the surflet. The notation $p_s$ determines one arbitrary surflet of a surflet-pair $s$.

**Step 1: Pre-selection**

To reduce crosstalk resulting from background two strategies are proposed which demand a slightly different first step:

**Strategy A: Exclusion**

The first possibility addresses a simple exclusion. It demands the algorithm to operate in an environment known in detail, which has to be constant for all measurements. One example would be an application, where a robot gets new parts in its well known working cell. The data could be kept clean of background in several ways. One possibility is to suppress the data acquisition from regions which are defined to be background. Another way is to erase the background after acquisition, for example by deleting the scene without objects from the complete scene.

The algorithm of the exclusion strategy, starts with drawing a set

$$S = \{s_1, \ldots, s_n\}$$  \hspace{1cm} (5.1)
of \( n \) surlet-pairs \( s_i \). With regard to the \( k \) models of the database

\[
\Omega = \{ M_1, \ldots, M_k \},
\]

the maximal diameter \( \delta_{M_j, \text{max}} \) of the largest model \( M_j \in \Omega \) limits the distance allowed in a surlet-pair.

Subsequently, the features \( f_s \) are calculated. At this stage it is not known whether both surlets are part of the same object or rather describe a false relation which is crosstalk of two different objects. Consequently, a second reduction step has to be applied which determines the sample set

\[
S^*_i = \{ p_s | \exists s \in S : p_s \in s \wedge H_M(h_M(f_s)) \geq \tilde{H}_M \wedge [H_M(h_M(f_s)) \geq H_M(h_M(f_s)), \forall j \neq i] \},
\]

for each model \( M_i \) individually. The threshold

\[
\tilde{H}_M = \frac{\lambda_H}{m} \sum_{\forall f_s} H_M(h_M(f_s)), \text{ where } m = \text{card}\{f_s|H_M(h_M(f_s)) \neq 0\},
\]

restricts \( S^*_i \) to the significant surlets \( p_s \) of the model \( M_i \). It can be adjusted by a positive scalar\(^1\) \( \lambda_H \). The higher the value \( \lambda_H \), the more characteristic it is regarding to an object of the database. The notation \( \text{card} \) is used for the cardinality of the set. Condition (5.4) limits the set to features, which respond most for the preferred model.

### Strategy B: Identify and Ignore

A more flexible solution is to identify and ignore the background. Then models are built from typical structures of the background using the same routines as for regular objects in the database. As a consequence, it is possible to identify potential background surlet-pairs and to ignore them. This strategy becomes necessary, whenever the background has undetermined degrees of freedom. An example might be a service robot with the task to pick objects from unknown shelves in a supermarket. In a scenario like this, it would be complicated to realize a shelf recognition and pose estimation with state of the art algorithms. Instead using the proposed algorithm needs only a planar patch to approximate a shelf. Missing information on the shelf's pose has no impact because of the intrinsic model invariance in this aspect.

Again the set

\[
S = \{ s_1, \ldots, s_n \}
\]

represents the database of \( k \) object models expanded with a set

\[
\Omega^* = \{ M_{k+1}, \ldots, M_l \}
\]

\(^1\)In the experiments the scalar was parameterized to \( \lambda_H = 1.3 \).
of \( l - k \) background models. After calculation of the features \( f_s \) the second reduction step follows. In addition to strategy A the set

\[
S^*_M = \{ p_s \mid \exists s \in S : p_s \in s \land [H_M(h_M(f_s)) \geq H_M(h_M(f_s)) \forall M_j \in \Omega^*] \land (5.9)
\]

\[
H_M(h_M(f_s)) \geq \tilde{H}_M \land (5.10)
\]

\[
[H_M(h_M(f_s)) \geq H_M(h_M(f_s)), \forall j \neq i] \}
\]

contains the further condition (5.10). It eliminates, all features that are most likely background. The corresponding surflets need no further processing and affect classification no longer.

**Step 2: Cluster Seeds Initialization**

It is assumed that surflets in \( S^*_M \) show an inhomogeneous distribution in 3D-space. The intention is that they build spatial accumulations for objects represented by the model \( M_i \). These contemplations lead to:

**DEFINITION 5.7:** A cluster is a spatial accumulation of oriented points in a model specific interpretation \( S^*_M \) of 3D-space. All points of a cluster originate from a common instance of the corresponding object \( M_i \).

To approximate a cluster two aspects are important:

1. It is most likely that the set \( S^*_M \) will contain some surflets from the desired object, if drawing a sufficiently large number \( m \) of random samples.

2. It can be expected that the neighborhood of surflets being part of the desired object, contains further valid surflets.

Both aspects build the basis for the following definition:

**DEFINITION 5.8:** A cluster seed \( C_{M_i} \) is a model specific local neighborhood of radius \( r_{M_i,\text{max}} \) around a center \( c \). The model-dependent set

\[
C^*_{M_i} = \{ C_{M_i,1}, \ldots, C_{M_i,m} \} \subseteq S^*_M
\]

defines cluster seeds, where the corresponding centers \( c \) are equal to the position part of surflets \( p_s \in S^*_M \). The sphere around a cluster seed contains the local subset

\[
S^*_{c,M_i} = \{ p_s \in S^*_M \mid r_{M_i,\text{max}} \geq \| p_s - c \| \} \subseteq S^*_M
\]

The term cluster seed is used to emphasize the preliminary state as compared to a cluster. According to the definition these \( m \) cluster seeds are initialized with random samples drawn from \( S^*_M \).

**Step 3: Cluster Seed Refinement**

It cannot be supposed that all cluster seeds are part of an object. Those who are, usually contain crosstalk. To filter out reliable information, two alternating steps are iteratively applied for refinement.
5.4. THE SEGMENTATION ALGORITHM

Regrouping: Assuming that all surlets \( p_s \) belong to the same object, a recombination

\[
T(S'_{e,M}, \frac{1}{2}\text{card}S''_{e,M})
\]

(5.14)

[Chap. 2.17, p. 23] of the surlets \( p_s \) should produce valid features \( f_s \). On the other hand, false surlets that previously built a valid feature by chance are now likely to cause crosstalk. The features of the recombined surlets are calculated and compared to the histogram \( H_{M_i} \) of the model. Invalid features that point to a zero value \( (H_{M_i}(h_{M_i}(f_s)) = 0) \) are labeled as crosstalk.

Crosstalk Elimination: A cluster seed is valid if it contains no crosstalk. To reach this state all surlets causing crosstalk are erased. One single detection of crosstalk starts a subsequent call of regrouping.

Both steps are applied until no crosstalk occurs or no surlet remains. Cluster seeds without surlets are deleted. The surlet sets of the remaining cluster seeds are referred to as

\[
S''_{e,M_i} \subset S'_{e,M_i}
\]

(5.15)

Step 4: Cluster Seed Merging

Due to the large number of initial cluster seeds in relation to the number of expected objects, many cluster seeds accumulate. They are subsets of a common cluster and belong to the same object. Cluster seeds of the same object share a large number of surlets. These cluster seeds should be merged to approximate the common cluster.

Unfortunately, the surlet sets of the different cluster seeds are not identical. A good estimation of a cluster is found, whenever a significantly large number of surlets is common for all cluster seeds.

To achieve good cluster it is advisable to start with the best cluster seeds. Therefore, a criterion is needed to rate the distribution quality of each cluster seed and its surlet set \( S''_{e,M_i} \). In [SC00] and [WHH03] it is shown that the logarithmic likelihood outperforms other criteria like Kullback-Leibler divergence and \( \chi^2 \)-test in processing time and reliability. Accordingly, the criterion

\[
L(M_i|S''_{e,M_i}) = \frac{1}{\text{card}S''_{e,M_i}} \sum_{p_s \in S''_{e,M_i}} \ln H_{M_i}(h_{M_i}(f_s))
\]

(5.16)

is applied.

Higher votes of the criterion \( L(M_i|S''_{e,M_i}) \) refer to better model fits and a larger number of valid surlets. This circumstance is used to sort the cluster seeds by their matching quality with respect to the database model.

Getting \( k \) valid cluster seeds, the set \( \{C_{M_i,1}, \ldots, C_{M_i,k}\} \) is obtained, where every cluster seed \( C_{M_i,j} \) contains an individual surlet set \( S''_{e,M_i,j} \). Subsequently, the best ranked cluster seed is compared to all lower-ranked seeds. Each time the number of coincident surlets exceeds a certain threshold\(^2\) \( \lambda_0 \), the common surlets are stored in a cluster \( \hat{c} \). Finally, the surlet set

\[
S'''_{e,M_i} = \{p_s | p_s \in S''_{e,M_i,1} \cup S''_{e,M_i,j} \wedge \\
\lambda_0 > \text{card}(S''_{e,M_i,1} \cup S''_{e,M_i,j}) \wedge \\
\forall j \in \{2, \ldots, k\}\}
\]

(5.17)

\(^2\)30% of co-occurrence shows good performance.
builds a cluster.

The merging of cluster seeds into one cluster is a conservative process. The re-
striction to surflets contained in both merged cluster seeds reduces the likelihood of
undetected crosstalk. Nevertheless, such crosstalk still occurs when a cluster seed is
placed in between two object instances of the same model. Then, one half of the set
may point to one object instance and the other half to the second object instance.

Merged cluster seeds are deleted from the sorted list. If a cluster seed remains,
the algorithm will start again at the top of the list. Otherwise, the merging process is
finished.

**Step 5: Distribution Checkup**

The surflet set of a cluster is the intersection of the surflet sets of its cluster seeds.
Although every cluster seed held a distribution of surflets being very likely to the esti-
mated object, it is not certain that the new set does the same. Thus, after merging the
current set of surflets $S_{c\cdot M}$ has to be checked again.

As previously done for the cluster seeds in **Step 3**, the surflets of a cluster are
regrouped and the features as well as their distribution histogram are calculated. These
clusters should contain nearly no crosstalk. Moreover, they are supposed to be similar
to the hypothesized database distribution. Thus, the measured cluster distribution is
compared to the database. Additionally, the highest distribution response is demanded
when applying criterion (3.16) on the expected model. If the cluster distribution fails
to meet the condition, the cluster will be deleted.

**Step 6: Cluster Similarity**

Only a few clusters $\hat{C} \in \hat{C}_M = \{\hat{C}_1, \ldots, \hat{C}_m\}$ reach this level where crosstalk seems
to be eliminated and the surflet distribution meets the expectations. However, misclas-
sifications are rare but still possible since similar parts of different objects may show
similar distributions. An example for this phenomena is the similarity between the
bottom of a cup and the bottom of a bottle, when both radii are nearly equal.

Fortunately, the size of a cluster contains hints to the correctness of classification
that could be described by two conditions:

1. The cluster size has to exceed a minimum.
2. All clusters of a model type should be of similar size.

The size is derived from the number of surflets in a cluster. It strongly depends on the
object, the density of the measurement, and the resolution of the subdivision space. To
prevent a collection of thresholds for size evaluation, the size is measured in terms of
the normalized number $p_C$ of contained surflets with respect to the surflet number $p_{\text{max}}$
of the largest cluster. The size must be in the interval $[\lambda_C, 1]$, where $\lambda_C$ is a threshold
for the minimum relative number of surflets.

As in the previous steps all clusters are deleted if they are not able to fulfill the
conditions.

**Step 7: Cluster Competition**

The last evaluation step prevents multiple classifications of an object. This situation
will occur, if an object is separated into multiple disjunct surflet sets. Therefore, the
5.5. EXPERIMENTS: SCENE INTERPRETATION IN PRACTICE

The scene interpretation algorithm as proposed in Chap. 5, p. 75 is built upon the conclusions of the previous two sections. The classifying function with respect to the database is the likelihood criterion due to best recognition rates and lowest processing costs. For the representation of three-dimensional space the octree is the favored approach. Although, it inlicts additional memory costs due to nodes that are necessary for navigation, it is still acceptable since the used scenes completely t into the working memory. Hence the scene interpretation algorithm benefits from the octree’s best access performance.

The experiments section for scene interpretation is organized into two major parts. The first part discusses results achieved with synthetic data, while the second part takes a closer look at results derived from real data.

5.5.1 Scenario

For the sake of comparability all scene interpretation experiments are based on the same scenario. It is a plain surface with some rigid objects placed upon. The whole scenario is given as a dense cloud of oriented points.

centroid $u_{C_1}$ of a cluster $C_1$ is calculated and compared to the centroids of all other remaining clusters of all model assumptions, e. g. the centroid $u_{C_2}$ of cluster $C_2$. A collision will occur, if

$$\| u_{C_1} - u_{C_2} \| \leq \max(\bar{r}_{M_i,C_1}, \bar{r}_{M_j,C_2}),$$

(5.18)

is true, where $\bar{r}_{M_i,C_1}$ and $\bar{r}_{M_j,C_2}$ are the mean distances of the surlets to the centroids $u_{C_1}$ and $u_{C_2}$ [Chap. 2.3.1, p. 25 and Chap. 2.3.2, p. 26]. Two cases have to be distinguished:

1. Both clusters are of the same model type ($M_i = M_j$). In this case both clusters are merged, i.e. the two disjunct sets of surlets are joined and build a larger new set.

2. The clusters are of different model types ($M_i \neq M_j$). Thus, the cluster $C_1$ of model $M_i$ remains, if it fulfills at least two of the three following conditions:

$$p_{C_1} > p_{C_2},$$

(5.19)

$$\frac{p_{C_1}}{n_{C_1}} > \frac{p_{C_2}}{n_{C_2}},$$

(5.20)

and

$$\frac{n_{C_1}}{n_{\text{max},C_1}} > \frac{n_{C_2}}{n_{\text{max},C_2}}.$$  

(5.21)

and vice versa. Condition (5.19) favors the cluster with the larger number of surlets with respect to its model-specific ranking, which was already calculated in Step 6. Condition (5.20) considers the number of surlets $p_{C_i}$ in relation to the number $n_{C_i}$ of merged cluster seeds per cluster. Finally, Cond. (5.21) addresses the number of merged cluster seeds $n_{C_i}$ in relation to the maximum number $n_{\text{max},C_i}$ of merged cluster seeds per model.

All clusters passing the last step compose the final result of the scene interpretation process and provide sets of segmented object surlets.

5.5 Experiments: Scene Interpretation in Practice
CHAPTER 5. SCENE INTERPRETATION

Goals

Two goals are of interest:

1. The approach has to separate objects, i.e. to segment a point cloud which contains points from only one instance of an object. Notice that it is not desired to select all points of an object. A subset of points is much faster to process and it is still sufficient for further applications like pose estimation.

2. Each separated and valid point cloud should be classified with respect to a database.

It will be demonstrated that the presented approach is capable to fulfill both requirements at once.

Measurement of Performance

To measure the performance of the approach we distinguish between the goals and the achieved success. According to the previous enumeration it is postulated:

**DEFINITION 5.9:** The classification rate symbolizes the measured percentage that a segmented cluster of points bears the correct object label. Therefore, at least one correct object point must be in the cluster.

**DEFINITION 5.10:** The classification accuracy describes the quality of a cluster. It is the term for measuring the percentage of correct object points in a correctly classified cluster.

5.5.2 Synthetic Data

To identify the characteristics of an approach it is advisable to keep unknown influences as small as possible. Thus, the first experiments concentrate on synthetic data, where all scene aspects can be controlled.

An adequate database is selected which allows to evaluate specific questions of scene interpretation. In principle, the shape representation supports a large number of objects, but a large database either provokes more misclassifications and inflicts additional memory management problems. Moreover it would be questionable if more meaningful results really demand for more objects in a database than one would typically be faced with in robotics scenarios. Finally, it was decided to use a synthetic database of seven entries, which is a sufficiently large number for most applications. The database contains models of a bunny, a bottle, a carafe, a cup, a dragon, a horse, and a wine glass. Figure 5.2 depicts all trained shapes. All objects are given as oriented point clouds of about 80,000 to 120,000 entries.

All objects are randomly placed on the plane in an upright position. Therefore, a random point cloud was initially generated for the plane. Then the translated object point clouds were added to the scene. This procedure allows to decide the origin for every point. Hence, it simplifies the determination of the classification accuracy. To avoid that objects intersect each other, a minimum distance

\[ d_{\text{min}}(O_i, O_j) = \min\{r_{O_i, M, \text{max}}, r_{O_j, M, \text{max}}\} \]  

(5.22)

from one object \(O_i\) to every other object \(O_j\) is defined, where \(r_{O_i, M, \text{max}}\) and \(r_{O_j, M, \text{max}}\) are the maximum radii of object subscribing spheres. Figure 5.3 shows one exemplary
setting. To allow for statistically reliable results, all experiments are repeated with 1000 arrangements of the scene.

Initially, the mean number of scene points starts with about 770,000 surface points, which the octree structure reduces to about 77,000 points. As described in Chap. 4, p. 55 the reduction step provides two advantages. First, an inhomogeneous acquisition of the scene and the resulting variance of local point densities is restricted to an upper bound. Second, the smoothing that occurs in the leafs of the octree reduces disturbances. Subsequently, the algorithm draws about 464,000 surflets from the scene yielding to 232,000 features. Once the features have been generated, the evaluation splits in the different concepts of background treatment.

**Ignoring the Background**

The first experiment aims to interpret the scene strictly with respect to the database of the seven previously trained objects. It is assumed that the presence of any background does equally affect the object’s feature distributions. Hence, the algorithm should receive weaker object responses than without background, but every response is still supposed to be characteristic.

Currently, the results do not support this assumption. The common classification

![Image of objects](image)

**Figure 5.2:** The first experiments concerning the scene interpretation approach rely on a database of seven synthetic objects, shown above. The object sizes are optimized for the sake of visualization and do not depict the real scale-relations.
rate reaches an unacceptable level of 49.6%, while the classification accuracy achieves moderate 80.2%. Interestingly, the classification rate does not drop for all objects. Some objects seem to be "immune" to the presence of background and are reliably classified (>90%), while others seem to be unidentifiable.

Analyzing this inhomogeneous behavior in classification leads to meaningful knowledge. All table points are background and produce surflets, too. Of course, the calculation of the corresponding features yields too much crosstalk, but not in the expected manner. Some object’s distributions contain surflet combinations which are significant for the object as well as for the background.

Contrary to intuition, the table most resembles the bottle. Both shapes are dominated by planar structures. In fact, the body of the bottle is a plain bend around the symmetry axis, which is aligned to the z-dimension. Therefore, all bottle surflet-pairs of the body with the same x- and y-coordinates have parallel surface normals characteristic for the table, too. Accordingly, it can be concluded that the bottle is most similar to the background and collects the most of crosstalk. Such a large number of false features exceeds the capabilities of the algorithm and yields only 3.3% of correct classifications.

The conclusion of this experiment is that knowledge about the structure of the background has to be applied.

**Suppressing the Table**

In many cases background is not only stationary but a priori known, too. This knowledge could be given in form of an initial background sensing before placing any objects or in form of a working space definition excluding the working cell. For both versions the effect is the same: sensed points are initially judged whether being background and
are added to the scene only if they are not. Thus, the scene consists of object points exclusively.

Here, the initial data without background, consisting of about 370,000 points, is reduced to about 42,000 points via octree, from which 100,000 features are generated. Final clusters remaining after all steps of the scene interpretation algorithm have an average size of 1,600 surflets each. The common classification rate raises to 97.2%. Remaining misclassifications stem from clusters wrongly placed in between two objects.

The quality of clusters perform at an accuracy of 99.7% correctly classified surface points [Tab. 5.1]. Figure 5.4(a) shows a final cluster point cloud received for the bunny.

Including the Table Model into the Database

Eliminating the background wouldn’t be that easy if only the structure is known but the exact location and orientation is missing. In such cases the situation for the background is the same as it is for the desired objects. Thus, it is consequent to use the scene interpretation algorithm for background elimination, too.

The model generation for the background is very similar to the one for an object. First, a characteristic surface patch is sampled. The only condition is that the maximum diameter has to be beneath the one of the largest database object. A larger diameter would bring no benefit, but increases the amount of useless\(^3\) point pairs drawn at the beginning of the scene interpretation. Then, the background model is included in the database as well.

Taking a scene with background, the amount of initial data points increases to the initially mentioned amount of 77,000 points in the reduced scene cloud. Here, the number of drawn samples must be larger than without background since probability for an crosstalk grouping is much larger and yields to a large loss of points.

The first step of the algorithm allocates the features to the most likely source. Here, a modification is added to the algorithm. It is to ignore all point-pairs that build features which are mostly supposed to be background. Accordingly, the number of point pairs decreases to 115,000 corresponding features that remain as possible candidates for the objects. Compared to the case without background present, the algorithm eliminates many more points. The mean size of a cluster is now only about 330 surflets [Fig. 5.4(b)].

Though the mean cluster size is significantly smaller than for the previous experiment, the algorithm still achieves a classification rate of 97.2% with respect to the clusters. The quality of the final clusters remain high with an accuracy of 96.1% correctly classified points.

One important observation is that the probability for a correct object classification and for the segmentation accuracy strongly depends on the shape. The horse for example is hardest to be recognized. This effect seems to derive from the disadvantageous point distribution. The mass of surflets build clusters in the torso, while the thin legs widen the object diameter. As a consequence, the possibility to draw a non-object point is much larger than for compact shapes like the bunny. Accordingly, the amount of crosstalk increases and the remaining cluster points decrease.

The current treatment of background is more generic than the two versions in the previous settings. That makes it suitable for more applications. The larger relevance

---

\(^3\)Remember that point pairs with a distance larger than the maximum diameter of the largest object cannot be assigned to any valid object [Chap. 5.4, p. 78].
demands for further performance measurements in this case. Therefore, the processing times on an Intel Xeon 2.8GHz processor with 2GB RAM was observed. The measurement includes the elapsed time from the initial drawing of point-pairs up to the complete evaluation of all clusters, where a final cluster consists of an object label, a point cloud, and the centroid. The algorithm performed one complete run in averaged 6.8 seconds. Tabular 5.1 summarizes the results of all three experiments.

Table 5.1: Classification results retrieved for scenes consisting of upstanding objects randomly positioned on a table.

<table>
<thead>
<tr>
<th>treatment of background</th>
<th>ignored</th>
<th>suppressed</th>
<th>modeled</th>
</tr>
</thead>
<tbody>
<tr>
<td>correct object classification</td>
<td>49.6 %</td>
<td>97.2 %</td>
<td>97.2 %</td>
</tr>
<tr>
<td>correctly segmented points / classification accuracy</td>
<td>80.2 %</td>
<td>99.7 %</td>
<td>96.1 %</td>
</tr>
</tbody>
</table>

Critical Distance Between Objects

The next experiment addresses the influence of the distance \( d \) between two objects \( O_1 \) and \( O_2 \) on classification and accuracy. The distance is defined as the translational length

\[
d = \| c_{O_1} - c_{O_2} \|
\]  

(5.23)

between the two according centroids \( c_{O_1} \) and \( c_{O_2} \). For the reasons of comparability, all results are measured with respect to the mean distance

\[
\bar{r} = \frac{1}{N} \sum_{i=1}^{N} \| c_O - p_i \|
\]  

(5.24)

of a surflet \((p_i, n_i)\) to the centroid \( c_{O_2} \). The set of \( N \) surflet samples drawn from an object is \( \{(p_1, n_1), \ldots, (p_N, n_N)\} \).

To identify the correlations of the algorithm’s performance and the distance between two objects, the scenes are restricted to two instances of one object with and without background. Two point clouds of an object are positioned and the distance is varied in 20 steps from total overlay to \( 5\bar{r} \). Each translation is repeated 100 times with random rotations.

In the first case, two identical objects are placed in free space. The applied rotation and translation use all six degrees of freedom. The second case includes a plane as background in addition to the objects. As for the other experiments, the objects are positioned in an arbitrary upstanding position, i.e. a rotation around the axis parallel to the table normal. Three degrees of freedom still remain. Since the algorithm’s cluster competition step [Chap. 5.4, p. 82] would suppress all multiple recognitions within the range of an object, it is disabled for this test. Unfortunately, this comes along with a common decreased classification performance.

The results in Fig. 5.5 show increasing classification rates and accuracy for larger distances in both tests, where (a) represents scenes without background while (b) summarizes scenes with a background plane. The observation that the classification performance for total overlay with background outperforms the case without background
5.5. EXPERIMENTS: SCENE INTERPRETATION IN PRACTICE

Figure 5.4: The plots show the segmented surface points of the correctly classified bunny. Picture (a) is the result of suppressing the background. In this case crosstalk will occur only if two objects are positioned very closely. As a consequence most surflet combinations are valid and survive the steps of cluster refinement. The final cluster contains a large number of points. Plot (b) depicts the result when the approach uses a model of the background. Due to much more crosstalk with the table, the number of surflets decreases significantly during the refinement steps. Thus, the point clouds of final clusters are significantly smaller in the presence of background.
might be confusing. The reason is a perfect alignment due to less degrees of freedom, where all rotational shapes lead to the classification of at least one object instance. On the other hand, the perfect alignment causes a large amount of crosstalk, such that accuracy stays low. Slight translations destroy the effect and the performance suffers from massive crosstalk.

A conclusion derived from the experiment is that for general usage, objects should at least have a distance of 2.5 to 3 times $\tilde{r}$ between them. This means that the approach can handle objects that nearly touch but do not intersect each other. For example, two cups will be expected to be separated if standing side by side, but not if they...

![Figure 5.5: Both plots show the dependency of object classification and accuracy on the distance between two objects. Crosses depict the measurements of correct object classification. The probability of segmenting a correct cluster point representing the accuracy of classification, is symbolized with triangles. Plot (a) demonstrates an object-pair in free space with random rotation and an increasing translation, while plot (b) shows results of an object-pair in upstanding position.](image-url)
are piled up. The cluster competition step uses this information to suppress multiple classifications or misclassifications for objects with insufficient spatial distance.

5.5.3 Experiments on Real Data

The experience derived from all previous experiments forms the base of the actual evaluation of the scene interpretation algorithm, which is finally applied to real data. Therefore, the conditions the algorithm has to face are varied while the implementation stays the same.

Due to much denser point clouds than for the previous tests on synthetic data, the parameterization is adapted according to the number of samples drawn and the minimum number of necessary samples in a valid cluster. The other parameters stay untouched.

Figure 5.6 depicts the used database. It consists of four objects:

- a toy phone,
- a wooden train,
- a bust of Zeus,
- and a bottle.

In the training phase the objects were sampled with the DLR multi-sensory 3D-Modeler [Chap. 1.5.1, p. 14,[SH04]]. Due to lowest surface failures, the decision has been made to use the laser range-sensor. To avoid background samples in the training data, every object was placed on a box. Since the box is covered with green fleece, it is invisible to the sensor and contributes with no measurements to the point clouds. The surfaces are sampled at once. Acquisition of views or registration of surface parts is not necessary, since all measurements are obtained in world coordinates. According to that procedure, the bottom cannot be part of any model.

In contrast to tests on synthetic data, the real scenes exhibit noise and incomplete surfaces. The reasons for incomplete surfaces and clutter are

- surface parts that are unreachable for the sensor, for example when the necessary sensing position would cause a collision of the 3D-modeler with other objects or with the table,
- occlusions,
- and regions that produce no valid information due to local reflectance behavior of the surface.
Figure 5.6: The real objects’ database: (a) a toy phone, (b) a wooden train, (c) a bust of Zeus, and (d) a bottle.
Figure 5.7: The six pictures (a) to (f) and the three (g) to (i) following in Fig. 5.8 show nine scenes as surface meshes, which were calculated from the 3D-data acquired with the laser range-scanner and the laser-stripe profiler. Holes in the surfaces result from unreachable regions, occlusions, and disadvantageous surface conditions (color, transparency, etc.).
CHAPTER 5. SCENE INTERPRETATION

Laser Range-scanner and Laser-stripe Profiler

The initially discussed data sources are the laser range-sensor and the laser-stripe profiler. Both devices are integrated in the DLR multi-sensory 3D-modeler. The laser range-scanner contributes with seven data sets and the laser-stripe profiler with two data sets in sampling the 3D-scenes. The two sensors are very similar in their principle of depth measurement and in the acquisition characteristics with respect to density and noise.

Nevertheless, scenes are acquired more easily with the laser range-sensor. Here, the emitted energy is much lower. Thus, one gets less noise from background and one does not have to apply any safety requirements, which are necessary for the laser-stripe profiler to prevent eye damages to unaware laboratory personnel.

The surface meshes shown in Fig. 5.7 (a) to (f) and in Fig. 5.8 (g) to (i) are the reconstructions of the point clouds presented to the scene interpretation algorithm. The approach resulted in correct objects in all nine scenes. Due to a better ratio of object points to background points the number of drawn samples is lower than for the experiments on synthetic data. Accordingly, the processing times on an Intel Xeon 2.8GHz processor with 2GB RAM was only about 1.57 seconds. The mean separated cluster consisted of about 495 surflets. Figure 5.9 shows an exemplary result of the bottle.
Although it is not possible to exactly determine the classification accuracy, the performance was inferior to the synthetic results. Especially the train showed a huge number of false object points. One reason is the similarity of the top of the cabin with the table. Thus, the train and the background share a lot of common surfllet-combinations, which cannot be distinguished by the algorithm.

Any performance differences according to laser range-scans or laser-stripe profiling were not observable.

Structured Light

In industrial environments structured light approaches are very popular to acquire 3D-shape information. It is an active measurement for fast planar (depth maps) acquisition of 3D-environments with a very simple principle. An emitter projects several known images on a focused scene, where an image consists of monochrome black and white lines or colored lines. Then, at least one camera in a calibrated relation to the projector senses the scene. In the next step each line is identified. Since the corresponding plane to every line is known, it is possible to calculate the depth from the observed projection onto the scene.

In this thesis five scenes were acquired with the structured light device of ABW GmbH [abw06] that consists of a central projection unit flanked with two lateral cameras. To generate scenes from multiple points of view, the objects have to be positioned on a turntable. During the sensing process, the table performed twelve rotations of 30°. For each step the illuminated scene was sampled from both two cameras.
Figure 5.10: The pictures show the surface meshes calculated from the data acquired with the ABW-3D structured light device. The degree of details is much higher than for the laser-based devices. On the other hand the amount of artefacts is larger, too. Taking the scenes (a), (b) and (c) the objects mirror in the blank metallic surface of the turntable. If changing the point of observation and using a board to prevent reflections, the device will collect wrong points in the background.
The acquired scenes are shown in Fig. 5.10. They contain the same objects as used for the previous experiments concerning the laser range-sensor and the laser-stripe profiler. Characteristics of this measurement method are very dense and detailed point clouds with very few holes. On the other hand, the number of artefacts is significantly larger. One source of noise is identified as artefacts raising from the turntable. Such disturbances mostly consist of ghost clouds, deriving from mirrored light due to the metallic surface or shifted fragments due to the rotation of the scene. Especially Fig. 5.10(d) and Fig. 5.10(e) show heavy accumulations of noise.

The sensor concept allows to vary the observed parts of a scene. This advantage is used to investigate three different stages of surface completeness:

1. The scene interpretation algorithm receives only the depth information of one single view. The input contains the same information as a depth map would. Apart from the single view point the shape data are very incomplete. This is a result from occlusion, shadowing, the opening angle of the camera, and different positions of emitter and receiver.

2. The next stage of visibility merges two depth images taken from opposing viewpoints. Hence, we get multiple - often unconnected - surfaces per object.

3. The final stage unifies all 24 views in one 3D-shape.

As we have already forecast for synthetic data [Chap. 3.6.6, p. 42], the classification rate suffers from incomplete surfaces. Thus, it is not surprising that the algorithm performs worst for the single view case at a rate of only 59.7%. Taking the antipodal view into account, the classification rate increases significantly and reaches a mean level of 76.0%. However, using two antipodal views still includes many invisible areas, which mostly result from occlusions. Further, building a merged scene out of all 24 views increases the classification rate to 82.4%. Tabular 5.2 summarizes the obtained results for running the algorithm five times per scene and visibility stage.

Comparing the results to the ones of the two laser-based approaches shows a classification rate beneath the expectations, even if taking the information of all views. There are multiple reasons for this phenomenon. First, the structured light scene is compared to database models acquired via laser-range scanner data, ignoring the different characteristics with respect to typical surface densities. Next, the environmental conditions, i.e. the small turntable, demand the situation of objects closer than in the laser tests. Accordingly, the minimum distance was violated in some cases. But most of all the algorithm suffered from the improper background model. The used plane was not capable to correctly describe the turntable and the artefacts resulting from the metallic surface and the rotational process of data acquisition.

Especially the train is difficult to detect. Here, the algorithm suffers from the conditions as it is the smallest object in the database and unfortunately it resembles the plane the most because of several planar parts. Finally, it must be stated that ignoring the environmental conditions, especially of the background negatively affect classification. Moreover, a complete background increases the danger of getting ambiguous features correlated with a larger amount of undetected crosstalk. However, the final performance depends on the dissimilarity and number of the database models.

**Stereo**

A very popular method for the acquisition of 3D-data is the use of stereoscopic devices. This technology can be found in many biological and biologically inspired vision sys-
CHAPTER 5. SCENE INTERPRETATION

Table 5.2: Correct classifications of objects on a turning table with respect to the ABW-3D structured light sensor.

<table>
<thead>
<tr>
<th>Scene Description</th>
<th>Number of Evaluated Views</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single</td>
</tr>
<tr>
<td>scene 1 (bust, phone)</td>
<td>60.0%</td>
</tr>
<tr>
<td>scene 2 (bottle, phone, train)</td>
<td>63.2%</td>
</tr>
<tr>
<td>scene 3 (bottle, bust, train)</td>
<td>52.6%</td>
</tr>
<tr>
<td>scene 4 (bottle, bust, train)</td>
<td>58.6%</td>
</tr>
<tr>
<td>scene 5 (bottle, bust, phone, train)</td>
<td>64.3%</td>
</tr>
<tr>
<td>mean</td>
<td>59.7%</td>
</tr>
</tbody>
</table>

tems. According to theory, a device analyses the views of two observers of known pose and relation. Technical systems are mostly realized as a pair of cameras. If an object is in the sensed area of both observers the depth of a surface point will be calculable via triangulation.

In practice, it is difficult to find the corresponding point of the first camera in the image of the second camera. Some of the reasons are that

- a point observed in one image is not necessarily present in the second image as well,
- due to reflections and illumination, the appearance of a surface can change a lot with respect to the viewpoint,
- and patches of homogenous reflectance offer no information to identify a single surface point.

To overcome the stereoscopic problems, most algorithms rather correlate boxes than single points. Unfortunately, this comes along with a loss of details massively corrupting the surface normals. A characteristic of many stereo images is a stairs structure instead of the original object surface.

Since surface normals are an essential information in our object description, it is not realistic to assume good results under these conditions. It might be possible to correct the stairs-effect by interpolating between several depth images, but this would required research far beyond this work.
GENERALIZATION with respect to object representation and scene interpretation is an important motivation in this thesis. So far, the presented results fulfilled this claim. This chapter discusses extensions and refinements of the algorithms with respect to other aims and conditions. It is shown that the main concepts are transferable to related applications.

Most of these topics are in a preliminary state of development and are not researched in depth. Of course, they leave space for speculation or ongoing investigation.

6.1 Positioning of the Clusters

One essential question of the scene interpretation approach is where to position the cluster seed nodes. This decision determines the final quality of the interpretation result. This work spots a robotics application, where processing time is typically kept as short as possible. Other applications, especially if done offline, will not have this requirements. Thus additional processing can be done to improve the scene interpretation.

The simplest and fastest method to initialize cluster seeds is to use surface points, which work quite well in practice. Nonetheless, this method prevents a segmented point cloud to include the complete object surface. In fact, the reason is the size of a cluster seed’s support region. It is restricted to a sphere of the maximum object radius $r_{\text{max}}$. Therefore, intersecting support regions of different cluster seeds could never contain the whole object volume. Under this condition, the center $q$ is the only point which is able to collect all surface points $p$.

However, to increase the support region is not advisable since it is unknown how much additional space is adequate and since it increases the likelihood of more candidates for crosstalk. In the following, two promising alternatives are presented.
6.1.1 The Floating Cluster Seeds Algorithm

One way to optimize the initial cluster seed positions is to move them into the assumed center. Further, this method is called the floating cluster seeds algorithm (FCSA).

Therefore, a further parameter is established in the model. It is the minimum object radius

\[ r_{\text{min}} \leq ||q - p_i||, \forall p_i \in O \]  

that defines the maximum area in an object which is free of surface points.

With a given scene respecting the pre-selection according to object \( O \) [Chap. 5.4, p. 78], it can be assumed that the point cloud is significantly denser at the objects’ physical locations. Further, at least one cluster seed can be expected being initialized on the surface of a correct object cluster [Chap. 5.4, p. 80].

The idea is to move the cluster seed and the corresponding region of interest \( F(r, q) \) [Chap. 4.13, p. 62] until a maximum number of points is included. A general way to solve such a task is to maximize an energy function by degrading the gradients.

Following this idea, the energy function

\[ E(q, S_M) = \sum_{p_i \in S_M} E_{\text{pot}}(p_i, q) \]  

is defined, where \( S_M \subseteq F(r, q) \) is the set of drawn surflet samples and \( E_{\text{pot}}(p_i, q) \) is the potential of a surflet \( p_i \) with respect to the center \( q \).

A surflet’s potential energy will be large in the borders of regions, if the distance to the center is close to \( r_{\text{min}} \) or \( r_{\text{max}} \). In other cases the contribute is very low or zero. This can be achieved by combining the two sigmoidal functions

\[ E_{\text{min}}(p, q) = (1 + e^{s(r_{\text{min}} - \delta(p, q))})^{-1} \]  

and

\[ E_{\text{max}}(p, q) = (1 + e^{-s(r_{\text{max}} - \delta(p, q))})^{-1}. \]

Finally, one can postulate the formulation

\[ E_{\text{pot}}(p, q) = E^{-1}_{\text{norm}}(q)(E_{\text{min}}(q, p) + E_{\text{max}}(q, p) - 1). \]

The term

\[ \delta(p, q) = ||p - q|| \]

denotes the distance between the center \( q \) and a surflet \( p \). The parameter \( s \) specifies the shape of the sigmoidal. Small values yield to low gradients such that the graph resembles a linear function. Otherwise, large values trim the sigmoidal towards the shape of a step-function. The divisor

\[ E_{\text{norm}}(q) = E_{\text{min}}\left(\left(\frac{1}{2}(r_{\text{max}} - r_{\text{min}}, 0, 0)^T - q\right), q\right) + E_{\text{max}}\left(\left(\frac{1}{2}(r_{\text{max}} - r_{\text{min}}, 0, 0)^T - q\right), q\right) - 1 \]

normalizes Eq. (6.5). It only depends on \( q \). Subsequently, \( E_{\text{pot}}(q, p) \) is limited to the codomain \([0, 1]\).

Optimizing the cluster seed position \( q \) is equal to the maximization of \( E(q, S_M) \). This is achieved by calculating the derivatives of \( E(q, S_M) \) with respect to the components of the vector \( q \). Then \( q \) is corrected with its weighted negative partial derivatives.
Repeating the correction for some iterations moves the cluster seed in an enhanced position, where the weight determines the final degree of correction. However, finding the global maximum is not guaranteed.

Since the cluster seed optimization is a computational intensive procedure, it is advisable to separate the equation in often used parts. First, the distance $\delta(p, q)$ is determined. Next, the exponential functions

$$
\lambda_{\text{min}}(\delta(p, q)) = e^{s(r_{\text{min}} - \delta(p, q))} \tag{6.8}
$$

and

$$
\lambda_{\text{max}}(\delta(p, q)) = e^{-s(r_{\text{max}} - \delta(p, q))} \tag{6.9}
$$

are calculated. Then it is possible to insert Eq. (6.8) and Eq. (6.9) into

$$
\Lambda_{\text{min}}(\delta(p, q)) = \frac{\lambda_{\text{min}}(\delta(p, q))}{\delta(p, q)(1 + \lambda_{\text{min}}(\delta(p, q)))^2} \tag{6.10}
$$

and

$$
\Lambda_{\text{max}}(\delta(p, q)) = -\frac{\lambda_{\text{max}}(\delta(p, q))}{\delta(p, q)(1 + \lambda_{\text{max}}(\delta(p, q)))^2} \tag{6.11}
$$

respectively. The derivative to the x-dimension is given by

$$
\frac{\partial}{\partial x} E(q, S_M) = \frac{s}{E_{\text{norm}}(q)} \sum_{p_i \in S_M} (\Lambda_{\text{min}}(\delta(p_i, q)) - \Lambda_{\text{max}}(\delta(p_i, q))) (p_{i,x} - q_x). \tag{6.12}
$$

Analogously, one can determine $\frac{\partial}{\partial y} E(q, S_M)$ for the y-dimension and $\frac{\partial}{\partial z} E(q, S_M)$ for the z-dimension. At this stage, using the partial derivatives is possible for the correction of the cluster seed center $q$. Following the assumption that every correction gets the center closer to the optimum position, it is necessary to steer the correction. A center estimated in the $(k + 1)$th iteration is defined by

$$
q_{k+1} = q_k + \frac{r_{\text{min}, M}}{k + 1} \left[ \left( \frac{\partial E(q, S_M)}{\partial x}, \frac{\partial E(q, S_M)}{\partial y}, \frac{\partial E(q, S_M)}{\partial z} \right)^T \right]. \tag{6.13}
$$

In general the correction vector does not translate the cluster seed to the real center. Moreover, it is possible to move too far such that the cluster seed position gets even worse. This effect can be reduced by scaling the correction vector with a small weight that depends on the minimum object radius $r_{\text{min}, M}$ and the actual iteration. Accordingly, movements into the correct direction shift the cluster seed closer to the optimum.

### 6.1.2 Results

Concerning processing speed, the scene interpretation algorithm with the simple cluster seed initialization was satisfactory. Thus, all experiments with respect to processing speed are very coarse. The experiment and the results presented for the FCSA are very preliminary and cover only a few working aspects.

The experiment is based on synthetic data which is a database of four objects. It includes a bottle, a carafe, a cup, and a wine glass. The test scene consists of a plane and one instance of each database object, placed in an upstanding random position without any intersections to other objects or the plane. Figure 6.2 shows an example for one test.
Figure 6.1: The image shows an example of a reduced point cloud of a table scene with four objects.

The investigated questions are: how much does the FCSA improve the classification and what is the required additional processing cost? Therefore, the algorithm examined 100 scenes and iteratively optimized the cluster seeds’ positions in 25 steps. Figure 6.2 plots the movements of the cluster seeds for the exemplary scene of Fig. 6.2.

The cluster seeds of the two large objects (bottle and carafe) move much more than ones of small objects (cup and wine glass). This is determined by the weighted maximum correction vector [cf. Eq. (6.13)]. Hence, the larger objects fulfill the condition to change significantly.

The experiments concerning the FCSA were not performed using the latest version of the scene interpretation algorithm. Therefore, a presentation of the mean processing time and classification rate is of minor evidence. Nevertheless, a coarse preliminary summary of the results concludes that the number of false classifications is 10 times less, while the processing time is 10 times larger. Regarding the high performance without FCSA, strong arguments are necessary to justify the additional massive processing expense of the FCSA.

6.1.3 Repetitive Clustering

Another possibility to optimize the position of a cluster seed is to repeat the cluster algorithm twice. In the first part, the scene interpretation algorithm segments the point cloud as usual. The result is a point cloud, which contains those points being in mean distance or closer to the center. Points with a distance near the maximum radius are very unlikely. This results from the circumstance that a final cluster is the product of an intersection of multiple cluster seeds, which are positioned on the object surface.

As demonstrated, the classification rate is very high and significantly reduces the number of object candidates [Chap. 5.5, p. 83]. Accordingly, it is opportune to repeat
6.2. INCREASING THE NUMBER OF CLUSTER POINTS

For some applications the segmented objects might contain too little points. Autonomously grasping an object with an artificial hand is one example for such an application. Here, a stable grasp depends on the correct positioning of every finger, which demands for a dense sampling of object surfaces.

The first approach to solve this type of density problem is to increase the number of initial point pair drawings, but some experience has shown that this approach is not successful. Two circumstances are responsible. First, a larger set of drawn points
yields a larger amount of crosstalk. Therefore false positive cluster seeds survive for a longer time and more iterations are necessary to clear the set of cluster seed points from crosstalk. Second, the higher number of iterations also yield a loss of many desired surface points. Thus the increase in processing cost is overproportional to the increase in resulting surface points.

A promising alternative is to repeat the scene interpretation algorithm and to collect the results until the desired amount of surface points is reached. This procedure offers several advantages:

1. It is possible to get a linear relation between processing costs and segmented surface points. Thinking about an application which has different lengths of recognition phases, it is further possible to segment the maximum number of points per available time gap.

2. Since every new run of the scene interpretation draws another initial point set, multiple hypotheses can be unified into an improved classification.

3. Modifying the algorithm with respect to repetitive clustering [Chap. 6.1.3, p. 102] allows to speed up subsequent runs.

6.3 Parallelism and Online-Recognition

Actually, semiconductor technology has reached a level where multi processor systems and dual core processors get cheap enough to be interesting for a large public. The computer game industries’ pursuit for realistic computer graphics and sound has already established multi-processor graphic cards and sound cards in usual personal computers. Surely, the development will not end at this stage.

This situation in mind, an efficient algorithm should be able to take advantage of parallelism. The proposed scene interpretation allows multi-processing at different levels. Rather than with many programming languages, the term *thread* is used in a more abstract sense. It describes a process working parallel to others. The task is independent and produces no side effects, so that a migration to an arbitrary processing unit is feasible. Questions concerning common memory versus private memory are not topic of this discussion.

6.3.1 Parallelized Model Evaluation

In an early phase of the scene interpretation algorithm, the initial set of surface point pairs is divided into subsets, where each subset contains the pairs with respect to the most likely model. Every further evaluation concerning one model subset is independent to those of other models. This condition fails after building the final clusters. That way, it is not only possible to spend one thread per model but is also thinkable to run the evaluation of every cluster seed on a single thread. Under the assumption of hardware that provides massive parallelism most processing is performed simultaneously. Only a few tasks like the final cluster competition [Chap. 5.4, p. 82] have to run in a single processor mode to coordinate and judge the results of the parallel units.

6.3.2 Online-Recognition

In robotics, fast analysis of a scene is of essential importance to maximize economic aspects in industry and to increase acceptance in service applications. Some scenarios
6.4. POSE ESTIMATION

won’t allow to completely acquire a scene at once. One example is a robot bearing a 3D-sensing device and the task to identify some objects in its environment. To explore the neighborhood, the robot scans with the sensing device.

Assuming a rigid environment, it is possible to take advantage of the proposed scene interpretation. In the example the robot collects more and more information. Subsequently the knowledge about the world is completed. Running the scene interpretation on incomplete data allows to generate hypotheses about the position of interesting objects. Thus it is possible to steer the exploration by placing attractors wherever the scene interpretation detects objects and needs more information to correctly classify them. On the other hand, distractors can be used to inhibit further exploration of already examined areas.

Coupling the proposed exploration and online-recognition with the suggestions proposed in the previous section, very fast working cycles can be in reached.

6.4 Pose Estimation

Using an object representation which is invariant with respect to translation and rotation seems to be disadvantageous for applications that rely on a good pose estimation. An example might be a service robot with the task to clear a table of bottles. For grasp planning one can further assume that the robot needs the exact pose of a bottle.

Standard solutions for this problem design a specialized recognition for specific objects. An often used approach is to generate models, which allow to recognize a bottle and its pose at once. An algorithm like this is typically very fast. Also it typically needs adaptation (if at all possible) for other objects. The required knowledge for these changes mostly demands for an expensive expert. At the same time, this automatically limits the economic attractiveness of the robot.

In contrast, this work has presented a framework for a simple generation of a reference database and an easy to use scene interpretation. It allows for rapid segmentation and classification of point clouds of an object at once. Even if the final point clouds often contain a certain percentage of false surface points, it is an accurate separation from the scene. This makes the scene interpretation algorithm well suited as a pre-computational step with a succeeding pose estimation step.

Since further processing profits from the fast segmentation and classification, it is possible to develop these benefits into generic methods for the determination of the pose. Now the conditions are less sophisticated. Thus, it is possible to select from a large variety of approaches. Some of them only determine an object pose. Others, mostly so called view-based approaches, classify an object and calculate its pose in one step. Of course the classification is already done, however the additional assumption could be used as a last verification.

All these pose estimation approaches need their own object representation, so that they have to be embedded in a second database. Nevertheless, such a two-step design is attractive due to the generic concept.
Conclusion

This work focuses on a novel generic approach for fast interpretation of rigid 3D-scenes. It presents several new methods starting with free-form object modeling, followed by classification, efficient 3D-region access and finally a scene decomposition.

The basis for modeling 3D free-form objects is a compact, rotational and translational invariant description of shape. It relies on a four-dimensional distribution of surface point pairs, which is capable of representing objects without any restrictions of their structure. Moreover, the model unifies local and global aspects of shape. Experiments have shown that the object description satisfies the requirements for moderate disturbances. In detail noise, occlusion, clutter, and deformation with respect to classification were investigated. In fact, experimental results proof a strong correlation between the performance of the approach and the applied criterion for classification. The likelihood criterion outperformed a number of several popular criteria. It is characterized by the highest rate of correct classification as well as the lowest processing cost.

Since the interpretation efficiency of spatial information strongly depends on the organization of 3D-data and the realized access methods, this work developed a novel combination of 3D-indexing and an algorithm for spherical 3D-region search. The results demonstrate the potential of spatial subdivision trees when used with octal keys for referencing. The region search algorithm itself benefits from a two step approach. The first step realizes a coarse approximation of the focused volume by descending the tree to branches which possibly contain the desired region. This operation is very fast due to restriction on simple integer comparisons. Only for the exact determination of distances to the focus, the region search algorithm has to apply processing intensive floating-point operations. Therefore, the algorithm performs best for small and medium spheres in a 3D-scene. This circumstance stems from a large search space restriction imposed by the first step and a subsequent concretization by the second step.

The last part of this thesis addresses the scene interpretation problem itself. It includes object segmentation, which is the separation of a local subset from a large scene.
point cloud as well as the classification of this subset. A novel approach is discussed, based on a cluster refinement algorithm running on object specific point cloud derivatives. It starts with random surflet sets and iteratively sorts out likely misclassifications. Each time the surflet sets are examined with respect to new aspects of their feature distribution. Simultaneously, the required similarity to the expected model is raised. It is demonstrated that this method allows to segment and classify surflet sets in parallel.

The combination of all discussed approaches results in a framework for generic object description and for generic scene interpretation. Since it provides very simple model generations in training phase and rapid, high accurate results in interpretation phase, it is highly suited for autonomous systems. The special focus lies on robotics applications, where time and processing are limited resources. Several experiments have shown the framework’s capability to classify if knowledge about the background is available. Whenever this is not true, the classification process behaves very object specific in terms of quality. The same behavior is observed whenever several objects are grouped below a critical minimum distance.

Considering applications where the setting often changes, while the life-cycle is short, robots actually suffer from expensive adaptation costs for changing the vision software and acquiring reference models. Here, the proposed methods could help to encourage new challenges and to conquer new markets.
In the following, the most relevant parameters, notations, and operators are summarized for a quick review. Generally, vectors are represented by bold Latin letters, while functions and scalars use normal Latin or Greek letters. Sets are symbolized by capital letters from the Latin or Greek alphabet.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>The cross product of two vectors.</td>
</tr>
<tr>
<td>◦</td>
<td>The scalar product of two vectors.</td>
</tr>
<tr>
<td></td>
<td>\cdot</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### APPENDIX A. NOTATION

<table>
<thead>
<tr>
<th>Function</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma$</td>
<td>The function that maps a surflet-pair $s$ to a feature vector $f_s$.</td>
</tr>
<tr>
<td>$\Upsilon$</td>
<td>The function generates a set of random surflet samples drawn.</td>
</tr>
<tr>
<td>$h_M$</td>
<td>The function that maps a feature vector $f_s$ to a histogram bin $b$.</td>
</tr>
<tr>
<td>$\text{card}$</td>
<td>The cardinality function.</td>
</tr>
<tr>
<td>$\phi_{h_M, \text{max}}$</td>
<td>The function that maps a set of sampled feature vectors to a histogram $H$.</td>
</tr>
<tr>
<td>$\mathcal{E}$</td>
<td>The Euclidian distance between two histogram vectors.</td>
</tr>
<tr>
<td>$\cap$</td>
<td>The intersection criterion.</td>
</tr>
<tr>
<td>$\chi^2_1$</td>
<td>The parameter represents the first notation of the $\chi^2$-test.</td>
</tr>
<tr>
<td>$\chi^2_2$</td>
<td>The parameter represents the second notation of the $\chi^2$-test.</td>
</tr>
<tr>
<td>$\mathcal{K}$</td>
<td>The parameter represents the Kullback-Leibler divergence.</td>
</tr>
<tr>
<td>$\mathcal{L}$</td>
<td>The parameter represents the likelihood criterion.</td>
</tr>
<tr>
<td>$m$</td>
<td>The function maps a given surflet to an index used to reference its spatial position.</td>
</tr>
<tr>
<td>$c$</td>
<td>The function maps a given index to the according spatial cell.</td>
</tr>
<tr>
<td>$\mathcal{F}(r, q)$</td>
<td>The function describes the region of interest (ROI).</td>
</tr>
<tr>
<td>$\hat{i}(I, d)$</td>
<td>The function determines the prefix derived from the Index $I$ of desired length $d$.</td>
</tr>
<tr>
<td>Variable</td>
<td>Explanation</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>$s$</td>
<td>The tuple defines a surflet pair consisting of two surflets that consist of a surface point and a surface normal themselves.</td>
</tr>
<tr>
<td>$u, v, w$</td>
<td>The orthogonal vectors define a coordinate frame in relation to a surflet.</td>
</tr>
<tr>
<td>$f_s$</td>
<td>The four-dimensional feature vector is derived from a surflet-pair.</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>The first parameter of the feature vector $f_s$. It represents an azimuthal angle of a polar angle.</td>
</tr>
<tr>
<td>$\beta$</td>
<td>The second parameter of the feature vector $f_s$. It represents an azimuthal angle of a polar angle.</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>The third parameter of the feature vector $f_s$. It represents an azimuthal angle of a polar angle.</td>
</tr>
<tr>
<td>$\delta$</td>
<td>The fourth parameter of the feature vector $f_s$. The parameter encodes the Euclidean norm of two surflets.</td>
</tr>
<tr>
<td>$\delta_{M,\text{max}}$</td>
<td>The parameter describes the maximum object diameter that is the distance between the farthest two points of Object $M$.</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>The parameter names the initial set of surflets used in the training phase.</td>
</tr>
<tr>
<td>$\Psi$</td>
<td>The parameter represents a set of four-dimensional feature vectors.</td>
</tr>
<tr>
<td>$n_f$</td>
<td>The scalar describes the maximum number of surflet pairings that can be derived from a given surflet set.</td>
</tr>
<tr>
<td>$d$</td>
<td>The scalar describes the size of a histogram, i.e. the number of bins.</td>
</tr>
<tr>
<td>$H$</td>
<td>The parameter denotes a feature histogram.</td>
</tr>
<tr>
<td>$M$</td>
<td>The parameter denotes an object model.</td>
</tr>
<tr>
<td>$\bar{r}_M$</td>
<td>The scalar subscribes the mean distance between points and the center of an object.</td>
</tr>
<tr>
<td>$r_{M,\text{max}}$</td>
<td>The scalar denotes the distance between the object point being farthest from the object center.</td>
</tr>
</tbody>
</table>
### APPENDIX A. NOTATION

<table>
<thead>
<tr>
<th>Variable</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vec{n}$</td>
<td>The vector represents the surface normal derived from a local neighborhood.</td>
</tr>
<tr>
<td>$A$</td>
<td>The scalar denotes the area of a triangle.</td>
</tr>
<tr>
<td>$\Omega$</td>
<td>A database of objects.</td>
</tr>
<tr>
<td>$O$</td>
<td>The parameter defines an object.</td>
</tr>
<tr>
<td>$\hat{O}$</td>
<td>The notation for an unknown object.</td>
</tr>
<tr>
<td>$\mathbb{N}_{i,j}$</td>
<td>This notation summarizes a set of natural numbers starting from $i$ to $j$, inclusively.</td>
</tr>
<tr>
<td>$S$</td>
<td>The parameter symbolizes a set of surflet-pairs.</td>
</tr>
<tr>
<td>$\hat{\epsilon}$</td>
<td>The minimum Euclidian distance with respect to a database.</td>
</tr>
<tr>
<td>$\hat{\cap}$</td>
<td>The minimum intersection criterion with respect to a database.</td>
</tr>
<tr>
<td>$\hat{\chi}^2_1$</td>
<td>The parameter minimum database response with respect to the first notation of the $\chi^2$-test.</td>
</tr>
<tr>
<td>$\hat{\chi}^2_2$</td>
<td>The parameter minimum database response with respect to the second notation of the $\chi^2$-test.</td>
</tr>
<tr>
<td>$\hat{K}$</td>
<td>The parameter represents the minimum database response with respect to the Kullback-Leibler divergence.</td>
</tr>
<tr>
<td>$\hat{L}$</td>
<td>The parameter represents the maximum database response with respect to the likelihood criterion.</td>
</tr>
<tr>
<td>$H_{\text{min}}$</td>
<td>The parameter defines the value of the lowest non-zero bin in a histogram.</td>
</tr>
<tr>
<td>$H'_{\text{MF}}$</td>
<td>The parameter represents the modified histogram that is free of zero bins.</td>
</tr>
<tr>
<td>$n_{d_{\text{max}}}$</td>
<td>The parameter denotes the maximum storage capacity of a subdivision tree.</td>
</tr>
<tr>
<td>$\epsilon_{d_{\text{max}}}$</td>
<td>The parameter represents the edge size of an atomic cell, given the tree depth $d_{\text{max}}$.</td>
</tr>
<tr>
<td>Variable</td>
<td>Explanation</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>$I$</td>
<td>The value contains the index of a spatial cell.</td>
</tr>
<tr>
<td>$C$</td>
<td>The parameter represents a spatial cell.</td>
</tr>
<tr>
<td>$n_{\text{INT}}$</td>
<td>The parameter defines the number of required integers, which is necessary to generate an index of desired length.</td>
</tr>
<tr>
<td>$d_{\text{maxOT}}$</td>
<td>The value stands for the maximum depth of an octree.</td>
</tr>
<tr>
<td>$d_{\text{maxBTT}}$</td>
<td>The value stands for the maximum depth of a balanced binary tree.</td>
</tr>
<tr>
<td>$I_{\mathcal{F}}$</td>
<td>The parameter denotes the index according to the cell, which is large enough to contain the region of interest.</td>
</tr>
<tr>
<td>$I^*$</td>
<td>The parameter represents the super-key correlated to a spatial position.</td>
</tr>
<tr>
<td>$S^*_{M_i}$</td>
<td>The parameter symbolizes the sample set after the pre-selection step.</td>
</tr>
<tr>
<td>$C_{M_i}$</td>
<td>The parameter defines a model-specific local neighborhood around a center point.</td>
</tr>
<tr>
<td>$S'_{e,M_i}$</td>
<td>The parameter symbolizes the sample set included in a cluster seed in the end of cluster seed initialization.</td>
</tr>
<tr>
<td>$S''_{e,M_i}$</td>
<td>The parameter symbolizes the sample set included in a cluster seed in the end of cluster seed refinement.</td>
</tr>
<tr>
<td>$S'''_{e,M_i}$</td>
<td>The parameter symbolizes the sample set included in a cluster seed in the end of cluster seed merging.</td>
</tr>
<tr>
<td>$s$</td>
<td>The parameter symbolizes the saturation metric used for the degree of space occupancy.</td>
</tr>
<tr>
<td>$t_{\text{rel}}$</td>
<td>The parameter symbolizes the relative processing time used for the comparison of region search algorithms.</td>
</tr>
</tbody>
</table>
Bibliography


