Shape Retrieval Methods for Architectural 3D Models

Dissertation

zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Dipl.-Inf. Raoul Henrik Joseph Frédéric Wessel

aus Koblenz

Bonn, April 2013

Universität Bonn
Institut für Informatik II
Friedrich-Ebert-Allee 144, D-53113 Bonn

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

Dekan: Prof. Dr. U.-G. Meißner

1. Referent: Prof. Dr. Reinhard Klein
2. Referent: Prof. Dr. Tobias Schreck

Tag der mündlichen Prüfung: 17.01.2014
Erscheinungsjahr: 2014

Contents

Zusammenfassung **v**

Abstract **vii**

Acknowledgements **viii**

1 Introduction **1**
- 1.1 Motivation 2
- 1.2 Goals 3
- 1.3 Contributions 4
- 1.4 Outline 4
- 1.5 Preliminaries 6
  - 1.5.1 3D Object Retrieval as a Special Case of Information Retrieval 6
  - 1.5.2 Leave-one-out Tests 7
  - 1.5.3 Retrieval Metrics 9
  - 1.5.4 Robust Estimation of Conditional Probabilities 12

I Feature-based Shape Retrieval for 3D Architectural Context Models **21**

2 Learning Distinctive Local Object Characteristics **23**
- 2.1 Introduction 23
- 2.2 Related Work 25
  - 2.2.1 Comparing Global Shape Descriptors 25
  - 2.2.2 Comparing Local Shape Descriptors 27
  - 2.2.3 An Overview on Shape Descriptors 28
  - 2.2.4 Supervised Learning in Shape Retrieval 33
  - 2.2.5 3D Shape Benchmarks 34
- 2.3 Class Distribution Descriptors 35
  - 2.3.1 Combining Class Distribution Descriptors 35

2.3.2 Comparing Class Distribution Descriptors 39
2.4 Results on Princeton Shape Benchmark 39
2.4.1 Experimental Setup 39
2.4.2 Evaluation 41
2.5 A Benchmark for 3D Architectural Data 47
2.5.1 Classification Schemes 48
2.5.2 Benchmark Models 49
2.5.3 Retrieval results 50
2.6 Conclusion 52

3 Learning the Compositional Structure of Man-Made Objects 55
3.1 Introduction 55
3.2 Related Work 57
3.3 Feature Selection and Descriptor Computation 58
3.3.1 Feature Selection 58
3.3.2 Descriptor Computation 59
3.3.3 Integrating Feature Locations 61
3.3.4 Spatial Relationship between Features 62
3.3.5 Modified Feature Vectors and Kernel Functions 62
3.3.6 Modified Combination of Class Distribution Descriptors . 64
3.4 Results 64
3.4.1 Experimental Setup 64
3.4.2 Evaluation 65
3.4.3 Timings 66
3.5 Conclusion 67

4 Beyond Shape: Groups, Materials, and Text for 3D Retrieval 71
4.1 Introduction 71
4.1.1 Generalization Issues 74
4.1.2 Contribution 74
4.2 Intrinsic Groupings for Feature Localization 74
4.3 Material Descriptors 77
4.4 Textual Annotations 77
4.5 Combining Shape, Material, Text, and Different Localization Strategies 79
4.6 Conclusion 80

II Graph-based Shape Retrieval for 3D Architectural Building Models 83

5 Analyzing and Indexing Building Models 85
5.1 Introduction 85
5.2 Room Connectivity Graphs 86
5.2.1 Node Attributes 87
5.2.2 Edge Attributes 87
5.3 Related Work 88
5.3.1 Model Graphs 90
5.3.2 Skeleton Graphs 91
5.3.3 Reeb Graphs 93
5.3.4 Summary 94
5.4 Room Connectivity Graph Extraction 94
5.4.1 Automatic Story Segmentation 94
5.4.2 Floor Plan Generation 98
5.4.3 Room Detection 99
5.4.4 Door and Window Detection 105
5.4.5 Detection of Vertical Connections and Room Refinement . 105
5.5 Searching for Structures in Room Connectivity Graphs 107
5.6 Results 107
5.7 Conclusion 108

6 Retrieval and Classification with Room Connectivity Graphs 119
6.1 Introduction 119
6.2 Related Work 120
6.2.1 Edit Distances 121
6.2.2 Graph Kernels 121
6.2.3 Graph Embeddings 123
6.3 Method Overview 123
6.4 Node and Edge Attributes 125
6.4.1 Node Attributes 125
6.4.2 High-level Node Attributes 126
6.4.3 Edge Attributes 128
6.5 Approximate Graph Edit Distances 129
6.5.1 Algorithm 129
6.5.2 Cost Functions 130
6.6 Bag-of-Subgraphs Construction 132
6.6.1 Subgraph Mining 133
6.6.2 Codebook Generation 133
6.6.3 Subgraph Embeddings 134

6.7 Evaluation 134
6.7.1 Methods and Parameters 135
6.7.2 Influence of Attributes 136
6.7.3 Retrieval Results 136
6.7.4 Classification Results 137
6.7.5 Timings 137
6.8 Conclusion 138

III Closure 145

7 Conclusions 147
7.1 Summary 147
7.2 Future Work 149

Bibliography 151

ZUSAMMENFASSUNG

In dieser Arbeit werden neue Methoden zur inhaltsbasierten Suche nach 3D Modellen aus dem Bereich der Architektur vorgestellt. Dabei werden grundsätzlich zwei Typen von Architekturmodellen unterschieden. Der erste Typ umfasst sogenannte *Kontextobjekte*, die für die detaillierte Ausgestaltung eines neuen Gebäudeentwurfs verwendet werden. Hierzu zählen beispielsweise Inneneinrichtungsgegenstände wie Möbel, sowie Modelle zur Umgebungsgestaltung wie z.B. Pflanzen oder Zäune. Der zweite Typ von Modellen umfasst die eigentlichen Gebäudemodelle. Um eine effiziente und auf das Anforderungsprofil der Nutzer zugeschnittene inhaltsbasierte Suche für beide Modelltypen zu ermöglichen, ist die Entwicklung von individuellen Suchmechanismen notwendig. Kontextobjekte wie z.B. Einrichtungsgegenstände, die eine bestimmte, gemeinsame Funktion erfüllen (wie z.B. Sitzmöbel) weisen oftmals eine global ähnliche Form auf. Nichtsdestotrotz werden sie aus architektonischer Sichtweise als unterschiedlichen Objektunterklassen zugehörig angesehen (z.B. Sessel, Drehstuhl, Lehnstuhl). Die Unterscheidung wird oft anhand kleiner geometrischer Details getroffen und ist bisweilen nur einem Experten auf dem Gebiet der Architektur möglich. Gebäude auf der anderen Seite werden meist anhand der Struktur ihrer zugrundeliegenden Grundrisse und Raumpläne unterschieden. Topologische Raumplaneigenschaften sind beispielsweise der Ausgangspunkt, um Wohngebäude von Gewerbebauten zu unterscheiden.

Der erste Beitrag dieser Arbeit ist ein neuer Metadeskriptor zur Suche nach Kontextobjekten, der unter Verwendung eines überwachten Lernansatzes verschiedene Typen lokaler Formdeskriptoren miteinander kombiniert. Der Ansatz ermöglicht die Unterscheidung von Objektklassen anhand kleiner geometrischer Abweichungen und integriert zugleich Expertenwissen aus dem Bereich der Architektur. Die Methode wird zunächst anhand einer Datenbank bestehend aus allgemeinen 3D Objekten getestet. Im zweiten Schritt erfolgt eine Evaluation anhand von 3D Objekten aus dem Architekturbereich. Im Folgenden wird der Ansatz um eine neue Methode zur geschickten räumlichen Lokalistation von Formdeskriptoren erweitert. Zusätzlich wird Wissen über räumliche Anordnungen von Objektkomponenten ausgenutzt, um die Suchergebnisse weiter zu verbessern. Im zweiten Teil der Arbeit wird mit dem *Raumverbindungsgraphen* (RVG) ein Konzept zur effektiven Beschreibung eines Gebäudes anhand seiner Grundrisse und Raumpläne vorgestellt. Zunächst wird erläutert, wie ein RVG aus einem 3D Gebäudemodell erzeugt werden kann. Im Anschluss wird diskutiert, wie gezielt und effizient nach Substrukturen in diesem Graphen gesucht werden kann. Abschließend wird ein als *Bag-of-Subgraphs* bezeichneter neuer Deskriptor eingeführt, bei dem ein attributierter Graph mithilfe von Subgrapheinbettungen in eine Vektorrepräsentation überführt wird. Die Suchperformanz dieses Deskriptors wird dann anhand einer Datenbank von Modellen mit verschiedenen Grundriss- und Raumplantypen evaluiert.

Alle in dieser Arbeit vorgestellten Methoden wurden mit dem Ziel entwickelt, eine möglichst automatisierte Indexierung und Suche zu gewährleisten, die so wenig wie möglich menschliche Interaktion erfordert. Dementsprechend sind für alle Verfahren lediglich Polygonsuppen als Eingabe erforderlich, die nicht manuell repariert oder strukturiert werden müssen. Der menschliche Arbeitsaufwand beschränkt sich auf die Erstellung von Groundtruth für die verwendeten überwachten Lernverfahren in Form manueller Annotation von 3D Objekten, sowie der Bereitstellung von Informationen über die Orientierung von Gebäudemodellen und der zur Modellierung verwendeten Maßeinheit.

Abstract

This thesis introduces new methods for content-based retrieval of architecture-related 3D models. We thereby consider two different overall types of architectural 3D models. The first type consists of *context objects* that are used for detailed design and decoration of 3D building model drafts. This includes e.g. furnishing for interior design or barriers and fences for forming the exterior environment. The second type consists of actual *building models*. To enable efficient content-based retrieval for both model types that is tailored to the user requirements of the architectural domain, type-specific algorithms must be developed. On the one hand, context objects like furnishing that provide similar functions (e.g. seating furniture) often share a similar shape. Nevertheless they might be considered to belong to different object classes from an architectural point of view (e.g. armchair, elbow chair, swivel chair). The differentiation is due to small geometric details and is sometimes only obvious to an expert from the domain. Building models on the other hand are often distinguished according to the underlying floor- and room plans. Topological floor plan properties for example serve as a starting point for telling apart residential and commercial buildings.

The first contribution of this thesis is a new meta descriptor for 3D retrieval that combines different types of local shape descriptors using a supervised learning approach. The approach enables the differentiation of object classes according to small geometric details and at the same time integrates expert knowledge from the field of architecture. We evaluate our approach using a database containing arbitrary 3D models as well as on one that only consists of models from the architectural domain. We then further extend our approach by adding a sophisticated shape descriptor localization strategy. Additionally, we exploit knowledge about the spatial relationship of object components to further enhance the retrieval performance. In the second part of the thesis we introduce attributed *room connectivity graphs (RCGs)* as a means to characterize a 3D building model according to the structure of its underlying floor plans. We first describe how RCGs are inferred from a given building model and discuss how substructures of this graph can be queried efficiently. We then introduce a new descriptor denoted as *Bag-of-Attributed-Subgraphs* that transforms attributed graphs into a vector-based representation using subgraph embeddings. We finally evaluate the retrieval performance of this new method on a database consisting of building models with different floor plan types.

All methods presented in this thesis are aimed at an as automated as possible workflow for indexing and retrieval such that only minimum human interaction is required. Accordingly, only polygon soups are required as inputs which do not need to be manually repaired or structured. Human effort is only needed for offline groundtruth generation to enable supervised learning and for providing information about the orientation of building models and the unit of measurement used for modeling.

ACKNOWLEDGEMENTS

First of all I would like to thank my supervisor Prof. Dr. Reinhard Klein who put trust in me when assigning me to the PROBADO project in order to face the challenges arising from the domain of architectural shape retrieval. I can hardly think of anybody else who has that much of an ability to inspire people for new ideas. The many controversial yet always fruitful and salutary discussions will never be forgotten.

I am also very grateful to Prof. Dr. Tobias Schreck who kindly agreed to serve as an external reviewer.

When I joined the Bonn Computer Graphics group my office mates were Dr. Jan Meseth and Dr. Ákos Balasz, whom I want to deeply thank for their gentle welcome. I would also like to thank Ferenc Kahlesz and Dr. Gero Müller who additionally to my aforementioned office mates ensured a life besides research, especially in my first years as a PhD student. I always enjoyed the atmosphere in the research group consisting of so many nice and brilliant colleagues over all these years that I can hardly thank someone else in particular - except for Roland Ruiters, who helped me so many times by discussing all sorts of research questions not only from the field of computer graphics, but also from cosmology, nuclear physics, thermodynamics, and Matrioshka brains - thanks a lot.

Additionally, I would like to thank my co-authors, Rafael Baranowski, René Berndt, Dr. Ina Blümel, Dr. Marcin Novotni, Sebastian Ochmann, Dr. Ruwen Schnabel, Richard Vock, and Roland Wahl.

I will not forget to mention the German Research Foundation for funding most of the work that I conducted for this thesis as part of the PROBADO project under grants GZ 554975(1) Oldenburg BIB 48 OLof 01-02, INST 3299/1-1, and KL 1142/8-2. Additional thanks for funding go to the University of Bonn and to the German National Library of Science and Technology, especially to Dr. Irina Sens.

Last but not least I would like to thank my family for enabling me to pursue an academic career - thank you.

CHAPTER 1

INTRODUCTION

During the recent five decades, architectural drafting has undergone a major paradigm shift from analog to digital techniques [Fal98]. Traditionally, the process of planning a new building was built on paper-based drawings as well as on physical scale models. The first milestones of the paradigm shift were marked by the development of graphical human/machine interfaces at MIT in the mid-1960s. Building on ideas of computer graphics pioneer Ivan Sutherland that first became manifested in Sketchpad [Sut64], these computers with highly specialized interfaces already allowed simple architectural and engineering drafts in 2D and 3D. The motivation was to simplify the drafting process by minimizing the amount of repetitive drawing, by allowing to make changes to existing drafts, and by supporting geometric constraints for primitive generation (e.g. perpendicular/parallel planes). Additionally, command-driven frameworks like the Integrated Civil Engineering System [Roo65] were invented. These academic developments were quickly picked up by companies from the fields of architecture, engineering, and construction (AEC), amongst them e.g. the famous architecture and engineering firm Skidmore, Owings & Merrill LLP (SOM) with the Building Optimization Program, or General Motors with the Design Augmented by Computer (DAC-1). Although at this point 3D drafting was intensively researched at universities and used by some larger AEC companies, early commercial CAD products only allowed 2D drafting until the late 1970s and early 1980s.

With the introduction of the personal computer starting its triumphant advance in the late 1970s, sophisticated graphics hardware became more and more affordable. Consequently, this led to a mushrooming of available commercial 2D and 3D CAD software in the 1980s, including the introduction of products like AutoCAD (Autodesk), Microstation (Bentley), CATIA (Dassault Systèmes), or Allplan (Nemetschek). Although 3D modeling software was available from this point on, it took more than another decade until architectural drafting in 3D finally became the preferred method of choice over 2D digital drafting, which might have been at least partially caused by architects' customization to traditional analog 2D drafting boards.

Today, 3D drafting software following the guidelines of *Building Information Modeling* (BIM) covers the complete lifecycle of a building, starting from design drafts over design development, construction documentation, production, documentation of the current condition up to building operation [ETSL08]. Modern BIM software is sometimes referred to as 4D, 5D, 6D, or even 7D CAD, as additionally to the three geometric dimensions, parameters including schedule time, cost-related information, energy and sustainability concerns, and as-built facility management information are integrated into the drafting process [Hol11]. Nevertheless, 3D drafting and the resulting building models remain the centerpieces of AEC planning processes and can be considered as the virtual basis of modern construction industry.

Apart from the pure geometry of the building itself, 3D drafting also includes interior design aspects and shaping of the surrounding exterior environment. Especially when filing a new draft as a tender, buildings are enriched by encapsulated detailed 3D models representing furniture, greening, or functional elements like doors and windows. In contrast to the building itself, these elements are usually not modeled from scratch by the architect, but are rather taken from respective databases in order to cheapen and shorten the drafting process. Currently, there exists a large amount of freely available models, but also a lot of high-quality collections that are brought to the market by specialized firms.

1.1 Motivation

Integrating already existing 3D models representing furniture, greening, or functional elements in a new draft is an efficient means to facilitate, cheapen, and shorten the architectural design process, see [Blü13]. The main ingredients for successful reuse are efficient methods to search and browse model collections. The straightforward solution to this task would be to use manually assigned metadata to enable textual search and retrieval. However, there are several reasons that render this approach rather intractable in the long term. First, the number of free as well as commercially available 3D models is growing at an increasing rate, making manual metadata generation more and more expensive in general. Second, when looking at existing databases with manual annotations[1] it can be noted that the available metadata is often inconsistent or incomplete. Additionally, it is not precise enough to represent fine-grained architectural classification schemes like e.g. the widely used Getty Art & Architecture Thesaurus [Pet94]. Coping with this problem would require a huge number of high-salaried AEC experts to annotate the growing stockpile of models. Considering the reasons hindering

[1]See e.g. the ArchibasePlanet.com project by Daniil Placida, created in 2001, available at `http://www.archibaseplanet.com/` [last accessed on 22 January 2013]

efficient and cheap manual metadata generation it seems necessary to develop retrieval systems that require no or only minimum human preprocessing and yet provide satisfying search results. One such approach is the usage of *content-based* retrieval, i.e. a system in which retrieval is conducted based on the data contained in the document *itself* instead of manually generated metadata. For the case of 3D models, this means retrieval is relying on the object geometry and, optionally, on additional content like e.g. textures or surface materials.

Apart from complete integration of existing models into new drafts, searching building collections for inspirational as well as teaching purposes is of particular interest to the AEC community, see [Blü13]. This leads to the necessity to also be able to search and browse building models in a meaningful way. In contrast to the above mentioned context objects like furniture or functional elements there exist a huge number of possibilities how to categorize building models, e.g. according to the shape of the ground plan, 3D form characteristics, form typology/building type, or building function (for more detailed overviews we refer to [Neu05, MB97, Pet94]). Consequently, fine-grained manual annotation of building models would require even more effort than annotation of the context objects. Additionally, buildings are largely defined by the structure of their underlying floor plans in terms of the spatial arrangement of floors and rooms. Starting the preliminary design of a building with a given schedule of spaces, architects arrange rooms, floors, and their connections in graphs representing topological structures. These topologies strongly characterize buildings and express their internal organization. However, such structures can hardly be described textually at all. Therefore, there is also the necessity for a content-based retrieval system tailored to the specific properties of building models particularly incorporating means to integrate the search for topological structures.

1.2 Goals

The overall goal of this thesis is to facilitate drafting processes in the field of architecture by providing the designer tools for efficient search of 3D architectural context objects and building models, either for integration into the new draft or for inspirational purposes. This should be realized using content-based retrieval systems for 3D models that take into account the differing requirements to search and browse for context objects on the one hand and building models on the other hand. To allow an as automated as possible ingest of models into the retrieval system we aim at a framework that is able to handle largely varying quality of geometry representations, which is a challenge that always comes along when dealing with real world 3D data. The first step to reach the overall goal is to develop a robust shape descriptor for context objects that is able to address the challenges imposed by fine-grained architectural classification schemes. A wide range of global and local 3D shape descriptors has been developed and successfully tested in the past, see e.g. [JH99, KFR03, Nov03, OOFB08]. Our goal is to enhance the performance of these descriptors by additionally exploiting knowledge about the underlying AEC domain. The second step to reach the overall goal is to make 3D building models searchable according to the underlying structure of their floor plans, i.e. the topology of rooms and floors. Retrieval methods building on comparison of topological properties of 3D shapes have been presented in the past, e.g. in [HSKK01, ZTS02, EMM03b, SSGD03]. In contrast to these approaches, we aim at characterizing building models according to an automatically computed segmentation into high-level semantic entities like rooms and stories instead of low-level geometric primitives. Apart from the pure search task, such a characterization is also beneficial to a designer to get a deeper understanding of built architecture.

1.3 Contributions

The contributions of this thesis can be summarized as:

- **A new meta-descriptor for more efficient 3D object retrieval.** The *class distribution descriptor* allows to combine arbitrary local and global shape descriptors and incorporates domain specific expert knowledge.

- **A new method for component relationship aware shape retrieval.** We introduce a method for learning the distinctiveness of spatial relationships between object components.

- **A new topological descriptor for 3D building models.** We present *room connectivity graph* as a means to capture the topological arrangement of rooms and stories in a building. We show how to enrich this graph with certain attributes to enable targeted retrieval.

- **A new efficient method for graph-based retrieval using Bags-of-attributed Subgraphs.** We convert attributed graphs into a vector-based representation using embeddings of subgraphs which accelerates similarity search.

1.4 Outline

Part I of this thesis deals with efficient search for architectural context objects. We first show how the incorporation of knowledge about a certain data domain like e.g. architecture can be exploited to boost the retrieval performance of state-of-the-art shape descriptors. To this end we introduce the *class distribution descriptor* (CDD). Given an arbitrary shape descriptor originating from some object, this meta descriptor states the probabilities for the object belonging to certain classes. Domain knowledge is incorporated by estimating the conditional probabilities using a supervised learning approach. Furthermore, we show how CDDs built from different (local) shape descriptors can be combined and evaluate the improved retrieval performance both on a set of general 3D models and on a set with particularly architecture-related models. To further exploit domain-specific knowledge about architectural 3D models we make use of the fact that such man-made objects are mostly comprised of certain geometric primitives i.e. planes, cylinders, cones, spheres, and tori. We use this observation to learn which spatial relationships of certain (partial) primitives are most significant for certain object classes to further enhance the descriptiveness of the CDDs. In the end of Part I we demonstrate the versatility of CDDs by building meta descriptors from non-shape related model features, i.e. texture and textual information.

In Part II of this thesis we concentrate on retrieval of 3D building models based on their floor plans. We first introduce attributed *room connectivity graphs* as a means to characterize room and floor topology of a building. We then develop algorithms to infer RCGs from 3D building models represented as unstructured polygon clouds requiring only minimal human interaction. The suitability of RCG building representations for retrieval purposes is shown by searching building databases for attributed query graphs in terms of subgraph isomorphisms. Additionally to this approach relying on exact graph matching we develop a method for fast and fuzzy similarity computation between any two RCGs. The algorithm is based on an embedding of the RCGs into a finite vector space. By that, graph similarity determination boils down to easy and quick comparison of two vectors. Furthermore, we use this vector-based representation for floor plan classification. We evaluate the results and compare them to the performance of a human classifier.

All algorithms presented in this thesis are designed to work on static 3D models represented as unstructured polygon soups. Currently, parametric 3D CAD models that play an increasing important role in the AEC industry (see [SEL04] for an overview) are not supported explicitly. However, it is possible to derive a static instance from the parametric model and apply the developed algorithms to it. Except for the content of Chapter 4, most of the methods and algorithms described in this thesis have already been published at international conferences in the field of computer graphics, multimedia indexing, and architecture [WBK08b, WBK08a, WBK09, WK10, WOV$^+$11a]. Additionally, a more detailed description of our methods presented in [WOV$^+$11a] was published as a technical report [WOV$^+$11b].

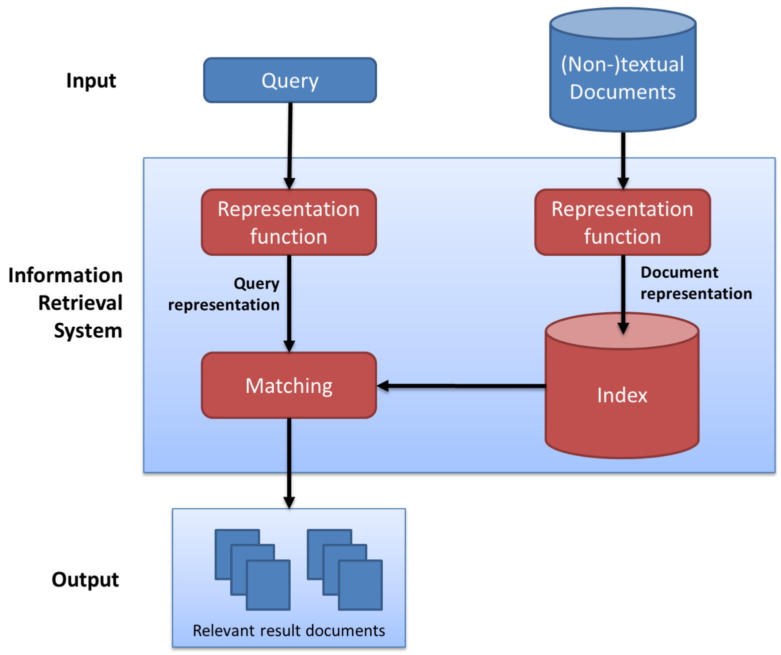


Figure 1.1: Simplified scheme of a general IR system according to [BYRN99].

1.5 Preliminaries

In this Section we will first introduce basic concepts about information retrieval and performance evaluation of information retrieval systems, followed by a brief introduction to density estimation with kernel classifiers, which is a prerequisite for both parts of this thesis.

1.5.1 3D Object Retrieval as a Special Case of Information Retrieval

Figure 1.1 shows the simplified concept of an information retrieval system (IR system) [BYRN99]. The purpose of an IR system is to make a set of documents searchable in a meaningful way as well as to provide interfaces to do so. The term *documents* thereby is not restricted to textual documents but applies to all types of data including e.g. images, audio, video, and 3D graphics. Provided the documents from the IR system's repository, certain *representation functions* are used to construct the internal *document representation*[2] from the raw data. This

[2]Depending on the context, the document representation is often denoted as (sets of) descriptor(s) or feature(s).

document representation must provide a compact characterization of the underlying document. The *index* finally subsumes all document representations. In classic text retrieval, it might e.g. consist of inverted lists [BYRN99], for image retrieval it might e.g. consist of a database of SIFT-based Bag-of-Features descriptors [Low04].

Retrieval in an IR system starts with the user formulating a query. It is important to note that the query format and the format of the indexed documents do not necessarily need to be identical. For example, if the indexed documents are images and the representation function is some sort of classifier assigning each image a textual label, then the query can easily be formulated in terms of a search string. In cases where document format and query format are in fact identical, one speaks of *query by example*[3]. Once the query is formulated, the internal *query representation* is computed. This representation can be but does not necessarily need to be identical to the *document representation*. Finally, given the query representation, the IR system searches the index for matching documents and delivers them to the user, starting with the most relevant one.

In the following we will briefly discuss the particular components of an IR system in our special case of 3D architectural object retrieval (see Figure 1.2). The documents to be indexed consist of 3D models that are represented as unstructured polygon soups. They are characterized using various global and local shape descriptors (see Section 2.2.3 for further details), and topological descriptors that describe the spatial arrangement of rooms and stories of 3D building models. The index itself consists only of a simple data structure maintaining the descriptors in memory. Queries are mainly formulated following the query by example paradigm, i.e. either a 3D model or, like in the case of search for topological substructures in buildings, a 2D floor plan sketch is provided. The type of matching procedure that is used to find document representations that are similar to the query representation depends on the descriptors involved. For vector-based descriptions like e.g. Zernike moments or the class distribution descriptor, similarity can be determined using e.g. L_2 distance or the χ^2 metric. For graph-based representations we will use approximate graph edit distances, Bags-of-attributed-Subgraphs, graph kernels, and constrained subgraph isomorphism computation.

1.5.2 Leave-one-out Tests

Shape retrieval performance is usually tested with the help of a dataset containing pre-classified models. We denote the set of model classes by $\mathcal{C} = \{C_1, ..., C_{|\mathcal{C}|}\}$, where $|\mathcal{C}|$ denotes the number of classes. Classes C_i are sets themselves contain-

[3]Note that this term does not mean the homonymous query language for relational databases.

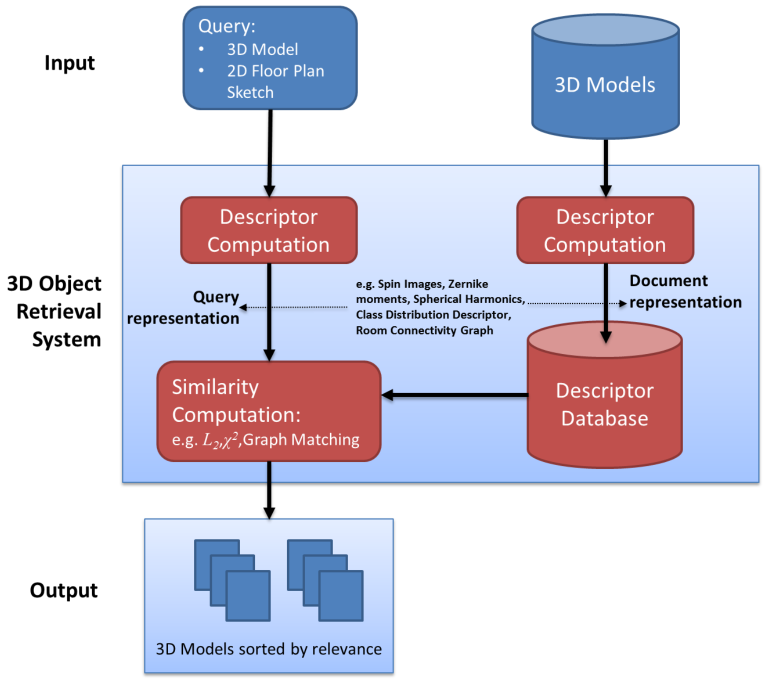

Input
Query:
3D Model
2D Floor Plan Sketch
3D Models
3D Object Retrieval System
Descriptor Computation
Descriptor Computation
Query representation
e.g. Spin Images, Zernike moments, Spherical Harmonics, Class Distribution Descriptor, Room Connectivity Graph
Document representation
Similarity Computation:
e.g. L_2,χ^2,Graph Matching
Descriptor Database
Output
3D Models sorted by relevance

Figure 1.2: Scheme of a typical 3D object retrieval system.

ing a certain number of models. The size of class C_i, i.e. the number of models it contains, is indicated by the cardinality $|C_i|$. The performance of the retrieval system is evaluated using a series of $\sum_{i=1}^{|\mathcal{C}|} |C_i|$ leave-one-out tests. For each test, exactly one model is selected to function as query object for the rest of the remaining dataset. Let $\mathcal{M} = \{m_1, \dots, m_{|\mathcal{M}|}\}$ denote the set of models in the database, let m_q denote the query object, and let $\mathcal{M}^{\setminus mq} := \{m | (m \in \mathcal{M}) \wedge (m \neq m_q)\}$ denote the set of all models in $\mathcal{M}$ except for the current query object. Then the retrieval system computes the similarity between m_q and all remaining models $m \in \mathcal{M}^{\setminus mq}$ and delivers a result list RL_{mq} in which all remaining database objects are sorted according to their similarity to the query object, starting with the most similar one. The term $RL_{mq}[i]$ denotes the i-th object in the retrieval list. For each retrieval list resulting from the leave-one-out tests, a retrieval quality measure can be computed (see below). The performance of the complete system is measured by either averaging the retrieval quality measures of each single leave-one-out test (micro averaging) or by first computing the average retrieval quality per model class and then averaging over all classes (macro averaging), see [Shi08]. If not stated differently, all retrieval performance evaluations in this thesis are quality measures averaged over all models (micro averaging).

1.5.3 Retrieval Metrics

Precision and Recall

Precision-Recall plots are common tools to efficiently visualize the performance of a retrieval systems [BYRN99]. In contrast to the retrieval metrics described below, the quality is not measured by a single scalar value but by a set of several value pairs, each consisting of a precision and a recall value[4]. To understand the measure, we first define the sets of *relevant* as well as *found* objects in a retrieval list RL_{mq} considering the k objects most similar to the query object. The set $\mathcal{F}^k_{mq}$ of *found* objects simply denotes the first k objects in the retrieval list. The set $\mathcal{R}_{mq}$ of *relevant* objects denotes those objects that belong to the same class as the query object, i.e. $\mathcal{R}_{mq} = \{m | m \in \zeta(m_q) \wedge (m \neq m_q)\}$, where $\zeta(m) \in \mathcal{C}$ denotes the class that object m belongs to. We can then define precision and recall at the k-th

[4]Note that this description of precision and recall is motivated by our application to performance evaluation of a document retrieval system. For definition and usage of precision and recall in the classification context where both are single-value metrics, we refer to [Pow11].

position of the retrieval list:

$$precision(RL_{mq}, k) = \frac{|\mathcal{F}^k_{mq} \wedge \mathcal{R}_{mq}|}{|\mathcal{F}^k_{mq}|}, \tag{1.1}$$

$$recall(RL_{mq}, k) = \frac{|\mathcal{F}^k_{mq} \wedge \mathcal{R}_{mq}|}{|\mathcal{R}_{mq}|}. \tag{1.2}$$

Recall denotes the fraction of relevant objects found when considering the first k results with respect to the overall number of relevant objects. Precision denotes the fraction of relevant objects amongst the first k results. Intuitively speaking, recall answers to the question: "How many of the relevant documents were found?" Precision answers to the question: "Besides the relevant documents, how many non-relevant documents were found?"

Precision and recall are computed for all positions k at which the retrieved object belongs to the same class as the query object. This will result in a set of $|\zeta(m_q)| - 1$ pairs of precision-recall values. Let us consider an example to get an intuitive understanding of a certain precision-recall value pair, namely $\{(0.8), (0.5)\}$: The set of retrieved documents for query object m_q that contains 50% of all documents relevant to the query, namely those belonging to class $\zeta(m_q)$, contains 80% of relevant results and 20% of non-relevant results.

To allow comparison between different retrieval lists, precision-recall values are usually interpolated at a set of certain distinct recall values, like e.g. $\{0.05, 0.10, \ldots, 1.00\}$. Precision-recall at recall values that are smaller than $\frac{1}{|\zeta(m_q)|}$ is undefined which causes the resulting plot to start always at a recall value above 0. A perfect retrieval result would lead to a precision-recall plot consisting of a horizontal line with constant precision values of 1.

Precision-recall diagrams are usually equipped with an additional plot that visualizes the hypothetical retrieval performance of a randomized retrieval system, i.e. the order in the retrieval list was determined at random. To generate this plot, several hundred randomized results are usually averaged. The random plot helps to get an impression of the complexity of the underlying retrieval problem. The more classes are involved, the worse the performance of the randomized retrieval system becomes, indicating that the retrieval problem becomes harder and harder.

Single-Value Retrieval Metrics

Although less expressive than a complete precision recall plot, single-valued retrieval metrics are nonetheless often used for evaluation of a document retrieval system. They come in especially handy if certain parameters of the system must be tuned, as the retrieval performance of different parameter settings can be easier compared automatically. Before describing the particular retrieval metrics, we first define an indicator function

$$\chi(RL_{mq}, i) = \begin{cases} 1, \text{if } RL_{mq}[i] \in \zeta(m_q), \\ 0, \text{else,} \end{cases} \tag{1.3}$$

stating whether the i-th element in the retrieval list belongs to the class of the query object.

First Nearest Neighbor (1-NN): This metric simply compares the first result in the retrieval list with the query object regarding their class memberships:

$$\Delta_{1-NN}(RL_{mq}) = \chi(RL_{mq}, 1). \tag{1.4}$$

The 1-NN performance of a retrieval system is identical to that of a one-nearest neighbor classifier that contains the remaining database objects as training examples and is used to classify the left out query object [DHS01].

Tiers: The k-Tier denotes the fraction of relevant documents that have already been found amongst the first $k \cdot (|\zeta(m_q)| - 1)$ results, which is exactly the same as the recall of the retrieval list at the $k \cdot (|\zeta(m_q)| - 1)$-th position (c.f. Equation 1.2):

$$\begin{aligned} \Delta_{k-Tier}(RL_{mq}) &= \frac{|\mathcal{R}_{mq} \wedge \{RL_{mq}[1], \ldots, RL_{mq}[k \cdot (|\zeta(m_q)| - 1)]\}|}{|\mathcal{R}_{mq}|} \\ &= \frac{|\mathcal{R}_{mq} \wedge \mathcal{F}_q^{k \cdot (|\zeta(m_q)|-1)}|}{|\mathcal{R}_{mq}|} \\ &= recall(RL_{mq}, k \cdot (|\zeta(m_q)| - 1)). \end{aligned} \tag{1.5}$$

Usually, 1-Tier and 2-Tier are used for evaluation. The 1-Tier can thereby be thought of as a measure stating how close the result is to that of a perfect retrieval system, which would require only the first $|\zeta(m_q)| - 1$ positions in the retrieval list to provide full recall.

Discounted Cumulative Gain (DCG): DCG [JK00] takes a typical human behavior into account that occurs when examining retrieval result lists. In general, only the first view results are important, results in the middle or at the end of the list are usually not examined at all. This phenomenon is most obviously revealed when thinking of results of web search engines. DCG pays attention to this behavior by *discounting* results that are farther away from the top of the retrieval list. The DCG then reads:

$$\Delta_{DCG}(RL_{mq}) = \frac{\chi(RL_{mq}, 1) + \sum_{i=2}^{|\mathcal{M}|-1} \frac{\chi(RL_{mq}, i)}{\log(i)}}{1 + \sum_{i=2}^{|\zeta(m_q)|-1} \frac{1}{\log(i)}}. \tag{1.6}$$

As can be seen, the discount for results at the end of the retrieval list is achieved by dividing by a logarithmically increasing weight. The denominator represents the highest achievable DCG for the particular class $\zeta(m_q)$ and functions as a normalizer.

For all single value retrieval metrics, $\Delta_{(.)} \in [0,1]$ holds, where 0 corresponds to the worst possible result and 1 corresponds to the best possible result. Like in the case of precision and recall, the single value retrieval performance measurements of each leave-one-out test are finally averaged. For micro averaging, the performance of the retrieval system then reads

$$\Delta_{(.)}^{micro} = \frac{1}{|\mathcal{M}|} \sum_{mq \in \mathcal{M}} \Delta_{(.)}(RL_{mq}), \tag{1.7}$$

for macro averaging, it reads

$$\Delta_{(.)}^{macro} = \frac{1}{|\mathcal{C}|} \sum_{i=1}^{|\mathcal{C}|} \frac{1}{|C_i|} \sum_{mq \in C_i} \Delta_{(.)}(RL_{mq}). \tag{1.8}$$

1.5.4 Robust Estimation of Conditional Probabilities

Estimating conditional probabilities $p(C|x)$ denotes the problem of determining the probability that given an observation[5] $x \in \mathcal{X}$ which was inferred from some document, this document belongs to class C. Computation of such densities is no easy problem. In general, there are two approaches, namely parametric and non-parametric density estimation [DHS01]. *Parametric* density estimation methods explicitly model the class-conditional probabilities $p(x|C)$ (e.g. using a Gaussian Mixture Model) and optimize parameters of the function such that they best fit the data (e.g. using Maximum Likelihood estimation). *Non-parametric* estimation denotes purely data-driven approaches to estimate the density functions, i.e. the class-conditional probabilities are not modeled.

One way to estimate the conditional probabilities is to use modified supervised kernel hyperplane classifiers. Considering binary classification (i.e. $\mathcal{C} = \{C_+, C_-\}$), the original purpose of a classifier is to find a discriminant function $g : \mathcal{X} \to \{+1, -1\}$ that predicts the belonging of a document to a certain class, given an observation x that was derived from this document:

$$g(x) = \begin{cases} +1, \text{ if } x \text{ indicates } C_+, \\ -1, \text{ else.} \end{cases} \tag{1.9}$$

[5]alternative names are *cases*, *inputs*, *instances*, or *patterns* [SS01]

Kernel hyperplane classifiers can be modified such that they not only predict hard class assignments, but provide a class probability. In the following we will briefly discuss two popular and robust hyperplane classifiers, namely Support Vector Machines (SVMs) [SS01] and Nonlinear Kernel Discriminant Analysis (NKDA) [RS99]. We will show how these classifiers can be used to robustly estimate conditional probabilities in a binary as well as a multicategory scenario.

Hyperplane Classifiers

Consider the set of empirical training data

$$\{(x_1, y_1), \cdots, (x_m, y_m) \in \mathcal{X} \times \{\pm 1\}\}, \tag{1.10}$$

where $x_i \in \mathcal{X}$ denotes the observation and y_i denotes the class label[6] of the document that x_i was derived from. Let us consider the case that the observations belong to a finite dimensional vector space, which means $\mathcal{X} = \mathbb{R}^d$. The idea of a hyperplane classifier is to learn an optimally separating hyperplane from a set of hyperplanes in a vector space $\mathcal{H}$ in which an inner product $< ., . >$ is defined, such that the decision function for an unknown observation $x \in \mathbb{R}^d$ can be expressed in terms of

$$g(x) = \text{sgn}(\langle w, x \rangle + b), \text{ where } w \in \mathcal{H}, b \in \mathbb{R}. \tag{1.11}$$

Support Vector Machines (SVMs)

SVMs belong to the category of *discriminative* classifiers in the sense that they directly model the decision boundaries. In SVM learning, the separating hyperplane is supposed to be optimally in the sense that it maximizes its distance to all training observations (maximum margin criterion, see Figure 1.3). Let us consider those observations x_+ and x_- that are closest to the hyperplane on either side

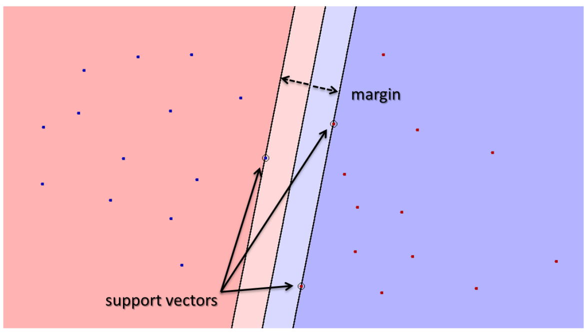


Figure 1.3: **Linear Support Vector Machine** The separating hyperplane maximizes the margin between the support vectors (encircled dots) of both classes.

[6]alternative names are *targets*, *outputs*, or *observation* [SS01]

of the decision boundary. The
hyperplane is then to satisfy the following equations:

$$\langle w, x_+ \rangle + b = +1 \text{ and } \langle w, x_- \rangle + b = -1. \tag{1.12}$$

For any observation x, we can express the distance from the origin to x along the hyperplane normal as

$$\frac{\langle w, x \rangle}{||w||}. \tag{1.13}$$

By that, the distance between x_+ and x_- along the hyperplane normal and thereby the margin of the classifier can be written as

$$\frac{\langle w, (x_+ - x_-) \rangle}{||w||} = \frac{\langle w, x_+ \rangle}{||w||} - \frac{\langle w, x_- \rangle}{||w||} \tag{1.14}$$

$$= \frac{1-b}{||w||} - \frac{-1-b}{||w||} = \frac{2}{||w||}. \tag{1.15}$$

It is obvious that maximizing the margin is equivalent to minimizing $||w||$. Additionally, the constraints imposed by the training set must hold and guarantee $g(x_i) = y_i$. This leads to the following constrained optimization problem:

$$\underset{w \in \mathcal{H}, b \in \mathbb{R}}{\text{minimize}} \frac{1}{2} \langle w, w \rangle \text{ subject to } y_i(\langle w, x_i \rangle + b) \geq 1, i = 1, \cdots, m. \tag{1.16}$$

Solving this quadratic programming problem leads to the following Lagrangian:

$$L(w, b, \alpha) = \frac{1}{2} \langle w, w \rangle - \sum_{i=1}^{m} \alpha_i (y_i(\langle x_i, w \rangle + b) - 1), \tag{1.17}$$

where α_i denote the Lagrange multipliers. Those observations x_i for which the corresponding Lagrange multiplier evaluates to a value different from zero are called *Support Vectors*, see Figure 1.3. Solving the dual optimization problem (see [SS01, Bar12] for details) finally leads to the following decision function:

$$g(x) = \text{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i \langle x, x_i \rangle + b \right). \tag{1.18}$$

Nonlinear Kernel Discriminant Analysis (NKDA)

NKDA is inspired by classical *Linear Discriminant Analysis* (LDA), which subsumes a category of *informative* linear classifier. Informative in this context means that in contrast to *discriminative* classifiers, the underlying generative statistics of the classes, i.e. the class-conditional probabilities $p(x|C)$, are explicitly modeled. Suppose the densities can be parameterized, i.e. they read $p_{\theta_j}(x|C_j)$, then parameter estimation is conducted by maximizing the (log)likelihood:

$$\begin{aligned}\hat{\theta} &= \arg\max_{\theta\in\Theta} \prod_{i=1}^{m} p(x_i|y_i) \\ &= \arg\max_{\theta\in\Theta} \sum_{i=1}^{m} \log p(x_i|y_i). \end{aligned} \tag{1.19}$$

In classical LDA, the observations are assumed to be distributed according to two different multivariate Gaussian distributions with identical covariances but differing means, i.e. $p_{\theta_j}(x|C=j) \propto \mathcal{N}(x;\mu_j,\Sigma)$. For binary classification, the decision function can be written as

$$\begin{aligned} g(x) &= \operatorname{sgn}\left[\log\left[\frac{p(C_+|x)}{p(C_-|x}\right]\right] \qquad (1.20) \\ &= \operatorname{sgn}\left[\log\left[\frac{p(x|C_+)p(C_+)/p(x)}{p(x|C_-)p(C_-)/p(x)}\right]\right] \\ &= \operatorname{sgn}\left[\log\left[\frac{p(x|C_+)}{p(x|C_-)}\right] + \log\left[\frac{p(C_+)}{p(C_-)}\right]\right]. \end{aligned}$$

Assuming that the class priors $p(C)$ are uniform and inserting the parameterized form $p_\theta(x|C)$ we arrive at

$$\begin{aligned} g(x) &= \operatorname{sgn}\left[\log p(x|C_+) - \log p(x|C_-)\right] \\ &= \operatorname{sgn}\left[\log \mathcal{N}(x;\mu_+,\Sigma) - \log \mathcal{N}(x;\mu_-,\Sigma)\right] \\ &= \operatorname{sgn}\left[\frac{1}{2}(x-\mu_-)^T\Sigma^{-1}(x-\mu_-) - \frac{1}{2}(x-\mu_+)^T\Sigma^{-1}(x-\mu_+)\right] \\ &= \operatorname{sgn}\left[x^T \underbrace{\Sigma^{-1}(\mu_+-\mu_-)}_{=:w} + \underbrace{\frac{1}{2}(\mu_-^T\Sigma^{-1}\mu_- - \mu_+^T\Sigma^{-1}\mu_+)}_{=:b}\right] \\ &= \operatorname{sgn}\left[x^Tw+b\right] = \operatorname{sgn}\left[\langle w,x\rangle + b\right]. \end{aligned} \tag{1.21}$$

Note that this is exactly the form of the hyperplane classifier decision function introduced in 1.11. A common way to determine a separating hyperplane is to force the transformed training observations to be as close as possible to the training labels:

$$\{\hat{w},\hat{b}\} = \arg\max_{w\in\mathcal{H},b\in\mathbb{R}} \sum_{i=1}^{m} ||y_i - (\langle x_i,w\rangle + b)||^2. \tag{1.22}$$

For the sake of simplicity, we will integrate the offset b into the linear mapping by using homogenous coordinates, i.e. $x'_i = (x_{i1}, \cdots, x_{id}, 1)^T$ and $w' = (w_1, \cdots, w_d, b)^T$. By defining $X := (x'_1, \cdots, x'_m)^T$ and $y := (y_1, \cdots, y_m)^T$, Equation 1.22 can then be reformulated as

$$\begin{aligned} \hat{w}' &= \underset{w \in \mathcal{H}, b \in \mathbb{R}}{\arg\max} \sum_{i=1}^{m} ||y_i - \langle x_i, w' \rangle ||^2 && (1.23) \\ &= \underset{w \in \mathcal{H}, b \in \mathbb{R}}{\arg\max} ||y - Xw'||^2 && (1.24) \end{aligned}$$

The optimizing weight vector can be written as a linear combination of the training observations (see [RS99] for further details), i.e. $w' = \sum_{i=1}^{m} x'_i \alpha_i = X\alpha$. Equation1.23 then reads

$$\hat{w}' = \underset{w \in \mathcal{H}, b \in \mathbb{R}}{\arg\max} ||y - XX^T\alpha||^2. \qquad (1.25)$$

The Gram or kernel matrix $K := XX^T$ can now be redefined according to $K_{ij} = \left\langle x'_i, x'_j \right\rangle$. The optimal weights $\hat{\alpha}$ can be determined by computing $\hat{\alpha} = K^{-1}y$. The decision function finally reads

$$g(x) = \text{sgn} \left[\sum_{i=1}^{m} \langle x', x'_i \rangle \hat{\alpha}_i \right], \qquad (1.26)$$

where $x' \in \mathbb{R}^{d+1}$ denotes an unknown query observation $x \in \mathbb{R}^d$ converted to homogenous coordinates.

The Kernel Trick

Equations 1.18 and 1.26 indicate that SVM as well as NKDA decision functions are formulated in terms of dot products, or positive semidefinite *kernels*. Let us consider a case in which the training observations cannot be separated well by the described hyperplane classifiers (Figure 1.5a). Although there is no optimal linear decision boundary in low-dimensional space, there might exist a mapping $\Phi : x \rightarrow \Phi(x)$ into a high-dimensional (or even infinite) feature space in which the observations become linearly separable, see Figure 1.5b. In the depicted example, the mapping reads $\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3 : (x_x, x_y) \rightarrow (x_x^2, x_y^2, x_x x_y)$. While in 2D, the observations require a circle-shaped decision boundary for optimal separation, the transformed observations in 3D only need a linear one. However, in general such a mapping is not known and cannot be computed explicitely. The idea behind the kernel trick is that this actual mapping does not need to be computed explicitly, but only the dot products it induces, i.e. $\langle \Phi(x), \Phi(x') \rangle$. For any algorithm that is

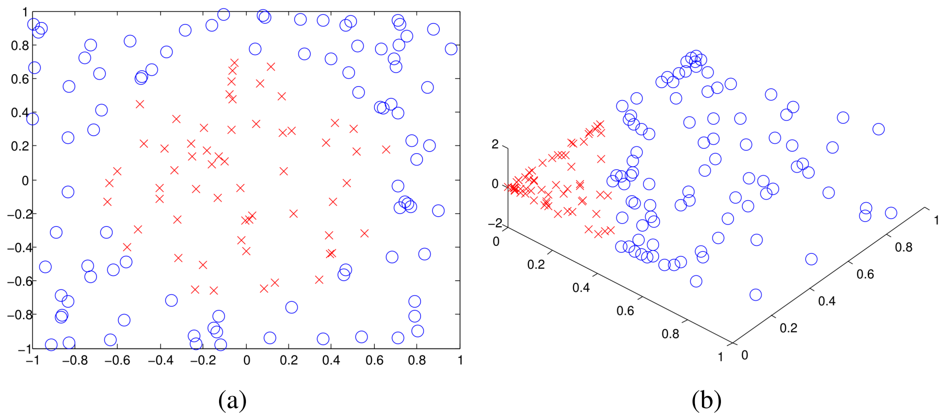


Figure 1.4: **Observation mapping to higher dimensional feature space to enable linear separation.** a) Original 2D observations. The two classes are not linearly separable. b) Observations after transformation to 3D by virtue of the mapping $\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3 : (x_x, x_y) \rightarrow (x_x^2, x_y^2, x_x x_y)$. In the new feature space, the data is linearly separable. This example is taken from [SS01].

formulated in terms of positive semidefinite kernels it is possible to construct an algorithm that operates on an alternative positive semidefinite kernel. The hope is that by choosing the right kernel, the observations will become linearly separable.

Probabilistic Classifier Ouput

So far, the described classifiers are able to produce binary output $g(x) \in \{+1, -1\}$ indicating the class that observation x is assumed to stem from. In the following we will briefly describe how to infer probabilistic predictions.

SVM case For probabilistic SVM output, several methods have been proposed [Vap98, Wah99, Pla99]. We will briefly describe the latter approach by Platt et al. since it is widely used and provides good results with respect to concurring methods. Platt et al. first define a continuous function $g'(x)$ by dropping the signum operator in Equation 1.18:

$$g'(x) = \sum_{i=1}^{m} y_i \alpha_i \langle x, x_i \rangle + b. \tag{1.27}$$

Platt's idea for computing the class probabilities is to analyze SVM outputs on real-world datasets. To this end he approximates the conditional decision function probability $p(g'|C)$ by histograms. His findings are that on either wrong side of the margin (i.e. the histogram entries corresponding to $p(g'|C_+) \leq -1$ and $p(g'|C_-) \geq +1$), the distribution of $g'(.)$ is that of an exponential and can therefore be modeled according to

$$p(g'(x)|C_\pm) \propto \lambda_\pm \exp(-\lambda_\pm(1 - g'(x))). \tag{1.28}$$

With this model at hand the conditional probabilities can be computed by

$$p(C_\pm|x) = \frac{1}{1 + \exp(\pm(A \cdot g'(x) + B))}, \tag{1.29}$$

where $A = -(\lambda_+ + \lambda_-)$ and $B = \lambda_+ - \lambda_- + \log \frac{p(C_-)}{p(C_+}$. Parameters A and B are fit using maximum likelihood estimation from the training data, for further details we refer to the original paper.

NKDA case For NKDA, the probability can be easily derived by taking a look at the definition of the LDA decision function (Equation 1.20) as a likelihood ratio:

$$g(x) \quad = \quad \text{sgn}\left[\log\left[\frac{p(C_+|x)}{p(C_-|x)}\right]\right]. \tag{1.30}$$

Analogously to the SVM case, we drop the signum operator and arrive at a continuous function $g'(x)$ from which the probabilistic predictions can be easily deduced after a few rearrangements:

$$\begin{aligned} g'(x) &= \log\left[\frac{p(C_+|x)}{p(C_-|x}\right] && (1.31)\\ \Leftrightarrow \exp(g'(x)) &= \frac{p(C_+|x)}{p(C_-|x)} = \frac{1 - p(C_-|x)}{p(C_-|x)} = \frac{1}{p(C_-|x)} - 1 && (1.32)\\ \Leftrightarrow p(C_-|x) &= \frac{1}{1 + \exp(g'(x))}. && (1.33) \end{aligned}$$

Analogously, $p(C_+|x) = \frac{1}{1+\exp(-g'(x))}$ holds.

Multicategory Classification

So far we introduced two robust methods for (probabilistic) classification that are applicable to the binary two-class scenario. The overall strategy to extent classification algorithms to the multicategory case is to divide the problem into several binary subproblems. There are two main approaches [DHS01]:

- **One-versus-all method** For each class, one model is trained, resulting in $|\mathcal{C}|$ classifiers in total. Positive training data are the observations from the class itself, negative training data consists of the observation from all other classes. For classification, the *winner-takes-it-all* strategy is used, i.e. the unknown observation x is assigned the label of the classifier that provided the highest value $g'(x)$.

- **One-versus-one method** For each pair of classes, one model is trained, resulting in $|\mathcal{C}| \cdot (|\mathcal{C}| - 1)/2$ classifiers in total. Positive and negative training data is provided by the two respective classes. For classification, the *max-wins voting* is used, i.e. the unknown observation x is assigned the label of the class that was predicted by most classifiers.

It is sometimes argued that the one-versus-one method might lead to better results, as binary subproblems intuitively seem to be easier solvable than those in the one-versus-all setting, see e.g. [RT01]. While this might be true for relatively weak classifiers like classical LDA, there is no empirical evidence which approach works better if non-linear SVMs are used as classifiers [bDK05]. Nevertheless, depending on the classifiers in use and the amount of data involved, the one-versus-one method might be advantageous during training. Although there are more classifiers to train ($\mathcal{O}(|\mathcal{C}|^2)$ instead of $\mathcal{O}(|\mathcal{C}|)$), the number of negative examples in each training step can be tremendously smaller than in the one-versus-all method, which might render a large classification problem computable in the first place.

Multicategory Probabilistic Classification

The goal is to determine a discrete distribution over the conditional probabilities, i.e. one wants to compute $p(C_1|x), \cdots, p(C_{|\mathcal{C}|}|x)$. Analogously to the two approaches for hard category assignment, there are two according methods for probabilistic classification.

- **One-versus-all method** Decision functions $g'(.)$ are evaluated as described above. For each class, the according probability $p(C_i|x)$ is computed. Finally, all probabilities are normalized to sum up to 1.

- **Pairwise coupling** For each pair of classes $\{C_+, C_-\}$, conditional probabilities $p(C = C_+|x, C \in \{C_+, C_-\})$ and $p(C = C_-|x, C \in \{C_+, C_-\})$ are computed. In an iterative scheme, these probabilities are combined to finally arrive at $p(C = C_i|x, C \in \mathcal{C})$.

While the first approach is straightforward, the second needs further explanation. We thereby follow the description by Hastie and Tibshirani from their according paper [HT98]. The observations of the conditional probability are first abbreviated by $r_{ij} := p(C = C_i|x, C \in \{C_i, C_j\})$. Then they define

$$\mu_{ij} := \frac{p(C = C_i|x, C \in \mathcal{C})}{p(C = C_i|x, C \in \mathcal{C}) + p(C = C_j|x, C \in \mathcal{C})}. \tag{1.34}$$

The idea behind pairwise coupling is now to estimate the sought-after probabilities $\hat{p}(C = C_i|x, C \in \mathcal{C})$ such that the resulting $\hat{\mu}_{ij}$ are close to the observed conditional probability r_{ij}. The similarity between $\hat{\mu}_{ij}$ and r_{ij} is determined by the Kullback Leibler divergence [CT06], which is often thought of as a distance between distributions:

$$\mathcal{D}^{\mathcal{KL}} = \sum_{i<j} r_{ij} \log \frac{r_{ij}}{\hat{\mu}_{ij}} + \sum_{i<j} (1 - r_{ij}) \log \frac{1 - r_{ij}}{1 - \hat{\mu}_{ij}}. \tag{1.35}$$

Optimal probabilities that minimize the Kullback Leibler divergence can be found by using the following iterative scheme:

Algorithm 1 Pairwise Coupling

```
1: Input Conditional probability observations $r_{ij}$.
2: Output Conditional probability estimates $\hat{p}(C = C_i|x, C \in \mathcal{C})$.
3: Initialize $\hat{p}(C = C_i|x, C \in \mathcal{C}) = \frac{1}{|\mathcal{C}|}$.
4: while $\mathcal{D}^{\mathcal{KL}}$ not small enough do
5:     Compute $\mu_{ij} = \frac{p(C=C_i|x,C\in\mathcal{C})}{p(C=C_i|x,C\in\mathcal{C})+p(C=C_j|x,C\in\mathcal{C})}$.
6:     Update $\hat{p}(C = C_i|x, C \in \mathcal{C}) \leftarrow \hat{p}(C = C_i|x, C \in \mathcal{C}) \cdot \frac{\sum_{i\neq j} r_{ij}}{\sum_{i\neq j} \hat{\mu}_{ij}}$
7:     Renormalize $\hat{p}(C = C_i|x, C \in \mathcal{C}) \leftarrow \frac{\hat{p}(C=C_i|x,C\in\mathcal{C})}{\sum_{i=1}^{|\mathcal{C}|} \hat{p}(C=C_i|x,C\in\mathcal{C})}$
8: end while
```

Part I

Feature-based Shape Retrieval for 3D Architectural Context Models

CHAPTER 2

LEARNING DISTINCTIVE LOCAL OBJECT CHARACTERISTICS

2.1 Introduction

Most of the methods that were developed in the early days of 3D shape retrieval are based on *global* shape representations like spherical harmonics descriptors [SMKF04], Zernike moments descriptors [Nov03], or view-based descriptors [CTSO03, Vra04, MD06]. Global in this context means that one single descriptor characterizes an entire 3D model. Global descriptors that are encoded as vectors allow fast and easy object comparison by computing some distance between the associated vectors. As the affiliation of many 3D objects to a certain category is not solely defined by global shape attributes, *local* features were introduced. In contrast to global ones, local descriptors only characterize a part of the underlying object's geometry. The results presented in [MGGP06, GCO06], and especially in [OOFB08] indicate that local features are able to further enhance the retrieval performance. Additionally, to bridge the semantic gap between low level geometric descriptors and high level user intended object categories, supervised learning approaches have been successfully used to improve global and local feature based approaches [HLR05, SF06, FS06, ASYS08].

Summarizing the results of recent work on 3D shape retrieval, it has become obvious that *"...no single descriptor is capable of providing fine grain discrimination required by prospective 3D search engines"*[ASYS08], which especially holds for detailed architectural object classification schemes. It therefore seems promising to investigate how different types of descriptors can be combined to boost the performance, additionally to the use of local features and the incorporation of supervised learning methods.

Regarding approaches relying on local features, there are mainly two different ways to measure the similarity of two sets of local descriptors. The first approach relies on determining a mapping between the descriptors of two objects. Object similarity is then computed in terms of compatibility of corresponding descriptors, or in terms of the distortion caused by a geometric transformation that is induced by the descriptor correspondences [NDK05, FS06]. The second approach uses histograms that approximate the global distribution of local descriptor or other local properties of an object for comparison [LZQ06, OOFB08]. Both of these methods face drawbacks. Establishing feature correspondences usually involves the exhaustive use of non-trivially to determine thresholds on descriptor similarity, position, orientation and spatial arrangement. Additionally, it can become quite time-consuming due to combinatorial reasons. Histogram-based approaches, although allowing easy comparison of two objects using e.g. the Kullback-Leibler divergence, suffer from the drawback that certain highly discriminating local features might have a relative low impact due to the descriptor agglomeration in the histogram. Additionally, the spatial relationship between local features can not be expressed appropriately.

Another problem arising from the use of local features is the impact of scale. A priori it is not obvious *how local* a feature descriptor should be in order to be most distinctive. Common approaches either rely on the usage of a single fixed scale [MGGP06], a combination of several fixed scales [SF06, FS06], or on the detection of a built-in feature scale [GCO06, OOFB08]. Most of these approaches face the problem that accidentally choosing a less distinctive scale might lead to decreased retrieval performance.

To overcome these drawbacks we introduce the new *class distribution descriptor* (CDD). The CDD is a meta-descriptor that transforms geometric features like spherical harmonics [SMKF04], Zernike moments [Nov03], or spin-images [Joh97] into a representation stating how strongly the feature indicates certain object classes. To this end we will use the robust conditional probability estimation described in 1.5.4. The resulting CDD is uncoupled from the geometric feature it was inferred from: First, it is independent of the feature type, and second it is rather independent of the actual local feature geometry. Both of these properties render CDDs appropriate for easy combination and comparison of different local features without the need to take care of position and spatial arrangement.

Note that in contrast to other approaches (e.g. [FS06, LN07]) our method is not restricted to similarity measurements between unknown query objects and the objects contained in a training database, but it also allows comparison of two unknown query objects. Additionally, our CDD combination scheme enables the usage of multiple local feature scales and at the same time solves the above mentioned problems arising from less distinctive scales. In our experiments using different types of local shape features we show that our method is superior to common 3D shape retrieval approaches.

Summarizing the key contributions of this chapter, they are:

- A supervised learning approach allowing the easy use of arbitrary features for 3D shape retrieval by avoiding the problem of generating feature correspondences
- Combination of arbitrary features of different scales with no drawbacks caused by accidentally chosen less distinctive scales
- A new benchmark containing architectural 3D context objects
- Experimental evaluation of our approach on a standard 3D model benchmark [SMKF04] as well as our new architecture benchmark

2.2 Related Work

In the following we will give an overview of the related work on 3D shape retrieval. There are several different schemes for grouping the approaches, see e.g. [BKS$^+$05, TV08]. The domain a descriptor operates on, e.g. the boundary surface or interior properties, the preservation degree, i.e. how exactly the original model can be reconstructed from the descriptor, and the type of represented information, i.e. for example geometry or material properties are among the grouping criterions. Additionally, one usually distinguishes *global* and *local* descriptors. While global descriptors characterize the complete shape of an object, local descriptors only represent a part of it. Most shape descriptors can be used in a global as well as in a local manner. Depending on whether global or local descriptors are used, determining the similarity between two objects poses quite a different challenge. Before summarizing the most important shape descriptors we will briefly describe methods for descriptor comparison. We will thereby only discuss comparison of vector-valued descriptors. For an overview on graph-based approaches we refer to Part II of this thesis.

2.2.1 Comparing Global Shape Descriptors

A crucial ingredient for efficient comparison of global shape descriptors are certain invariance properties towards geometric transformations, which means for example that similarity evaluation between two shapes should provide the same result regardless of the objects' current position in space with respect to translation and rotation. We will briefly list the most important invariance properties and describe how they can be achieved. For a more detailed introduction, we refer to [FK04].

- **Invariance under Translation** The object's center of gravity is usually used as a reference point for descriptor computation, e.g. for subsequent volume discretization or as target point of the optical axis for view-based descriptors. The center of gravity is relatively robust towards small changes in the object's geometry.

- **Invariance under Rotation** There are two basic ways to assure invariance under rotation. One possibility is to transform the object into a somewhat canonical orientation before descriptor computation. This method was especially popular in the early years of research on 3D shape retrieval. The most common method is to align the object along its principal axes. However, this method has two drawbacks. First, the principal axes are brittle regarding even small changes in an object's geometry. Second, the result is not unique; taking into account flips along the planes spanned by the principal axes there are 2^3 possible orientations, which calls for additional heuristics for exact pinpointing (see e.g. [KPNK03]). The other possibility for achieving rotational invariance is not to manipulate the object itself but to instead construct the descriptor in a way such that it becomes rotation invariant. A popular way is to describe the shape by a function developed in a Fourier basis. By carefully designing the descriptor in a way that a rotation of the object would result in a time shift within the Fourier representation, one can achieve invariance under rotation by dropping the phase information in the descriptor.

- **Invariance under Scaling** The simplest way to achieve this invariance is to isotropically scale the object such that its bounding box touches the unit cube. However, this method is not very robust, as even a tiny bit of additional geometry can dramatically change the bounding box. A more efficient way is to scale the object such that the average distance of all points to the center of gravity is constant for all objects.

- **Invariance under Pose Variation** This property is especially interesting for natural objects like human beings or animals. A common way to achieve this invariance is to describe it in terms of intrinsic surface properties like geodesic distances, as they are more stable under pose changes than properties measured with Euclidean geometry.

Invariance of a vector-valued descriptor ensures that each of its entries is comparable to the according entry in another descriptor. Accordingly, comparing two objects with respect to the underlying global descriptors usually consists of evaluating some metric between the corresponding vectors, including e.g. L_1, L_2, or χ^2-distance. A more sophisticated approach that is feasible in the case of histogram descriptors is the earth mover's distance [RTG98]. It does not only incorporate comparison between corresponding descriptor entries (i.e. the i-th entry in the first descriptor is compared to the i-th entry in the second descriptor) but it rather computes the cost for an optimal rearrangement of the entries in one descriptor such that they would best match those of the other one.

2.2.2 Comparing Local Shape Descriptors

The comparison of two objects that are characterized by sets of local features is a far more challenging task than in the case of global descriptors. When presented only two global descriptors, it is easy to see that it is exactly these two descriptors that must be compared. In the case of local descriptors it is not clear at first glance how the comparison must be conducted. In the following we will describe the two main approaches for tackling this problem

Establishing feature correspondences The idea behind this approach is to determine a mapping between two sets of local descriptors that takes descriptor similarity as well as spatial relationships into account. Depending on the amount of distortion that is induced by the mapping, the similarity between the two underlying objects is determined. Körtgen et al. [KPNK03] try to find optimal correspondences by minimizing a rather complex energy function that amongst others contains terms for descriptor similarity and descriptor position. This leads to a weighted bipartite graph matching problem that is solved using the Hungarian algorithm [Mun57]. Similar approaches are introduced in [NDK05] and [WNK06], where object similarity is defined in terms of a thin-plate spline bending energy induced by previously determined pairwise feature correspondences. Funkhouser et al. [FS06] use a heuristic similarity measure involving spherical harmonics (SH) descriptor distances and similarity of spatial relationships. Focusing on recognition of small vehicles in point clouds from laser range scans, RANSAC based approach for the detection of small compatible feature sets are presented in [SMS$^+$04] and [JH99].

Methods based on geometric hashing [LW88] are extremely popular in computer vision but have also been applied to 3D shape retrieval [GCO06]. Although this approach takes spatial relationships of features into account, it faces two major drawbacks. First, the memory consumption for storing the hash tables is rather high. Second, the degree of discretization of the transformation space and the Euclidean space at which high quality retrieval results can be achieved is rather hard to determine. Despite their ability to include spatial relationships of local features into the object similarity measure, the described methods require to manually define a lot of pruning thresholds on descriptor similarity and spatial consistency, rendering it hard to achieve good generalization results.

Histograms Methods based on histograms either approximate distributions of local shape descriptors or they approximate distributions of relationships between local shape descriptors.

The first approach is commonly known as the Bag-of-Features (BoF) paradigm. BoF-based methods have recently gained increasing attention in the 3D shape retrieval community [LZQ06, LGW08, OBBG09, BBGO11]. The idea behind this approach is inspired by the common Bag-of-Words approach [Har54] which is used for text retrieval and classification. First, a codebook of local features is selected with respect to a set of training objects. New objects are then characterized by describing their local feature occurrences with respect to the before established codebook. By that, local features are mapped into a single histogram, allowing for easy comparison of two 3D objects. BoF-based descriptors are invariant under rotation by construction[1] as they lack the ability to represent positional information of local features and their according descriptors as well as the spatial relationship between several features. Loosely speaking, the resulting histogram only states how often a certain descriptor appears in an object with respect to the total number of descriptors. In [LGW08], Li et al. try to alleviate this shortcoming by additionally taking the distance between the object center and the local feature into account. However, the exact spatial relationship between tuples of features cannot be represented appropriately by a BoF approach[2].

In contrast to the BoF method, the goal of the second approach is instead of approximating the descriptor distribution directly, to rather characterize the relationship between any pair of descriptors in terms of similarity and dissimilarity, respectively [PRM$^+$00, OFCD02, ILSR02, IRSS03, OMT03]. A very basic example of this type of method, the D2-descriptor, was introduced in [AKKS99]. In this approach the "local descriptors" simply consist of points randomly sampled from the object's surface. A histogram is built that approximates the distribution of the pairwise Euclidean distances between all points. Like in the BoF approach, this second type histogram descriptor is invariant under rotation.

2.2.3 An Overview on Shape Descriptors

In the following we will briefly describe the most important types of shape descriptors. Especially for the older methods we mostly summarize the vast descriptions that can be found in [BKS$^+$05] and [TV08]. We loosely follow the grouping scheme in [BKS$^+$05].

[1]Assumed that the underlying local descriptors are rotational invariant

[2]Nevertheless it is possible to concatenate several descriptors to a single one, add the according information about spatial relationships, and build BoFs from these combined descriptors, see e.g. [OB07]. However, regarding the exponential nature of the underlying combinatorial problem it is doubtful that the resulting statistics can be expressed by a histogram appropriately.

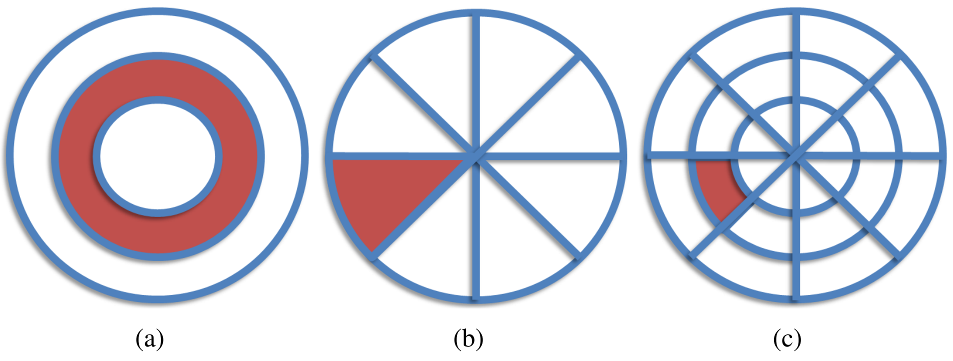


Figure 2.1: **Volume discretization.** a) Shell bins. b) Sector bins. c) Combined shell-sector bins. This example is taken from [BKS$^+$05].

Simple Statistical Descriptors Statistical descriptors try to capture probability distributions of global and local properties of 3D objects. In contrast to the surface descriptors described further below, local surface properties of the object like normals or curvature are not incorporated directly. Paquet et al. [PRM$^+$00] propose object volume to bounding box volume ratios as well as the distance between the bounding box center and the center of mass. The resulting simple descriptor is invariant under translation. Corney et al. [CRC$^+$02] use the deviation of the object's actual surface and its convex hull to characterize an object. *Shape distributions* have been proposed by several researchers, including [PRM$^+$00, OFCD02, ILSR02, IRSS03, OMT03]. The common idea is to construct histograms of local and global geometric properties including the angle distribution of surface point triples, the distribution of distances between random surface point pairs, or the area distribution of triangles spanned by random surface point triples. Another way to compare distributions of geometric properties is to characterize them according to their moments instead of histograms, see e.g. [PRM$^+$00, ZC01, SV01, ETA02].

Volume-Based Descriptors The idea behind volume-based descriptors is to first discretize the volume an object is located in. For each resulting bin, a function is evaluated that locally describes the object's surface (or in some cases the volume). Ankerst et al. [AKKS99] propose to divide the volume into shells, sectors, or a combination of both, see Figure 2.1. The bin models are aligned around the object's center of gravity. A number of points are randomly sampled from the object surface. It is then counted how many points fall into each bin. By that, the shell model (Figure 2.1a) results in a rotation invariant descriptor while the other two variants require pose normalization first. Suzuki et al. [SKO00] propose a regular grid-like volume subdivision instead. By grouping certain bins, rotation invariance at 90 degree intervals is achieved. A similar approach is described in [VS01]. By applying a Fourier transformation to the descriptor, high frequencies can be pruned leading to increased robustness towards small changes in the geometry. Ohbuchi et al. [OOIT02] slice the object into a fixed number of pieces along each principal axis and compute statistics on sampled surface points per slice including the moment of inertia, average distance to the particular axis and the according variance.

There are several approaches that do not characterize the surface of an object but rather the enclosed volume. In most cases, such descriptors can only be applied to a model whose surface is closed and orientable by construction or is appropriately converted by using a suitable surface reconstruction technique. Keim [Kei99] and Novotni et al. [NK01] propose to measure object similarity in terms of overlapping object volumes. Heczko et al. [HKSV02] introduce a method that does not require a closed surface. The overall idea is to construct tetrahedrons formed by each surface triangle and the object's center of mass. The resulting tetrahedrons are then intersected with a volume subdivision structure like the one in Figure 2.1 and an according shape descriptor is constructed.

While most of the above presented volume-based descriptors require pose normalization of the object to achieve rotation invariance, spherical harmonics (SH) descriptors as introduced by Funkhouser et al. [FMK+03] provide this feature intrinsically. The starting point is the combined volume subdivision scheme as depicted in Figure 2.1c. While keeping the strict separation of the shells, the separation between sectors in one shell is relaxed. Instead of considering each bin isolated, the surface point counts in each bin are considered as nodes of a continuous spherical function located on the particular shell. The function is fitted using a spherical harmonics basis. By dropping the phase information of the resulting complex SH coefficients one arrives at one rotation invariant descriptor per shell. A scale-invariant version of the local SH descriptor was presented in [NDK05]. The idea of considering continuous functions instead of isolated bins is followed to its logical conclusion by Novotni et al. in their work about Zernike moments descriptors [Nov03]. Instead of only using orthonormal Legendre polynomials and Fourier basis functions to characterize each shell separately, Novotni adds the Zernike polynomial as a third basis function to continuously characterize the volume across shell boundaries. A similar approach relying on Krawtchouk instead of Zernike moments is presented by Mademlis et al. in [MAD+06].

Symmetry-based descriptors are presented in [KCD+03], [PSG+06], and [Rus07a]. The idea is to parameterize an object according to its (local) degree of symmetry. To this end, the desired space of symmetric transformations (e.g. all planar reflections) is discretized. Additionally, a voxelized version of the object is created. By documenting the object's degree of symmetry with respect to each of these transformations, a descriptor is computed which can be used to determine object similarity.

Surface geometry descriptors The idea behind these descriptors is to locally characterize the object's surface at (randomly) selected locations. Paquet et al. [PR00] compute histograms over the angle distribution between each of the first two principal axes of a face and the according face normal. The histogram over the normal angle distribution is the starting point for descriptor development in [IW02] and [WCI04]. In [ZP01], the local principal curvatures are approximated via quadric fitting and subsequently mapped onto a single value called shape index. The resulting descriptor is denoted as shape spectrum and is inferred by computing a shape index histogram. In [ZP02], the space of planes intersecting the model is first discretized. For each triangle it is computed how much it contributes to any of the intersection planes in terms of area overlap and angle consistency. A non-rotation invariant descriptor is constructed from the aggregated contributions.

The idea behind *spin images* [Joh97] is to efficiently characterize local surface geometry. For some surface point, the associated spin image is oriented along the surface normal. The surrounding local space is discretized using cylindrical shells centered on the normal and slicing planes that are perpendicular to the normal. The surface point count inside each of the resulting bins is evaluated. By construction, spin images are rotation invariant around the normal. They have been used as local descriptors in numerous retrieval efforts, including e.g. [ADBP04, LZQ06, MGGP06]. Körtgen et al. [KPNK03] introduce a straightforward extension of 2D shape contexts [BMP02] to three dimensions. For a number of surface points, a histogram is computed that characterizes the angle and distance distribution to points in the local neighborhood. In [GCO06], geometrically non-trivial parts of the mesh are represented by local descriptors incorporating area and curvature. The descriptors are by construction scale invariant. Additionally, in contrast to most other descriptors, the area of description is not restricted to a spherical support.

Although providing certain invariance properties, none of the surface descriptors presented so far is invariant towards isometric transformation. To alleviate this drawback, several approaches have been developed that rely on intrinsic surface properties which do not change under isometric transformation, e.g. geodesic distances. Accordingly, many methods use the Laplace-Beltrami operator (LBO) and its approximations on a triangle mesh as a starting point. Reuter et al. [RWP06] introduce *Shape-DNA*, a descriptor consisting of the first k eigenvalues of the mesh's Laplace-Beltrami spectrum. Rustamov picks up this approach and presents global point signatures that consist of each vertex' corresponding row in the Laplace-Beltrami spectrum scaled by the square roots of the corresponding eigenvalues

[Rus07b]. A global descriptor is computed by constructing a histogram of the pairwise distances of all global point signatures. These methods also triggered the development of spectral methods that additionally incorporate the object's interior, see e.g. [RWSN09, RLF09, Rus10]. However, these approaches are not pose invariant any longer, since they depend on the embedding of the surface in the Euclidean space. Descriptors based on heat kernels are closely related to the LBO-based ones, as computing the heat kernel on a mesh actually boils down to little more than evaluating the LBO. In contrast to the global point signatures by Rustamov [Rus07b], multiscale *Heat Kernel Signatures* introduced by Sun et al. allow for a certain localization of the descriptor's support, depending on how much time has passed since the beginning of the heat diffusion process. Improvements regarding scale invariance and applications of this approach have been presented in [OBBG09, BK10, DLL$^+$10, OMMG10, BBGO11]. Note that all descriptors relying on an approximation of the LBO require a more or less intact mesh in terms of connectivity, as otherwise the necessary computation of geodesics would be faulty. For an excellent overview including additional descriptors for non-rigid 3D object retrieval as well as an exhaustive comparison of state-of-the-art methods we refer to [LGB$^+$12].

View-based descriptors The basic idea of view-based descriptors is to transform one instance of the 3D similarity problem into several instances of 2D similarity problems. Heczko et al. [HKSV02] compute orthogonal projections along each principal axis of the model. From the resulting silhouettes, a constant number of points are sampled. The resulting distances from its center to the sampled points are stored in a vector. By applying a Fourier transformation and subsequently dropping the phase information, a rotation invariant descriptor is constructed. Song et al. [SG02] render the model from different viewpoints. The resulting 2D silhouettes are characterized by the degree of circularity, 2D Fourier descriptors, moment invariants, and curvature scale space. For any two rendering viewpoints, the similarity of the according representations is computed. The whole model is then represented and compared by a histogram built from these distances. Alternatively, 2D Zernike moments, curvature histograms, and color information have been proposed to characterize 2D views [Löf00, AVMD04]. Instead of only three projections along the principal axes, the light field descriptor introduced by Chen et al. [CTSO03] is composed of all ten unique 2D views that result from renderings from the corners of a dodecahedron looking through the object's center. Each silhouette is represented by according Zernike moments and 2D Fourier descriptors. To compare two objects, heuristics are used to find the optimal rotation that minimizes the induced descriptor distances. An improvement of this method is introduced in [MD06]. In this work, the optimal rotation between the shapes is found using fast Fourier transformation, an approach similar to those presented in [WNK06] and [Kaz07] for optimal alignment of SH descriptors and the underlying model, respectively. Depth buffer descriptors are introduced in [HKSV02] and [Vra04]. The model is again rendered from different viewpoints, but instead of a flat image, the resulting content of the depth buffer is used to characterize the particular view using a 2D Fourier descriptor. Depth buffer images are also the starting point for Ohbuchi's method [OOFB08], who uses them to locate SIFT descriptors and subsequently constructs one global BoF descriptor. A similar approach is presented in [LGS10], where one BoF descriptor per view is computed and object similarity is defined in terms of the best matching descriptor pair.

View-based shape descriptors also play an important role in sketch-based 3D shape retrieval which has recently regained attention, see e.g. [YSSK10, ERB+12]. For an overview and evaluation of state-of-the-art methods from this domain we refer to [LSG+12]. For a more thorough introduction to view-based 3D shape retrieval methods in general we refer to the recently published exhaustive survey by Liu [Liu12].

2.2.4 Supervised Learning in Shape Retrieval

In [NPWK05], Novotni et al. use a Support Vector Machine (SVM) and Kernel-based methods to implement relevance feedback for 3D shape retrieval using global Zernike descriptors. In an iterative process, the user selects relevant and non-relevant retrieved objects and is afterwards presented results that are improved by the learning process. Relevance feedback is also addressed in [ASYS08], where Akgül et al. present a linear score fusion approach. In the training step they use an SVM in order to learn how to optimally combine similarity values between two objects arising from different global shape descriptors. Hou et al. [HLR05, HR07] also make use of SVMs. In the training step, they learn conditional object class probabilities given a set of global shape descriptors. For a query object, global shape descriptors are computed and the conditional class probabilities are predicted based on the learned knowledge. Training objects belonging to the most likely class are returned as retrieval results. The approach presented in [LN07] combines elements of the supervised learning method presented in [HR07] with view-based Light Field Descriptors [CTSO03]. In [SF06], the appearance likelihoods of local SH descriptors are mapped to their distinctiveness in a training step. Given a query object, distinctive descriptors are selected according to the learned mapping. The similarity between two objects is then determined by the minimum Euclidean distance between all pairs of selected descriptors. Taking into account spatial relationships between local features, Funkhouser et al. further improve this method [FS06]. Amongst a large number of randomly chosen surface features, a small set containing the most distinctive ones is determined in a learning step for a database of training objects. Given a query object, randomly located surface features are computed and correspondences to the distinctive database features are established with respect to certain spatial constraints. Note that this method as well as the above described one [LN07] is restricted to searching similar objects only amongst the set of pre-classified training objects.

2.2.5 3D Shape Benchmarks

Along with the increasing importance of 3D shape retrieval methods, a number of benchmarks differing in the field of interest the models belong to, in the object representation, or in the type of classification scheme has been developed. The well-known Princeton Shape Benchmark [SMKF04] contains objects from various fields (e.g. planes, animals, plants, buildings). The models are represented as triangle meshes without consistent normal orientation and connectivity. Its hierarchic classification schemes include coarse levels which rather include functional aspects as well as fine-grained levels corresponding to rather form-based classifications. Another benchmark containing generic model classes is presented in [FGLW08]. The included triangle meshes are pose-normalized according to their principal axes. The proposed classification mainly follows a form-based scheme, but there are also some classes that rather reflect functional aspects (e.g. `MusicalInstrument,MilitaryVehicle`). In [JKIR06], the PRECISE Engineering Shape Benchmark is presented, focusing on 3D objects related to engineering. The models are represented as triangle meshes. The classification scheme includes semantic- as well as shape-related categories. The McGill 3D Shape Benchmark [ZSMS05] was especially designed for retrieval of articulated objects, and consequently contains mostly animals and humans. Objects are represented by voxel grids as well as by triangle meshes. The classification scheme is non-hierarchic and rather related to semantics than to form.

Several other benchmarks have been developed for different tracks of the SHREC Shape Retrieval Contest as a part of the Shape Modeling International conference (from 2006 to 2008, see [VRS$^+$06, VtH07, VtH08]) and the Eurographics Workshop on 3D Object Retrieval, respectively (since 2009, see [VtH09, DSS$^+$10, LSF$^+$11, SBBF12][3]). These benchmarks include datasets focusing on miscellaneous aspects like watertight models, face models, partial models, etc. The models contained in these benchmarks are collected from publicly available 3D repositories, from other benchmarks, or were generated for this event. The

[3]Please note that SHRECs from 2009 to 2012 include a total of 20 different tracks for each of which exists an according publication. We therefore only cite the summary paper (2009) and the proceedings (2010-2012), respectively.

underlying schemes mainly comprise form-based categories.

2.3 Class Distribution Descriptors

Overview The class distribution descriptor is an abstract meta-descriptor that can be inferred from an arbitrary vector-based descriptor[4]. It states how strongly the underlying descriptor indicates the object's belonging to each element of a predefined set of classes in terms of conditional probabilities. The idea is related to the approach presented in [HR07]. In this paper, supervised learning methods are used to predict conditional object class probabilities given global descriptors computed for a query object. The pre-classified objects in the database that belong to the most likely predicted class are returned as results. In our approach, we follow a different way enabling us to take advantage of local feature descriptors. Instead of using the conditional class probabilities as a classifier, we treat them as a new meta-descriptor: Given a set of object classes $\mathcal{C} = \{C_1, ..., C_{|\mathcal{C}|}\}$, the class distribution descriptor $\mathcal{D}(x)$ originating from an arbitrary local descriptor $x \in \mathcal{X} = \mathbb{R}^d$ is the result of a transformation Φ mapping x into a space of conditional probabilities:

$$\mathcal{D}(x) = \Phi(x) = \begin{pmatrix} p(C_1|x) \\ \vdots \\ p(C_{|\mathcal{C}|}|x) \end{pmatrix}. \tag{2.1}$$

To point out the advantage of this representation over an underlying geometric descriptor like SH or Zernike moments, let us consider a 3D model of an airplane, for which two local descriptors are computed, one approximately located at the tail and one in the cockpit. Although the geometry at these locations largely differs, the two resulting CDDs will probably look similar, as both, the tail as well as the cockpit, is highly significant for an airplane. This example also demonstrates how the CDDs are uncoupled from local geometry and feature localization, rendering feature correspondence determination unnecessary.

2.3.1 Combining Class Distribution Descriptors

Given CDDs that are derived from a set of l (local) descriptors $x_1, ..., x_l$ characterizing an object, we need to construct a representation that enables easy object comparison. To this end we aim at computing one single CDD $\mathcal{D}(x_1, ..., x_l)$ that

[4]In this thesis we only consider vector-based descriptors as a starting point for CDD inference. However, provided with an appropriate kernel it would be possible to derive CDDs from structured data like graphs or strings as well.

incorporates the previously created CDDs. From a statistical point of view, this descriptor should reflect the distribution of conditional class probabilities, given the descriptors $x_1, ..., x_l$. It should therefore read

$$\mathcal{D}(x_1, ..., x_l) \quad := \quad \begin{pmatrix} p(C_1|x_1, ..., x_l) \\ \vdots \\ p(C_{|\mathcal{C}|}|x_1, ..., x_l) \end{pmatrix}.$$

Combinations of probabilistic classifier outputs have been widely discussed in literature, see e.g. [Ued00, TvBDK00, HR07]. Supposing the local features to be statistically independent and adopting the Bayesian point of view[5] leads to a widely used combination method known as *product rule*. In the following we will show how the desired descriptor $\mathcal{D}(x_1, ..., x_l)$ can be computed by from descriptors $\mathcal{D}(x_1), ..., \mathcal{D}(x_l)$ using the product rule. Let us first consider only a single class C_k. According to the Bayes theorem, we know that

$$\begin{aligned} p(C_k|x_1, ..., x_l) \cdot p(x_1, ..., x_l) &= p(C_k, x_1, ..., x_l) \qquad (2.2) \\ &= p(x_1, ..., x_l, C_k) = p(x_1, ..., x_l|C_k) \cdot p(C_k) \end{aligned}$$

holds. Rearranging and demarginalizing provides

$$\begin{aligned} p(C_k|x_1, ..., x_l) &= \frac{p(x_1, ..., x_l|C_k) \cdot p(C_k)}{p(x_1, ..., x_l)} \qquad (2.3) \\ &= \frac{p(x_1, ..., x_l|C_k) \cdot p(C_k)}{\sum_{j=1}^{|\mathcal{C}|} p(x_1, ..., x_l|C_j) \cdot p(C_j)}. \qquad (2.4) \end{aligned}$$

As the x_i are supposed to be statistically independent (Naïve Bayes), the equation can be simplified in the following way:

$$p(C_k|x_1, ..., x_l) \quad = \quad \frac{p(C_k) \cdot \prod_{i=1}^{l} p(x_i|C_k)}{\sum_{j=1}^{|\mathcal{C}|} p(C_j) \prod_{i=1}^{l} \cdot p(x_i|C_j)}. \qquad (2.5)$$

[5]Note that this assumption of a *Naïve Bayes* probabilistic model [Bar12] is only a heuristic. As local features of one object might overlap, the independence assumption will become violated.

By expanding the fraction by $1/\prod_{i=1}^{l} p(x_i)$, we arrive at

$$\begin{aligned}
p(C_k|x_1,...,x_l) &= \frac{\frac{p(C_k)\cdot\prod_{i=1}^{l} p(x_i|C_k)}{\prod_{i=1}^{l} p(x_i)}}{\frac{\sum_{j=1}^{|\mathcal{C}|} p(C_j)\cdot\prod_{i=1}^{l} p(x_i|C_j)}{\prod_{i=1}^{l} p(x_i)}} && (2.6)\\
&= \frac{p(C_k)\cdot\prod_{i=1}^{l}\frac{p(x_i|C_k)}{p(x_i)}}{\sum_{j=1}^{|\mathcal{C}|} p(C_j)\cdot\prod_{i=1}^{l}\frac{p(x_i|C_j)}{p(x_i)}} && (2.7)\\
&= \frac{p(C_k)\cdot\prod_{i=1}^{l}\frac{p(C_k|x_i)}{p(C_k)}}{\sum_{j=1}^{|\mathcal{C}|} p(C_j)\cdot\prod_{i=1}^{l}\frac{p(C_j|x_i)}{p(C_j)}} && (2.8)\\
&= \frac{\frac{1}{p(C_k)^{l-1}}\cdot\prod_{i=1}^{l} p(C_k|x_i)}{\sum_{j=1}^{|\mathcal{C}|}\frac{1}{p(C_j)^{l-1}}\cdot\prod_{i=1}^{l} p(C_j|x_i)}. && (2.9)
\end{aligned}$$

As a priori all classes in $\mathcal{C}$ are supposed to be equally likely, i.e. $p(C) = \frac{1}{|\mathcal{C}|}\forall C \in \mathcal{C}$, the equation can be further simplified:

$$p(C_k|x_1,...,x_l) = \frac{\prod_{i=1}^{l} p(C_k|x_i)}{\sum_{j=1}^{|\mathcal{C}|}\prod_{i=1}^{l} p(C_j|x_i)}. \quad (2.10)$$

It follows that the conditional probability for class C_k given a set of independent features $\{x_1,...,x_l\}$ is distributed according to the product of the conditional probabilities for class C given the single features x_i, i.e.

$$p(C_k|x_1,...,x_l) \propto \prod_{i=1}^{l} p(C_k|x_i). \quad (2.11)$$

By that, we can now aggregate the individual CDDs into a single one:

$$\begin{aligned}
\mathcal{D}(x_1,...,x_l) &= \begin{pmatrix} p(C_1|x_1,...,x_l) \\ \vdots \\ p(C_{|\mathcal{C}|}|x_1,...,x_l) \end{pmatrix} = \begin{pmatrix} \frac{\prod_{i=1}^{l} p(C_1|x_i)}{\sum_{j=1}^{|\mathcal{C}|}\prod_{i=1}^{l} p(C_j|x_i)} \\ \vdots \\ \frac{\prod_{i=1}^{l} p(C_{|\mathcal{C}|}|x_i)}{\sum_{j=1}^{|\mathcal{C}|}\prod_{i=1}^{l} p(C_j|x_i)} \end{pmatrix} && (2.12)\\
&= \frac{\bigotimes_{i=1}^{l}\mathcal{D}(x_i)}{\sum_{j=1}^{|\mathcal{C}|}\left(\bigotimes_{i=1}^{l}\mathcal{D}(x_i)\right)[j]}, && (2.13)
\end{aligned}$$

where $\bigotimes_{i=1}^{l}\mathcal{D}(x_i)$ denotes element-wise vector multiplication, and $v[j]$ denotes the j-th element of vector v. As no other combination method seems to clearly outperform this strategy [TvBDK00, HR07], we compute combinations of CDDs in this simple way. Choosing the product rule for CDD combination additionally offers an essential advantage over methods like probability averaging, which is another commonly used combination method. The product rule automatically favors distinctive CDDs (those containing low entropy in the object class distribution) over less distinctive CDDs (those containing high entropy in the object class distribution). This is especially useful when combining heterogeneous descriptors because a priori, it is not obvious which descriptor types or scales are most distinctive. In the following we will take a closer look at this particular behavior of combined CDDs:

Maximum entropy descriptors Given a set of CDDs $\{\mathcal{D}(x_1), ..., \mathcal{D}(x_l)\}$, without loss of generality let us suppose the class distribution of the l-th descriptor has maximum entropy, i.e. $\mathcal{D}(x_l) = (\frac{1}{|C|}, ..., \frac{1}{|C|})^T$. Then the combined CDD reads

$$\begin{aligned}\mathcal{D}(x_1, ..., x_l) &= \begin{pmatrix} p(C_1|x_1, ..., x_l) \\ \vdots \\ p(C_{|C|}|x_1, ..., x_l) \end{pmatrix} = \begin{pmatrix} \frac{\frac{1}{|C|}\prod_{i=1}^{l-1} p(C_1|x_i)}{\sum_{j=1}^{|C|}\frac{1}{|C|}\prod_{i=1}^{l-1} p(C_j|x_i)} \\ \vdots \\ \frac{\frac{1}{|C|}\prod_{i=1}^{l-1} p(C_{|C|}|x_i)}{\sum_{j=1}^{|C|}\frac{1}{|C|}\prod_{i=1}^{l-1} p(C_j|x_i)} \end{pmatrix} \\ &= \mathcal{D}(x_1, ..., x_{l-1}).\end{aligned}$$

This shows that adding an arbitrary amount of non-meaningful CDDs to an object description does not change the resulting combined CDD in any way.

Minimum entropy descriptors Given a set of CDDs $\{\mathcal{D}(x_1), ..., \mathcal{D}(x_l)\}$, without loss of generality let us suppose the class distribution of the l-th descriptor has minimum entropy, i.e. the conditional class probability reads 1 for exactly one class (denoted by $\hat{C}$)) and 0 for all remaining classes, $\mathcal{D}(x_l) = (0, ..., 0, 1, 0, ..., 0)^T$. Then the combined CDD reads

$$\begin{aligned}\mathcal{D}(x_1, ..., x_l) &= \begin{pmatrix} p(C_1|x_1, ..., x_l) \\ \vdots \\ p(C_{|C|}|x_1, ..., x_l) \end{pmatrix} = \begin{pmatrix} \frac{0\cdot\prod_{i=1}^{l-1} p(C_1|x_i)}{1\cdot\prod_{i=1}^{l-1} p(C_{\hat{C}}|x_i)} \\ \vdots \\ \frac{0\cdot\prod_{i=1}^{l-1} p(C_{\hat{C}-1}|x_i)}{1\cdot\prod_{i=1}^{l-1} p(C_{\hat{C}}|x_i)} \\ \frac{1\cdot\prod_{i=1}^{l-1} p(C_{\hat{C}}|x_i)}{1\cdot\prod_{i=1}^{l-1} p(C_{\hat{C}}|x_i)} \\ \frac{0\cdot\prod_{i=1}^{l-1} p(C_{\hat{C}+1}|x_i)}{1\cdot\prod_{i=1}^{l-1} p(C_{\hat{C}}|x_i)} \\ \vdots \\ \frac{0\cdot\prod_{i=1}^{l-1} p(C_{|C|}|x_i)}{1\cdot\prod_{i=1}^{l-1} p(C_{\hat{C}}|x_i)} \end{pmatrix} \\ &= (0, ...0, 1, 0, ..., 0)^T = \mathcal{D}(x_l).\end{aligned}$$

This shows that adding a distinctive descriptor to an object description strongly changes the resulting CDD towards the new distinctive prediction.

Numerical aspects Multiplying a large number of values smaller than 1 which is a prerequisite for CDD combination might inflict numerical problems. At a certain point, intermediate results might run below the limit of floating point precision. We alleviate this problem by transforming the probabilities into the logarithmic domain. We subsequently consider the sums of the log-probabilities instead of the product and finally retransform the result into the original domain by computing the exponential.

2.3.2 Comparing Class Distribution Descriptors

To compute the similarity of two 3D objects, we must define a distance measure between the two according combined CDDs. As CDDs represent a discrete probability distribution over the object classes, the natural similarity measure between them is given by the Kullback-Leibler (KL) divergence, also known as relative entropy [CT06]. Let $p(C)$ and $q(C)$ denote two probability distributions over the set of object classes $\mathcal{C}$, then the KL-divergence reads:

$$D_{KL}(p,q) = \sum_{j=1}^{|\mathcal{C}|} p(C_j) \log \frac{p(C_j)}{q(C_j)}. \tag{2.14}$$

As we are interested in a symmetric object similarity measure, we use the modified version of KL-divergence in our experiments :

$$D_{KLsym}(p,q) = D_{KL}(p,q) + D_{KL}(q,p). \tag{2.15}$$

2.4 Results on Princeton Shape Benchmark

In this Section we will first give an overview of our experimental setup describing the 3D objects and descriptors we use. We will then present our results and compare them to the method described in [FS06]. We will further discuss the influence of descriptor types and feature scales.

2.4.1 Experimental Setup

Dataset For our experiments we use a subset of the 3D objects contained in the Princeton Shape Benchmark (PSB) [SMKF04]. The PSB is separated into two sets. The training set contains 907 objects divided into 90 classes and the test set contains 907 objects divided into 92 classes. Merging both subsets leads to 1814 objects in 161 classes. To ensure a certain amount of robustness when estimating conditional class probabilities using the methods described in 1.5.4, we restrict our evaluation to those PSB classes containing at least 20 objects in the merged set. This leads to 739 objects divided into 21 classes, see Table 2.1. We divide

class	objects	class	objects
biplane	28	commercial airplane	21
fighter jet	100	helicopter	35
enterprise like	22	human standing	100
human arms out	41	sword	31
face	33	head	32
two story home	21	city	20
dining chair	22	shelves	26
table rectangular	51	handgun	20
vase	22	potted plant	51
tree barren	22	ship	21
sedan	20		

Table 2.1: PSB classes used in our experiments.

this set into a training set and a test set. For the training set, we randomly select 16 objects of each class, resulting in a total number of 336 training objects. The remaining 403 objects are put into the test set. For all experiments involving our CDDs we use the training set for learning the discriminant functions; retrieval is done exclusively on the test set. Three methods are used for comparison, global spherical harmonics descriptors [KFR03], global Zernike descriptors [Nov03] and partial matching with priority driven search [FS06]. The two methods using global shape descriptors do not require any training and are only evaluated on the test set. Partial matching requires a leave-one-out classification. Query results can only be found amongst the training objects, so for each query object, the remaining 402 models of the test set are used as training data.

Descriptors We use three different types of local shape descriptors as a basis for CDD computation: SH descriptors [KFR03], Zernike descriptors [Nov03] and spin images [Joh97]. Descriptor locations are randomly selected on the underlying triangle mesh with respect to the triangle areas. All triangle meshes are normalized to their bounding boxes. To compute SH and Zernike moments, the triangle meshes are transformed into voxel grids of size $128 \times 128 \times 128$ in a preprocessing step. SH and Zernike moments are computed on the voxel grid around all randomly selected points. The resolution of the SH descriptors is set to eight shells, each of which is represented by 16 coefficients. For each Zernike descriptor we use 156 coefficients. To generate spin images, the meshes are transformed into point clouds, each of it containing 250000 points per surface unit, leading to approximately 100000 to 700000 points per object. Spin images are computed on this point cloud around all randomly selected points. The required normals are derived from the underlying triangles. Note that we do not use oriented normals to build the spin-image, as the PSB objects do not provide consistent face orientations in general. The spin image resolution is set to 16×16 bins. All feature scales mentioned in the next section describe the radius of the computed local descriptor. All scales are measured with respect to the radius of the underlying object. We compute 64 local shape descriptors of every type and scale. For comparison to other methods and to show the impact of local CDDs, we also compute global SH and Zernike moments located at the object centers. The resolution for both of them is the same as for the according local descriptors.

CDD Computation For estimation of the conditional class probabilities we use Nonlinear Kernel Discriminant Analysis as described in Section 1.5.4. As a kernel, we use the standard radial basis function reading

$$k(x, x') = \exp\left(-\frac{|x - x'|^2}{2\sigma^2}\right). \qquad (2.16)$$

The optimal kernel width σ is determined separately for every of the $|\mathcal{C}|(|\mathcal{C}|-1)/2$ NKDA classifiers using 8-fold cross-validation [DHS01], each time leaving out the descriptors of two of the 16 training models of both object classes. Note that it is important to consistently assign all descriptors of a single model to either the training set or to the validation set during cross-validation. Otherwise, overfitting would result, as the overlapping descriptors of one single model are not independent and identically distributed (see [DHS01]).

2.4.2 Evaluation

Comparison to other methods We compare our method to three other shape retrieval approaches. On the one hand, classic approaches relying on global features without any supervised learning are represented by global spherical harmonics (GSH) and global Zernike moments (GZM) descriptors. Partial matching with priority driven search (PDS) on the other hand represents the method that is most related to our approach as it incorporates local descriptors and supervised learning. As can be seen from the precision-recall plot in Figure 2.2, PDS and our best performing method using local Zernike moments combined with local spin images (LZM-LSI) perform at almost the same precision for the first recall value of 0.05. For larger recall values, our approach clearly outperforms PDS as well as the other methods. Table 2.2 shows the performance of all four methods regarding additional quality criteria for shape retrieval as described in Section 1.5.3. The retrieval measures given in Table 2.2 additionally show the superior performance of our method.

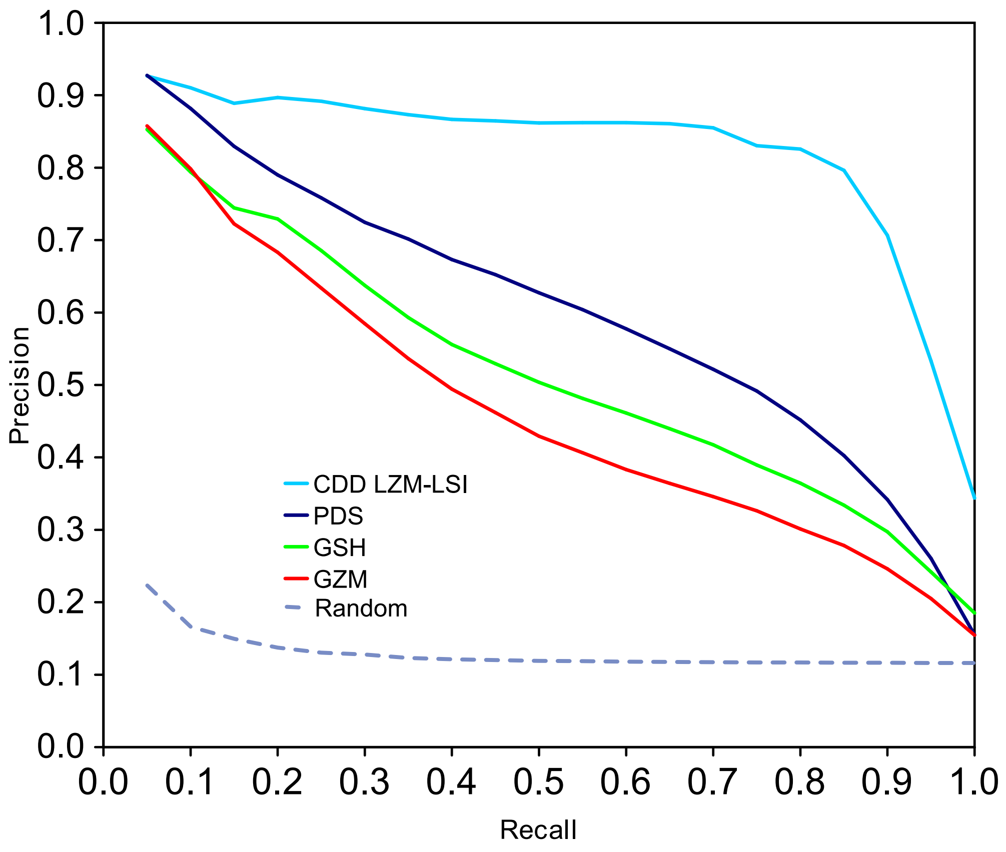


Figure 2.2: **Comparison to other methods.** The precision-recall plot shows a comparison of our approach (CDD LZM-LSI) using a combination of local Zernike moments descriptors and spin images to three other shape retrieval methods. *Partial Matching with Priority Driven Search* is denoted by PDS. Retrieval by one single global descriptor is represented by GSH (using spherical harmonics) and GZM (using Zernike moments).

Impact of local descriptors In order to show the importance of using local features we compare the performance of a classic global Zernike moment descriptor (GZM) and a CDD derived from one global Zernike moment descriptor (CDD

Method	1-NN	1-Tier	2-Tier	DCG
CDD LZM-LSI	**0.844**	**0.745**	**0.850**	**0.892**
PDS	0.789	0.543	0.677	0.795
GSH	0.705	0.427	0.598	0.731
GZM	0.732	0.390	0.540	0.714

Table 2.2: **Comparison to other methods.** Evaluation of our approach (CDD LZM-LSI) with respect to three other shape retrieval methods using standard retrieval metrics. Our algorithm shows superior performance regarding all quality criteria.

Method	1-NN	1-Tier	2-Tier	DCG
CDD LZM	**0.757**	**0.636**	**0.767**	**0.834**
CDD GZM	0.747	0.499	0.677	0.782
GZM	0.732	0.390	0.540	0.714

Table 2.3: **Impact of local descriptors.** The table shows three different methods using CDDs based on local Zernike moments (CDD LZM) computed on scale 1.0, CDDs based on one global Zernike moment (CDD GZM) and one global Zernike moment (GZM).

GZM) with our approach based on CDDs derived from local Zernike moments (CDD LZM). We include the CDD derived from the global descriptor in order to guarantee a fair comparison such that both, the local descriptors as well as the global one benefit from the supervised learning step. The results of the comparison are shown in Figure 2.3 and in Table 2.3. Although the performance of the global descriptor is definitely boosted by the CDD, it is still inferior to the CDDs derived from local descriptors.

Combination of different descriptor types To investigate the influence of descriptor type combinations, we select the best performing feature scales for every single descriptor type and combine the according CDDs with each other, see Table 2.4. It shows that the combination of Zernike moments and spin images (LZM-LSI) and the combination of SH descriptors and spin images (LSH-LSI) perform best and further improve the retrieval performance of each single descriptor type (see Figure 2.4 for the LZM-LSI combination). Although not reaching the quality that is achieved if spin images are involved, the combination of SH and Zernike moments (LZM-LSH) is superior to using these descriptors alone. The combination of all three descriptor types (LSH-LSI-LZM) does not improve the performance any further.

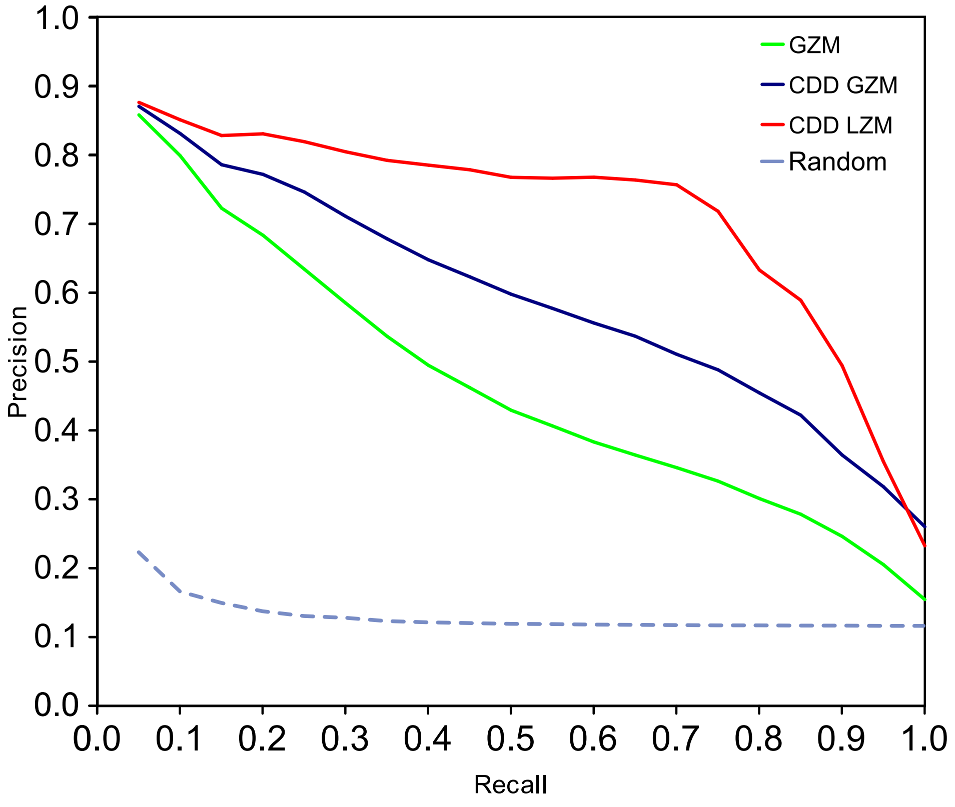


Figure 2.3: **Impact of local descriptors.** GZM denotes retrieval with one global Zernike moment. CDD GZM represents the same Zernike moment boosted by the CDD. CDD LZM shows retrieval results for 64 local Zernike moments at scale 1.0 using CDDs.

Selection of the right scale Figure 2.5 and Table 2.5 show retrieval results using CDDs from SH computed on three different scales (LSH 0.25, LSH 0.5, LSH 1.0). It also shows the retrieval results that are achieved by combining the resulting three CDDs into a single one (LSH combined). Like described in Section 2.3, the results show that choosing the product rule as combination strategy automatically favors distinctive feature. Without knowing which feature scale is most distinctive, combining leads to a result that is even a bit better than the one achieved by the optimal scale. This is due to the fact that although the overall performance of the scales 0.25 and 1.0 is worse than that of scale 0.5, these scales might be more distinctive for certain objects and thereby improve the performance if combined.

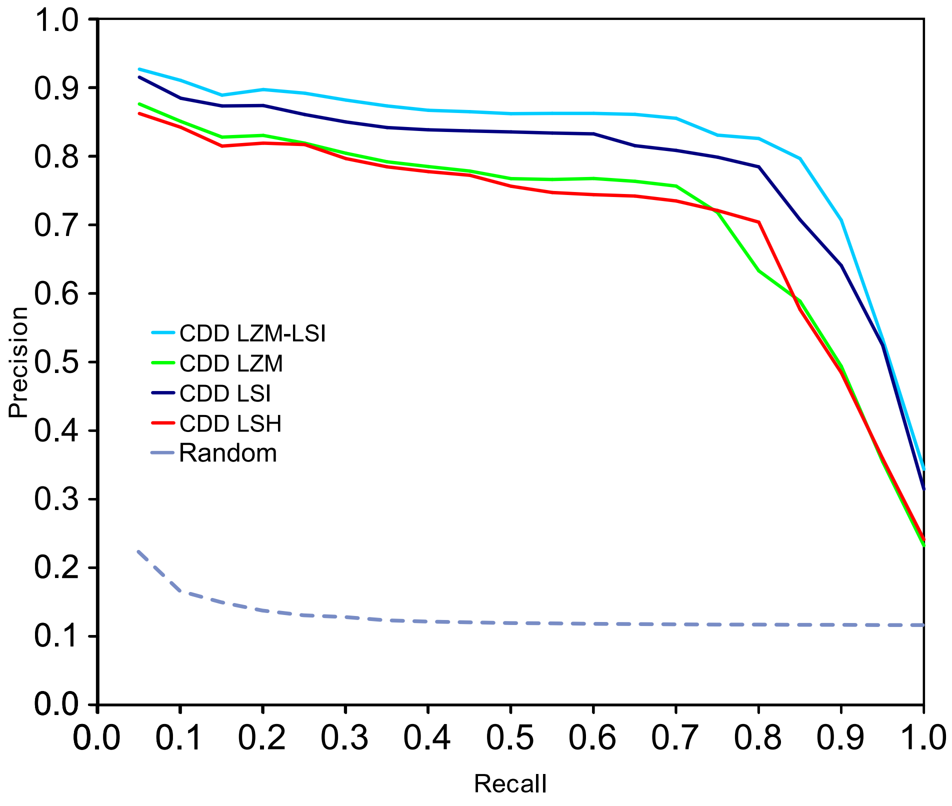


Figure 2.4: **Combining different descriptor types.** The combination of CDDs derived from Zernike moments and CDDs derived from spin images further improves the retrieval results.

Timings In Table 2.6 we provide information about the time consumption of our approach. All experiments are run on an AMD Athlon™X2 Dual Core[6] with 2.21 GHz and 2 GB RAM using Windows XP operating system (32 Bit). Parts of the training and the CDD computation are accelerated using an NVIDIA® GeForce®8800. The timings for preprocessing include the voxel grid computation (for SH and Zernike moments) and the point cloud generation (for spin images), respectively, as well as the computation of 64 local descriptors of a certain type. The timings for training and CDD computation are computed with respect to objects represented by 64 descriptors of one type on the scales shown in Table 2.4. The time required to query an unknown object given in a mesh representation is the sum of required preprocessing time and the CDD computation time. So far, the query time for one object is dominated by the descriptor computation during

[6]Despite the two available cores the timings were made using only one core.

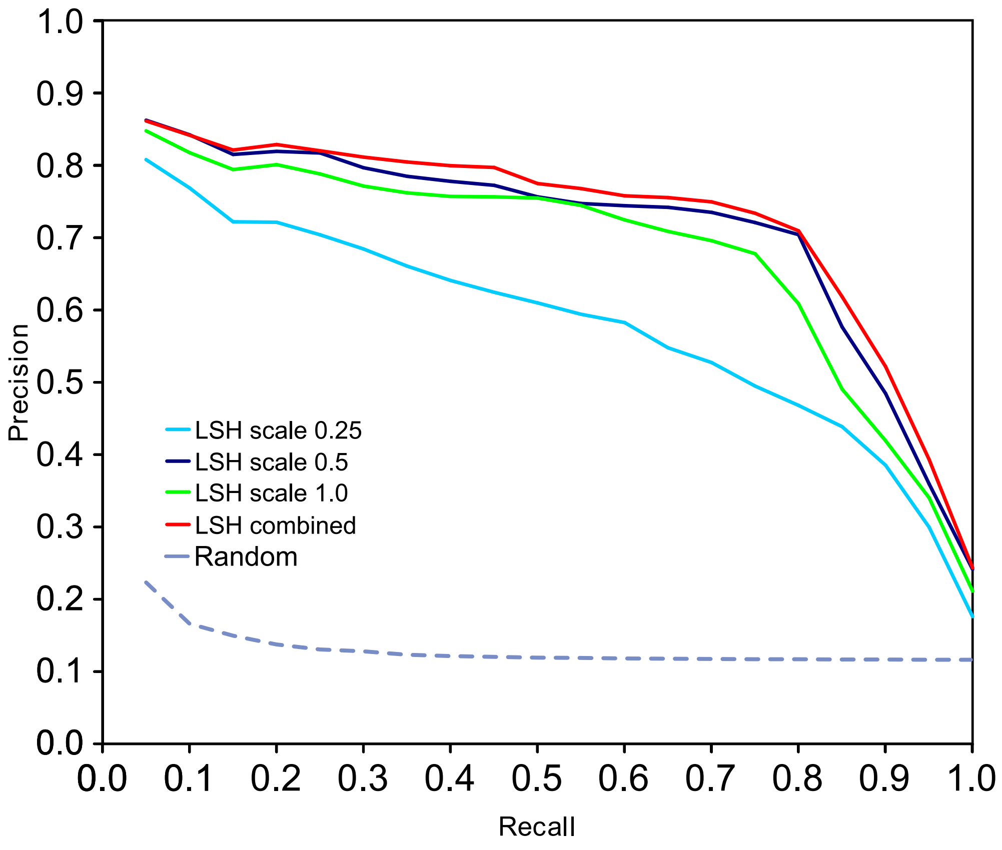


Figure 2.5: **Selecting distinctive feature scales.** The precision-recall plot shows CDDs derived from local SH descriptors computed at three different scales. Combination of the CDDs using the product rule leads to slightly better results than the best individual scale.

CDD Method	1-NN	1-Tier	2-Tier	DCG
LZM 0.5	0.757	0.636	0.767	0.834
LSI 2.0	0.824	0.716	0.810	0.880
LSH 0.5	0.742	0.633	0.746	0.835
LZM-LSI	0.844	0.745	**0.850**	0.892
LZM-LSH	0.769	0.678	0.782	0.856
LSH-LSI	**0.861**	0.744	0.845	**0.896**
LSH-LSI-LZM	0.821	**0.751**	0.841	0.893

Table 2.4: **Combining different descriptor types.** Results of experiments regarding combinations of CDDs derived from different types of local descriptors. The value behind the descriptor name denotes the scale on which it was computed.

CDD Method	1-NN	1-Tier	2-Tier	DCG
LSH 0.25	0.630	0.464	0.604	0.747
LSH 0.5	0.742	0.633	0.746	0.835
LSH 1.0	0.737	0.587	0.712	0.818
LSH combined	**0.777**	**0.655**	**0.771**	**0.847**

Table 2.5: **Impact of descriptor scales.** Retrieval results using CDDs derived from local SH descriptors computed at three different scales.

the preprocessing step. Note that the focus of our work is on retrieval quality rather than speed.

2.5 A Benchmark for 3D Architectural Data

Benchmarks consisting of 3D objects and classification schemes are a crucial prerequisite for developing shape retrieval methods, as only they allow to faithfully evaluate the retrieval performance of new algorithms. In Section 2.4.2 we evaluated our newly introduced class distribution descriptors using the generic Princeton Shape Benchmark [SMKF04] which is commonly used in the shape retrieval community. However, keeping in mind that we are mainly interested in retrieval methods tailored to 3D models from the architectural domain, the PSB seems no optimal choice for our purposes, as it contains models from various domains. Unfortunately, existing benchmarks in general are not well suited for our domain-specific retrieval. First, they either include only some models which are relevant for architecture like the PSB, or they are tailored to a completely different data domain (e.g. [JKIR06]). Second, the classification schemes of common benchmarks usually do not match an architect's requirements regarding model function

	LSH	LZM	LSI
Preprocessing training set	58 min	234 min	174 min
Preprocessing test set	71 min	284 min	216 min
Preprocessing per object (average)	10.53 s	42.05 s	31.75 s
Training	45 min	46 min	50 min
CDD computation test set	25 min	26 min	27 min
CDD computation per object (average)	3.78 s	3.85 s	4.05 s
Query of one object	14.31 s	45.90 s	35.80 s

Table 2.6: **Timings.**

and form.

To overcome these drawbacks, we introduce a new 3D shape benchmark that focuses on architectural data. It currently contains 2257 models consisting of building elements (i.e. doors, pillars), furnishing (i.e. chairs, shelves), and environment elements (i.e. plants, trees). The models are either part of different libraries of architectural CAD-applications or they belong to public services for architects of furniture manufacturers. Along with the shape database we provide classification schemes developed in close cooperation with architects to match their specific requirements. We will evaluate our previously introduced class distribution descriptor on the new benchmark and compare the results to those achieved on the Princeton Shape Benchmark.

This Section is based on the work presented in [WBK09]. It should be noted that the classification scheme presented in 2.5.1 was not contributed by the author of this thesis but by the co-author of [WBK09], Dr. Ina Blümel. We include the description of the classification scheme to ensure a better understanding of our retrieval results on our new benchmark.

2.5.1 Classification Schemes

One of the eminent interests of architects is the concept of form. Regarding a statement of the functionalism age, there is a particular relation between form and function in the design process of buildings and objects: *"[...] form ever follows function [...]"* [Sul96]. The architectural drafting process is both, form- and function-oriented. Architects are *"[...] working from abstract problem formulations to concrete solutions and splitting problems into subproblems iterative and recursive processes that rely upon anticipations of possible solutions [...]"* [CR92]. While drafting buildings, architects first concentrate on the function of rooms and components. Consequently, they mainly think of an object according to its function during this phase. After defining the functions, more and more details are added to the draft. During this phase, architects rather think according to form, provided that this form is suitable for the intended function. Figure 2.6 shows an example for the two different ways of classification.

Taking this into consideration, a component-classification has to be both form- and function-oriented to allow intuitive searching within any stage of drafting. Beside classifications for building construction and technology (CI/SfB [Roy68], BS ISO 12006-2/3 [Bri01, Bri07]) there exists a comprehensive, scientifically compiled vocabulary for the whole domain of art and architecture, the Getty Art & Architecture Thesaurus (AAT) [Pet94]. We use AAT as a starting point for developing our classification scheme because it

- contains *form-based and function-based* classifications of components,
- is hierarchical analogous to the needs of architects within the design process (from abstract problem formulations to concrete solutions),
- is currently consisting of around $34,000$ concepts, widely used by libraries and archives and in compliance with ISO-Standards.

Taking into account the two ways in which architects define 3D object similarity, we develop two hierarchic classification schemes, the first focusing on form and the second focusing on function. Regarding both schemes, the shape of an object serves as a starting point for retrieval and classification. Human beings are able to conclude about an object's function mostly by its form only. Consider e.g. a rectangular plate. By its form, it could be considered a table board, or a part of a wall, or a door. But as soon as a handle is attached at a specific position, it can be immediately recognized as a door, i.e. one can conclude about an object's function by inspecting its form. If necessary, additionally taking the surface material into account resolves ambiguities. On its finest level, the function-based scheme contains 183 classes, while the form-based scheme contains 180 classes. The classes are selected in a way such that most categories given in the AAT are represented. The schemes are provided in the same file format that is used for the PSB. Note that not all categories currently contain objects. In this first version of the benchmark we concentrate on furnishings considering especially the office domain regarding the eminent request for furnishing models within the design stages of detailing and visualization in architectural practice.

2.5.2 Benchmark Models

The original models collected for our benchmark exist in various file formats including e.g. `.3ds` or `.gsm`. All models are converted into triangle meshes and stored in the Wavefront OBJ format. The resulting files strongly vary in size,

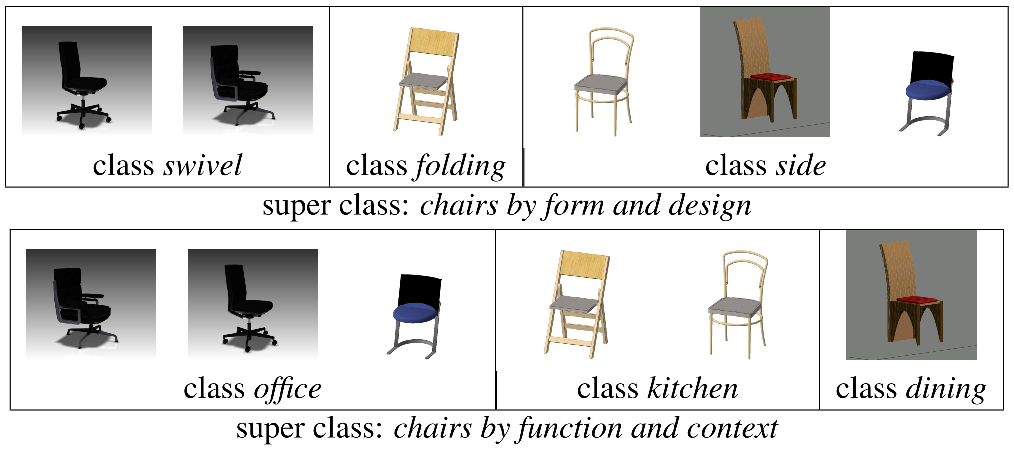

class *swivel*
class *folding*
class *side*
super class: *chairs by form and design*
class *office*
class *kitchen*
class *dining*
super class: *chairs by function and context*

Figure 2.6: **Form versus function.** Objects may either be classified by form or by function according to the perception within a certain stage of designing.

ranging from objects with only 12 triangles up to models with a total of 901300 triangles. As is usually the case when dealing with real-world data, the quality of the meshes differs strongly with respect to consistent face orientation. There is also no guarantee that the meshes are watertight.

2.5.3 Retrieval results

Given our new benchmark, we evaluate the retrieval performance of global Zernike moments descriptors (GZM) as well as CDDs derived from local spin images (LSI) and GZMs. To make sure that enough training data for each category is available when using supervised methods, we again select all objects belonging to categories containing at least 20 objects, resulting in 1817 models subdivided in 25 classes using the form-based classification scheme, and 1774 models subdivided in 26 classes using the function-based classification scheme. Parameters for feature computation are chosen identical to those described in Section 2.4.1. We randomly select 16 models from each category to build the training set. The remaining models form the test set. For each object in both sets we compute 64 spin images and one global Zernike moments descriptor. We then perform the learning step on the training dataset in the described way. Retrieval is conducted on the test set computing CDDs for all local and global features. In Figure 2.7 and 2.8 we show the results for both classification schemes. Additional retrieval performance measurements can be found in Table 2.7. We compare the retrieval performance on the test set using global Zernike moments (without learning), CDDs derived from global Zernike moments descriptors and CDDs derived from the local spin-images (both involving learning). To get an impression how much of a challenge

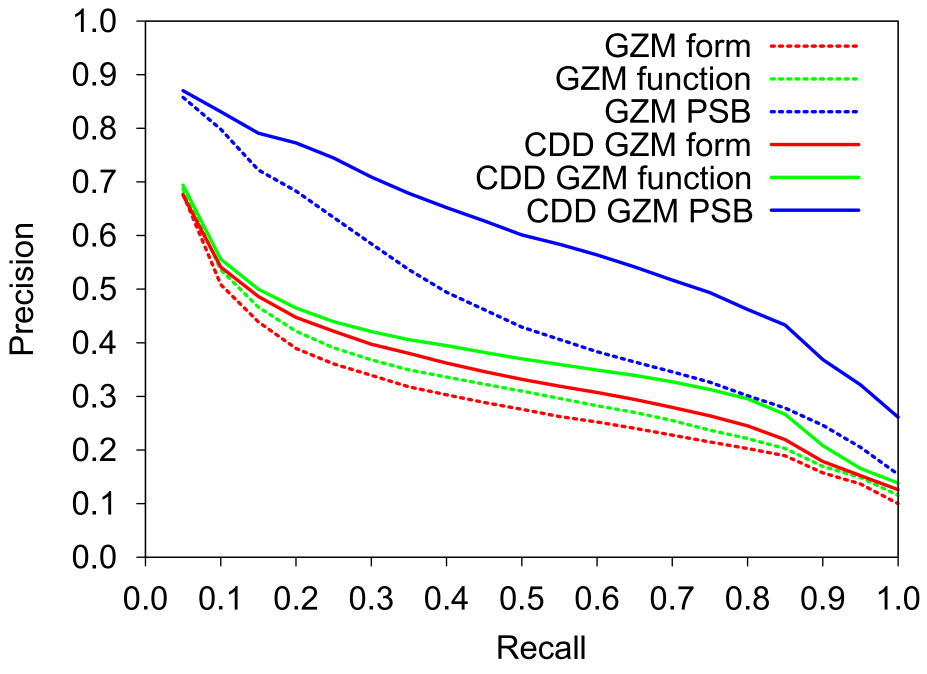


Figure 2.7: **Retrieval results using global Zernike (CDD) descriptors.** We compare the retrieval performance achieved on the PSB to that achieved on the form- and function-based classification scheme of our architecture benchmark.

our new benchmark poses, we compare the results to those achieved by the same retrieval algorithms on the PSB (see Section 2.4.2). As can be seen in Figures 2.7 and 2.8 as well as in Table 2.7, the retrieval performance of all algorithms running on the new benchmark is worse than that on the PSB, especially when considering the good results that were achieved on the PSB using CDDs inferred from local spin-images. This is most probably due to the fact that our benchmark is restricted to architectural data, including classes that are very similar to each other in terms of shape (see e.g. Figure 2.6). For both classification schemes, the intra-class variability of global model shapes is still quite large, leading to similar retrieval results using global Zernike moments descriptors (see Figure 2.7). As the intra-class variability of local model shapes on the other hand is believed to be smaller within the form-based classification than in the function based-one, local descriptors provide better retrieval results regarding the form-based classification (see Figure 2.8).

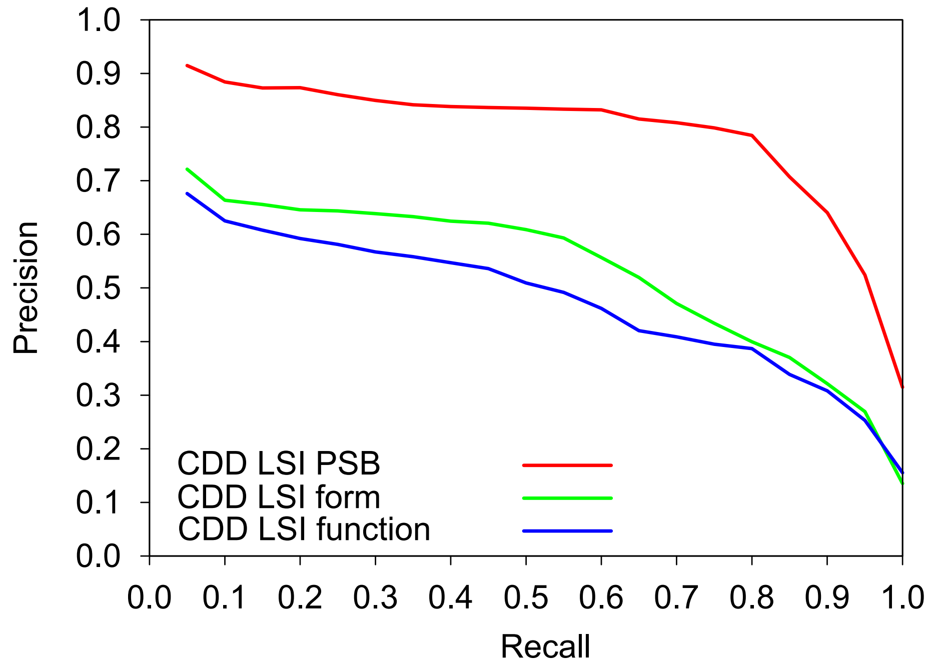


Figure 2.8: **Retrieval results using local spin-images and supervised learning.** We again compare the retrieval performance achieved on the PSB to that achieved on the form- and function-based classification scheme of our architecture benchmark.

2.6 Conclusion

Our first experimental evaluation of CDD performance on the generic Princeton Shape Benchmark shows encouraging results. Using CDDs, we are able to boost the performance of well-known global descriptors. Additionally, CDDs enable us to successfully combine local descriptors of different type and different scale into one single distinctive representation that outperforms CDDs derived from a single descriptor regarding retrieval performance. With the introduction of the new form- and function-based architecture benchmark providing very low inter-class variability we notice a strong drop in the retrieval performance of all evaluated retrieval methods. However, even in this much harder scenario, CDDs still prevail over the other methods.

Descriptor performance From a technical point of view it seems quite remarkable that CDDs from local spin images outperform CDDs from SH descriptors and from Zernike moments descriptors, considering that they can be constructed much more easily and do not require that much mathematical insight. Equipping

Descriptor	1-NN	1-Tier	2-Tier	DCG
GZM Form	**0.700**	0.279	0.404	0.682
CDD GZM Form	0.685	0.322	0.466	0.708
CDD LSI	0.592	**0.482**	**0.670**	**0.774**
GZM Function	**0.666**	0.296	0.412	0.681
CDD GZM Function	0.659	0.352	0.482	0.707
CDD LSI Function	0.561	**0.463**	**0.561**	**0.758**
GZM PSB	0.729	0.405	0.532	0.719
CDD GZM PSB	0.739	0.510	0.681	0.770
CDD LSI PSB	**0.824**	**0.716**	**0.810**	**0.880**

Table 2.7: **Comparison between retrieval results on PSB and architecture benchmark.** We evaluate global Zernike moments descriptors(GZM), meta descriptors from global Zernike moments (CDD GZM) and meta descriptors from local spin-images (CDD LSI).

a descriptor with invariance properties is usually accompanied by a loss of information and descriptiveness. In contrast to the completely rotational invariant SH and Zernike descriptors, spin images are only invariant under rotation around the associated normal. We think that the higher retrieval performance of spin images results from their better descriptiveness caused by less invariance demands.

Limitations and drawbacks The advantage gained by our supervised learning approach requires the additional effort of creating ground truth, i.e. a training set of 3D objects must be manually assigned to categories of some classification scheme. Depending on the intra- and inter class variability in the data, the amount of training data needed for improved retrieval can vary. Additionally, the query processing time is larger due to the necessary prediction of conditional class probabilities. However, regarding the results in Table 2.6 we believe that by further optimization interactive response times are within the realms of possibility.

Future work To gain a better understanding of the supposed impact of the supervised learning scheme we recommend to evaluate the performance of the proposed local descriptors (LSH, LZM, LSI) using a standard Bag-of-Features approach. The combined performance of the local descriptors should be evaluated by concatenating the resulting histograms of each single descriptor type. We also suggest repeating the experiments using a Support Vector Machine with probabilistic output, as available highly optimized state-of-the-art implementations of this classifier are very likely to outperform our self-implemented NKDA. It would also be interesting to investigate the applicability of the CDD approach to partial shape retrieval. Depending on the amount of missing object geometry, we suggest to use smaller descriptor scales in this scenario, if necessary.

Regarding the highly reduced effectiveness of shape retrieval methods on the architecture benchmark it seems necessary to better tailor algorithms to this specific domain. We will pick up this thought in Section 3 of this thesis by adjusting our feature localization method to the geometric properties of man-made objects.

Acknowledgements We would like to thank the following insitutions for providing 3D content: Leibniz University Hannover (Faculty of Architecture and Landscape Sciences, AIDA), Nemetschek AG, Samas GmbH & Co. KG, Graphisoft Deutschland GmbH, Vitra AG, Wilkening + Hahne GmbH & Co. KG. We would further like to thank MACE-Metadata for Architectural Contents in Europe for their cooperation within developing the classification schemes `http://www.mace-project.eu` *[last accessed 23 January 2013].* Additional thanks go to Dr. Philip Shilane for providing results on PDS.

CHAPTER 3

LEARNING THE COMPOSITIONAL STRUCTURE OF MAN-MADE OBJECTS

3.1 Introduction

Apart from the problems arising from similarity computation that we already discussed in Section 2.2.2, the usage of local features for 3D shape retrieval additionally bears the challenge of *feature selection/localization*, i.e. the decision which parts of the 3D object should be represented by a descriptor. In the previous chapter we rather ignored this question by randomly localizing the descriptors on the object surface and characterizing a surrounding sphere or a cylinder, respectively, of fixed radius. In this Section we will investigate a more sophisticated feature selection technique that is tailored to the domain of interest which consists of man-made architectural objects. Additionally, we will introduce a method that allows easy exploitation of information that is encoded in the spatial relationship of features. We thereby overcome the drawbacks of common BoF approaches (spatial relationship is discarded) as well as those of correspondence establishing methods (complicated thresholding and poor generalization).

Most feature selection methods are based on local geometric properties of the 3D object. The idea is to identify features as parts of the object that are *salient* in a geometric sense. Most approaches thereby focus on features that can be robustly detected under object transformations like scaling, rotation, shearing, and articulation, see e.g. [NDK05, GCO06, OOFB08, HH09]. The approach that we will present in this chapter includes a selection method that is especially tailored to 3D models representing man-made objects, like the previously introduced context models from architectural drafting. Due to common manufacturing processes, these objects mainly consist of building blocks that can be assembled from parts of certain shape primitives like planes, cylinders, spheres, cones and tori. These structures are the starting point for our feature selection. We use the algorithm presented in [SWK07] to decompose a 3D model into segments corresponding to shape primitives. For each segment, we compute a shape descriptor. Depending on the size of the underlying shape primitive, our algorithm produces features ranging from very local to rather global (see e.g. Figure 3.1). Our feature selection method is similar to the one presented in [FMA$^+$10] using the mesh segmentation algorithm presented in [AFS06]. However, in contrast to this method our approach does not rely on an intact mesh connectivity which is often not available when dealing with real world data. Instead we only require a point cloud which can be easily obtained by densely sampling the underlying mesh. In addition to plane-, cylinder-, and sphere-like shapes which are recognized in [AFS06], the algorithm in [SWK07] supports cone- and tori-like shapes.

In contrast to shape retrieval approaches based on global descriptors where object similarity can be determined in a straight forward way by computing the distance between global descriptors, there is no such easy way for methods involving local features, as we already described in Section 2.2.2. Using BoF-based approaches, the spatial arrangement of local features is lost as soon as they are described by histograms, just as it happens when objects arranged in a certain order are put into a bag. In contrast, methods based on *correspondence computation* take feature similarity as well as their spatial relationship into account, but in most cases they involve tricky manual parameter tuning, rendering it hard to achieve good generalization results. In this chapter, we try to overcome the drawbacks of the above mentioned approaches. We propose a probabilistic framework for learning the *compositional structure* of 3D objects which is inspired by an approach to 2D image retrieval by Ommer et al. [OB07, OB09]. In contrast to common BoF-approaches it incorporates the relative position of single features as well as the spatial relationship of feature tuples. To this end we extend the class distribution descriptor approach presented in Chapter 2. Learning the class distributions is then not only conducted with respect to a shape descriptor, but also with respect to positional information. Using a supervised learning scheme, we overcome the shortcomings of methods based on correspondence computation, as we do not need to manually enforce cumbersome thresholds on descriptor similarity or spatial distances. We finally compare our new approach to the results achieved by our approach presented in Chapter 2 on the 3D architecture benchmark. Summarizing the key contributions of this Chapter, they are:

- A new method for feature selection that is especially tailored to the domain of 3D models representing man-made objects.
- A supervised learning framework for efficient similarity computation of local feature sets incorporating their spatial relationship.
- An evaluation of our new approach using the architecture benchmark with a subsequent discusson of the impact of our feature localization technique

as well as of the proposed incorporation of spatial feature relationships.

3.2 Related Work

There is a huge amount of literature on feature localization for 3D shapes. In the following we will therefore concentrate on methods that have been used in the context of 3D shape retrieval.

Randomly selected uniformly distributed features Probably the easiest way for feature selection is to randomly select uniformly distributed points on the object surface as feature centers. Feature radii are then usually determined according to a manually chosen value. Mitra et al. [MGGP06] locally characterize 3D shapes by probabilistic shape signatures based on spin-images [Joh97] computed at randomly selected uniformly distributed points on the object's surface. Providing good results for automatic scan alignment, the retrieval performance of this method highly depends on the chosen local spin-image scale. Uniformly distributed surface points also serve as a starting point for multi-scale spherical harmonics descriptor computation and subsequent distinction-based feature selection in [SF06]. Furuya et al. [FO09] compute randomly localized dense multi-scale SIFT descriptors and aggregate them in a BoF.

Geometry-based feature selection In [GCO06], local salient regions are detected as mesh patches providing a high curvature relative to the surrounding area. A region-growing approach is used to subsequently augment small salient patches to larger regions. Shalom et al. [SSSCO08] use the shape-diameter function for both, segmentation and part signature definition. Ohbuchi et al. [OOFB08, OF08] introduce salient local visual features extracted as SIFT-features [Low04] from rendered depth images[1]. A generalization of the SIFT algorithm to three dimensions is presented in [NDK05]. In this work, Novotni et al. detect salient points on a 3D voxel grid as local extrema of the scale space Laplacian-of-Gaussian. For each detected salient point, they compute a local SH descriptor. Further approaches based on scale spaces are presented in [LVJ05] and [ZHDQ08]. In [HH09], Hu et al. present an approach to detect local salient mesh regions using extrema in the Laplace-Beltrami spectral domain of the mesh rather than in the usual spatial domain, rendering this localization algorithm invariant to isometric mesh deformations. An approach closely related to our own is presented in [SWWK08]. Primitive shapes like planes, cylinders, etc. are detected in 3D

[1]Interestingly, randomly localized dense SIFT feature computation as presented in [FO09] and subsequent BoF construction outperforms this saliency-driven approach.

laser range scans. In contrast to our approach, no local descriptors based on the point supports of the shapes are computed. The primitives are directly used as nodes in a graph-based algorithm searching for certain manually defined configurations of primitives, forming simple shapes like building roofs. A similar method restricted to the detection of configurations of planes is described in [VKH06]. For additional geometry-based feature selection methods and exhaustive stability evaluations we refer to the recently published comprehensive articles by Yu et al. [YWC12] and Dutagaci et al. [DCG12].

Distinction-based feature selection Selection based on the retrieval performance of local features is introduced in [SF06] and used in [SF07, FS06]. Considering a set of pre-classified training objects, a number of random surface points are sampled from all objects. For each of these points, a local SH descriptor is computed characterizing the local surface geometry with respect to a certain radius. For each descriptor, it is determined how well it is suited for efficient object retrieval. For new unknown objects, again local SH descriptors are computed around randomly sampled surface points. The knowledge acquired during the training step is used to predict the retrieval performance of these local descriptors and only the most distinctive ones are finally used for retrieval.

3.3 Feature Selection and Descriptor Computation

In this Section, we will first describe how features of 3D models representing man-made objects can be selected using shape primitives like planes, cylinders, etc. After that we will show how the supporting regions of shape primitives can be represented by descriptors.

3.3.1 Feature Selection

As a starting point for the detection of primitive shapes we use an unstructured 3D point cloud which can be obtained by randomly sampling from a 3D mesh. We employ the algorithm presented in [SWK07] which recognizes planes, spheres, cylinders, cones, and tori in the point cloud. In the evaluation conducted in [LSSK09], the segmentation provided by this approach showed increased robustness compared to the method presented in [WK05]. In contrast to [AFS06], the point cloud-based approach does not require an intact mesh connectivity and it is not restricted to planes, cylinders, and spheres. In this Section we will only give a very brief outline of the shape detection technique and the interested reader is referred to the original paper.

The data is decomposed into disjoint sets of points $\mathcal{S}_{\pi_i}$, each corresponding to a detected shape proxy π_i respectively, and a set of remaining points $\mathcal{R}$ that consists of outliers as well as areas of more complex geometry for which primitive shapes would give an inappropriate representation. For further processing, all remaining points are ignored. Points that are represented by a shape primitive are also called the *support* of a shape. Thus, given the point-cloud $\mathcal{P} = \{p_1, \ldots, p_{|\mathcal{P}|}\}$, the output of the shape detection is the following:

$$\mathcal{P} = \mathcal{S}_{\pi_1} \cup \ldots \cup \mathcal{S}_{\pi_l} \cup \mathcal{R}, \tag{3.1}$$

where each subset (the support) $\mathcal{S}_{\pi_i}$ is associated with a shape primitive π_i. All points in $\mathcal{S}_{\pi_i}$ constitute a connected component and fulfill the condition

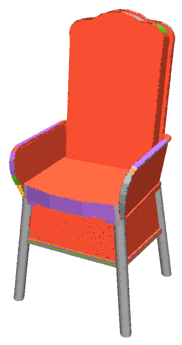
$$s \in \mathcal{S}_{\pi_i} \Rightarrow d(s, \pi_i) < \epsilon \wedge \sphericalangle(n_s, n(\pi_i, s)) < \alpha, \tag{3.2}$$

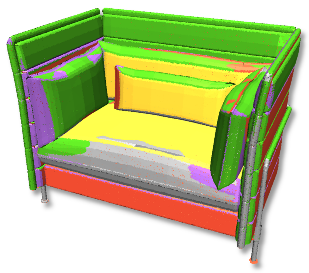
where n_s is the normal of point s, $n(\pi_i, s)$ denotes the normal of the primitive π_i at the point closest to s, and $d(s, \phi_i)$ denotes the Euclidean distance between s and π_i. The normals n_s are thereby estimated on the point cloud. The parameters ϵ and α are chosen by the user according to the sampling distance. The set $\mathcal{R}$ contains all remaining, unassigned points.

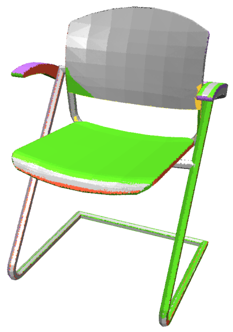
Examples for the decomposition of several objects from our architecture benchmark (see Section 2.5) can be found in Figure 3.1. For the choice of parameters concerning the primitive shape detection, we refer to Section 3.4.

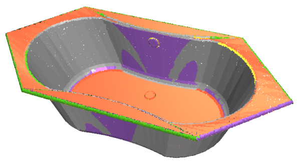
3.3.2 Descriptor Computation

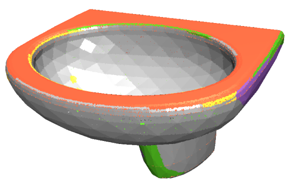
Theoretically, it would be possible to use the primitive shape type together with certain properties (e.g. radius and height for a cylinder primitive) as a shape descriptor. However, there are two reasons rendering this approach inefficient. First, the primitive shape detection is not robust with respect to the type of the detected primitive. For example, a set of points originating from a pipe might either be identified as part of a cylinder primitive or as part of a torus primitive with a very large radius (see e.g. the legs of the bench in Figure 3.1f). Second, such a descriptor would not incorporate the fact that the underlying support points might only represent a part of a primitive (e.g. only a hemisphere instead of a whole sphere). We therefore do not characterize the local object part by the primitive itself but rather by its support. Once the primitive shape is detected, we compute a spin


(a) armchair


(b) easy chair


(c) side chair


(d) bathtub


(e) sink

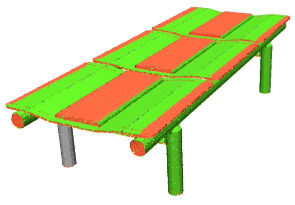

(f) bench

Figure 3.1: **Detection of primitive shapes.** Colors are chosen with respect to the partially detected primitive types including plane (red), cylinder (green), torus (grey), cone (purple), and sphere (yellow). Figure f) shows an example for the instability of the shape detection. Two legs are identified as a cylinder, but one is identified as part of a torus.

image [Joh97] representing the support points. By that, the discrete shape type is described in a more continuous way.

We align the spin image axis according to the Z-axis of the underlying object. Note that this representation is not invariant under rotations around the Z-axis of the object. However, 3D models representing man-made objects are mostly modeled in a way that their Z-axis is chosen according to the world's up-direction. Therefore, this choice does not put a severe restriction to our algorithm. Note that our framework for learning the compositional structure of 3D objects is not restricted to the usage of spin images. It would also be possible to use e.g. spherical harmonics descriptors.

3.3.3 Integrating Feature Locations

Let us briefly recall the definition of the class distribution descriptor from Section 2.3. For a given set of classes $\mathcal{C} = \{C_1, ..., C_{|\mathcal{C}|}\}$, the CDD $\mathcal{D}(x)$ of an arbitrary local descriptor $x \in X = \mathbb{R}^d$ reads

$$\mathcal{D}(x) = \begin{pmatrix} p(C_1|x) \\ \vdots \\ p(C_{|\mathcal{C}|}|x) \end{pmatrix}.$$

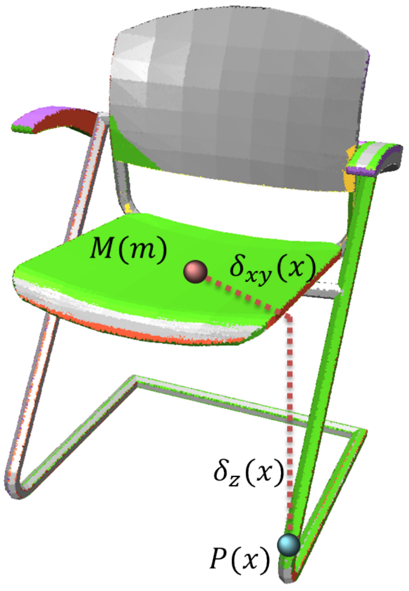


Figure 3.2: **Spatial relationship between center and feature.** The dotted lines visualize the two components of the spatial relationship $\Phi(M(m), x)$.

In the first step, we integrate the relative position of feature x inside the 3D object into the CDD. Let $M(m) \in \mathbb{R}^3$ denote the center of mass of model m, and let $\Phi(M(m), x)$ denote the spatial relationship between the position of feature x and the object center $M(m)$.There are several possibilities how to choose $\Phi(M(m), x)$. In a setting where the underlying object can be rotated in an arbitrary way, the natural choice would be $\Phi(M(m), x) := ||M(m) - P(x)||$, where $P(x) \in \mathbb{R}^3$ denotes the 3D position of x. However, as in our setting the Z-axes of the objects are consistently oriented, we follow another approach allowing us to integrate more precise information about the spatial relationship. We define

$$\Phi(M(m), x) := \begin{pmatrix} \delta_z(x) \\ \delta_{xy}(x) \end{pmatrix} \tag{3.3}$$

$$= \begin{pmatrix} M_z(m) - P_z(x) \\ \sqrt{(M_x(m) - P_x(x))^2 + (M_y(m) - P_y(x))^2} \end{pmatrix}, \tag{3.4}$$

where the subscripts denote the x, y, and z components of the 3D coordinates. We thereby decompose the spatial relationship between $M(m)$ and x into a signed distance along the z-axis which is represented by the first component of $\Phi(M(m), x)$. The second component describes the unsigned distance between $M(m)$ and x when both are projected into the xy-plane. Please see Figure 3.2 for a visualization of the two components.

So far, the size of the local feature is not incorporated. We therefore introduce an additional parameter $\gamma(x)$ describing the number of support points of feature x with respect to the total number of points in the object. By that, the modified CDD reads:

$$\mathcal{D}'(x) = \begin{pmatrix} p(C_1|x, \delta_z(x), \delta_{xy}(x), \gamma(x)) \\ \vdots \\ p(C_{|\mathcal{C}|}|x, \delta_z(x), \delta_{xy}(x), \gamma(x)) \end{pmatrix}. \tag{3.5}$$

3.3.4 Spatial Relationship between Features

In the second step, we will additionally consider the spatial relationship between feature tuples (x_i, x_j) consisting of two features from the same object. As the positions of $P(x_i)$ and $P(x_j)$ around the object center $M(m)$ are fixed by δ_z and δ_{xy} except for rotation around the Z-axis, we only need to additionally incorporate the distance $||P(x_i) - P(x_j)|| =: \delta(x_i x_j)$ between $P(x_i)$ and $P(y_i)$ into our framework. The according CDD is then given by:

$$\mathcal{D}'(x_i, x_j) = \begin{pmatrix} p(C_1|x_i, x_j, \delta_z(x_i), \delta_z(x_j), \\ \qquad \delta_{xy}(x_i), \delta_{xy}(x_j), \gamma(x_i), \gamma(x_j), \delta(x_i x_j)) \\ \vdots \\ p(C_{|\mathcal{C}|}|x_i, x_j, \delta_z(x_i), \delta_z(x_j), \\ \qquad \delta_{xy}(x_i), \delta_{xy}(x_j), \gamma(x_i), \gamma(x_j), \delta(x_i x_j)) \end{pmatrix}. \tag{3.6}$$

Intuitively speaking, it describes how likely it is that the currently considered object belongs to a certain object category, given the co-occurrence of features x_i and x_j in a certain spatial arrangement.

3.3.5 Modified Feature Vectors and Kernel Functions

As described in Section 3.3.2 we use spin image coefficients $x \in \mathbb{R}^d$ as a descriptor for the extracted shape primitives. Considering single features, this descriptor has to be combined with the additional information about feature location and size given by the parameters $\delta_z(x), \delta_{xy}(x), \gamma(x)$. Note that simply defining

$$x'_i := (x[1], ..., x[d], \delta_z(x), \delta_{xy}(x), \gamma(x))^T \tag{3.7}$$

and evaluating a kernel function $k(x'_i, x'_j)$ in order to train the probabilistic classifier would lead to instabilities as the coefficients for spatial relationship and relative feature size have a completely different meaning and scale compared to the spin image coefficients, let alone the ratio of 3 to 156 coefficients. Although kernel-based discriminant functions are known to be able to implicitly weight certain feature entries, stability can be increased by introducing weighting factors when considering feature entries that are measured on different scales. Therefore, we modify the original simple RBF kernel (see Equation 2.16) from Section 2.4.1 by introducing weights to properly balance all coefficients in x'_i. For single features, we define

$$k_s(x'_i, x'_j) := \exp\left(-\frac{(x'_i - x'_j)^T W_s (x'_i - x'_j)}{2\sigma^2}\right), \tag{3.8}$$

where W_s is a diagonal matrix of size $(d+3) \times (d+3)$ containing weighting factors for $\delta_z(x), \delta_{xy}(x), \gamma(x)$. The vector $D(W_s) \in \mathbb{R}^{d+3}$ constituted by the diagonal elements of W_s reads:

$$D(W_s) := (1, \cdots, 1, \alpha_\delta, \alpha_\delta, \alpha_\gamma). \tag{3.9}$$

Please note that both descriptor entries $\delta_z(x)$ and $\delta_{xy}(x)$ that describe the relative spatial position of the feature share the same weighting parameter α_δ. In contrast, the relative feature size $\gamma(x)$ is weighted by another parameter α_γ. Considering a feature pair (x_i, x_j), the underlying shape descriptors must be combined into one common vector. In this vector, we order x_i and x_j according to the size of their point support. Without loss of generality, let x_i be the local feature with the larger point support. Then, incorporation of the additional information about feature location, size, and spatial relationship leads to the following kernel input vector ${x'}_{ij}$:

$$\begin{aligned} x'_{ij} \;=\; & (x_i[1], ..., x_i[d], x_j[1], ..., x_j[d], \delta_z(x_i), \delta_z(x_j), \\ & \delta_{xy}(x_i), \delta_{xy}(x_j), \gamma(x_i), \gamma(x_j), \delta(x_i x_j))^T \end{aligned} \tag{3.10}$$

The according kernel function reads

$$k_t(x'_{ij}, x'_{i'j'}) := \exp\left(-\frac{(x'_{ij} - x'_{i'j'})^t W_t (x'_{ij} - x'_{i'j'})}{2\sigma^2}\right), \tag{3.11}$$

where W_t is a $(2d+7) \times (2d+7)$ diagonal matrix containing the weighting factors such that the diagonal $D(W_t) \in \mathbb{R}^{2d+7}$ reads:

$$D(W_t) := (1, \cdots, 1, \alpha_\delta, \alpha_\delta, \alpha_\gamma, \alpha_\delta, \alpha_\delta, \alpha_\gamma, \alpha_\delta). \tag{3.12}$$

We determine all weighting factors as well as the kernel width σ completely automatically using cross-validation. During the NKDA training process, a discriminant function for each pair of the $|\mathcal{C}|$ object classes is computed. This leads to different α_s and σ for each discriminant function, taking into account that feature size and relative position are of varying importance depending on the considered object categories.

3.3.6 Modified Combination of Class Distribution Descriptors

In order to finally compare the CDDs derived from different objects, the CDDs of each single feature and of each feature pair must be combined into one single CDD. To this end we use the combination technique presented in Section 2.3.1. The resulting descriptor can be determined by multiplying and renormalizing the CDDs computed so far:

$$\mathcal{D}'(x_1,...,x_d) \quad = \quad \frac{\bigotimes_{k=1}^{l}\mathcal{D}'(x_k)\bigotimes_{i=1}^{l}\bigotimes_{j>i}^{l}\mathcal{D}'(x_i,x_j)}{\sum_{c=1}^{|\mathcal{C}|}\left(\bigotimes_{k=1}^{l}\mathcal{D}'(x_k)\bigotimes_{i=1}^{l}\bigotimes_{j>i}^{l}\mathcal{D}'(x_i,x_j)\right)[c]}.$$

3.4 Results

For our experiments, we use the 3D architecture benchmark presented in Section 2.5 along with the form-based classification scheme. We compare the results of our newly introduced method involving sophisticated feature localization as well as the incorporation of spatial feature relationships to our previously introduced approach relying on CDDs computed on arbitrary localized spin images, see 2.5.3.

3.4.1 Experimental Setup

Dataset The dataset is identical to the one used in 2.5.3, i.e. all classes that contain at least 20 objects are used for the evaluation, resulting in 25 classes with a total of 1817 objects. We divide the data into a training set and a test set. For the training set, we randomly select 16 objects of each class, the remaining 1417 objects are put into the test set.

Preprocessing and Shape Detection A point cloud representation is the prerequisite for computing primitive shapes as well as spin image descriptors. We therefore normalize all meshes to the $[-1,-1,-1]\times[1,1,1]$ bounding box and randomly sample 50000 points per unit area on the surface from the underlying triangles. For the shape detection described in Section 3.3.1, we set $\alpha = 0.9$ and $\epsilon = 0.002$. Note that the same parameter setting is used for the whole dataset. Depending on the complexity of the underlying model, the number of detected shapes varies between 10 to 200. For further descriptor computation, we select those 32 shapes providing the largest point support. If less than 32 shapes are detected, all of them are used.

Descriptors To evaluate our approach, we compute spin image descriptors describing the point support of every selected shape primitive. The spin images are

Method	1-NN	1-Tier	2-Tier	DCG
Uniformly Selected Local SI	0.592	0.482	**0.670**	0.774
Shapes + SR	**0.763**	**0.523**	**0.670**	**0.810**

Table 3.1: **Comparison to random feature selection.** Our new algorithm including shapes and spatial relationships shows superior quality to random feature selection.

positioned at the center of gravity of the support points and oriented according to the Z-axis of the object. The radius is chosen with respect to the support point farthest from the center of gravity. For the comparison, we randomly select 64 uniformly distributed surface points as spin image centers. In this setting, spin images are oriented according to the surface normal. For both settings, the spin image resolution is set to 16×16 bins.

Feature Tuples For an object with l detected shape features, we select those $l/2$ features providing the largest support to generate $\binom{l/2}{2}$ 2-tuples. For the maximum number of $l = 32$ shapes, this leads up to 120 feature tuples.

3.4.2 Evaluation

The performance of our algorithm is shown in Figure 3.3 and in Table 3.1. Considering the precision-recall plot, our method *(Shapes and Spatial Relationships)* outperforms the approach based on spin images centered at randomly selected uniformly distributed surface points *(Uniformly Selected Local Spin Images)*[2]. Table 3.1 shows the performance of both methods regarding additional quality criteria. Again, our new method involving spatial relationships and feature selection according to shape primitives achieves a higher retrieval performance. However, despite the performance increase, the results are not satisfying yet, especially when thinking of the excellent findings on the PSB presented in Chapter 2.

[2]Please note that the curve for random feature localization in Figure 3.3 shows a worse performance than the according one in Figure 2.7. However, Tables 2.7 and 3.1 show identical results for the same experiment. Unfortunately, this behavior was caused by a bug contained in the evaluation utilities accompanying the PSB that we used for earlier evaluations, see `http://shape.cs.princeton.edu/benchmark/psb_util.zip` [last accessed 22 January 2013]. The routine computing precision-recall values is faulty in the sense that the query object is also contained in the target dataset, rendering the results too good. However, this bug is not contained in the computation of the other retrieval metrics.

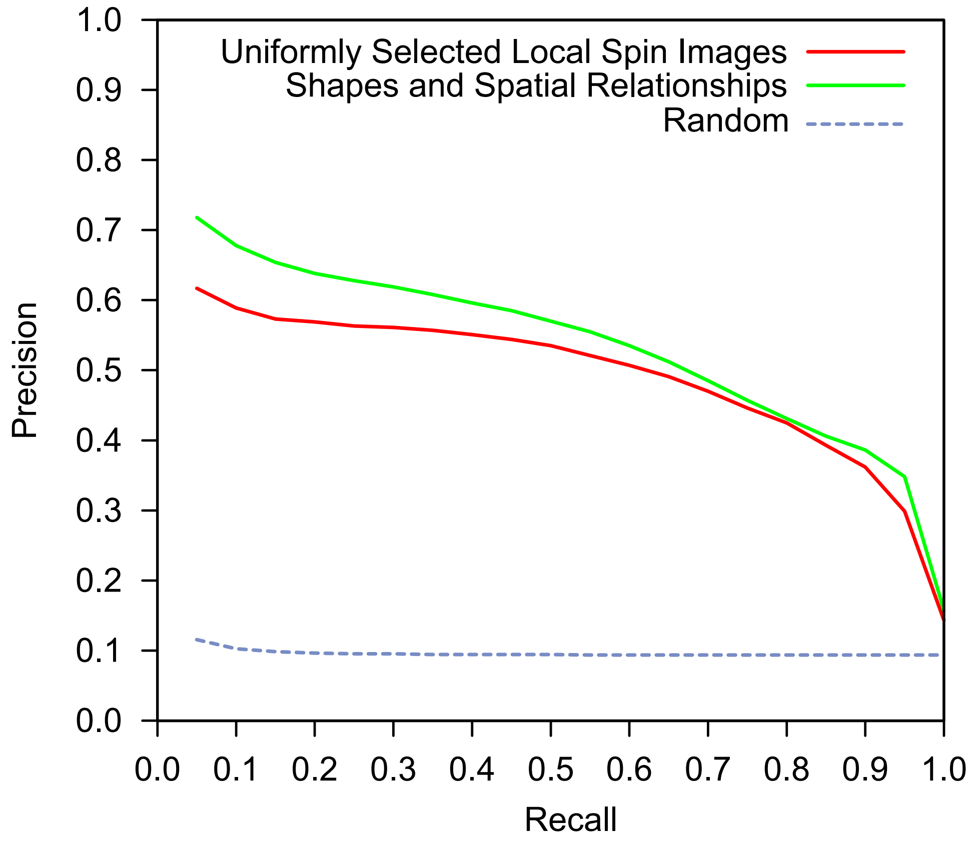


Figure 3.3: **Comparison between our new method and random feature localization.** The new algorithm including shapes and spatial relationships shows superior quality over random feature selection.

3.4.3 Timings

In Table 3.2, we provide information about the time consumption of our approach. All experiments run on an Intel®Core^{TM}2 Quad with 2.33 GHz and 4 GB RAM. Shape detection, training and CDD computation are parallelized using OpenMP. Training and CDD computation are additionally accelerated using an NVIDIA® GeForce®8800. Preprocessing timings include point cloud generation, shape detection and spin image descriptor computation for the shape and spatial relationships based approach and point cloud generation and spin image descriptor computation for the random feature selection based approach, respectively. Training and CDD computation take longer if spatial relationships are taken into account which is due to two reasons: First, feature tuples lead to an additional amount of training vectors. Second, cross-validation must be performed to determine the

	Uniformly Selected Local SI	Shapes and SR
Preprocessing	1 h 15 min	15 h 00 min
Prepr. per object	2.5 s	29.7 s
Training Shapes	44 min	9 h 24 min
CDD computation	55 min	2 h 13 min
CDD comp. per object	2.3 s	5.6 s
Query time	4.8 s	35.3 s

Table 3.2: **Timings.** Preprocessing times are with respect to the whole dataset including 1817 object. CDD computation times are with respect to the test set including 1417 objects. Please note that the large deviation between this preprocessing time for spin image generation and the one shown in Table 2.6 (about 12 times faster) results from the usage of a kd-tree for point range search as well as from parallelization to four cores.

weighting factors α_δ and α_γ.

3.5 Conclusion

In this Chapter, we introduced an improved feature localization technique. Exploiting the fact that man-made objects mainly consist of shape primitives, we used a decomposition algorithm to compute an according segmentation. Features are localized at the according segments. Additionally, we incorporated the relative position of single features as well as the spatial relationship of feature tuples into our retrieval framework. In our experiments we could observe increased retrieval performance for the architecture benchmark compared to our method presented in Chapter 2. Nevertheless, the architecture benchmark still remains a hard task for shape retrieval, which is most probably due to its fine granularity and the resulting low inter-class variance. Even with our improved algorithm, results are still way behind those achieved on the PSB, see Section 2.4.

Limitations While the incorporation of information about the relative position of single features only slightly adds to the CDD computation time when compared to the plain vanilla version from Chapter 2, the consideration of feature 2-tuples increases the time amount quadratically, which is the reason why in this Section we started with $l \leq 32$ descriptors and considered only the largest $l/2$ descriptors for tuple building. Although it would be desirable to take tuples consisting of more than two descriptors into account with additional descriptions of their spatial arrangement, such an approach is currently not feasible due to the ever increasing time consumption for both, training and querying.

Another limitation is the belief in a consistent Z-axis alignment of the models. Although this assumption holds quite well for architectural context objects regarding our experience, there is no guarantee that it generalizes when considering arbitrary objects. This problem could be dealt with by using completely rotational invariant shape descriptors. Accordingly, the description of relative position and spatial relationship would have to be changed. One possibility would be to use the description presented in [FS06] which works for rotational invariant descriptors. Another possibility would be keep the spin image as a descriptor and orient it along the axis through the surface point and the object's center of gravity.

Future Work It should be investigated how the above mentioned limitations regarding the number of feature tuples as well as their size in terms of contained features could be overcome. Instead of systematically considering all n-tuples, we propose to use a sort of importance sampling to keep the tuple numbers low, at least for the query step. The idea is to only consider features that have the potential to lead to a discriminating CDD (one with low entropy) when built into a tuple. This approach is somewhat similar to the method of distinctiveness-driven feature selection presented by Shilane et al. [SF06]. However, we suggest a stricter supervised learning approach. In the following we will briefly outline our idea.

First, the training set must be divided into two parts. The first set is used to train the conditional probability predictions like described above, including single features and feature tuples up to the desired tuple size. After training, CDDs are computed for features and tuples in the second set. For each resulting CDD, the entropy of the class distribution is determined as a measure of distinctiveness. Let us now consider one single tuple size $n = \hat{n}$. Suppose the underlying object contains a total of l features, then each feature is part of $\binom{l-1}{\hat{n}-1}$ tuples. Each feature is then assigned a score computed as the average entropy of all CDDs it contributes to. In the next step, support vector regression (SVR) is used to learn the relationship between single features and the resulting average entropy score. Entropy score assignment and SVR are repeated for all desired values of n. When presented a query object, SVR is used to predict one average entropy score for each feature at each tuple size. Based on the prediction, importance sampling can now be used to draw features for tuple construction according to a probability that reflects the average entropy score distribution of all features in the object. The sampling is to favor features that promise a low entropy for the given tuple size. Note that this approach assumes quite a simplification. Suppose we consider the distinctiveness of a feature tuple as a random variable conditioned on the descriptors, then our method would work best if the underlying features were statistically independent. However, we believe that despite this heuristic, the proposed procedure might be able to improve the retrieval performance and at the same time decrease time consumption, at least for querying.

Acknowledgements We would like to thank Prof. Dr. Björn Ommer for providing insights to his work in [OB07].

CHAPTER 4

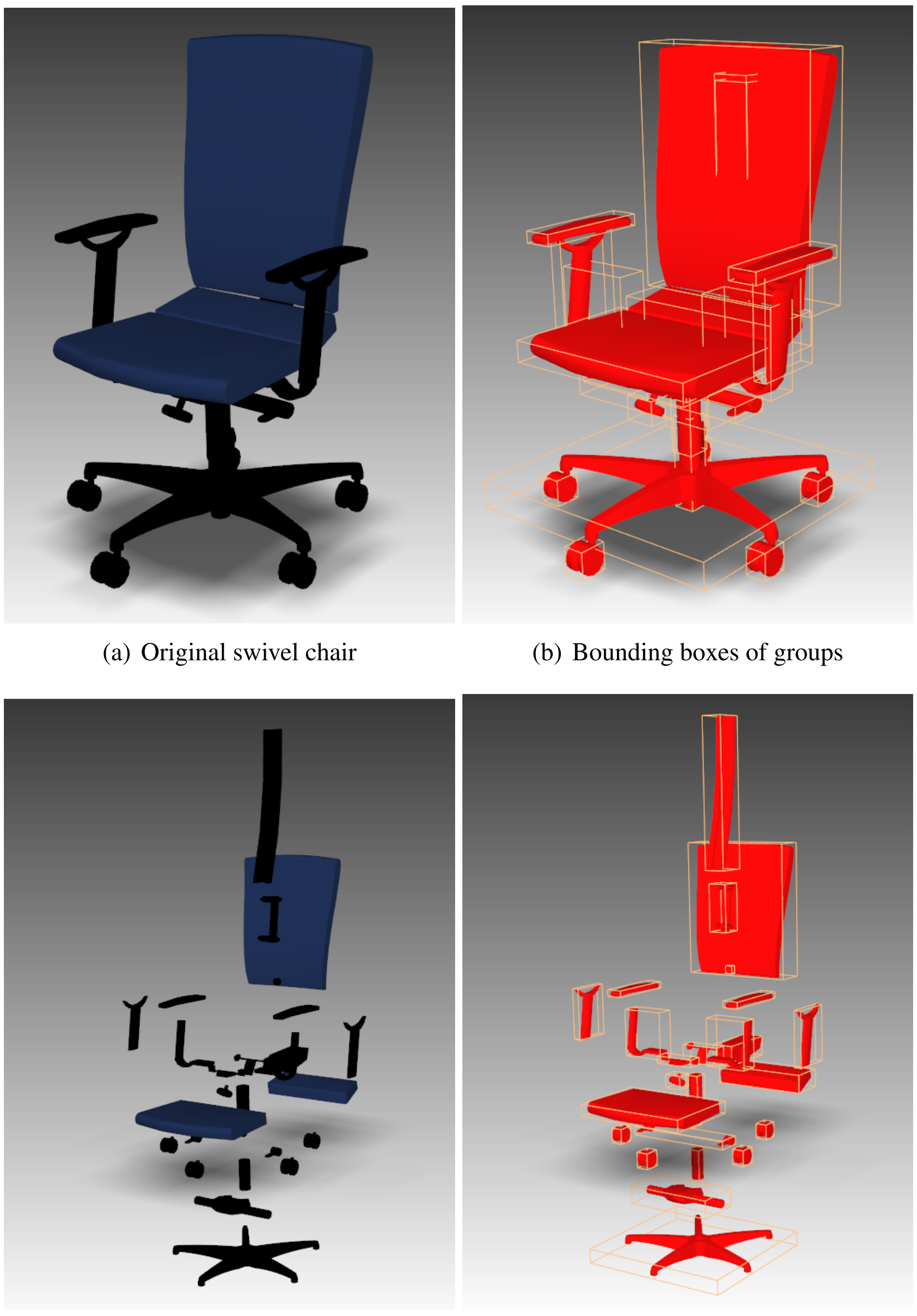
Beyond Shape: Groups, Materials, and Text for 3D Retrieval

4.1 Introduction

The methods for 3D retrieval that have been presented so far in this part of the thesis require only unstructured polygon soups as input, as in general there is no guarantee on the quality of a given mesh. However, regardless of the mesh quality, depending on the underlying manufacturing process, many models contain information exceeding pure geometric properties:

Grouping Several polygons might be assigned to a common group, see Figure 4.1. Depending on the underlying file format, hierarchical groupings might also be supported. Groupings mainly result from two possible causes. First, designers group polygons and existing groups to facilitate the modeling process as well as subsequent rendering, e.g. to apply the same transformation or texturing to all elements of the group. Second, existing (partial) 3D models might have been integrated to the new object. Each such part usually constitutes an additional group. An important aspects of the contained groups arises from the fact that they are not related to certain geometric aspects like e.g. the primitive shapes that we used for segmentation in the last chapter, but that they can also comprise arbitrary polygons that form some sort of *semantic* entity.

Materials Apart from the overall shape, surface materials are the second important property that enables human beings to unambiguously understand an object's function. Regarding our everyday experience, it is quite obvious that in most cases the belonging of a 3D model to a certain class and the materials it is made of are statistically dependent. Although there are only few examples in which the object class uniquely determines the material and virtually none for the opposite case, material properties at least give a hint to possible categorizations of an object.



(a) Original swivel chair (b) Bounding boxes of groups

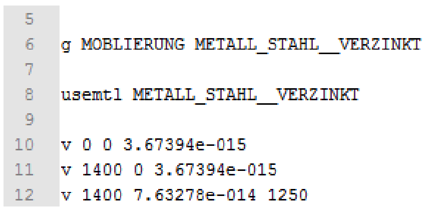
(c) Exploded drawing showing single groups (d) Exploded drawing showing group bounding boxes

Figure 4.1: **Polygon groups inside a swivel chair.** This example of a non-hierarchical grouping is taken from our architecture benchmark described in Section 2.5 (model id 5134).

There is a vast amount of methods to describe the reflectance properties of an object's surface materials. Depending on the underlying file format, 3D objects can include textures, simple Phong shading models [Pho75], more sophisticated bidirectional reflectance distribution functions [NRH+77], or even bidirectional texture functions [DvGNK99].

Textual Annotations Depending on the modeling process, a 3D object might contain a certain amount of textual annotations. These annotations can be stored in comment sections, but they may also be used as group or material identifiers, see Figure 4.2. Annotations result either from user interaction, e.g. to assign semantic meaning to a certain part of the object, or they result from the integration of existing partial models into the new draft. In either case, because of their hopefully semantic nature, such annotations might be helpful in order to identify the underlying object class.

In this chapter we will investigate how these additional features that come along for free with many but not with all 3D models can be used to further improve the performance of our CDD-driven retrieval approach. To this end we will first analyze how the intrinsic grouping performs as a segmentation and feature localization method and compare it to random localization (Chapter 2) as well as to shape-based localization (Chapter 3). In the second step, we describe material properties as well as the textual annotations in the graphics file using simple histogram descriptors. With the class distribution descriptor at hand, we are given a versatile tool that allows the easy incorporation of these descriptors. Furthermore, we will investigate how combinations of different feature localization methods, shape-related and non-shape-related descriptors contribute to improved retrieval performance.


```

g MOBLIERUNG METALL_STAHL__VERZINKT

usemtl METALL_STAHL__VERZINKT

v 0 0 3.67394e-015
v 1400 0 3.67394e-015
v 1400 7.63278e-014 1250
```

Figure 4.2: **Textual annotations in a graphics file excerpt.** In this example extracted from a Wavefront `.obj` file (Architecture benchmark model id 4287), a group identifier (`g [...]`) as well as a material identifier (`usemtl [...]`) contain a very precise textual annotation of the material itself, obviously *zinc coated steel / metal*.

Note that although most approaches on 3D object retrieval focus on shape-based similarity determination, the usage of material information and textual annotations is no entirely new concept but has already been proposed in several papers, see e.g. [SH95, Löf00, TS04, LZL+12] (color/material) and [MKF04] (textual annotations). Unfortunately, only [LZL$^+$12] and [MKF04] provide evaluations on standard benchmarks (McGill 3D Shape Benchmark and PRECISE in [LZL$^+$12] and Princeton Shape Benchmark in [MKF04]) that additionally investigate the performance gain caused by the usage of color/material and textual information, respectively. In both papers, a slight increase in the retrieval performance is reported in comparison to sole geometric descriptors.

4.1.1 Generalization Issues

When regarding the results of this Chapter, one should be aware of the fact that the above mentioned groupings and textual annotations are highly dependent on the modeling human being. For example, how to exactly group the parts of a swivel chair like the one depicted in Figure 4.1 is up to the designer. The same object might be grouped in two completely different ways by two different persons. If the objects in certain classes of a model database only originate from a relatively small group of designers (compared to the number of models), then it is very likely that CDD computation overfits towards specific grouping or annotation schemes. The trained CDD computer might therefore not generalize well when presented new data from different designers. Note that this problem is not due to inherent design flaws of our method but it arises from the available data which might be biased regarding groupings and textual annotations. Although this problem might also occur up to a certain degree with shape and material-based descriptors, we believe that in these cases it is more or less negligible, as in contrast to grouping or annotating, there seems to be more or less common sense about shape and material properties of certain object classes, even among different designers.

4.1.2 Contribution

Please note that the Bag-of-Words descriptors for textual annotations and particularly the material histogram descriptor that we use should not be regarded as the latest state of technology in either field of research. Instead of tweaking our shape retrieval system to the last bit, our goal is rather to show the versatility of our approach in terms of plugging in different segmentations and descriptors. The contribution of this chapter is therefore rather exploitation and evaluation of the methods presented in Chapter 2 and 3 than any fundamentally new ingredients.

4.2 Intrinsic Groupings for Feature Localization

In Section 3.3.1 we introduced primitive shape decomposition as a means for model segmentation and subsequent spin image localization. This approach can

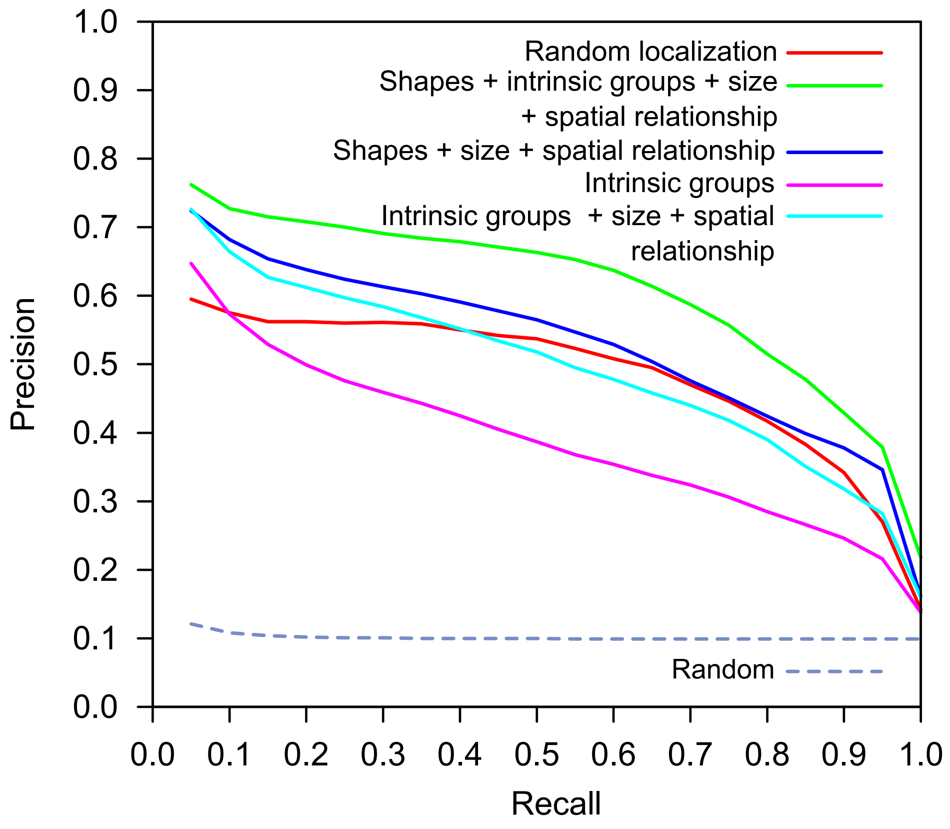


Figure 4.3: **Retrieval results using intrinsic grouping.** Please see Section 4.2 for detailed explanations.

be generalized to arbitrary segmentations like the intrinsic grouping in a straight forward way. We only consider non-hierarchical groupings in our experiments, as the available data from the architecture benchmark does not contain any hierarchies. Analogously to our experiments in Chapter 3, we compute a Z-axis aligned spin image for the support points of each group. We thereby consider up to 31 groups. If more groups are available, we only take the largest ones into account. Additionally, we compute one global Z-axis aligned spin image that comprises the complete model, as there is no guarantee that all parts of the object are assigned to a group. Note that if there is no intrinsic grouping available, this global descriptor is the only one characterizing the model.

In Figure 4.3 and Table 4.1 we depict the results of our experiments. For comparison, we include the performance graphs of the methods described in Chapter 2 (Random localization 🟥) and Chapter 3 (Shapes + size + spatial relationship 🟦). In our first experiment (Intrinsic groups 🟪) we do not include information about spatial relationships, feature tuples, or feature size into the descriptor to ren-

Method	1-NN	1-Tier	2-Tier	DCG
Random localization	0.592	0.482	0.670	0.774
Primitive Shapes with spatial relationships	0.763	0.523	0.670	0.810
Groupings no spatial relationships	0.750	0.384	0.511	0.744
Groupings with spatial relationships	**0.807**	0.486	0.608	0.793
Primitive Shapes and groupings with spatial relationships	0.790	**0.596**	**0.729**	**0.842**

Table 4.1: **Comparison of different approaches for descriptor localization and consideration of features relationships.** Combining orthogonal localization techniques and incorporation of spatial relationships provides superior retrieval performance.

der it more comparable to the random localization. For small recall values, this localization technique performs slightly better, but with increasing recall, random localization quickly gains the upper hand. Taking a closer look at the data as well as the feature localization, this behavior turns out to be quite reasonable. First, the architecture benchmark contains some classes that include objects that differ only very little from each other. For example, there are tables that are basically identical except for the dimensions of the table plate. Similarity determination between such objects benefits from the very similar grouping and the subsequent deterministic descriptor localization in contrast to the randomized one. Second however, the result shows that the lack of knowledge about the spatial feature relationships decreases the retrieval performance. Although not modeled explicitly, the spatial relationship is implicitly characterized by the relatively large and vastly overlapping randomly localized spin image descriptors leading to better generalization results for larger recall values. This fact becomes even more evident when augmenting the descriptor by information about the spatial relationship and considering descriptor tuples (Intrinsic groups + size + spatial relationship 🟦), using the exact same technique and parameters as described in Chapter 3. Although not reaching the result achieved with the primitive shape-based descriptor localization, the retrieval performance is highly increased.

Assuming that the geometrically motivated descriptor localization using primitive shapes and the rather semantically motivated one using intrinsic groupings both encode somewhat orthogonal information, we finally combine both descriptors (Shapes + intrinsic groups + size + spatial relationship 🟩). The resulting descriptors provide the best retrieval results of all methods so far.

Method	1-NN	1-Tier	2-Tier	DCG
Random localization	0.592	0.482	**0.670**	0.774
Primitive Shapes with spatial relationships	**0.763**	**0.523**	**0.670**	**0.810**
Material information	0.577	0.275	0.392	0.673

Table 4.2: **Material descriptor evaluation.** The material descriptor alone is not able to reach the performance of shape-based descriptors.

4.3 Material Descriptors

As already mentioned in the introduction, there exists a vast amount of more or less complex and sophisticated methods to represent materials on 3D objects. In the case of our architecture benchmark, the only available material descriptions are textures and simple Phong models. There has been intensive research on texture-based material retrieval and classification, see e.g. [VZ09, LSAR10]. Unfortunately, in our case textures are only available for a very small number of objects, rendering this type of information more or less useless in our scenario. However, the amount of models at least partially containing Phong model assigned surfaces is quite large.

Following our previously described approach it would be possible to treat the Phong coefficients of each single polygon together with its relative size as one local descriptor which can be used to infer a corresponding CDD. However, it appears that these local reflection coefficients can hardly separate any object classes at all. Instead, we construct a descriptor that characterizes the diffuse color distribution over the whole object. To this end we build a histogram containing $3 \times 16 + 1$ bins. Each of the three color channels is divided into 16 bins. The additional bin is used for non-shaded polygons. Polygons contribute to the individual bins according to their diffuse color and with respect to their relative size. The resulting histogram descriptor is then converted to a corresponding CDD. The retrieval performance of the material descriptor is depicted in Figure 4.4 (Material information 🟩) and Table 4.2. From the plot it becomes clear that material alone is a rather poor descriptor for 3D object retrieval. However, in Section 4.5 we will show its ability to improve the retrieval performance when combined with shape-based descriptors.

4.4 Textual Annotations

Deriving a CDD from each single textual annotation would require the usage of elaborated string kernels, see [Gär03] for an overview. Analogously to the mate-

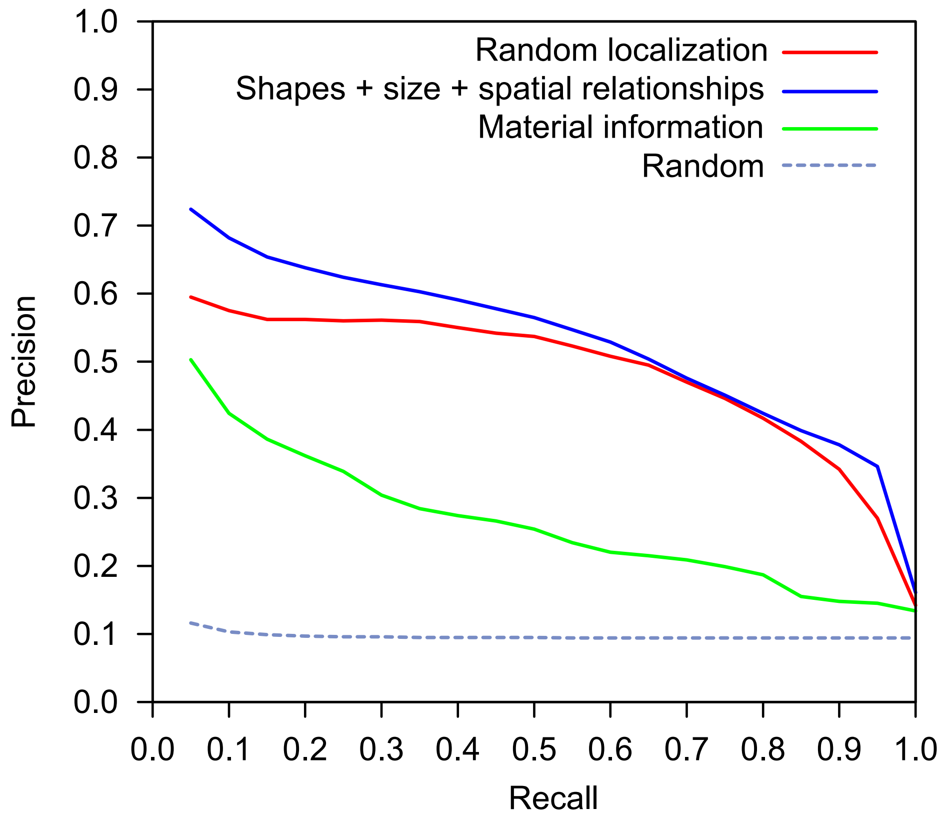


Figure 4.4: **Retrieval results using material information.** Please see Section 4.3 for detailed explanations.

rial description, we follow a simpler approach and characterize the global distribution of textual annotations inside a 3D object using a Bag-of-Words approach [Har54]. The idea is to first generate a codebook of words. For each document, the number of occurrences of all codebook words is determined and written into an according histogram. Finally, this histogram is normalized such that its entries sum up to one. The resulting normalized descriptor can be easily compared using standard kernels for vector data. Although there have been considerable research efforts on efficient codebook selection (see [YP97] for an excellent overview), this topic is considered less important today. First, modern hardware can easily handle feature vectors containing several hundreds or even thousands of entries. Second, with the introduction of highly efficient classification methods like SVMs it has become clear that in many cases even sophisticated feature reduction methods do not increase the performance [Joa98]. We therefore renounce feature selection in our experiments, also in consideration of the fact that the total amount of distinct annotations in the dataset is relatively small.

Method	1-NN	1-Tier	2-Tier	DCG
Random localization	0.592	0.482	**0.670**	0.774
Primitive Shapes with spatial relationships	**0.763**	**0.523**	**0.670**	**0.810**
Textual annotations	0.562	0.283	0.404	0.684

Table 4.3: **Bag-of-Words descriptor evaluation.** The descriptor is not able to reach the performance of shape-based descriptors.

We convert all textual annotations to lower case and split words at any non-letter symbol. We then remove all words that are obviously only due to the usage of certain modeling or conversion tools, like e.g. `Autocad` or `Deep Exploration`. For the training set, we arrive at a codebook containing 73 distinct words. The histograms describing each 3D model are computed by counting the exact codebook word occurrences[1]. CDD descriptors are trained and computed using an RBF kernel again. The results of our retrieval experiments using textual annotations are depicted in Figure 4.5 and Table 4.3. The retrieval performance is quite similar to that achieved with the material descriptors. We believe that this mainly results from the fact that a lot of textual annotations are used to describe materials, such that both descriptors encode similar, non-orthogonal information. This hypothesis is supported by our findings gained in the combination experiments described in the next Section.

4.5 Combining Shape, Material, Text, and Different Localization Strategies

In our final experiments we try to combine the presented feature localization strategies and the global descriptors characterizing material and textual annotations. To this end we must introduce a weighting scheme that balances the influence of the single global descriptors and the set of local ones. The idea is to treat the global descriptors as if there were many instances of them. We therefore define the modified descriptor

$$\mathcal{D}^m(x) \;=\; \frac{\bigotimes_{i=1}^{m} \mathcal{D}(x)}{\sum_{j=1}^{|\mathcal{C}|} \left(\bigotimes_{i=1}^{m} \mathcal{D}(x)\right)[j]},$$

where m denotes the number of virtual instances of descriptor x. This parameter can be automatically tuned during preprocessing and can then be applied to the

[1]The usage of more sophisticated methods including approximate (sub)string matching as well as mapping schemes for synonyms from different languages might additionally increase robustness and retrieval results.

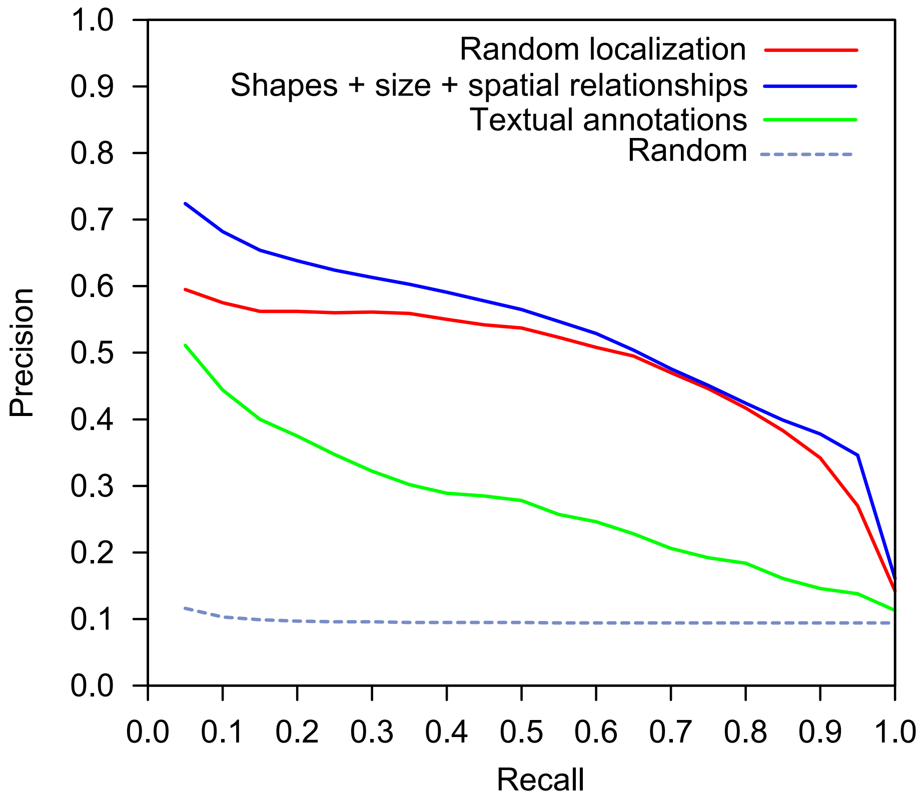


Figure 4.5: **Retrieval results using textual annotations.** Please see Section 4.4 for detailed explanations.

test data. The results of our combination experiments are shown in Figure 4.6 and Table 4.4. Adding textual information to our previously best-performing approach using a combination of primitive shape-based and intrinsic groups-based feature localization further improves the retrieval performance. However, adding the material information does not increase the accuracy any further, which we believe is due to the fact that the information encoded by material and Bag-of-Words descriptor is not that orthogonal.

4.6 Conclusion

In this Chapter we exploited our findings from Chapter 2 and 3 by using information that exceeds pure shape features. By incorporating material and textual annotations into our retrieval framework we were able to increase the overall performance of our system. Additionally, we took advantage of model-inherent

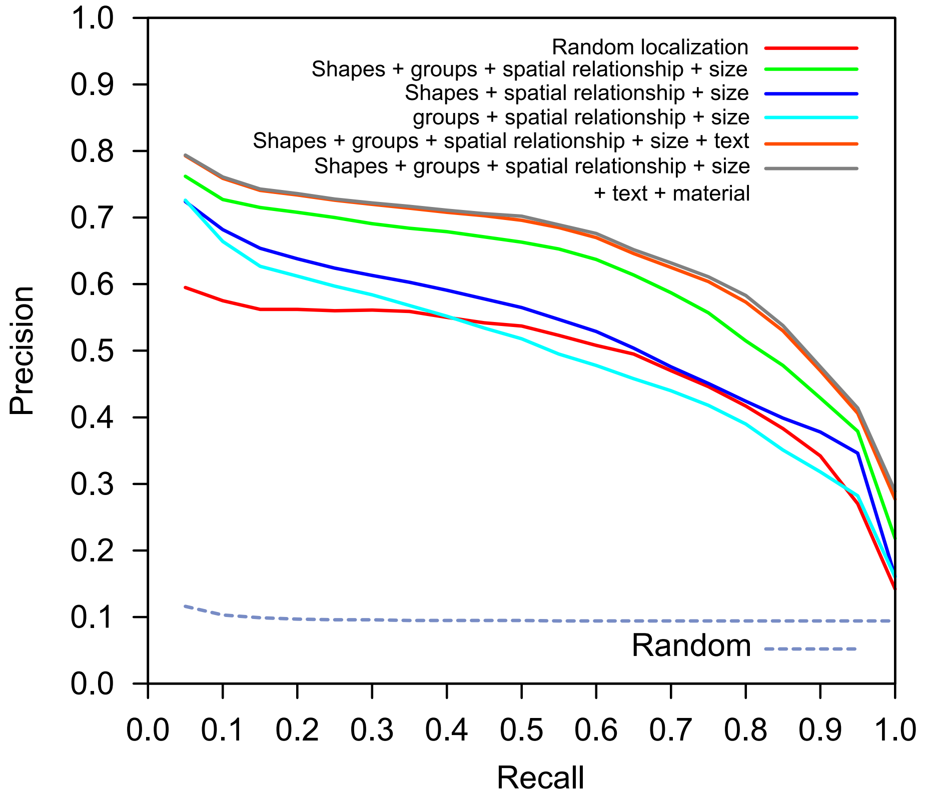


Figure 4.6: **Retrieval results using combined descriptors.** Please see Section 4.5 for detailed explanations.

semantics-driven groupings which also added to our overall results. Despite these improvements one should keep in mind that the data and the trained CDD computers are potentially biased towards designer-specific groupings and annotations as described in Section 4.1.1. Additionally, regarding the overall results on the architecture benchmark it is obvious that although our methods brought quite some progress, the shape retrieval problem for such fine-grained data still remains an open problem.

Future Work An interesting challenge would be to investigate the influence of hierarchical groupings on the retrieval performance if such data was available. Additionally, for textured models, the influence of more sophisticated material descriptors like the ones presented in [VZ09] should be analyzed. However, before such research can be conducted it is necessary to collect adequate data and build an according benchmark. Regarding our experience with the setup of our architecture benchmark we believe that acquiring a collection that homogeneously

Method	1-NN	1-Tier	2-Tier	DCG
Random localization	0.592	0.482	0.670	0.774
Primitive Shapes + spatial relationships	0.763	0.523	0.670	0.810
Groupings	0.750	0.384	0.511	0.744
Groupings + spatial relationships	0.807	0.486	0.608	0.793
Primitive Shapes + groupings + spatial relationships	0.790	0.596	0.729	0.842
Primitive Shapes + groupings + spatial relationships + text	**0.832**	0.629	**0.759**	0.858
Primitive Shapes + groupings + spatial relationships + text + material	0.830	**0.633**	0.758	**0.860**

Table 4.4: **Combination of all previously described features.** Combining orthogonal localization techniques, incorporation of spatial relationships and additional information about material or textual annotations provides superior retrieval performance.

provides high quality texturing and semantic grouping is going to be quite a challenge, especially if it is supposed to be freely available for other researchers.

Part II

Graph-based Shape Retrieval for 3D Architectural Building Models

CHAPTER 5

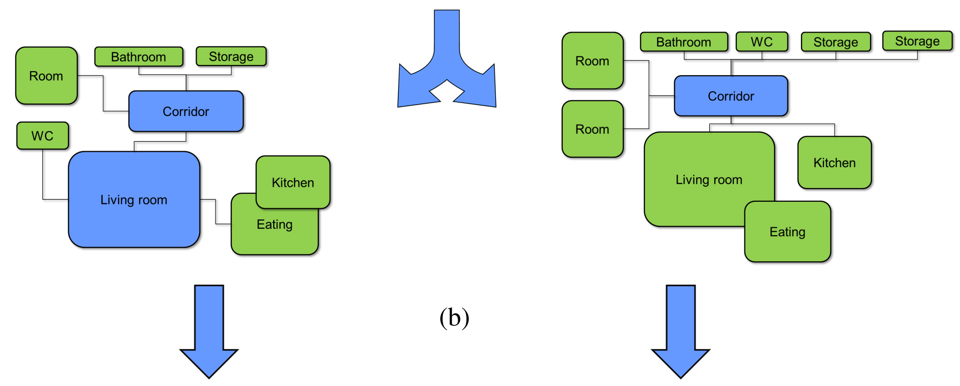
ANALYZING AND INDEXING BUILDING MODELS

5.1 Introduction

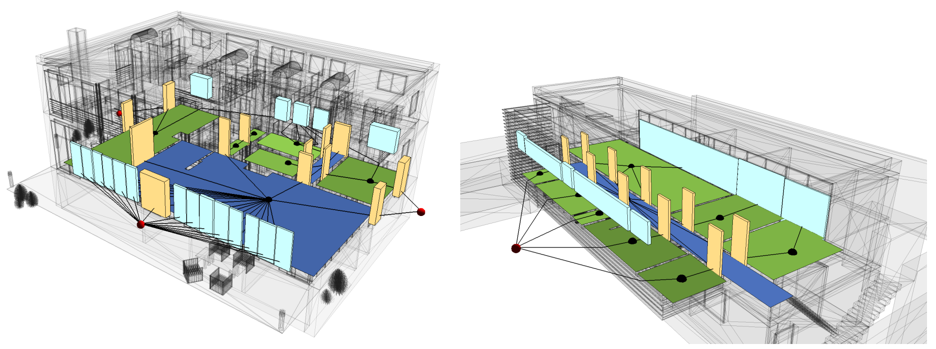
In the previous chapters we introduced shape retrieval methods tailored to *context objects* representing building elements, furnishing, and environment elements. In the second part of this thesis we focus on the building models themselves. We introduce means to analyze their structure and subsequently develop efficient methods for retrieval. While for context object retrieval the focus was on vector-based descriptors to enable shape retrieval, in the following second part we will mainly rely on structured shape representations, i.e. graphs. This is due to the reason that for architectural drafting, the spatial and topological arrangement of rooms and stories is most important. Such relationships can hardly be described using vector-based descriptors but rather require structured data representations. An important aspect in building retrieval is the interior building topology. To understand the development of building topologies in architects' practice, we depict an early part of the planning process in Figure 5.1. Starting the preliminary design of a building with a given schedule of spaces (5.1(a)), architects arrange rooms and their connections in graphs representing topological structures (5.1(b)). These topologies strongly characterize buildings and express their internal organization [Mit90]. In the next step, these abstract graphs are used as templates to develop concrete 2D floor plans, see Figure 5.1(c) (here overlaid with the 3D building models). As displayed in the figure, different floor plans can be generated out of the same schedule of spaces through early decisions concerning the topology and outer parameters like global building form. Taking all desirable drafting opportunities into consideration, floor plans are the most unambiguous means to show the spatial organization (development structure, topology and disposition of rooms, [Sch04]). Therefore they are a major ingredient for architectural drafting. Eventually, the structural patterns formed by rooms and their connections refer to building zones (e.g. private or public) and eventually to building types [MB97].

Dwelling, $140\ \text{m}^2$					
Living room	up to $60\ \text{m}^2$	WC	$3\ \text{m}^2$	Storage/Utility	$6\ \text{m}^2$
Room(s)	$12 - 14\ \text{m}^2$	Kitchen	$10 - 14\ \text{m}^2$	Corridor	as required
Bathroom	$7 - 10\ \text{m}^2$	Eating	$12 - 18\ \text{m}^2$		

(a)



(b)



(c)

Figure 5.1: **Architectural design chain.** a) Schedule of spaces. b) Two possible arrangements of spaces. c) Resulting buildings with floor plans.

5.2 Room Connectivity Graphs

Considering the importance of floor plans for describing and understanding architecture, it is obvious that state-of-the-art 3D shape retrieval methods solely based on global or local geometric features are less suited for the domain of architectural building models, as they cannot represent their topological structure.

To overcome these drawbacks, we propose attributed room connectivity graphs (RCGs) as floor plan descriptors for building models. To our best knowledge, the concept of simple non-attributed room connectivity graphs was first mentioned in the area of robotics for modeling navigation planning problems [Lau83]. We pick up this general concept and tailor it to the requirements of building and floorplan retrieval. An RCG $G = (V, E)$ consisting of nodes V and edges E characterizes a single story or a whole building in terms of the topology of the underlying floor plans. Rooms are represented by attributed nodes $v_i \in V$. There is a large variety of interesting room properties that might be useful for building retrieval and classification. It ranges from rather low-level geometric attributes like e.g. area, window area, height, perimeter, or a geometry descriptor characterizing the basic shape of the room, up to high-level semantic attributes like room type (corridor, distributor, balcony, etc.). We collect all important room attributes in a vector and attach it to the room node. Apart from the room nodes, each RCG contains exactly one additional node v_o that represents the *outside world*. This node is especially important to characterize which rooms allow access to the surrounding environment. Edges $e_j \in E$ represent connections between rooms. These edges can be assigned low-level attributes like width, height, geometry descriptors characterizing the basic shape of the connection, or high-level attributes like connection type (door, window, staircase, elevator shaft, etc.).

5.2.1 Node Attributes

We distinguish two kinds of low-level room attributes, *room types* and *room properties*. The room type attribute $a_{type}(v)$ is a value stating whether a node v represents a room or the outside world. It reads

$$a_{type}(v) = \begin{cases} room, & \text{if } v \text{ represents a room,} \\ world, & \text{if } v \text{ represents the outside world, i.e. } v = v_O \end{cases}$$

Room properties are only assigned to the actual rooms excluding the outside world node. Let us denote the set of nodes that does not contain the outside node v_o by $V \backslash \{v_o\}$. For each room node $v \in V \backslash \{v_o\}$, the vector

$$a_{prop}(v) = \begin{pmatrix} a_{prop}(v)[1] \\ \vdots \\ a_{prop}(v)[k] \end{pmatrix}.$$

represents scalar-valued room properties $a_{prop}(v)[i] \in \mathbb{R}$.

5.2.2 Edge Attributes

Edges can represent three different kinds of connections: doors, windows, and vertical connections including staircases and elevators. One can additionally distinguish connections within the building and connections that lead to the outside.

Accordingly, the edge type attributes read

$$a_{type}(e) = \begin{cases} door_{inside}, & \text{if } e \text{ connects two rooms by a door,} \\ door_{outside}, & \text{if } e \text{ connects a room to the outside world by a door,} \\ vertical, & \text{if } e \text{ connects rooms of adjacent stories,} \\ window_{inside}, & \text{if } e \text{ connects two rooms by a window,} \\ window_{outside}, & \text{if } e \text{ connects a rooms to the outside world by a window.} \end{cases}$$

Note that it would be possible to additionally characterize edges by their properties (e.g. window size) in a similar way as the room property vector. However, in our work with experts from the architectural domain we came to understand that vertical connections and doors play a far more important rule for understanding the structure of a building than windows. In fact we will not use edges representing windows in our retrieval experiments but will instead include them as room properties.

We use the representation of a 3D building model by attributed RCGs as a starting point for retrieval and classification. After a review of the related work on graph-based 3D shape retrieval and graph similarity computation in general, we will describe the necessary building blocks for simple retrieval with RCGs, which are:

- **Extraction of RCGs** We propose a method for error-tolerant extraction of RCGs that is robust towards modeling errors. It is applicable to building models in almost any format as it only relies on polygon soups.
- **Simple room structure retrieval** We show how simple room structures can be found in a building model according to the extracted RCG. Retrieval is based on node and edge attributes characterizing the type of room or connection (e.g. door or window).

Although in the following we will extract RCGs from existing 3D models using only geometric information contained in the associated polygon mesh, our concept of RCGs is completely uncoupled from the underlying object representation. As source, one might as well use a 2D sketch or a high-level BIM model that enables easy extraction of rooms, stories, and connecting elements along with high-level attributes.

5.3 Related Work

The idea behind graph-based 3D shape retrieval methods is to characterize an object according to the linkage (graph edges) of its constituting parts (graph nodes).

Depending on the particular method, the semantic level of such parts ranges from low-level triangles over higher-level shape primitives like cylinders and spheres up to high-level entities that consist of a textual description rather than of pure geometry. The linkage usually represents spatial proximity, i.e. it describes that certain parts are either touching, close to each other, or connected in a certain way. Graph-based representations can be relatively easy made invariant under translation, rotation, scaling, and in certain cases also isometric transformation. Additionally, they at least theoretically enable partial retrieval, which boils down to subgraph matching. While these are obvious advantages over most vector-based representations, the successful usage of graph-based methods requires to overcome two major obstacles, which in our opinion is the main reason for their limited success and decreasing impact in the 3D shape retrieval community:

- **Robust extraction** The extraction of features that constitute nodes must be extremely robust towards small changes in the geometry, noise, and the underlying representation (e.g. triangulation level), as adding just a single node or edge can dramatically change the entire graph topology and thereby faking structure that is not really present. Two similar objects might therefore result in quite different graph representations. In comparison, BoF methods can easily stand additional local features, as each one of them only marginally changes the resulting histogram distribution. Note however that there are some application scenarios in which the node extraction is extremely reliable, for example when using Constructive Solid Geometry (CSG) representations.

- **Similarity computation** Computing exact matchings by determining subgraph isomorphisms is known to be NP-complete [Coo71]. However, in case nodes are added attributes that strongly decrease the number of compatible pairs in the query and the target graph, the resulting early exits during solution tree exploration can render this problem tractable, at least for searches in rather small databases. Unfortunately, considering the above described problems of insufficiently robust node extraction, exact subgraph matching is useless anyway in most cases, as no match would exist due to clutter. Additionally, there are only few scenarios in which exact matchings are desired. Instead, fuzzy distance measures are required. Graph edit distances are considered a loophole for this purpose, but computing them is NP-hard as well [ZTW$^+$09]. They allow insertion and deletion of nodes and edges in either the query or the target graph, thereby enabling to determine a similarity value in the absence of exactly overlapping structures. However, the computational efforts required are high, which leads researchers to explore new methods for similarity determination. This includes graph kernels and methods that are based on comparison of graph statistics represented as vectors instead of the graphs themselves.

Retrieval methods relying on graph-based 3D object representations are usually divided into three groups, including *model graph*-based, *skeleton graph*-based, and *Reeb graph*-based methods. Please note that our newly introduced RCGs best fit into the category of high-level attributed model graphs. We will briefly introduce the most relevant methods of each group along with their particular graph comparison methods. Regarding the older methods, we mainly rely on the information contained in the survey by Tangelder et al [TV08].

5.3.1 Model Graphs

Model graph-based representations can usually be derived from 3D solid models, which mainly includes boundary representations (B-rep) and Constructive Solid Geometry (CSG). Elinson et al. [ENR97] construct attributed graphs from CAD models that encode machining features. The high-level textual node and edge attributes (e.g. `drilling/milling feature, cutting tool radius`) constitute *signatures* that are compared using isomorphism checks. A similar approach is presented in [CR01], where a *model dependency graph* is constructed and a heuristic largest common subgraph algorithm is used for similarity determination. Note that both approaches do not directly incorporate the underlying geometry as graph nodes but rather use attached high-level attributes. El-Mehalawi et al. [EMM03a, EMM03b] also extract graphs from CAD models, but the node attributes are more shape-related than in the preceding two approaches. A simple graph retrieval method using vectors that subsume statistics about certain graph properties is proposed, like e.g. the number of nodes of a certain type. For a more sophisticated inexact comparison, a heuristic using clique matching is presented. McWerther et al. [MPSR01] use similar node and edge attributes. They propose two methods for graph comparison. On the one hand, the *Invariant Topology Vector* (ITV) is introduced, representing statistical properties of the graph, e.g. the minimum and maximum degree. ITVs of two different graphs are compared using the Euclidean distance. On the other hand, the Eigenvalues of the graph Laplacian are used for comparison. By subsequently subdividing graphs using the Fiedler vector [Fie73], partial similarity is analyzed. Wang et al. [WLHZ10] use an approximation for graph edit distances [RB09a] to compute the similarity between attributed graphs. In contrast to the previously described approaches which lack a decent evaluation, a pre-classified database of 200 models is used to estimate the retrieval performance in terms of the standard methods described in Section 1.5.3. The new method shows superior results when compared to ITV and graph spectra, but also when compared to non-graph-based methods like light field descriptors

[CTSO03].

The methods described so far have in common that they are tailored to high-quality CAD data which renders the robust node and edge extraction process almost trivial. However, in many scenarios only low-level representations are available, including polygon soups and point clouds. There exist several approaches that try to overcome this drawback by extracting higher-level geometric primitives from low-level representations to subsequently construct a model graph. Zuckerberger et al. [ZTS02] use different segmentation strategies to partition watertight meshes. The best fitting basic shape (sphere, cylinder, cone, or plane) is assigned to each resulting patch, and neighboring patches are connected with edges. Verma et al. [VKH06] present a method to recognize roof shapes in aerial LIDAR data. A region growing approach is used to detect planes in the point data. Roof-topology graphs are defined to describe configurations of planes for some simple building forms shaped like I, L and U. These configurations are searched for in the set of the detected planes. In a second step, the detected simple shapes are extended to more complicated forms according to the plane configurations in the point-cloud. A similar approach is described in [SWWK08], where point clouds are partitioned into geometric primitives (planes, cylinders, spheres, cones, and tori) using a RANSAC method. Neighboring primitives are connected via edges. Previously defined proxies can then be search for in the resulting graph. Similar approaches have been proposed in the field of robotics to tackle the grasping problem, see e.g. [NSB+12].

5.3.2 Skeleton Graphs

The medial axis of a 3D shape is the set of points located at the centers of those spheres touching the boundary of the shape at at least two locations. The point set forms a skeleton graph that captures the essential geometrical and topological properties of the underlying shape. Dead ends and junctions constitute nodes in the graph, edges are added according to the nodes' respective connectivity on the skeleton. Skeletal graphs are mostly constructed by transforming a model into a volumetric representation like a voxel grid and by subsequently thinning the shape until only a skeleton is left. However, some methods also work directly on the polygon mesh. There exists a huge amount of papers proposing methods for skeleton extraction, for the most recent and comprehensive surveys on this topic we refer to [CSM07] and [BAB+08].

Sundar et al. [SSGD03] construct several skeletons by varying the thinning parameter which results in a hierarchical structure. Additionally, the graphs are made acyclic by only taking their minimum spanning tree into account. Each non-terminal node is assigned a vector that contains the eigenvalues of the graph Laplacian associated with the subgraph that is rooting in the particular node. Coarse similarity determination between graphs is conducted using these spectral signatures to restrict the search space for the subsequent graph matching process. Lou et al. [LJI$^+$03] attribute the extracted skeleton graph with additional properties (e.g. edge/loop) and use decision tree-based graph matching for similarity determination. An improved version of this method using multi-scale hierarchical skeleton graphs can be found in [IJL$^+$04]. There, local shape information is added to the graph representation and a genetic algorithm is used for similarity determination. The method by Brennecke et al. [BI04] uses a polygonal mesh instead of a volumetric representation to derive skeleton graphs. Similarity is computed using maximum common subgraphs. Cornea et al. [CDS$^+$05] represent the skeleton graph as a point set and additionally store the distance to the shape boundary for each point. Similarity is computed using the earth mover's distance (see Section 2.2). In contrast to the before mentioned skeleton-based methods, Cornea et al. evaluate their approach using a subset of the Princeton Shape Benchmark (PSB) containing 99 classes. Comparing the results to those achieved with various global shape descriptors in [SMKF04], this graph-based approach shows a rather poor performance that except for the 1-NN value is only slightly better than that of the D2 descriptor [AKKS99]. Interestingly, the 1-NN value is higher than in all experiments in [SMKF04]. However, it should be noted that the results are not completely comparable without further ado due to the differing PSB subsets that were used in the experiments. Siddiqi et al. [SZM$^+$08] assign a saliency value to each segment of the skeleton graph that depends on its reconstruction capability. Starting with the most salient segment, an acyclic directed graph is extracted. The graph Laplacian spectrum of the subgraph rooting each node is computed, which is quite similar to the method described in [SSGD03]. The sums of the eigenvalues of each spectrum are subsumed in a vector that serves for fast comparison between different graphs. The top-ranking results are further refined using a combination of bipartite graph matching and a greedy recursive algorithm. To this end, node similarity is determined according to the mean curvature histogram computed over the voxels in the particular segment. The method is evaluated using a subset of the McGill database for articulated objects. The results suggest that it does not outperform spherical harmonics (SH) descriptors or shape distributions. Additionally it should be noted that far better results have been achieved on this particular dataset using non-graph-based methods, see e.g. [LGF$^+$10]. Hayashi et al. [HRNS11] consider pairs of points lying on the skeleton and use feature distributions (e.g. pairwise distance, maximum distance to boundary) to compare two shapes. Bai et al. [BLYL13] characterize the shortest paths between all pairs of skeleton end points by a vector of fixed length that encodes the distance to the shape boundary along the path. Given two shapes, the similarity for all combination of end points is computed according to the vectors describing the attached shortest paths. An optimal matching of the two end point sets is subsequently computed using the Hungarian method.

5.3.3 Reeb Graphs

The term *Reeb graph* denotes the connectivity of level sets originating from a Morse function that is defined on a manifold. Each connected component of the level set represents a node. Spatially neighboring nodes (i.e. those originating from adjacent level sets) are connected by an edge. Hilaga et al. [HSKK01] make use of geodesic distances as a function for level set computation, which is especially suited for articulated objects, as the geodesics are isometric invariant. For each point, they compute the sum of geodesic distances to all other points on the shape. A multiresolution Reeb graph (MRG) is constructed from this function, i.e. the codomain of the function is dissolved using subdivision schemes ranging from coarse to fine. All nodes are assigned attributes characterizing the area of the part of the shape that is associated with the node and the range in which the length of shortest path originating from this node vary. A heuristic is used to compare two graphs starting at the coarsest and finally iterating to the finest level. Bespalov et al. [BRS03] exhaustively evaluate Hilaga's approach using solid CAD models. Their findings include that MRGs are prone to topological changes in general and also suffer from low-quality mesh representations with bad connectivity. Biasotti et al [BMM$^+$03] investigate alternative functions for constructing the level sets including Euclidean distances and geodesic distances from curvature extrema. They use error tolerant graph isomorphism for similarity determination. However, the brief evaluation does not provide any conclusive results regarding the superiority of one distance function. Tung et al. [TS04] introduce augmented Reeb graphs that include additional topological (consistency check regarding wether parents of the neighbors of two nodes are also matched in higher resolution level) and geometrical attributes (relative node position, volume, color, chords, curvature). An evaluation of this method can be found in the SHREC 2007/2008 tracks on articulated watertight models [VtH07, BA08]. The method provided comparatively good results in both tracks, proving superior when compared to several other feature-based approaches. In [BMSF06], Biasotti et al. focus on partial matchings using extended Reeb graphs which incorporate an additional SH descriptor that describes the shape associated to each node. A similar approach relying on pose-oblivious shape signatures instead of the SH descriptors is presented in [AK11]. Partial matching is also the focus of the work by Tierny et al. [TVD09]. They robustify the Reeb graph extraction by getting rid of structural distortions and achieve superior retrieval results when compared to [BMSF06].

5.3.4 Summary

Taking an overall look at the graph-based methods it can be noted that in contrast to feature-based approaches, many algorithms lack a decent evaluation regarding their retrieval performance. However, it seems that careful and robust extraction of nodes and edges can lead to powerful methods that are able to compete with feature-based methods, or might even lead to a better performance, like e.g. in the case of Wang et al. [WLHZ10].

5.4 Room Connectivity Graph Extraction

There exists a large variety of architectural 3D building models regarding quality of modeling, formats, structuring, and semantic annotations. It ranges from pure unstructured polygon soups without any semantics provided in simple triangle mesh formats up to highly structured BIM models containing semantically annotated hierarchies of building elements. Our goal is to provide a method for RCG extraction that is applicable to as many building models as possible, regardless of quality, format, and structuring. Our approach is therefore based on quite weak assumptions about the object. It only requires an unstructured polygon soup, which can be extracted from virtually any 3D representation. Additionally, the units of measurement of the model as well as the coordinate plane corresponding to the ground must be known. The extraction of RCGs is then conducted in three steps, including *segmentation into stories*, *room detection*, and *detection of connections between rooms*.

5.4.1 Automatic Story Segmentation

Assuming flat rooftops, building stories are in general bordered by a flooring and a ceiling, both of which consist of relatively large patches that are parallel to the ground. Our idea for story detection is to robustly identify those patches corresponding to flooring and ceiling. To this end, two main problems need to be solved: First, architectural models contain furniture and staircases that include patches which are also parallel to the ground and must not be mistaken as belonging to the flooring or ceiling. Second, the *floor construction*, i.e. the volume between a ceiling and the flooring of the room above, often contains additional large patches that are parallel to the ground, see e.g. the green planes in Figure 5.2. In the following we describe how to overcome these difficulties and identify those patches that correspond to floorings and ceilings.

Floor Construction Detection We first determine the set of polygons that are parallel to the ground. To this end, let $d_{up} = (d_x, d_y, d_z)^t$ denote the vector indicating the normalized up-direction of the scene containing the building model. In most cases, d_{up} equals $(0, 0, 1)^t$ as the XY plane usually corresponds to the ground. Let $\mathcal{P} = \{P_1, ..., P_{|\mathcal{P}|}\}$ denote the set of polygons constituting the scene, and let $n(P_i)$ denote the according face normal vectors. The set of all polygons that are nearly parallel to the ground is given by

$$\mathcal{P}_{||} = \{P_i \in \mathcal{P} : | < n(P_i), d_{up} > | \leq \epsilon_{||}\},$$

where $\epsilon_{||}$ governs the maximum deviation between $n(P_i)$ and d_{up} in terms of the cosine of the enclosed angle. Let $P_{\perp d_{up}}$ denote the plane that is perpendicular to d_{up} and passes through the origin $\mathcal{O} = (0, 0, 0)^t$. For all polygons $P_i \in \mathcal{P}_{||}$ we compute the distance to $P_{\perp d_{up}}$. All polygons that provide approximately the same distance to $P_{\perp d_{up}}$ (we use a maximum deviation of 1mm) are put into a common bin. For each resulting bin we compute the sum of the included polygon areas. Patches caused by furniture or staircases are much smaller than those caused by floorings, ceilings and floor constructions. We therefore discard all bins with an area sum below a certain threshold $t_{floorsize}$.

The thickness of flooring constructions is limited. It is generally determined by the span, the load and the construction material, see e.g. [Hol07]. In buildings up to four floors the distance between a ceiling and the flooring of the room above is usually less than 0.4 meters, whereas multistory buildings, industrial buildings or hall structures may have floor profiles up to two meters and more. As we are mainly dealing with residential buildings, we collect all polygons within the distance of 0.4 meters in a bin representing a *flooring construction*. Within one flooring construction bin, the polygons with the maximum distance d_{max} to $P_{\perp d_{up}}$ constitute a flooring (see e.g. the blue planes in Figure 5.2), the ones with the minimum distance d_{min} constitute a ceiling (see e.g. the red planes in Figure 5.2). Note that in cases of poor modeling, the flooring of the upper story and the ceiling of the lower story might be represented by the same polygons[1]. We represent the k-th flooring construction FC^k as a tuple $FC^k = (\mathcal{P}^k, d^k_{min}, d^k_{max})$ including the set of polygons $\mathcal{P}^k = \{P_i \in \mathcal{P} : d^k_{min} \leq d(P_i, P_{\perp d_{up}}) \leq d^k_{max}\}$ belonging to the flooring construction as well as the distances between the ceiling and flooring level to $P_{\perp d_{up}}$ which are denoted by d_{min} and d_{max}. For simple structured buildings with flat roofs the above described method provides exact estimates about the location of single stories between the detected flooring structures. However, there are several types of buildings that cannot be processed in this way:

[1]This would actually correspond to an infinitely thin ceiling which would never be constructed in real life.

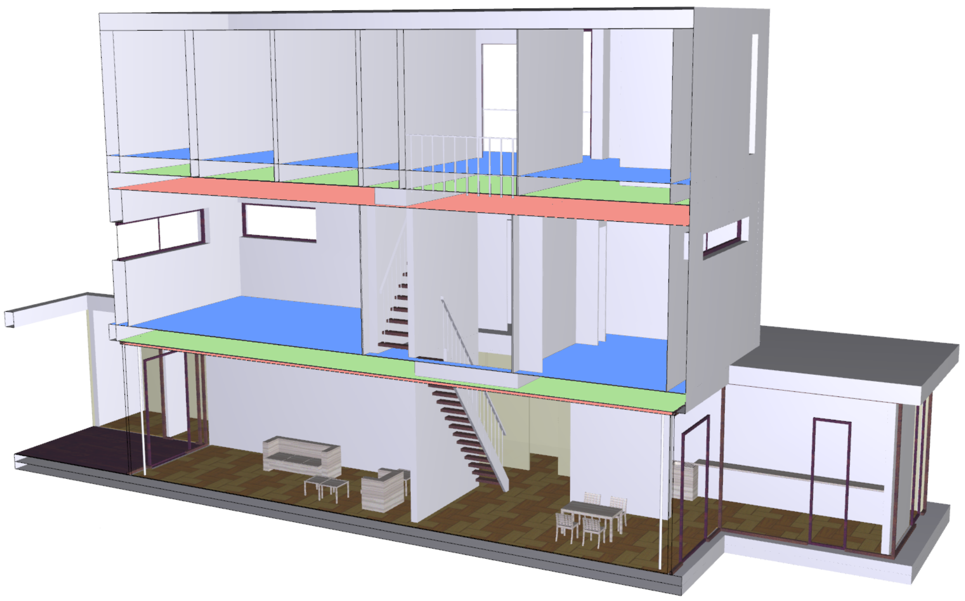


Figure 5.2: **Cross section through three-story residential building.** Flooring constructions located between ceilings (depicted in green) and floorings (depicted in red) contain additional polygons (depicted in blue).

Split-level buildings In split-level homes (see Figure 5.3), the floor levels of one or more parts of the house are located somewhere between the flooring and ceiling of other parts of the house. In this case there is no global floor level for each story but several local ones. As the Figure indicates, split-level structures can be rather complex. Although in principle our methods for segmentation into stories could be adopted to this challenge, we will concentrate on buildings with global floor levels in this work.

Missing floor planes Many building models lack the flooring of their lowest story, see e.g. Figure 5.4, preventing the story from being localized exactly. The occurrence of this problem can be detected relatively easy by determining whether the distance between the lowermost recognized horizontal plane and the lower side of the bounding box exceeds the typical height of a story (usually 2.40 meters). While for the depicted example it would be sufficient to assume that the flooring is located right at the lower side of the bounding box, this is no convenient strategy for arbitrary building models. In many cases there are some modeling artifacts that exceed the level down to which the actual building walls are reaching. An example is shown in Figure 5.5, where staircases and a small wall that is located outside the building exceed the floor level. To deal with this problem we simply propagate the height of the story located above assuming that both storys have similar height. This is of course only a heuristic method and should eventually be replaced by a more sophisticated approach.

Non-flat rooftops In buildings with non-flat roof tops, the uppermost story is not located between two flooring structures, see Figure 5.4. This might result in problems during the subsequent room localization. We will describe the consequences in more detail in Section 5.4.3.

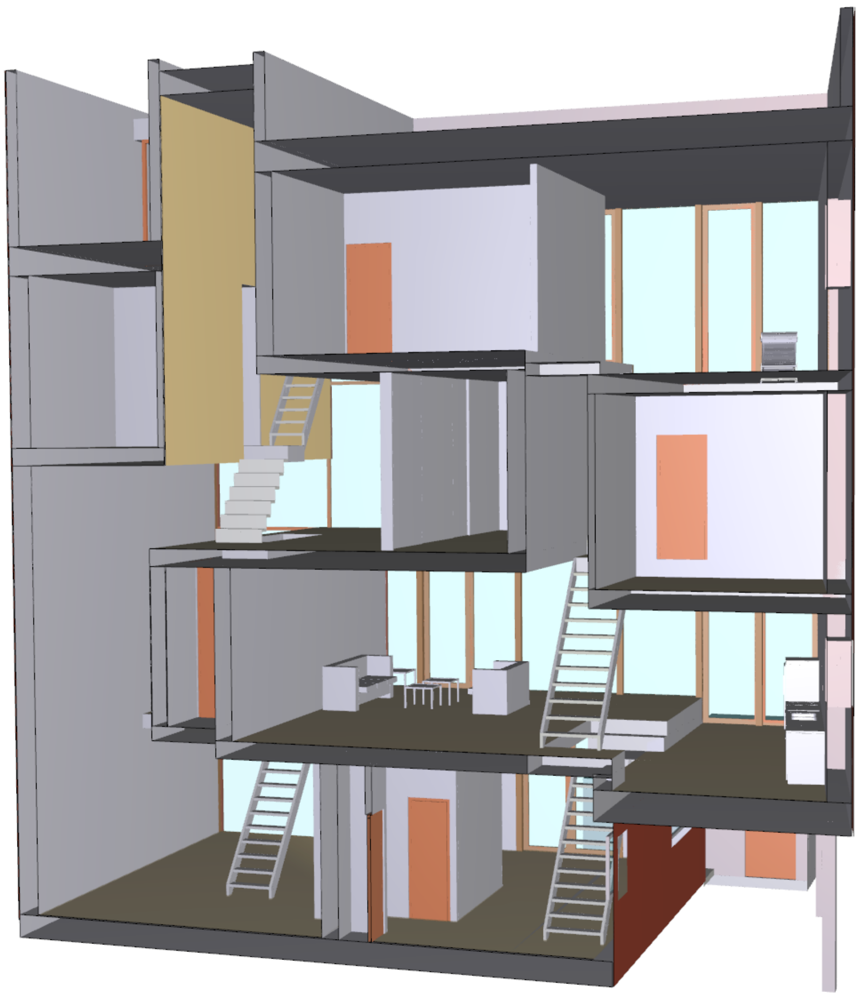


Figure 5.3: **Cross section through split-level home.** Instead of global floor levels there are several local levels.

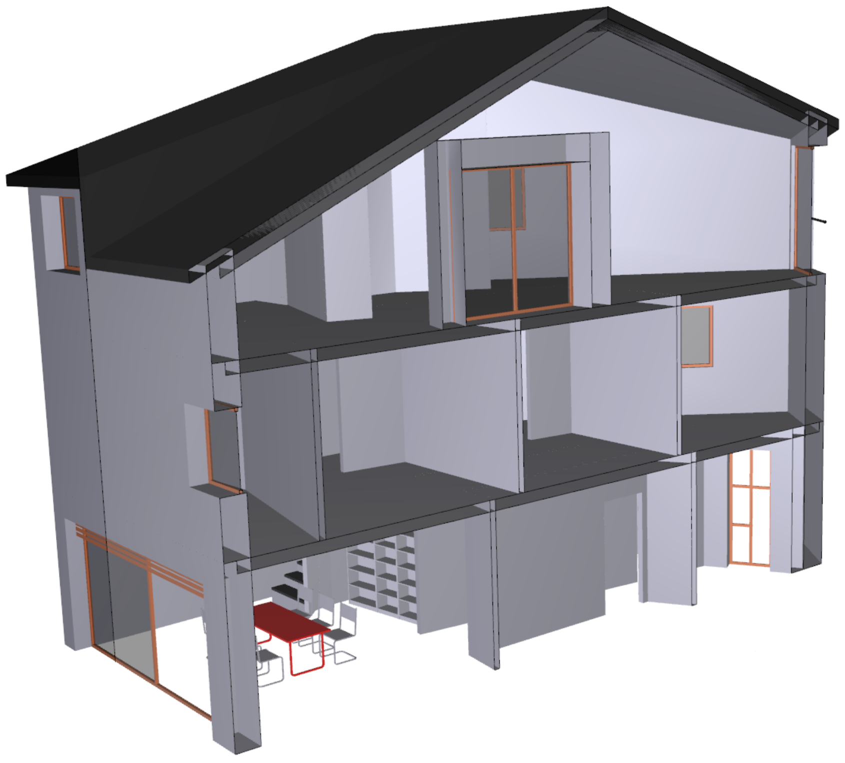


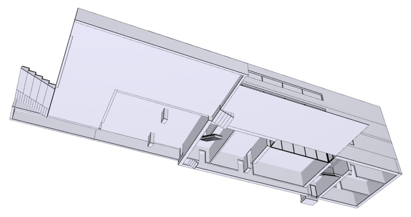
Figure 5.4: **Cross section through saddle roofed residential building with shed dormer.** Note that the lowermost flooring construction is not modeled.

5.4.2 Floor Plan Generation

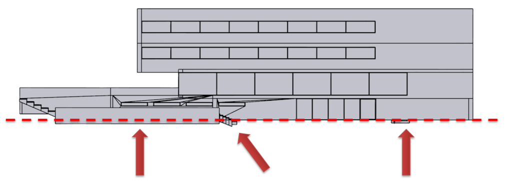
As in architecture 2D floor plans are used to unambiguously show the spatial organization of a story, we restrict ourselves to the use of horizontal cross sections instead of the complete 3D representation to efficiently extract rooms, windows and doors from building models. A horizontal cross section of a 3D polygon soup results in a set of two-dimensional line segments that do not provide any connectivity. We transfer this set into a two-dimensional halfedge representation in which the line segments are connected with respect to their spatial relationship. Computing three cross sections for each story at different height levels, we use the resulting halfedge structures to derive the RCG.

We represent each line segment that originates from the horizontal cross section by two vertices q_i and q_j, a halfedge h_{ij} leading from q_i to q_j and another halfedge h_{ji} leading from q_j to q_i. We denote the vertices by $\mathcal{Q} = \{q_1, ..., q_{|\mathcal{Q}|}\}$ and the set of halfedges by $\mathcal{H}$.

In the first preprocessing step, we determine the connectivity between the vertices. Two vertices q_i and q_j with $i \neq j$ are merged if the Euclidean distance



(a) **Missing floor plane.** In this building the lowermost flooring was not modeled.



(b) **Modeling artifacts.** Staircases and a small wall outside of the building (see arrows) exceed the level of the hypothetical flooring (dashed line)

Figure 5.5: **Story localization problems originating from missing lowermost floor plane.** Additional modeling artifacts prohibit the usage of simple localization heuristics relying on the bounding box.

between them is below a certain threshold $\epsilon_{\cup}$ (see Figure 5.6a). One often encounters buildings containing double-sided modeled polygons, resulting in two line segments that are very close to each other. After vertex merging, the directed connection from vertex q_i to vertex q_j might therefore be represented by more than a single halfedge. In this case we eliminate all halfedges from q_i to q_j but one. In the second step, we compute the connectivity between vertices and halfedges. For each halfedge h_{ij} we determine the Euclidean distance $d(h_{ij}, q)$ to every vertex $q \in \mathcal{Q} \setminus \{q_i, q_j\}$. If $d(h_{ij}, q)$ is below the above introduced threshold $\epsilon_{\cup}$, h_{ij} and its opposite halfedge h_{ji} are split into two halfedges. The resulting halfedges are connected to the vertex (see Figure 5.6b). In all our experiments we use $\epsilon_{\cup} = 1mm$. In the next preprocessing step we eliminate intersecting halfedges. Intersecting halfedges h_{ij} and their opposite halfedges h_{ji} are split and a new vertex is inserted at the point of intersection (see Figure 5.6c).

Note that the resulting graph still contains unintended gaps due to modeling errors like the one depicted in Figure 5.8a. In the beginning of the room detection described in the next paragraph we do not take care of these inconsistencies but resolve them only after a first version of the RCG is constructed.

5.4.3 Room Detection

Our idea for the precise localization of rooms is to compute a horizontal cross section located marginally below the story ceiling. In the resulting halfedge graph, the rooms correspond to faces. The concept can be best understood by taking a look at Figure 5.7. In 5.7a and 5.7b, the cross sections are localized slightly

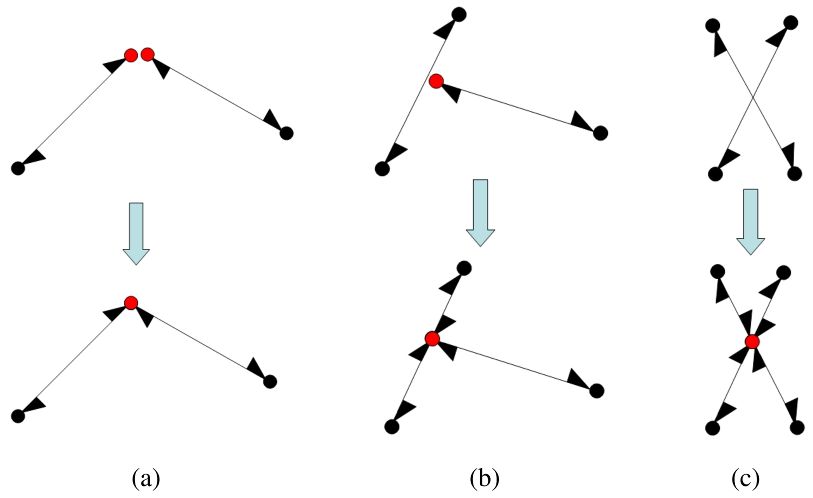


Figure 5.6: **Line segment processing steps.** a) Merging of close vertices. b) Halfedge split due to close vertex. c) Halfedge split and vertex insertion due to halfedge intersection.

above the flooring and at breast height. As can be seen, the red intersecting structure does not enclose single room faces but the whole story face. This is due to openings between the rooms. Only with a cross section located slightly below the ceiling as depicted in Figure 5.7c, the story face brakes up into individual room faces, as the openings usually do not reach that high. Given the halfedge graph representation of the cross section, we determine the graph faces and thereby the buildings of the room by walking along the halfedges, see Algorithm 2. However, there are also buildings which our method cannot be applied to successfully, see Figure 5.7d. Although the depicted cross section is located only millimeters below the ceiling, the openings reaching right up there prevent the detection of a decent room structure. Another problem arises from building with non-flat rooftops. As the ceiling is missing, the cross section cannot be localized in the above described way. Due to the slope of the rooftop, the area of the story resulting from the cross section becomes smaller and smaller with increasing height. The only area-preserving cross section can be conducted directly above the flooring. However, at this height openings between the rooms might prevent the detection of separated room faces. Note that in this work we do not deal with this problem but concentrate on buildings with flat rooftops.

Distinguishing Rooms, Walls, and Facades Regarding the extracted faces, two cases must be distinguished. A face can be circled on the inside (*inside face*) or

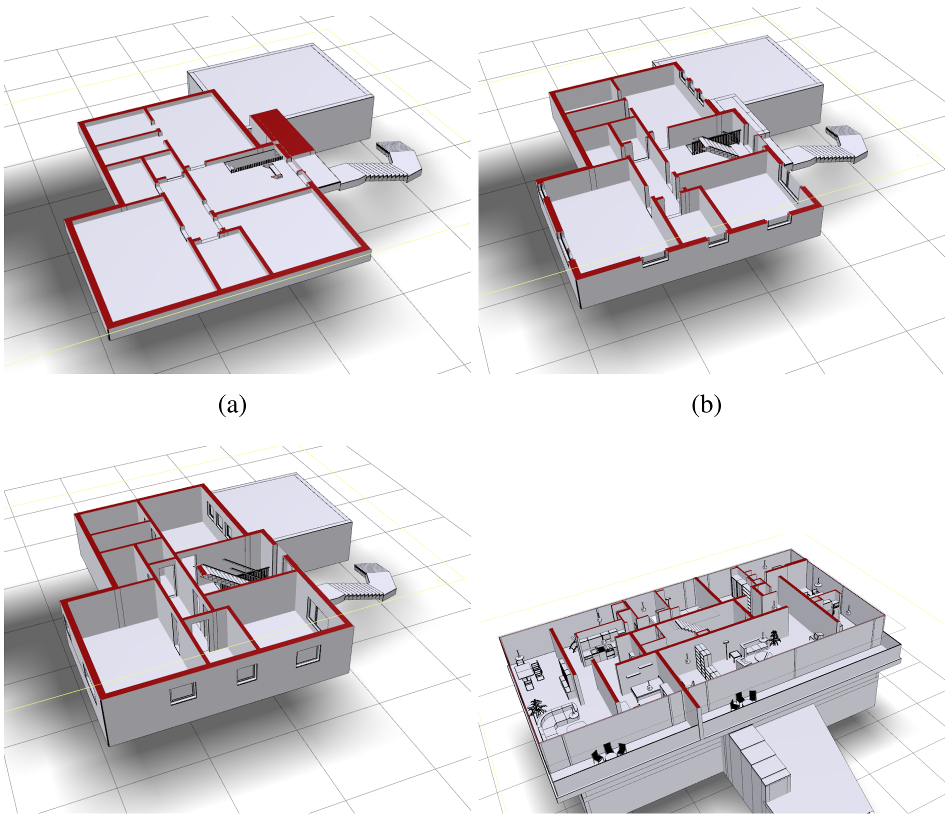

(a) (b) (c) (d)

Figure 5.7: **Cross sections for room detection** (Buildings in a-c are identical, Figure d shows another one). **a) Cross section above flooring.** Openings between rooms cause the red intersection structure to enclose the whole story instead of single rooms. No intersection with windows, but intersections with doors at this height level. **b) Cross section at breast height.** Same as a), but windows become intersected now. **c) Cross section below ceiling.** Openings do not reach high enough to prevent the story face from braking into room faces. Window and door intersections are not visible any longer. **d) Failure case.** Cross section below ceiling does not result in separated rooms due to high-reaching openings.

on the outside (*outside face*). Inside faces correspond to either rooms or walls[2]. Outside faces either represent rooms or walls lying completely inside another face without being connected to it or they correspond to the facade. To distinguish

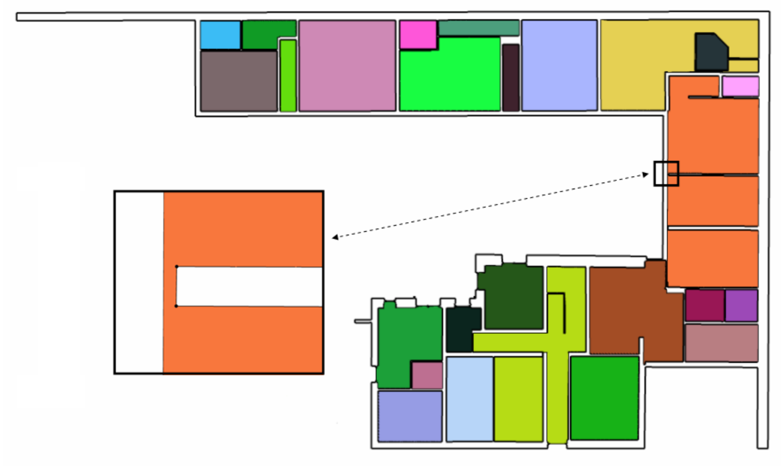
[2]Please note that theoretically our method might produce false rooms originating from atriums that are completely enclosed by the building. However, the data available to us did not contain such cases. In Section 5.4.5 we discuss how this problem could be dealt with.

Algorithm 2 Face Determination

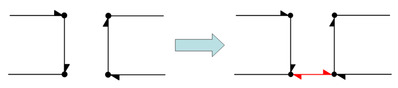
```
  Input Halfedge graph $HG$ = (nodes $\mathcal{Q}$, halfedges $\mathcal{H}$)
  Output Face set $\mathcal{F}$

  Function DetermineFaces(Halfedgegraph $HG$)
  for all $h_{ij} \in \mathcal{H}$ do
    if $h_{ij}$ was not visited yet then
      Create new face $f$
      Assign $f$ a pointer to $h_{ij}$
      TraverseHalfedge($h_{ij}$,$f$,$h_{ij}$)
      Add $f$ to $\mathcal{F}$
    end if
  end for
  Function TraverseHalfedge(halfedge $h$, face $f$, starting halfedge $h_{start}$
  Mark $h$ as visited
  Assign $h$ a pointer to $f$
  $q_{target} \leftarrow$ target vertex of $h$
  $h_{next} \leftarrow$ next outgoing halfedge of $q_{target}$ in clockwise direction with respect to $h$
  if $h_{next} \neq h_{start}$ then
    TraverseHalfedge($h_{next}$, $f$, $h_{start}$)
  end if
```

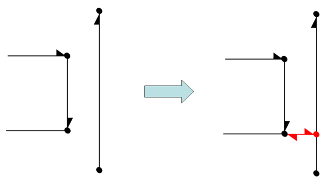
between inside and outside faces, we compute the angle between the incoming halfedge and the exiting halfedge with respect to clockwise direction for each of the n vertices during the face traversal. Inside faces correspond to an angular sum of $(n-2)\pi$, outside faces correspond to a sum of $(n+2)\pi$, where n denotes the number of vertices. To determine which outside face corresponds to the facade we pick an arbitrary vertex of each outside face and test whether it is located inside one of the inside faces. The facade face is located outside of all inside faces. Note that in case the building model consists of several tracts (see e.g. Figure 5.11), there also exist several facade faces. In the RCG, all facade faces are subsumed in the node representing the outside world, i.e. the one with attribute type $a_{type}(v) = world$ (c.f. Section 5.2.1). The inside faces represent either rooms or walls of the story building. In the next step we identify those faces that belong to rooms. First, all faces with an area of less than $1m^2$ are discarded, as according to DIN guidelines (see [Neu05]), rooms are larger in general. To eliminate the remaining wall-representing faces, we additionally consider the face perimeter. In general, walls have an elongated shape while rooms correspond to rather quadratic-shaped structures. Considering a wall of width w with a cross



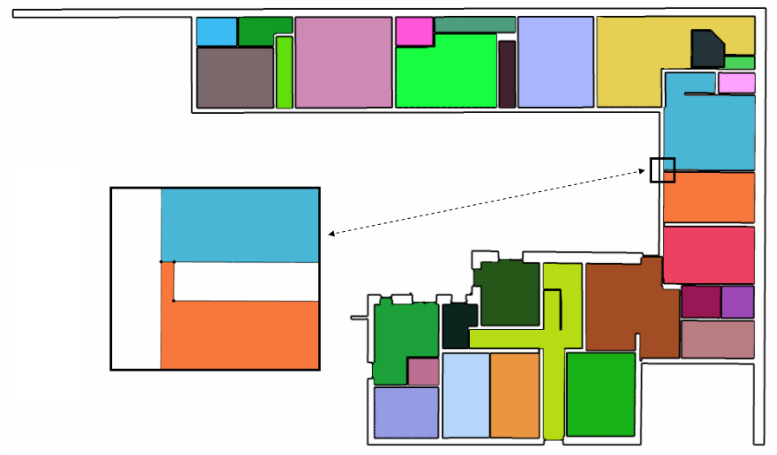
(a) **Result of room detection.** Unintended gaps between walls result in rooms that are not separated



(b) **Gap closing version I**. Insertion of two halfedges between existing nodes



(c) **Gap closing version II**. Insertion of a new node with subsequent insertion of two halfedges.



(d) **Result after gap closing.** Overall, four gap closing operations were performed.

Figure 5.8: **Room refinement.** Closing operations are applied to unintended gaps in the floor plan that caused a faulty room detection.

section area A and a quadratic room of the same area A, the perimeters P_{wall} and P_{room} are given by

$$\begin{aligned} P_{wall} &= 2\left(\frac{A}{w} + w\right), \\ P_{room} &= 4\sqrt{A}. \end{aligned} \tag{5.1}$$

Common wall widths range from 6cm to 40cm. Assuming the worst case of $w = 40$cm, the wall perimeter evaluates to $P_{wall} = \frac{5A}{m} + 0.8m$, which is much larger than P_{room} for the remaining face with areas $A \geq 1m^2$. One could exploit this observation and use area and perimeter as features to train a simple binary classifier in order to robustly learn a decision boundary between rooms and walls. However, it turns out that it is sufficient to use simple thresholding on the ratio $\frac{A}{P}$ between area and perimeter to robustly discriminate. We set the threshold to 0.25, faces with larger area-to-perimeter ratio are classified as rooms, the remaining are assigned to the wall class. Accordingly, for each detected room, a node v with attribute $a_{type}(v) = room$ is stored in the RCG.

Gap Closing Figure 5.8a shows the problem of gaps between walls creating connections between rooms that are actually intended to be separated. In the data we found gaps ranging to a size of about 10cm. A naive approach to deal with this problem would be to increase the threshold $\epsilon_{\cup}$ that is used for the construction of the halfedge graph. By that, gaps up to a size of $\epsilon_{\cup}$ would be closed. The drawback of this method is that the larger $\epsilon_{\cup}$ is chosen, the more vertices are moved and the floor plan shape changes. Walls would disappear due to this simplification and structures like windows and doors would become more and more unrecognizable.

We follow a different approach that is inspired by topological simplification techniques which were originally invented for mesh simplification, see [SZL92]. While this method concentrates on determining simplification operations that preserve the topology, our algorithm identifies actions that would cause significant topology changes, i.e. gap closing operations that eliminate unintended room connections. A gap-closing operation can either be the insertion of two halfedges between two vertices (Figure 5.8b) or the insertion of a new vertex on a halfedge including the split of this halfedge and a subsequent insertion of two halfedges, see Figure 5.8c. We denote a gap-closing operation to be *valid* if it splits the face into two faces that still satisfy the room conditions (i.e. minimum area of $1m^2$ and area-to-perimeter ratio larger than 0.25. For each face we perform a sequence of virtual gap-closing operations, until a valid one is found and conducted. The resulting faces are tested for valid gap-closing operations recursively. We consider gaps up to a size of 375mm (minimum window width according to DIN, see

[Neu05]), which is large enough to close most unintended gaps in our data but is still small enough to not close window openings unintentionally, see Figure 5.8d.

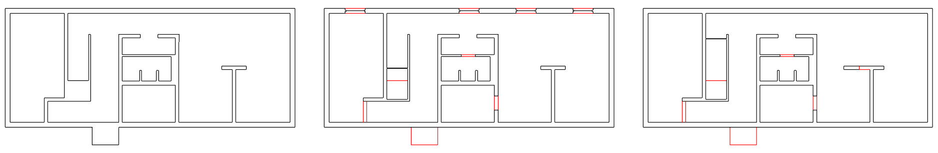
5.4.4 Door and Window Detection

Instead of treading the cumbersome way of recognizing doors and windows as geometric structures in the 3D model representation, we aim at identifying them as regularity breaking elements in 2D cross sections. To this end, we compute two more horizontal cross sections. The first one is located marginally above the story flooring, such that it will intersect doors but not windows, see Figure 5.7a. The second cross section is located at breast height (i.e. 1.40m above the flooring), such that it will intersect both, windows and doors, see Figure 5.7b. For both additional cross sections, we conduct the room detection in the above described way. Windows and doors produce geometric inconsistencies in a wall. By determining the inconsistencies between the three cross sections, these elements can be identified. Inconsistencies between the cross sections at breast height and the one below the ceiling indicate the presence of either windows or doors. If a corresponding inconsistency is also found in the cross section above the flooring, it indicates the presence of a door. In Figure 5.9 we show three cross sections of one story at the different height levels along with the inconsistent line segments that are contained in the high cross section but not in the lower ones.

Once the inconsistent line segments have been identified, we must determine whether they really correspond to doors or windows as there are also inconsistencies caused by other structural differences. Figure 5.9 shows that each door and window causes one pair of inconsistent segments in the two lower cross sections. According to DIN standards (see [Neu05]), there exist minimum widths for windows (375mm) and doors (55cm), such that inconsistent segments caused by these elements must provide according minimum lengths. All pairs of inconsistent line segments are tested whether they provide these characteristics. Once a pair of segments is identified to be a door or a window, we determine to which faces the according halfedges in the cross section below the ceiling belong (note that this might also be the face representing the outside world). We finally add an edge to the room connectivity graph connecting the room nodes representing the determined faces. As an attribute type $a_{type}(e)$, the edge is either assigned `door` or `window`, potentially together with the differentiation between `inside/outside`, c.f. Section 5.2.1.

5.4.5 Detection of Vertical Connections and Room Refinement

By a vertical connection we understand a structural element that allows a person to move between floor levels. Typical examples for such elements are staircases



(a) Cross section below the ceiling (b) Cross section at breast height (c) Cross section above the flooring

Figure 5.9: **Cross sections through the story of a building.** In b) and c) those line segments that are only contained in a) are shown in red. Inconsistent segments caused by doors are contained in both lower cross sections whereas inconsistent segments caused by windows can only be found b).

and elevator shafts. Note that all vertical connections that are located inside of a building require the presence of a hole in the flooring construction between the two floor levels to enable access. Our method for recognition of vertical connections is based on the detection of such holes. Note however, that despite being necessary for a vertical connection, the presence of a hole is not sufficient to indicate bidirectional accessibility between the floor levels.

One might think of localizing vertical connections by simply detecting holes in the flooring planes $\mathcal{P}^k$. However, if the hole caused by the vertical connection is located at an outside wall, this would only result in a reduction of the flooring plane area, but not in a change of its topology. Instead, we transform the vector-based representation of the flooring planes $\mathcal{P}^k$ into a discrete one by rendering them using an orthographic projection along d_{up}. Additionally, we render an orthographic projection of the facade face, i.e. the outside face that borders the entire story and perform a thinning operation by the number of pixels that correspond to the outer wall width, e.g. 40cm. We then subtract the flooring bitmask from the facade bitmask. Each remaining connected bitmask component represents a potential vertical connection. By pruning holes with too small area and perimeter one can further constrain the set of possible connections. The rooms which are connected by the new vertical component can be detected by projecting an arbitrary point located inside the hole to the upper and lower story.

The detected holes would allow for additional refinement of the detected room. Their shapes could be modified such that they would not contain the area that actually belongs to a hole. By that one could also eliminate the problem of faulty room detection originating from enclosed atriums, as the refined atrium room should at least theoretically have an area of zero. Note however that we did not implement such a refinement step.

5.5 Searching for Structures in Room Connectivity Graphs

The RCG represents a basic structure for retrieval of building models in databases. With this representation at hand, an attributed query graph can be searched for in a database of RCGs extracted from building models. An example for an interface for query graph definition is shown in Figure 5.10[3]. The search process requires to determine subgraph isomorphisms with respect to node and edge attributes and properties. Subgraph-isomorphism in general is known to be NP-complete [Coo71]. However, there are two aspects that make this approach feasible to this particular retrieval problem. First, the graphs we consider are in general not very large. Second, the attributes and properties that are assigned to the nodes and the edges efficiently accelerate the graph matching process.

While determining a subgraph isomorphism between an attributed query graph and a RCG, we only match nodes and edges of the two graphs if their attributes are similar. We therefore use global constraints describing the amount of similarity the nodes or edges must provide such that they can be matched. Considering the edge attributes we introduced, they result in a constraint involving the edge type: Two edges can only match if they represent the same structure, that is either door or vertical connection. Considering the node attributes, they result in two constraints: First, two nodes can only match if they represent the same structure, that is either a room or the outside world, second, two nodes can only match if the area of their associated rooms differs only by a certain amount of square meters. Although we restrict our experiments to these three constraints, the attributes introduced so far allow for the construction of further constraints involving for example the ratio between area and perimeter. Note that all constraints are global, i.e. they hold for all the nodes and edges, respectively.

In case the graphs to be matched both provide an outside world node, these nodes should be matched first to further accelerate the retrieval. If no such node exists in the query graph, an arbitrary node can be chosen.

5.6 Results

RCG Extraction For the room connectivity graph extraction all thresholds are chosen as described above. The threshold $t_{floorsize}$ describing the minimum story support size is set to $10m^2$. In Figures 5.11 to 5.15, examples of extracted RCGs

[3]Please note that the search interface was not developed by the author of this thesis but by the PROBADO project partners from Technische Universität Darmstadt, Germany, notably by René Berndt.

are shown. Each colored face represents a room node in the graph. Edges between room nodes are shown including the type connection (either `door` or `window`). Figure 5.15 shows a problem we discovered in some building models. The arrow points to a structure representing a window. In contrast to the other windows in this story, the window pane and the facade are exactly in one line. Therefore, the wall structure provides no inconsistency and the window cannot be detected.

Retrieval Figure 5.16 shows results of our retrieval experiments. As query we use a graph representing the structure of a typical apartment including a corridor that leads to four rooms (Figure 5.16a). The results shown in Figure 5.16b are generated using the constrained subgraph matching without considering the area and perimeter attributes. In Figure 5.17a we present timing results for subgraph search in a database containing RCGs from 138 3D building models. In this experiment we use the graph depicted in Figure 5.17b consisting of the outside node that is connected to a corridor which leads to four rooms. All rooms are assigned a size of 20 square meters. The value in the first column determines the required area ratio of two rooms to make the associated nodes match. The area ratio of two rooms is computed by dividing the smaller value by the larger one. With lower constraints on the matching, the number of buildings that can be found in the result increases along with the query time. We can see that for reasonable match ratios interactive search times can be achieved. Note that the number of result buildings does not directly correspond to the number of matched subgraphs, as in one building there might be more than just one match. All timings are with respect to a web-based search, i.e. they are somewhat conservative as they additionally include the times for data transfer to the server.

5.7 Conclusion

In this Chapter we introduced the room connectivity graph as a descriptor for compact characterization of buildings. RCGs describe the topological relationships of rooms and stories, but they can also be enriched to characterize the shape of single rooms for example in terms of area or height. RCGs are thereby an abstract concept that is decoupled from the actual representation of the building. RCGs might be automatically derived from sources including for example 2D drawings using an adapted room detection system like the on presented in Mace et al. [MLVT10], 3D laser scans, low-level 3D polygon data, or high-level BIM data. Another option would be to use semi-automated approaches like described in the work by Weber et al. [WLRB$^+$10].

Our system for automated extraction of RCGs uses 3D building models made of low-level polygon soups as input. We transform the 3D RCG extraction problem to several problems involving 2D projections of the building. We first decompose the building into its stories. By computing cross sections we are able to segment a story into single rooms. We try to improve the robustness of our method towards modeling errors by introducing a gap closing heuristic. Horizontal connections between rooms are found by detecting inconsistencies within the associated walls. In a last step we determine vertical connections between stories by searching for holes in the flooring planes. With the RCGs at hand, we investigated their eligibility for a simple retrieval task using attributed query graphs for constraint subgraph isomorphism search. We showed that by tightening thresholds on node compatibility in terms of matching room area, interactive response times could be achieved even for web-based searches on a database containing 138 RCGs.

Limitations During our development and experimentation we understood that RCG extraction for arbitrary building is an extremely complicated task. We therefore concentrated on a set of rather good-natured building models that are restricted in several ways:

- Our algorithm currently cannot process split-level buildings.
- Our algorithm will produce faulty results for buildings with non-flat rooftops considering the stories that are intersected by the roof.
- The detection of vertical connections between stories is based on recognizing holes in the flooring. While holes are necessary for vertical connections inside a building, they are not sufficient, and therefore vertical connections could be faked.
- Missing floor planes in the lowermost story might lead to problems with the room detection of that story.
- Cross sections at breast height might not cut through all windows. Especially in cellars, windows are often located above a level of two meters.
- Doors and windows that reach along the entire height of a wall are not detected as inconsistencies. Therefore, no connection between two rooms or one room and the outside world is detected. This problem occurs quite often in modern building with virtually seamless glass facades.

While our algorithm could be improved regarding some of the limitations like e.g. the segmentation of split-level buildings, we believe that non-flat rooftops, high reaching openings, doors, and windows as well as the reliable detection of conncetions via staircases might require an approach that operates on the entire

3D geometry instead of the 2D cross sections. Another possibility would be to rather rely on a semi-automated system that presents RCG extraction suggestions to the user which will then be improved iteratively.

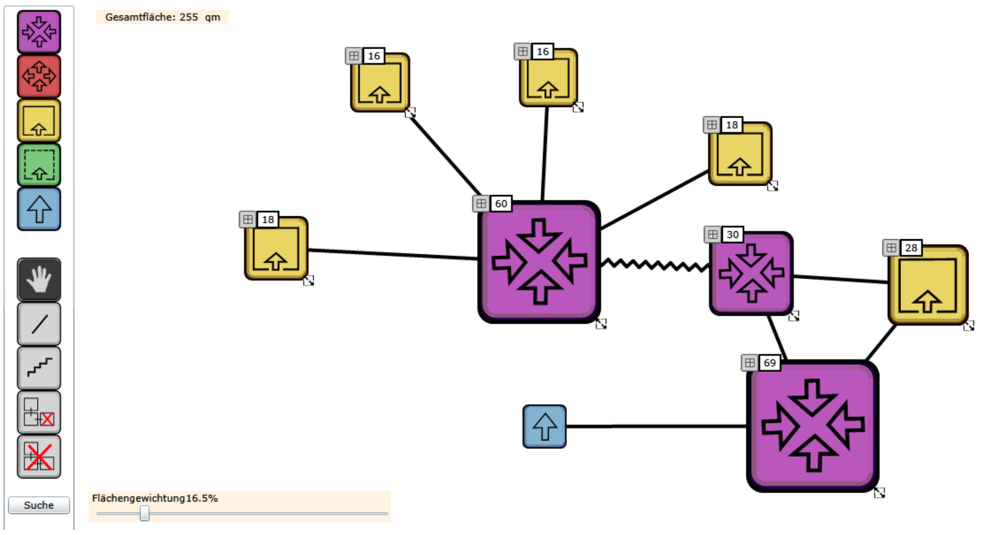
Future Work First of all at it would be most important to conduct a systematic evaluation of the extraction performance by comparing the automatically derived results to a manually created groundtruth. By that, the most urgent improvements could be determined. Additionally to overcoming the above mentioned limitations, there are several features which would add nicely to RCG extraction. It includes the detection of „rooms“ like terraces, balconies, and floors that are partially open to the outside. Additionally, vertical connections that are attached to the building but are located outside could be of interest. It would also be useful to better characterize the shape of rooms. To this end one could use 2D shape descriptors. Another method would be to classify rooms according to a certain set of shape proxies like e.g. `L-`, `C-`, or `T-`shaped. One could also introduce a more precise description of the spatial arrangement of different rooms.

Apart from RCG extraction, the comparison between two RCGs is an interesting topic. With the methods presented in this Chapter, so far we can only find topologically exactly matching subgraphs. For purposes like browsing or classification, a fuzzy distance measure is required. We will introduce such a notion of RCG similarity in the next Chapter.

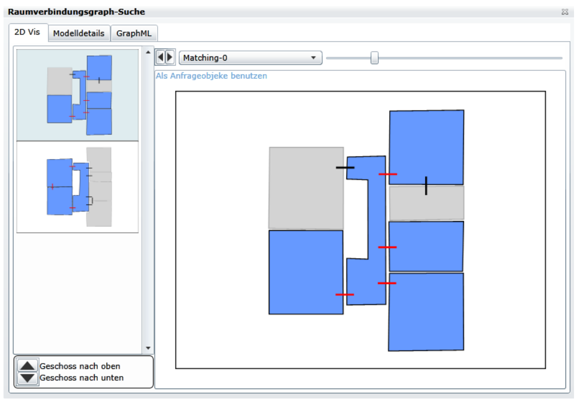
Future Prospects Regarding the current situation in the AEC sector, there are two developments that should be taken into consideration when thinking about future uses for RCGs. First, BIM is used to cover the complete lifecycle of a new buildings. Using according 3D BIM modeling tools results in building elements that contain a much higher level of semantics, including the location of and segmentation into rooms as well as connections between rooms. Inferring an RCG from such a high-level description therefore does not only seem to become more and more important due to the increasing usage of BIM, but it also seems to be much easier manageable than RCG extraction from a low-level polygon soup. Second, to allow the use of sophisticated retro fitting methods on legacy buildings for which no 3D description exist at all, digitalization by laser scanning has become the method of choice. To support this type of data, one could either adopt RCG extraction to this low-level representation, or one could hope for efficient tools that allow point cloud to BIM conversion[4]. However, there may be no free

[4]As a first step into this direction from the commercial sector, Autodesk© recently (2012) introduced the Point Cloud Feature Extraction for Autodesk© Revit© which allows the extraction of floor planes and walls from point clouds.

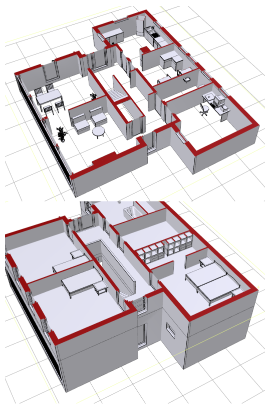
lunch after all, as to us it seems that many problems that are relevant to RCG extraction would also be relevant to solve the conversion problem.



(a) **Web-based query graph definition interface.** The size of the rooms can be easily varied by adjusting the size of the corresponding icon. The current size in square meters is shown for each room. Doors between rooms are represented by straight line, vertical connections are depicted by the jagged lines. The outside node is represented by the blue icon. The slider at the bottom can be used to steer how similar room sizes must be in order to make the associated nodes match. Note that the different room colors are only used to visually distinguish rooms of different connectivity level.



(b) **Simple web-based result visualization.** The rooms of the matched target graph are highlighted in blue. The two pictures on the left correspond to the two stories that are spanned by the query graph.



(c) **Corresponding 3D stories of the result building.** (2D and 3D representation are mirrored.)

Figure 5.10: **Web-based RCG search interface.** a) and b) show query graph definition and result visualization. The search for this rather complex graph took less than 2 seconds in a database of 138 buildings (including data transfer via Internet).

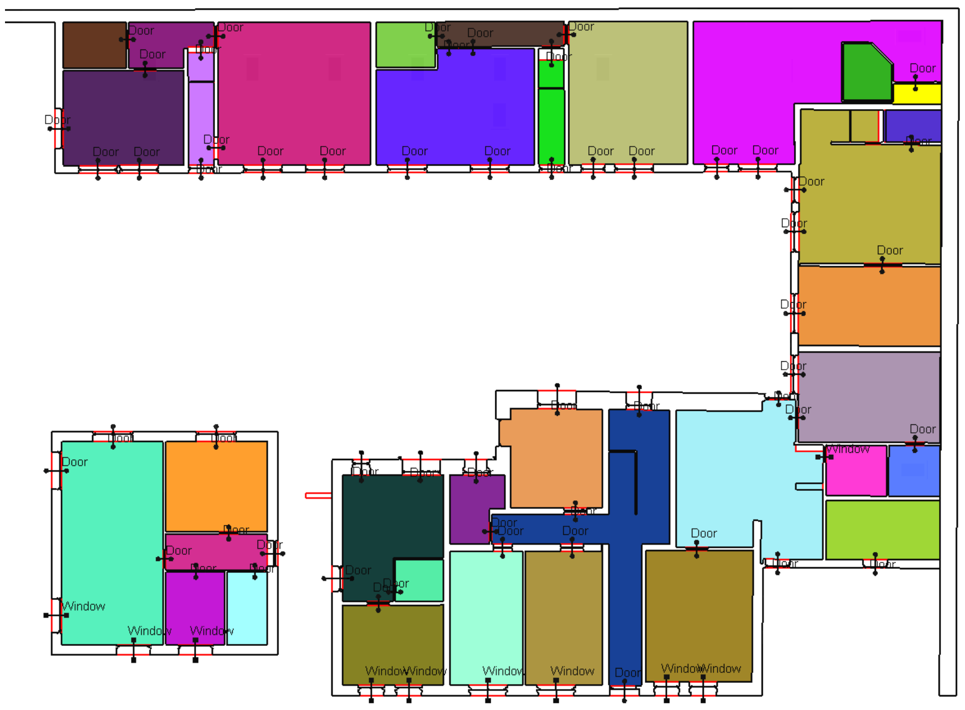


Figure 5.11: **Ground floor of a complex building containing two tracts.** The resulting room connectivity graphs consist of two connected components.

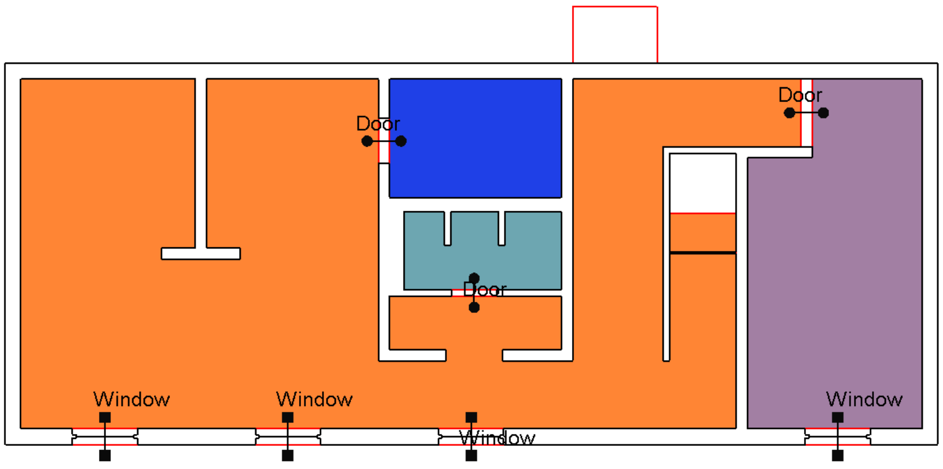


Figure 5.12: **Cellar of a residential building.**

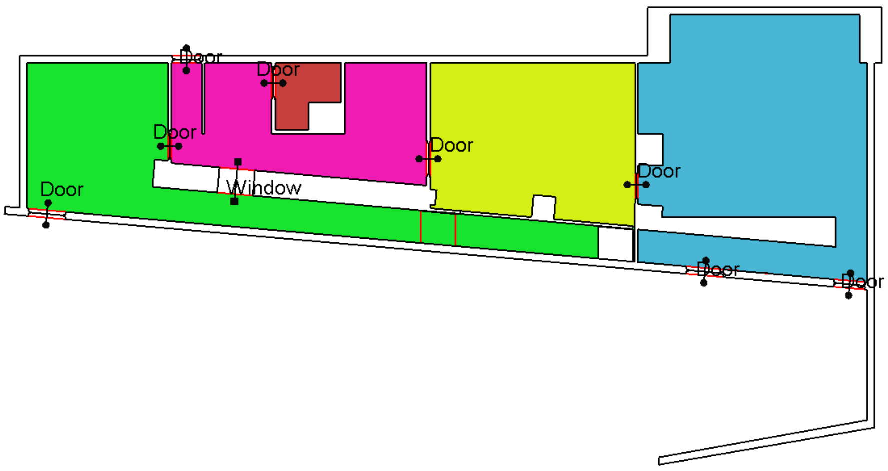


Figure 5.13: **Ground floor of a residential building.**

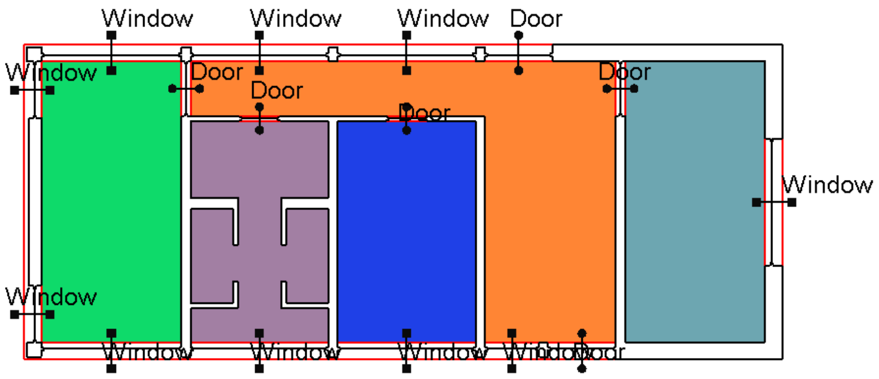


Figure 5.14: **Ground floor of a residential building.**

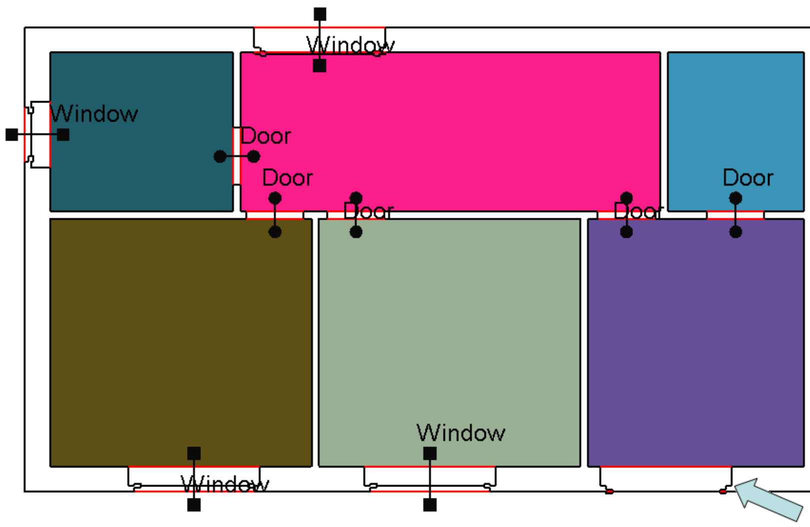
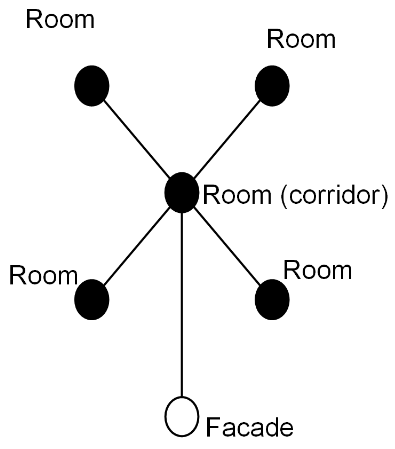
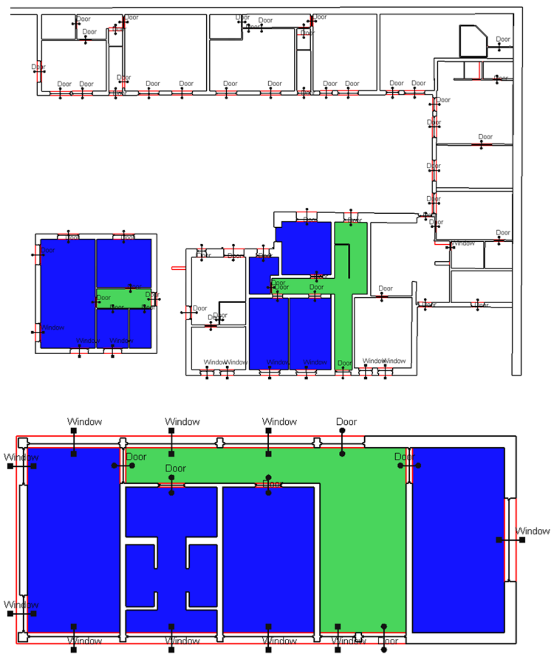


Figure 5.15: **First floor of a residential building.** The arrow points to a window that could not be detected by our method as the window pane is positioned exactly in one line with the facade.



(a)

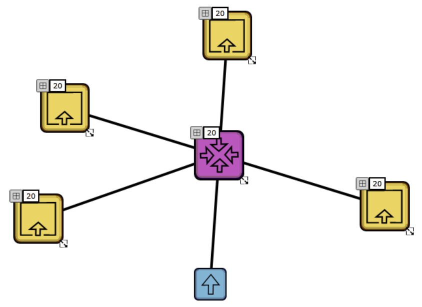


(b)

Figure 5.16: **Subgraph Retrieval Results.** a) Graph representing a typical apartment. All edges represent doors. b) Amongst the floor plans in Figure 5.11 to 5.15, the query graph is found twice in the floor plans of Figure 5.11 and once in that of Figure 5.14. The corridor room is shown in green, the other rooms are shown in blue.

Required room area match ratio	Found buildings	Search time
0%	32	3.7s
10%	24	2.5s
20%	16	2.0s
30%	10	1.5s
40%	5	0.9s
50%	2	0.2s
60%	1	< 100ms
70%	1	< 100ms
>80%	0	< 100ms

(a) **Timings for subgraph search.** The time consumption heavily depends on the desired area match ratio.



(b) **Query graph.** All rooms are assigned a size of 20 square meters.

Figure 5.17: **Subgraph search evaluation.** We investigate the influence of different room area match ratios on the performance when using the query graph depicted in b).

CHAPTER 6

RETRIEVAL AND CLASSIFICATION WITH ROOM CONNECTIVITY GRAPHS

6.1 Introduction

In the last Chapter we introduced room connectivity graphs as a means to characterize the structure of a building. Due to the use of node attributes, RCGs cannot only represent topological relationships, but they can also describe geometrical properties of rooms. In a few simple experiments we showed that retrieval can be conducted by defining an attributed query graph that is matched to RCG subgraphs in a database with respect to the defined attributes. While this type of query might be desirable to identify buildings that provide some well-defined, typical structure, the underlying search techniques using subgraph isomorphism causes several drawbacks. First, it is not suited to serve as a continuous similarity measure between two graphs, as it can only distinguish between either an exact topological match (which could be assigned an additional value describing the match quality) or no match at all. This is not only counterintuitive to the user, but it also prohibits the development of efficient browsing and classification methods. A solution to at least partially overcome the drawbacks would be to determine maximum common subgraph isomorphisms [GJ90]. However, this problem is known to be NP-hard as well. Second, although tighter constraints on the node matching criterions narrow the response time, the approach does not scale very well on larger datasets. Due to the lack of a metric, there is also no way to use standard acceleration structures to reduce the required time amount such that it would become sublinear in the number of RCGs in the database. Third, the more node properties are used, the harder it becomes to manually define up to which thresholds two nodes are still considered a match.

In this Chapter we introduce *subgraph embeddings* as a new method to determine fuzzy RCG similarity. The idea is to transform the structured graph representation into a vector-based one. To this end, we first decompose the RCG into a set of subgraphs. For each subgraph, we compute the similarity to a set of codebook graphs using a polynomial approximation on graph edit distances. Aggregating all similarity values finally provides us with a single vector for each RCG that enables fast retrieval, classification, and browsing. We additionally overcome the problem of manually defining node match values by transforming the low-level geometric room properties into semantically high-level room types. For evaluation we introduce a classification scheme that is based on typical structural patterns of single stories. We conduct retrieval and classification of single stories represented in according RCGs using different strategies, namely *approximate graph edit distances*, *Lipschitz embeddings*, *random walk graph kernels*, and similarity of our newly introduced *subgraph embeddings*. We thereby compare the impact of low-level geometric room attributes as well as the derived semantically higher-level room type features. We finally compare the retrieval performance of our method to that achieved by human architecture experts. Results show that our approach using subgraph embeddings and high-level attributes is superior to other methods in terms of retrieval and classification performance with respect to our suggested classification scheme. Summarizing the key contributions of this Chapter, they are:

- Automated generation of semantically high-level room type attributes that closely resemble an architect's understanding of floor plans (Section 6.4).
- A novel method for measuring the structural similarity of stories from modeled architectural buildings using attributed room connectivity graphs (Section 6.5).
- A new vector-based representation of RCGs that is based on embeddings of attributed subgraphs (Section 6.6).

This Chapter is based on the work presented in [WOV$^+$11a] and [WOV$^+$11b]. It should be noted that the room type classification scheme presented in 6.4.2 was not contributed by the author of this thesis but by the co-author of [WOV$^+$11a, WOV$^+$11b], Dr. Ina Blümel.

6.2 Related Work

In the following we briefly summarize the related work on fuzzy graph similarity including edit distances, graph kernels, and graph embeddings.

6.2.1 Edit Distances

The basic idea behind graph edit distances is to define graph dissimilarity by the minimum amount of deformation that is necessary to transform one graph into another [BA83, SF83]. The transformation thereby consists of a sequence of edit operations that usually include substitution, deletion, and insertion of both, nodes and edges, see Figure 6.1 for an example. Edit operations are assigned costs which can be defined with respect to node and edge attributes or labels. Graph dissimilarity is defined by the costs of the optimal least expensive edit path. While the approach represents a quite flexible method, determining the optimal edit path is NP-hard [ZTW$^+$09]. Several polynomial approximation methods have been proposed to overcome this drawback, see e.g. [EF84, RB09a, WZC94, ZTW$^+$09].

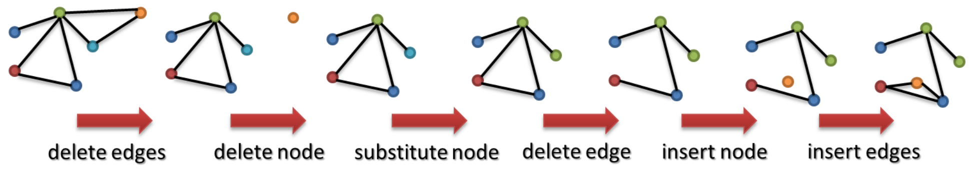
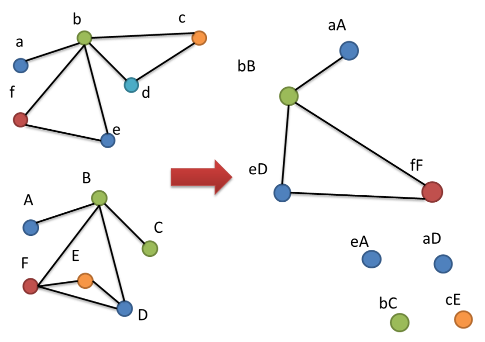


Figure 6.1: **Toy Example for Graph Edit Distances.** The Figure shows the gradual transformation of the graph on the left into the graph on the right. To this end, a sequence of insert/delete/substitution operations is conducted.

In this paper we use approximate graph edit distances introduced by Riesen et al. [RB09a] as a starting point. The idea is to find a mapping between two graphs such that only the nodes and their local surrounding edge structure are optimally matched instead of finding a globally optimal mapping. The approach has been applied to various retrieval and classification tasks including datasets of letters, fingerprints, images, molecules, proteins, and 3D CAD models [WLHZ10]. Its retrieval performance is close to exact graph edit distances while at the same time it only requires a fraction of the computation time. We will give a more detailed description of this prerequisite for our new method in Section 6.5.

6.2.2 Graph Kernels

The idea behind graph kernels is to develop similarity measures between graphs that are positive definite such that techniques originating from statistical learning theory which are defined in terms of inner products (e.g. classification with SVMs) become applicable to structured data. Convolution kernels for discrete structures including strings, trees, and graphs were introduced by Haussler [Hau99].



(a) **Product graph construction.** The two input graphs are shown on the left, the resulting direct product graph on the right.

	aA	aD	bB	bC	cE	eA	eD	fF
aA	0	0	1	0	0	0	0	0
aD	0	0	0	0	0	0	0	0
bB	1	0	0	0	0	0	1	1
bC	0	0	0	0	0	0	0	0
cE	0	0	0	0	0	0	0	0
eA	0	0	0	0	0	0	0	0
eD	0	0	1	0	0	0	0	1
fF	0	0	1	0	0	0	1	0

(b) **Adjacency matrix of the product graph.**

Figure 6.2: **Toy Example for Direct Product Kernel.** The two input graphs on the left are combined to form the product graph according to Equation 6.1.

This very generic type of kernel aims at generalizing a mathematical convolution to structured data. However, the adaption of this approach to concrete problem seems to be rather hard, especially for graphs that lack a certain ordering in contrast to e.g. strings. A common strategy in graph kernel design is to incorporate similarity of substructures like random walks, subtrees, cyclic patterns, or shortest paths [Gär05, Bor07]. In Figure 6.2 we depict an intuitive example showing a *direct product graph* that is constructed from two input graphs. Following the description in [Gär05], the product graph $G_1 \times G_2$ of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is constructed according to:

$$\begin{aligned} V(G_1 \times G_2) &= \{(v_1, v_2) \in V_1 \times V_2 : (v_1 \text{ matches } v_2)\}, \\ E(G_1 \times G_2) &= \{((u_1, u_2), (v_1, v_2)) \in V(G_1 \times G_2) : \\ &\quad (u_1, v_1) \in E_1 \wedge (u_2, v_2) \in E_2 \wedge ((u_1, v_1) \text{ matches } (u_2, v_2))\}, \end{aligned} \tag{6.1}$$

where the term *matches* denotes that the nodes or edges are in some way compatible. In Figure 6.2a we show an examples in which compatible nodes are indicated by matching color. There is no constraint on the edge compatibility. The Figure shows that substructures which are present in both input graphs are nicely connected in one component of the resulting product graph. In this graph, walks of a certain length l can be computed by raising the corresponding adjacency matrix (see Figure 6.2b) to the power of l. Accordingly, Gärtner constructs a kernel by first computing a weighted geometric series of the adjacency matrix and by then using the summed up entries of the result as a kernel value.

Graph kernels have been successfully applied to a wide range of data. Gärtner [Gär05] uses graph kernels relying on discrete node labels for molecule classification. These kernels can be computed in polynomial time. Borgwardt et al. [Bor07] extend this approach to incorporate arbitrary edge and node attributes for protein function prediction. They relax the binary adjacency matrix and allow continuous values to indicate partial compatibility. Segmentation graph kernels have been used for image classification [HB07]. In the domain of 3D objects, Fisher et al. [FSH11] use graph kernels to represent and learn the structural relationship between several 3D models belonging to one scene.

6.2.3 Graph Embeddings

Graph embeddings represent a method to transform structural graph data into a vector-based representation. Given a query graph, the similarity to a *fixed* number of codebook graphs is computed using some graph matching method. The resulting vector is normalized. The vector-based representation offers several advantages. Similarity search in a large database of embedded graphs can be conducted in terms of feature vector comparison which is usually far more cost-effective than any sort of graph matching. Additionally, acceleration structures can be used to further reduce search times.

Bunke et al. [BR08, RB09b] compute *Lipschitz embeddings* using the previously introduced approximate edit distance [RB09a]. In contrast to graph retrieval solely based on approximate edit distance, this method is much faster while at the same time it provides competitive retrieval performance. A similar method specialized on graphs with fixed number of vertices is proposed in [RVRB10]. Approaches based on embeddings of subgraphs instead of complete graphs are presented in [BHAT05] and [OA10]. First, interest regions in an image are detected and assigned discrete labels using a clustering method. Each region is represented by a node and its according label. Neighboring regions are connected by edges. Graph mining algorithms are then used to detect frequent subgraphs. Each image is finally represented by the number of occurrences of a set of codebook graphs. As a drawback, both approaches are restricted to discrete node labels and exact graph matchings. Note that this method using substructures is in fact a Bag-of-Features approach which is a popular strategy in text, image, and 3D retrieval.

6.3 Method Overview

Taking into account the shortcomings of building retrieval approaches based on exact subgraph matchings with RCGs as described in the introduction, we develop a search method that consists of three steps, generation of high-level room type attributes, fuzzy RCG dissimilarity computation, and subgraph embeddings of RCGs for efficient retrieval and classification.

High-Level Room Type Attributes RCGs contain automatically generated node attributes that describe the geometric shape of the underlying room. We propose a supervised learning method to derive semantically higher-level room type attributes like e.g. `corridor` or `access room` based on the low-level attributes. Our intuition is that these attributes much better resemble an architect's understanding of building topology and floor plans and finally lead to better retrieval and classification results.

Room Connectivity Graph Dissimilarity We introduce a fuzzy dissimilarity measure for RCGs that is based on approximate graph edit distances [RB09a]. It incorporates similarity of node and edge attributes. In contrast to previous methods for RCG retrieval, this algorithm requires only polynomial runtime, making the approach more feasible for larger databases.

Bag-of-Subgraphs We propose a new approach for generating a vector-based representation of RCGs that is inspired by Lipschitz embeddings of graphs [RB09b]: Based on the approximate edit distance, we represent RCGs as embeddings of their contained subgraphs into a codebook of fixed length. The codebook consists of subgraphs sampled from an RCG training set. This embedding is basically a Bag-of-Features representation, or loosely speaking, a *Bag-of-Subgraphs*. The benefit of this representation is threefold. First, the vector-based representation facilitates retrieval and classification. Similarity search in a large database of embedded graphs is reduced to feature vector comparison which is usually far more cost-effective than any sort of graph matching. It also enables the use of search acceleration methods like k-d trees. Additionally, standard methods from machine learning like e.g. support vector machines (SVMs) can be directly applied for RCG classification without the need for special graph kernels. Second, due to rather small subgraphs, the matching problem in general becomes better tractable. Finally, partial similarities can be found easier. Consider for example two graphs, each consisting of a single corridor with a largely different number of rooms attached. Although the overall similarity of the graphs is small due to the varying number of rooms, both graphs contain a considerable amount of very similar subgraphs.

Notational Conventions To better understand the algorithm we first introduce a few notational conventions:

- $G_S = (V_S, E_S)$ denotes the *source* graph consisting of a set of nodes V_S and a set of edges E_S. Analogously, $G_T = (V_T, E_T)$ denotes the target graph.

- v_s denotes an arbitrary node in V_S, i.e. $v_s \in V_S$, analogously, v_t denotes a node in V_T.
- e_s denotes an arbitrary edge in E_S, i.e. $e_s \in E_S$, analogously, e_t denotes an edge in E_T.
- $e_{ss'}$ denotes an edge in E_S that connects nodes v_s and $v_{s'}$, $v_s \in V_S$, $v_{s'} \in V_S$, analogously, $e_{tt'}$ denotes an edge in E_T that connects nodes v_t and $v_{t'}$
- $E_{adj}(v)$ denotes the set of edges that are adjacent to node v.
- $V_{adj}(v)$ denotes the set of nodes that are adjacent to node v.
- $|.|$ denotes the cardinality of any node or edge set.
- $A(v)$ denotes the area of the room associated to node v. $A(V)$ denotes summed area of all room nodes in V, i.e. $A(V) = \sum_{v \in V} A(v)$.
- $P(v)$ denotes the perimeter of the room associated to node v.

6.4 Node and Edge Attributes

In this Section we describe which low-level node and edge attributes are used as a starting point for our new retrieval method. We will then describe how these attributes can be transformed into high-level attributes representing architecturally meaningful room types to enhance the retrieval and classification performance.

6.4.1 Node Attributes

Analogously to our description in Section 5.2.1 we distinguish two node types including *room* nodes and the *world* node:

$$a_{type}(v) = \begin{cases} room, & \text{if } v \text{ represents a room,} \\ world, & \text{if } v \text{ represents the outside world, i.e. } v = v_O \end{cases}$$

For each room node $v \in V \backslash \{v_o\}$, we represent the following room properties in a feature vector:

1. Relative number of doors in this room:

$$a_{prop}(v)[1] = \frac{|E_{adj}(v)|}{|E|}$$

2. Room area with respect to the total story area:

$$a_{prop}(v)[2] = \frac{A(v)}{A(V \backslash \{v_o\})}$$

3. Room area with respect to areas of adjacent rooms:

$$a_{prop}(v)[3] = \frac{A(v) \cdot a^1_{prop}(v)}{A(V_{adj}(v))}$$

4. Room contains windows:

$$a_{prop}(v)[4] = \begin{cases} 1, & \text{if } v \text{ has windows,} \\ 0, & \text{else} \end{cases}$$

5. Perimeter-to-area ratio:

$$a_{prop}(v)[5] = \frac{P(v)}{\sqrt{A(v)}}$$

Property (1) represents how much the room contributes to the connectivity of the story. Attribute (2) globally characterizes the area of the room with respect to the area of the whole story, while (3) describes the area with respect to the surrounding rooms. Property (4) is an indicator for the amenity value of the room. For example, living rooms always contain windows while corridors or lumber-rooms are often windowless. Property (5) is useful for the identification of corridor-like rooms. In many cases, corridors serving as distributors in a story are stretched relatively long but are not very wide. In contrast, rooms with high amenity value tend to be more square-shaped. This difference can be characterized by computing the perimeter-to-area ratio.

6.4.2 High-level Node Attributes

For classifying buildings there exist several criteria, e.g. building geometry or building functions [MB97]. In this Chapter we will focus on distinguishing floor plans. Accordingly, we consider classes based on structural patterns of single stories as well as on properties of single rooms. In the following we will introduce five classes of floor types that are of great importance to architectural planning processes because they notably comprehend the crucial question of room access and, based on this, the zoning of floors in interactive and private spaces. They follow common guidelines of this field like discussed e.g. in [Sch04] and [Hen09].

Floor Types The `corridor type` (Figure 6.3(a)) is characterized by a corridor which is the dominant path through the story. The corridor's only purpose is to connect rooms, it has no amenity value. The `distributor type` (Figure 6.3(b)) is similar to the corridor type. However, a large room instead of a corridor serves as the connecting element. This room is often the biggest one of the story, with amenity values itself. In a `loft type` floor (Figure 6.3(c)), the whole story mainly consists of one large room which can contain several adjustable zones with different functions. In a `hierarchical` floor (Figure 6.3(d)), there exists a separation between a dominant area which leads to further areas, where a second distribution is conducted. This second distribution can be either the previously defined corridor or distributor type. `Chain type` (Figure 6.3(e)) floors are characterized by *access rooms* that are arranged like a chain, so that the connection only exists between one space and the next.

Like Schneider [Sch04] accurately described with the statement *"After all, the truly exciting solutions often lie on the line between two or more of these categories[...]"*, it has to be taken into account that some of the floors cannot be assigned to a category with absolute certainty, neither by machine nor human classifiers, which is supported by our findings in Section 6.7.3.

Predicting Room Types The introduced floor classes can be easily described using architectural terms that characterize room types like corridor, access room, or distributor with amenity value. Our idea is to transform low-level geometric room attributes into high-level room type attributes. By that we are able to better reflect an architect's understanding of floor plans and could probably improve retrieval and classification performance. We identify corridors, distributors, access rooms, and one-door spaces to be the high-level room types that are crucial for floor characterization. To automatically derive the high-level room type from the extracted low-level attributes we suggest to use supervised learning methods. Let

$$\begin{aligned}\mathcal{C} &= \{C_1, ..., C_4\} \\ &= \{\texttt{corridor}, \texttt{distributor}, \texttt{access room}, \texttt{one-door space}\}\end{aligned}$$

denote the set of room classes. Given a training set of low-level room attributes $a_{prop}(v_i)$ with assigned labels $y_i \in \mathcal{C}$ we use a Support Vector Machine (LIBSVM package [CL01]) to learn a discriminant function that maps geometric room attributes to room classes. For increased robustness, we assign the rooms in the testing set a vector a_{label} representing the predicted distribution over the room classes rather than only the most likely one. The prediction reads

$$a_{label}(v) = \begin{pmatrix} p(C = \texttt{corridor} & \mid & a_{prop}(v)) \\ p(C = \texttt{distributor} & \mid & a_{prop}(v)) \\ p(C = \texttt{access room} & \mid & a_{prop}(v)) \\ p(C = \texttt{one-door space} & \mid & a_{prop}(v)) \end{pmatrix}.$$

Ground Truth Generation For the above described learning approach we need ground truth, i.e. a set of low-level training attributes with assigned labels $y_i \in \mathcal{C}$ must be available. A possible way would be to assign the labels manually. However, a set of 100 RCGs might easily contain several thousand rooms, rendering this method cumbersome. We instead generate pseudo ground truth for a large number of rooms automatically. All graphs in the training set are assigned to one of the classes described in 6.4.2. By that, we can infer the assignment of single nodes to room types:

1. Rooms in the `corridor` class with high connectivity (more than two doors) are most likely corridors.
2. Rooms in the `distributor` class with high connectivity (more than two doors) are most likely distributors.
3. Rooms with exactly one door not belonging to the `loft` class are one-door spaces.
4. Rooms with exactly two doors not belonging to the `loft` class are access rooms.

Note that we do not assign labels to all rooms because of ambiguities. E.g., highly connected rooms in the `hierarchical` floor class might be either corridors or distributors, and the central room of a loft could be regarded as any of the above defined room types depending on its connectivity. Our assumptions are of course simplifications and only apply to RCGs that are perfectly and uniquely assignable to a single class. Although most of our data are not that ideal, our assumptions seem to be valid as shown by the results in Section 6.7.

6.4.3 Edge Attributes

As described in Section 6.4.2, our RCG classification scheme focuses on single stories instead of complete buildings. As a consequence, we do not consider vertical connections between stories but only doors. We distinguish doors connecting two rooms and doors connecting the outside world and the building. Accordingly, we define the edge type attributes of a subset of those presented in Section 5.2.2, namely

$$a_{type}(e) = \begin{cases} door_{inside}, & \text{if } e \text{ connects two rooms,} \\ door_{outside}, & \text{if } e \text{ connects a room to the outside world.} \end{cases}$$

6.5 Approximate Graph Edit Distances

Riesen et al. [RB09a] introduce a polynomial approximation on the computation of globally optimal edit paths. The idea is to find a mapping between two graphs such that only the nodes and their local surrounding edge structure are optimally matched (or deleted / inserted if necessary). This problem can be reduced to bipartite graph matching. In the following we will describe this method in more detail as it is an important building block for our subgraph embeddings.

6.5.1 Algorithm

With respect to the basic operations of edit distances, the following costs are defined:

- $c_{v_s v_t}$ for substituting source node v_s by target node v_t,
- $c_{\epsilon v_t}$ for inserting a node in the source graph G_S that is matched to v_t,
- $c_{v_s \epsilon}$ for deleting node v_s in the source graph if it is not matched against any node in V_T.

Node deletions and insertions can be interpreted as mapping a node to an additional ϵ-node. The above defined costs can be represented in a matrix $C_{V_S V_T}$ of size $[|V_S| + |V_T|] \times [|V_S| + |V_T|]$:

$$C_{V_S V_T} = \left[\begin{array}{cccc|cccc} c_{11} & c_{12} & \cdots & c_{1|V_T|} & c_{1\epsilon} & \infty & \cdots & \infty \\ c_{21} & c_{22} & \cdots & c_{2|V_T|} & \infty & c_{2\epsilon} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \infty \\ c_{|V_S|1} & c_{|V_S|2} & \cdots & c_{|V_S||V_T|} & \infty & \cdots & \infty & c_{|V_S|\epsilon} \\ \hline c_{\epsilon 1} & \infty & \cdots & \infty & 0 & \cdots & \cdots & 0 \\ \infty & c_{\epsilon 2} & \cdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \infty & \vdots & & \ddots & \vdots \\ \infty & \cdots & \infty & c_{\epsilon|V_T|} & 0 & \cdots & \cdots & 0 \end{array}\right].$$

Each node can only be matched to one ϵ-node, which is enforced by the off-diagonal values in the upper right and lower left part of the matrix reading infinity. Note that all node matching costs which are not set to infinity are normalized such that they lie[1] between 0 and 1. Using this matrix, the optimal mapping can be computed in $\mathcal{O}((|V_S| + |V_T|)^3)$ using Munkres' algorithm [Mun57] (also known as Hungarian method). Once this mapping is computed, the approximate graph edit distance $d_{AGE}(G_S, G_T)$ can be computed by adding all costs for the individual nodes mappings and then normalizing by $(|V_S| + |V_T|)$.

6.5.2 Cost Functions

Following [RB09a], node substitution costs $c_{v_s v_t}$ consist of a term reflecting the compatibility of the attached node attributes as well as a term describing the costs to match the adjacent edges. We denote attributes by $a_{v_i} = \{a_{type}(v_i), a_{prop}(v_i)\}$ and describe the compatibility of the node attributes by $c(a_{v_s}, a_{v_t})$. By the term $c(E_{adj}(v_s), E_{adj}(v_t))$ we denote the costs for matching the adjacent edges of v_s and v_t. Both terms are aggregated in the following equation:

$$c_{v_s v_t} = \lambda_1 \cdot c(a_{v_s}, a_{v_t}) + (1 - \lambda_1) \cdot \frac{c(E_{adj}(v_s), E_{adj}(v_t))}{|E_{adj}(v_s)| + |E_{adj}(v_t)|}. \tag{6.2}$$

The weighting factor λ_1 balances the influence of node attributes and edge matching costs. The edge matching costs which are defined with respect to attached edge attributes are again computed using Munkres' algorithm in a way analogously to the above defined one, using edge substitution, insertion, and deletion and according costs $c_{e_s e_t}$, $c_{\epsilon e_t}$, $c_{e_s \epsilon}$. Deletion and insertion costs can be defined as either constant or dependent on the inserted/deleted node's or edge's attributes. For further details on the algorithm we refer to [RB09a]. In our particular setting, we use room and edge attributes to define appropriate cost functions.

Node Substitution Costs For low-level attributes, we define node substitution in terms of the room type attribute $a_{type}(v)$ as well as the room property vector $a_{prop}(v)$ (see Section 6.4.1):

$$c_{low-level}(a_{v_s}, a_{v_t}) = \begin{cases} \frac{1}{\sqrt{5}} ||a_{prop}(v_s) - a_{prop}(v_t)||_2, & \\ & \text{if } a_{type}(v_s) = a_{type}(v_t) = room \\ 0, & \text{if } a_{type}(v_s) = a_{type}(v_t) = world \\ \infty, & \text{if } a_{type}(v_s) \neq a_{type}(v_t), \end{cases}$$

where $||.||_2$ denotes the Euclidean distance. The motivation for the normalization factor of $\sqrt{5}$ will be given in Section 6.6.2. The above definition ensures that both

[1]Actually, the matching costs might exceed 1 under certain circumstances, c.f. Section 6.6.2.

outside nodes are matched onto each other $(a_{type}(v) = world)$, and that no room is matched against an outside node $(a_{type}(v_s) \neq a_{type}(v_t))$. The distance between two nodes is defined in terms of their associated room property vectors.

For high-level room type attributes, we use the predicted label vector $a_{label}(v_s)$ instead of $a_{prop}(v_s)$:

$$c_{high-level}(a_{v_s}, a_{v_t}) = \begin{cases} \chi^2(a_{label}(v_s), a_{label}(v_t)), & \\ \qquad \text{if } a_{type}(v_s) = a_{type}(v_t) = room & \\ 0, \quad \text{if } a_{type}(v_s) = a_{type}(v_t) = world & \\ \infty, \quad \text{if } a_{type}(v_s) \neq a_{type}(v_t), & \end{cases}$$

Note that we now apply the χ^2 distance instead of the Euclidean one, as a_{label} represents a probability distribution.

Edge Substitution Costs We analogously define the edge matching costs

$$c(a_{e_s}, a_{e_t}) = \begin{cases} 0, & \text{if } a_{type}(e_s) = a_{type}(e_t) = 0 \\ \infty, & \text{if } a_{type}(e_s) \neq a_{type}(e_t), \end{cases}$$

which ensures that doors leading to the outside are not matched onto doors inside the building.

The algorithm presented so far finds a graph matching based on very local structures constituted by a node and its adjacent edges. However, in architectural planning the very important concepts of room access and connectivity can often only be understood by considering larger structures that are spanned over several rooms and doors. We therefore extend the edge substitution costs by an additional term. Consider the node substitution of v_s by v_t. Let $e_{ss'} \in E_{adj}(v_s)$ be an edge adjacent to v_s and let $e_{tt'} \in E_{adj}(v_t)$ be an edge adjacent to v_t. Instead of only considering the similarity of edge type attributes $a_{type}(e_{ss'})$ and $a_{type}(e_{tt'})$, we additionally incorporate the outdegree similarity of nodes $v_{s'}$ and $v_{t'}$ into $c_{e_{ss'}e_{tt'}}$. By that, we assure that matches preserve the topology more globally. To additionally control the balance of topology preserving and attribute preserving room matches, we introduce a weighting factor λ_2. Our modified edge substitution costs then read

$$\begin{aligned} c_{e_{ss'}e_{tt'}} &= c(a_{e_{ss'}}, a_{e_{tt'}}) \\ &+ \lambda_2 \cdot \frac{\min\{|V_{adj}(v_{s'})|, |V_{adj}(v_{t'})|\}}{\max\{|V_{adj}(v_{s'})|, |V_{adj}(v_{t'})|\}} \\ &+ (1 - \lambda_2) \cdot c(a_{v_s}, a_{v_t}). \end{aligned}$$

Edge Insertion and Deletion Costs We define insertion and deletion costs for edges connecting rooms in terms of how much accessible area the two adjacent rooms would gain or lose:

$$c_{\epsilon e_{vv'}} = c_{e_{vv'}\epsilon} = \frac{1}{2}\left[\underbrace{\frac{A(v')}{A(V_{adj}(v))}}_{\text{loss/gain at } v} + \underbrace{\frac{A(v)}{A(V_{adj}(v'))}}_{\text{loss/gain at } v'}\right].$$

Consider e.g. a room that has only one door. If this door would be removed, the room would lose all accessible adjacent area. Such a deletion operation should be expensive. Furthermore, consider a corridor with a large number of rooms attached. If this corridor would be cut off from one room, the impact should be small, because there still remains a large accessible area. On the other hand, if an additional room would be attached to such a corridor, the accessible area is also not changed too much. Insertion and deletion should be cheap in such a case.

The costs for insertion / deletion of a door that connects to the outside world are set to a constant, i.e.

$$c_{\epsilon e_{vv_o}} = c_{e_{vv_o}\epsilon} = \kappa \in [0,1]\,.$$

Note that we currently do not check wether a room that is about to be connected to the outside world is bordered by an outside wall at some point.

Node Insertion and Deletion Costs First, consider the deletion of a node v_t in the target graph. Along with the deletion of the associated room, all doors in this room vanish, resulting in the deletion of associated edges. Second, consider insertion of a new node in the target set which is subsequently matched against a node v_s in the source set. As the new room contains no doors, an edge insertion operations must be conducted such that it can be matched against the already existing node v_s and its edges. As a consequence, we define node insertion and deletion costs with respect to the according edge insertion and deletion costs:

$$c_{\epsilon v_t} = \frac{\sum_{t' \in V_{adj}(v_t)} c_{\epsilon e_{tt'}}}{|V_{adj}(v_t)|} \text{ and } c_{v_s \epsilon} = \frac{\sum_{s' \in V_{adj}(v_s)} c_{e_{ss'}\epsilon}}{|V_{adj}(v_s)|}.$$

6.6 Bag-of-Subgraphs Construction

In the following we introduce the two building blocks for constructing the Bag-of-Subgraphs, *decomposition into subgraphs*, *codebook generation*, and finally *subgraph embedding*.

6.6.1 Subgraph Mining

Following the results on efficient subgraph mining presented in [LF06], we use two different sampling strategies, *Forest Fire (FF)* and *Random Walk (RW)*. In the FF method, nodes are sampled in a pattern that is similar to a spreading fire. First, a seed node is randomly chosen and burnt. From each node that is burnt, the fire spreads to a random number of adjacent not-yet-burnt nodes which is chosen with respect to a certain "burning" probability p_{FF}. The procedure is iterated until the desired number of nodes is burnt. The burnt nodes and the connecting edges constitute the subgraph sample. Graphs mined in this way tend to be similar to the node visiting patterns that result from breadth-first search. For our domain of RCGs, FF is especially suited to extract distributors and their adjacent rooms (see Figures 6.3(a) and 6.3(b)). The first step in the RW method also consists of randomly selecting a seed node and marking it as visited. From every visited node, the algorithm either jumps back to the seed node with probability p_{RW}, or moves to an adjacent unvisited node with probability $1 - p_{RW}$. Once the desired number of nodes has been visited, the sampling is constituted by the visited nodes and the connecting edges. Random walks are effective to reflect chain-like structures (see Figure 6.3(e)) similar to those generated by depth-first search. In Section 6.7 we describe how to choose the probabilities p_{FF} and p_{RW} as well as the different sample sizes. Note that for both approaches the resulting subgraphs can overlap.

6.6.2 Codebook Generation

For codebook generation, we first split the set of RCGs into two approximately equally sized sets. The *training set* is used for codebook generation and for computing normalization offsets and scale factors, the *testing* set is used for evaluation. In some cases, the RCGs extracted from different stories of the same building are very similar to each other, sometimes even identical. To avoid a bias in our evaluation, we therefore put all stories of the same building in either the training set or the test set. However, this might induce a slight overfitting problem during high-level attribute learning, as identical RCGs could end up in different cross-validation sets. In the next step, we sample a certain number of subgraphs from the graphs in the training set. A common way to generate a codebook of fixed size from a set of features is to conduct a clustering. For each resulting cluster, one representative is selected and assigned to the codebook. In our setting, we use a k-medoid method instead of k-means which is often used for vector-based features, because computation of a "mean" graph is not straightforward[2]. For computing

[2]A possible solution to this problem is described in [GB02]. There, the weighted sum of two graphs is defined in terms of applying a corresponding number of steps of the optimal edit sequence between them.

the distance during the clustering we use approximate edit distances as described in Section 6.5. We denote the codebook of subgraphs by $\mathcal{Z} = (Z_1, ..., Z_k)$. In Section 6.7, Figure 6.4 we show the influence of the codebook size k onto the retrieval result.

When computing the low-level matching costs $c_{low-level}(a_{v_s}, a_{v_t})$, the similarity of the associated property vectors must be evaluated. As we have no prior knowledge if some of the properties are more important for ensuring good retrieval results with respect to our classification scheme, all entries should have the same influence. To this end we normalize each dimension of the property vector such that the entries regarding the complete training set lie between 0 and 1. We later apply the normalization offsets and scale factors to the test data. By dividing the Euclidean distance between the property vectors consisting of five entries by $\sqrt{5}$ we ensure that the low level matching costs lie between 0 and 1 if they are different from infinity, supposed the training data generalizes to the test data.

6.6.3 Subgraph Embeddings

Consider a single RCG from the testing set. We first sample a certain number of subgraphs in the above described way. We denote the resulting set by $\mathcal{S} = (S_1, ..., S_{|\mathcal{S}|})$. Rather than characterizing each subgraph by its closest codebook graph, we describe it by a Gibbs distribution over the codebook for increased robustness. Let $d_{AGE}(S_i, Z_j)$ denote the approximate graph edit distance between S_i and Z_j, then we compute the probabilistic codebook graph assignment variable a_i by

$$p(a_i = j|S_i) = \frac{\exp(-d_{AGE}(S_i, Z_j))}{\sum_{j'=1}^{k} \exp(-d_{AGE}(S_i, Z_{j'}))}.$$

Finally, we integrate the assignment variable into one vector b_t which is the Bag-of-Features. Each entry of $b_t(j), j = 1, ..., k$ represents a multivariate distribution over the codebook. We can now compute b_t by defining

$$b'_t := \sum_{i=1}^{n} \left(p(a_i = 1|S_i), ..., p(a_i = k|S_i)\right)^T,$$

and by computing entries $b_t(j) = \frac{b'_t(j)-\mu_j}{\sigma_j}$, where μ_j and σ_j are chosen in a way such that $b'_t(j)$ is normalized to zero mean and unit variance of the training set.

6.7 Evaluation

We evaluate our newly introduced approach based on subgraph embeddings against three other strategies regarding RCG retrieval performance using the classification scheme introduced in Section 6.4.2 and Figure 6.3. In the following we briefly summarize the method and describe parameter settings.

6.7.1 Methods and Parameters

For *approximate graph edit distances*, we compute the distance between two RCGs based on the algorithm described in 6.5. For *Lipschitz embeddings*, we first split the set of graphs into a training and a testing set. All graphs in the training set constitute the codebook. We then compute the distance of all graphs in the testing set to all codebook graphs using approximate graph edit distances, resulting in a vector of fixed length for each graph. The dissimilarity of two graphs is finally measured using the L_2 distance of the embeddings. For *subgraph embeddings*, we follow the description in Section 6.6 to compute a vector-based Bag-of-Features description for RCGs. Comparison of two RCGs is again based on the L_2 distance of the embeddings. We additionally evaluate the performance of graph kernels for RCG comparison. Examining geometric random walk kernels as well as exponential random walk kernels [Bor07], the latter provides a slightly better retrieval performance, we therefore only comment on this kernel type. The similarity function of two graphs G_S,G_T using an exponential random walk kernel k reads

$$k(G_S, G_T) = \sum_{i,j=1}^{|V(G_S \times G_T)|} \left[\sum_{k=0}^{\infty} \frac{(\beta A)^k}{k!} \right]_{ij}, \tag{6.3}$$

where $V(G_S \times G_T)$ denotes the nodes of the product graph $G_S \times G_T$ and A denotes the adjacency matrix describing the similarity of connected node pairs in $V(G_S \times G_T)$ (c.f. Section 6.2.2). To provide an evaluation that is comparable to the methods described before, we define this similarity using the node matching costs defined in Equation 6.2:

$$A[(v_s, v_t), (v_{s'}, v_{t'})] = \begin{cases} \exp(c_{v_s v_t}) \cdot \exp(c_{v_{s'} v_{t'}}), \\ \qquad\qquad\qquad \text{if } ((v_s, v_t)(v_{s'} v_{t'})) \in E(G_1 \times G_2) \\ 0, \text{ else.} \end{cases}$$

In our experiments the best results are achieved using $\beta = 4$ and evaluating the sum in Equation 6.3 up to $k = 4$.

We use parameters $\lambda_1 = 0.6$ (except for one setting described below) to balance node and edge matching costs and $\lambda_2 = 0.75$ to weight topology and attribute preservation (see Section 6.5).

For subgraph mining, we choose parameters as suggested in [LF06], leading to a burning-probability $p_{FF} = 0.5$ (which means that the fire spreads to two adjacent nodes on the average) for FF sampling and to a backward jump probability

$p_{RW} = 0.15$ for RW sampling. For each graph $G = (V, E)$, we sample up to 6 subgraphs of all sizes between 1 and $|V|$ nodes if possible. For codebook clustering, we randomly select 3000 sampled subgraphs from the training set, a larger number does not improve the results. In Figure 6.4 we show the influence of the codebook size on the retrieval performance, which increases up to a codebook size of about 100. We therefore use a codebook size of 100 in all other experiments.

Our dataset consists of 199 RCGs. The training and testing set are chosen in a way such that both contain approximately half of the graphs. To reduce the influence of a potentially biased data configuration we conduct 8-fold cross-validation using eight different splits of training and test data in all our experiments. Evaluation is always done exclusively on the testing set. The training set is used to generate the codebook (for Lipschitz and subgraph embeddings) and for learning high-level room types.

6.7.2 Influence of Attributes

For each of the above described methods we investigate the influence of room attributes. We evaluate the complete absence of room attributes (i.e. $a_{prop}(v) = const\ \forall v$), low-level geometric attributes like described in Section 6.4.1, and high-level room types introduced in Section 6.4.2. We additionally examine the influence of neglecting edges when evaluating approximate graph edit distances, i.e. setting $\lambda_1 = 1$ such that only room similarity is considered during matching. Note however that there is implicit knowledge about adjacent edges integrated in the low-level and high-level room attributes.

6.7.3 Retrieval Results

We first evaluate the retrieval performance of each of the four described methods separately regarding the influence of attributes and topology. The results are visualized in precision-recall plots 6.5, 6.6, 6.7, and 6.8. Note that for the graph kernels (Figure 6.8) we do not evaluate neglecting edges because construction of the adjacency matrix A (see Equation 6.3) would then not be possible. As can be seen in all four plots, using high-level room types instead of low-level attributes boost the retrieval performance. We attribute this to the fact that the learned room types are much closer to an architect's understanding to the concept of floor plans than its geometric properties alone. Incorporation of edges during the matching improves the results in most cases to a performance that cannot be achieved using only nodes and the implicit connectivity information encoded in the attributes.

In Figure 6.9, we compare the retrieval performance of all four methods using the individually best setting with high-level room types and edges. Taking into account the observation described in Section 6.4.2 that floors cannot be assigned to a category absolutely certain, neither by machine nor human classifiers, we additionally provide the averaged retrieval performance achieved by three human classifiers all having an educational background in architecture.

Approximate graph edit distances perform better than Lipschitz embeddings for small recall values, for larger recall this tendency is inverted, which is consistent with the results described in [RB09b]. The exponential random walk graph kernel performs worst, except for quite large recall values it is superior to approximate graph edit distances. Our newly introduced method using subgraph embeddings perform better than the three other methods, especially with increasing recall. The human classifiers' average retrieval performance is slightly worse than the automatic methods for small recall values but unambiguously superior for recall values larger than 0.2. Note that the human classifiers are always superior in the settings where no room types are learned.

6.7.4 Classification Results

Additionally to pure retrieval, we evaluate the classification performance of vector-based subgraph and Lipschitz embeddings using standard C-SVM with an RBF kernel provided in the LIBSVM package [CL01]. Like in the retrieval evaluation section, we analyze the influence of attributes and connectivity. We use the training set to learn discriminant functions, predictions are conducted on the test set. We use cross-validation on the training set to determine the necessary SVM parameters. Results are again computed for eight different partitionings of training and test set additionally providing error bars.

Table 6.1 gives an overview of our results. Note that for every feature setting, subgraph embeddings perform better than Lipschitz embeddings. Using predictions of room types based on geometric room attributes boosts the performance. Additionally, combining topology and predicted room types leads to better results than using only topology or only room attributes or their derived floor types.

6.7.5 Timings

In Table 6.2 we show timings for all four retrieval methods using room type predictions and edges. Codebook computation as well as attribute learning is a preprocessing step that only needs to be conducted once. For the Lipschitz embeddings, we use the complete training set as codebook, which is why we do not need to perform a clustering like for the subgraph embeddings. However, with an increasing amount of data, clustering for codebook generation would be required as well. Our newly introduced subgraph embeddings require a rather large amount of time which is due to the relatively expensive embedding step. However, the embedding must only be computed once which means that the query time should

	Lipschitz embeddings	Subgraph embeddings
Low-level attributes | no edges	56.9 ± 2.4	61.1 ± 2.1
No attributes | edges	63.3 ± 3.7	63.4 ± 3.5
Low-level attributes | edges	64.9 ± 3.0	65.0 ± 3.6
High-level attributes | no edges	75.9 ± 2.9	78.0 ± 3.2
High-level attributes | edges	77.0 ± 3.9	79.5 ± 2.6
Human classifier	79.1 ± 3.4	

Table 6.1: **Classification results.** The table shows floor classification rates using graph embeddings and subgraph embeddings with respect to different attribute and connectivity settings.

	AGED	Lipschitz	Subgraphs	Graph kernels
Codebook creation	-	-	6.58 min	-
Attribute learning	10 sec			
Query time	55 ms	64 ms	1.39 s	1.875 s

Table 6.2: **Timings.** AGED denotes approximate graph edit distances. Codebook generation and attribute learning times are with respect to a training set of approximately 100 RCGs. Query time per object is averaged over the whole testing set.

nevertheless scale well with increasing database size. In contrast to approximate graph edit distances and graph kernels, subgraph as well as Lipschitz embeddings only require a vector comparison for retrieval, rendering these approaches suitable for large databases.

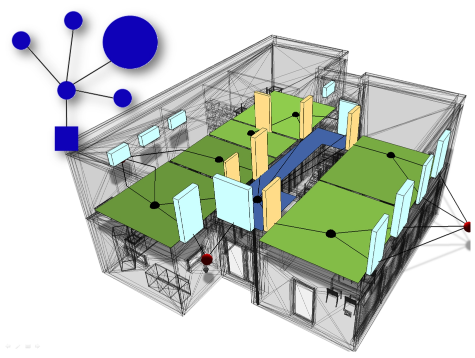
6.8 Conclusion

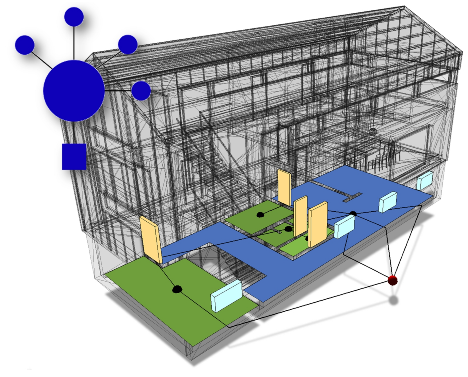
In this Chapter we introduced a new method to efficiently compute similarity between large numbers of RCGs. Our approach relies on the embedding of the graph in a finite vector space using a Bag-of-Subgraphs. The similarity between feature subgraphs and a set of codebook graphs is computed with a modified version of approximate graph edit distances [RB09a]. We use low-level room properties describing the connectivity as well as the shape as node attributes. Furthermore, we use a supervised learning approach to transform these low-level geometric and topological features into high-level room types that better reflect an architect's understanding of a room. In our experiments we could show that our new method using subgraph embeddings performs slightly better than related approaches, including pure approximate graph edit distances (AGED), Lipschitz embeddings, and an exponential random walk graph kernel. However, from our point of view, the more interesting lesson learned is that obviously, using sophisticated methods for measuring node similarity seems to be far more important than the choice of the overall graph matching algorithm. A large improvement of retrieval and classification performance of all methods could be noted when using the high-level attributes instead of the low-level ones. As the computation of the subgraph embeddings is rather expensive, our approach currently requires more time than computing AGED and Lipschitz embeddings. However, the overall computation time is not that large (1.39 seconds on average) and for larger datasets, AGED will at some point take longer to compute than the subgraph embeddings, especially if acceleration structures are used.

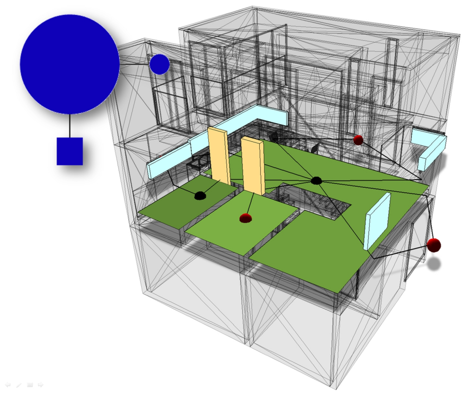
Limitations Despite the good performance of the newly introduced subgraph embeddings we can hardly conclude on the hypothetical performance of this approach if applied to other retrieval and classification problems involving graphs. This is due to the fact that the story type scheme does not seem to allow a consistent classification even by a human being, which is supported by the retrieval and classification results achieved by our architecture experts (see Figure 6.9 and Table 6.1). Another limitation is represented by our current subgraph sampling scheme which consists of randomly sampling up to six subgraphs of sizes between 1 and $|V|$ nodes. It is obvious that the larger the graphs become in terms of nodes, the less stable our method will perform, as sampling a particular graph of a certain size will become less and less likely due to the exponential growth of the binomial coefficient. However, for the current dataset containing rather small RCGs this problem does not seem to occur.

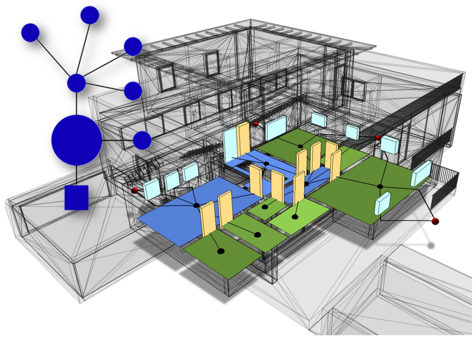
Future Work Future work should investigate the extension of our method from RCGs representing single stories to graphs representing complete buildings. Analogously, the classification scheme should be extended to floor plan classes covering several stories (e.g. maisonette, split-level). When considering more than one unit within one story, the classification has to be amplified towards outer access. Additionally, the influence of further geometric attributes could be examined. Apart from the RCG specific application, we recommend to investigate the performance of subgraph embeddings on other graph data in order to get a better estimation of their real potential. Regarding the above mentioned sampling problem that will inevitably occur when processing larger graphs like for example proteins, we suggest to either develop some importance sampling strategy to counteract the amount of randomness in the selection, or to combine it with a rather completely non-probabilistic strategy like spectral graph decomposition using the Fiedler vector [Fie73]. Such a decomposition could be used to construct a first coarse segmentation. The resulting parts could then be processed using the probabilistic sampling. However, the robustness and consistency of a decomposition for graphs of the same object class might highly vary depending on the classification scheme.

Acknowledgements We would like to thank Tobias Pohlmann from TIB, Torben Kohls from Leibniz University Hannover, and Anja Henkel for acting as human reference classifiers. Sincere thanks are given to the Faculty of Architecture and Landscape Sciences of Leibniz University Hannover as well as to the Faculty of Architecture of Technische Universität Braunschweig for providing the 3D building models.


(a) corridor type


(b) distributor type


(c) loft type


(d) hierarchical type

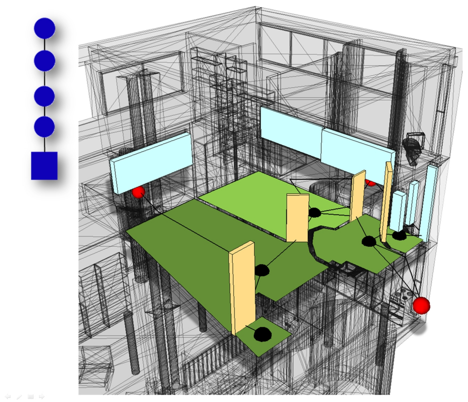

(e) chain type

Figure 6.3: **Floor types.** Rooms serving as corridor or distributor are depicted in blue, all other rooms in different shades of green. Doors are visualized in yellow, windows in light blue. The pictograms in dark blue show ideal examples of the different room types. Red spheres represent the outside world node (multiple occurrences in one image are only for clearer visualization).

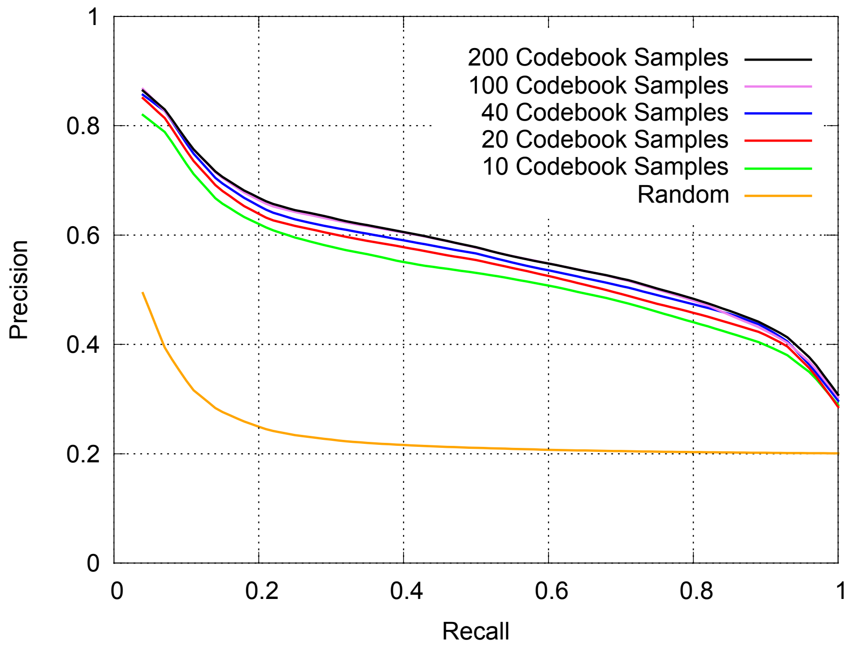


Figure 6.4: **Codebook size variation.** The plot shows retrieval performance of subgraph embeddings using high-level attributes and topological information und varying codebook size.

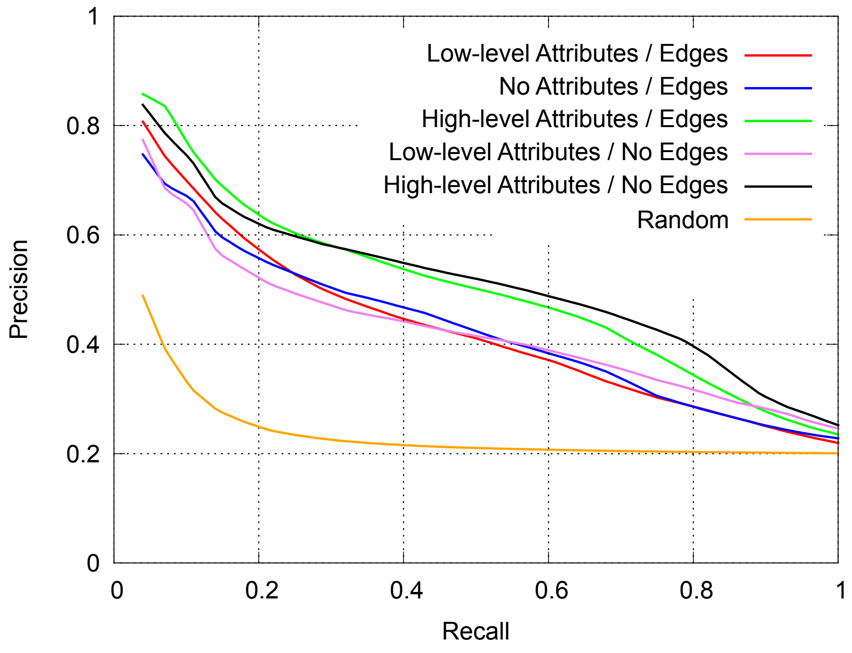


Figure 6.5: Retrieval results for **approximate graph edit distances**.

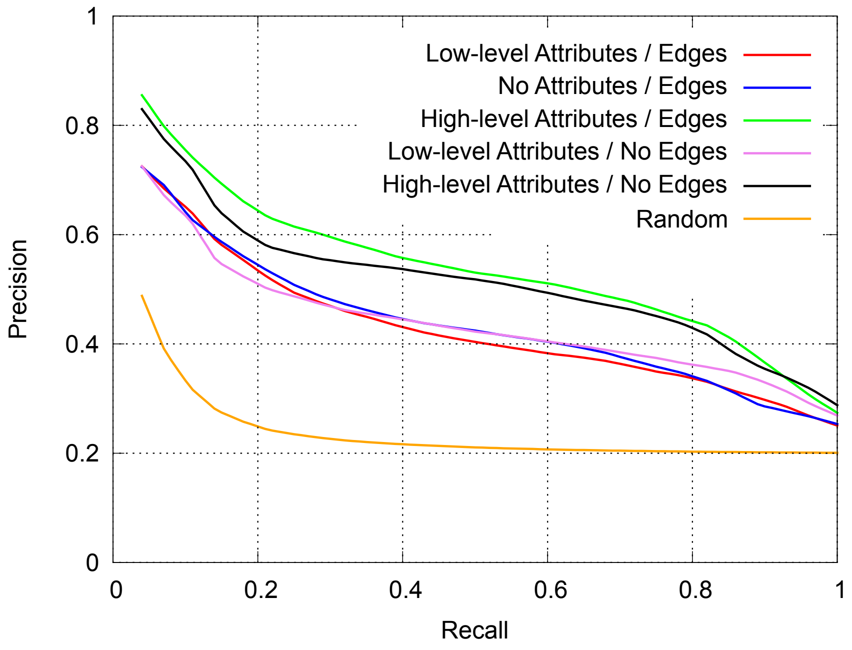


Figure 6.6: Retrieval results for **Lipschitz embeddings**.

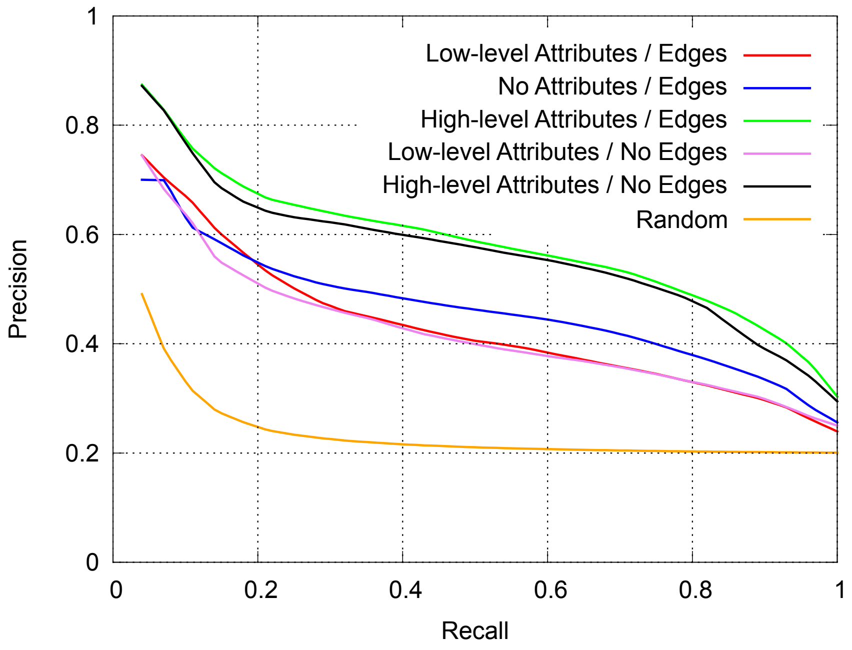


Figure 6.7: Retrieval results for **subgraph embeddings (our method)**.

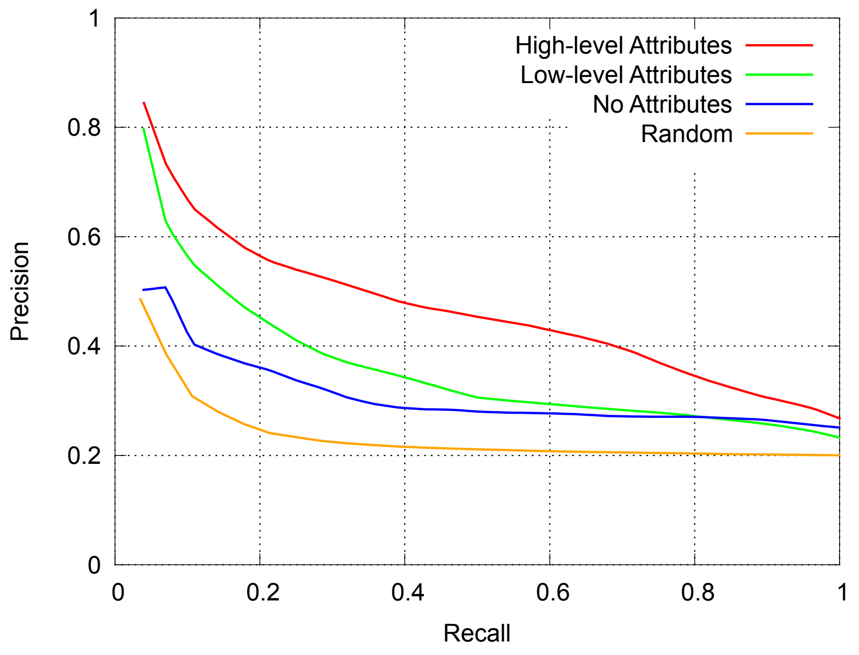


Figure 6.8: Retrieval results for **exponential random walk kernels**.

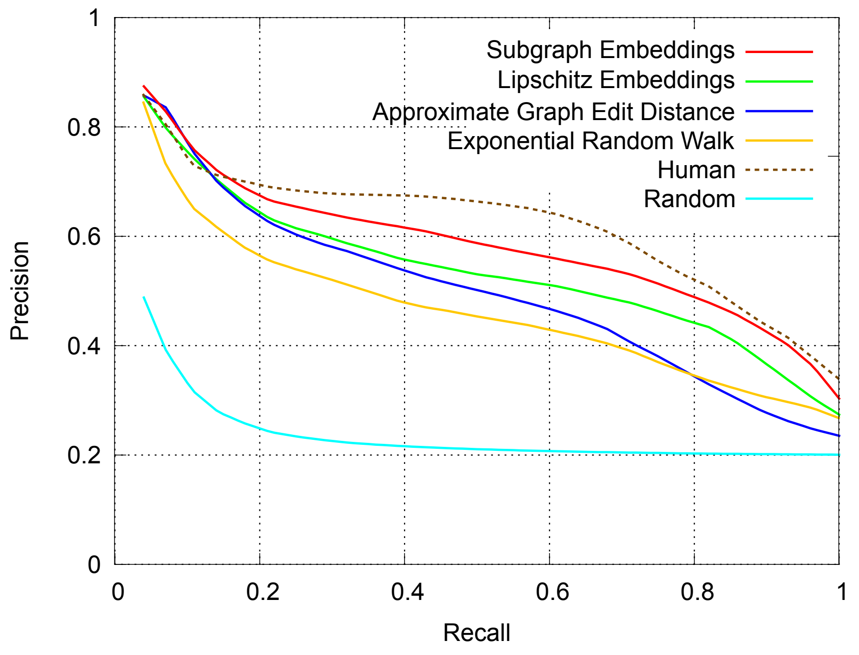


Figure 6.9: **Comparison of best performing strategies** as well as human classifier performance. All methods use room type predictions and topological information.

Part III

Closure

CHAPTER 7

CONCLUSIONS

7.1 Summary

The main motivation for our research on shape retrieval methods for objects from the architectural domain was the occurring shift from 2D analogue to 3D digital drafting. In order to efficiently reuse existing resources either for integration into new drafts or for inspirational purposes, 3D content must be made searchable. Context objects and building models thereby require different approaches, which resulted in this thesis being comprised of two parts. While for context objects, shape-related properties are most important for similarity determination, building models are highly dominated by the topological arrangement of rooms and stories.

Part I: Context Models In the first part of this thesis we dedicated ourselves to the challenge of improving shape retrieval by combining methods from supervised learning and local shape descriptors. We thereby tried to overcome the disadvantages that arise from the standard approaches involving local features: On the one hand, establishing feature correspondences is a very cumbersome process. It often requires to manually choose several parameters which makes it hardly automatable. On the other hand, Bag-of-Features representations can hardly include information about the spatial relationship between descriptors. Additionally, a single, actually highly descriptive feature might not have enough weight in the histogram representation. As a solution, we proposed class distribution descriptors, a representation that is uncoupled from the concrete type of the underlying feature. We first showed that CDDs can boost the performance of virtually any local or global descriptor due to the use of shape knowledge acquired in a training step. After showing superior results on the generic Princeton Shape Benchmark we applied our new method to dataset consisting only of architectural context object. Due to the fine granularity, the retrieval results were less persuading than on generic data, but yet our method performed better than the approaches we compare it to.

To further improve retrieval results we extended our method for computing

CDDs such that it did not only comprise information about the geometry represented by local feature but also about its position relative to the object center and relative to other features. To this end we used a decomposition of the object into geometric shape primitives. In our experiments on the architectural benchmark we could demonstrate enhanced retrieval performance. As a drawback of this method we identified the computational complexity in the case of incorporation of spatial relationship within larger feature tuples.

In the last Chapter of the first part we demonstrated the flexibility of our newly introduced approach by computing CDDs from non-shape related features including text and material properties. We additionally used intrinsic segmentation information as a feature localization method that is presumably orthogonal to the decomposition into primitive shapes. By a smart combination of the resulting CDDs from different features and segmentation strategies we were able to further improve the retrieval quality. However, even when using the combined force of all features, there is still room for improvement regarding the retrieval performance on the architectural context object benchmark.

Part II: Building Models While in the first part of this thesis we could build on the tremendous work on 3D shape descriptors that had been conducted over the recent decade, the need for a description to capture the structural characteristics of a building model required us to develop rather new methods. We introduced the attributed room connectivity graph to represent the arrangement of rooms and stories in a building as a graph with attributed nodes and edges. After introducing this rather abstract concept we invented methods to extract RCG representations from 3D building models consisting of polygon soups. We thereby had to restrict ourselves to a subset from the large variety of buildings, as certain shapes and structures would require additional and more sophisticated extraction methods. We then demonstrated the feasibility of the RCG concept and our extraction methods by searching a database of graphs for certain room substructures.

In the following Chapter we introduced several methods to compute a real similarity value between two RCGs instead of only being able to search for substructures. Based on approximate graph edit distances, we developed subgraph embeddings as an efficient means to transform the graphs into a vector-based representation that allows fast retrieval and browsing in even large databases. We concluded by suggesting to investigate the applicability of our new method to other retrieval problems involving graphs and pointed out the possible problems that might occur if the graphs consist of a large number of nodes.

7.2 Future Work

We already discussed rather concrete, technical future work that arises from our research at the end of every Chapter. In the following we would therefore like to briefly discuss future work in terms of a bit more abstract visions.

Since its introduction in the late 1990s, the field of shape retrieval in general has been mostly concentrated on content that was generated using some standard modeling software. This is a big difference when compared to the area of computer vision, where the focus has ever since been on aquired images showing real-life objects. The reason is as simple as obvious: While digital 2D acquisition devices have been extremely cheap and easy to handle for a long period of time, and even before digitalisation of analogue photos was widely used, 3D acquisition devices have only recently become affordable to end users, like e.g. the Microsoft Kinect. Despite improvements in object reconstruction from such devices like e.g. presented in [IKH$^+$11], they are still far less handy than their 2D counterparts. However, we believe that 3D geometry acquisition covering all thinkable levels of quality will at some point become similarly important. Already even today, laser scanning plays an ever increasing role. The resulting 3D reconstructions of existing objects require a different approach regarding the requirements that allow retrieval. While usually a CAD file contains one more or less semantically well-defined object, a 3D scan contains all sorts of objects, parts of objects, and a lot of clutter. Retrieval and recognition require some sort of prior segmentation in this case. By that, 3D shape retrieval becomes somewhat more similar to the problems in computer vision.

Apart from acquiring 3D geometry, highly sophisticated modeling tools following the BIM paradigm have and will become ever more important in covering the complete lifecycle of a modern building. This also includes highly flexible parametric models representing context objects whose shape can be adopted to the requirements of the particular building situation. Taking into account our above remarks, we are confronted with a realm of 3D objects that breaks down into two extrema: On the one hand, we acquire highly cluttered scenes consisting of unsegmented geometric fragments, on the other hand we have access to semantically highly enriched parameterized 3D BIM content. From our point of view, one of the central challenges for the future is to bridge this gap, i.e. we must develop methods to put more intelligence into e.g. point clouds, such that they eventually become as handy as BIM data. This will enable to efficiently handle legacy buildings for which no sophisticated BIM model exists in terms of retrofitting, but it will also ensure the searchability of this data for any thinkable purpose. The pressing need to focus on these upcoming challenges in the domain of architecture is underlined by two recent publications that address the problem of indoor scene segmentation and understanding from 3D laser scans, see [NXS12] and [KMYG12].

BIBLIOGRAPHY

[ADBP04] Jürgen Assfalg, Alberto Del Bimbo, and Pietro Pala. Spin Images for Retrieval of 3D Objects by Local and Global Similarity. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3 - Volume 03*, ICPR '04, pages 906–909, Washington, DC, USA, 2004. IEEE Computer Society.

[AFS06] Marco Attene, Bianca Falcidieno, and Michela Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22(3):181–193, 2006.

[AK11] Warawit Areevijit and Pizzanu Kanongchaiyos. Reeb graph based partial shape retrieval for non-rigid 3D object. In *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, VRCAI '11, pages 573–576, New York, NY, USA, 2011. ACM.

[AKKS99] Mihael Ankerst, Gabi Kastenmüller, Hans-Peter Kriegel, and Thomas Seidl. 3D Shape Histograms for Similarity Search and Classification in Spatial Databases. In *Proceedings of the 6th International Symposium on Advances in Spatial Databases*, SSD '99, pages 207–226, London, UK, UK, 1999. Springer-Verlag.

[ASYS08] Ceyhun Akgül, Bülent Sankur, Yücel Yemez, and Francis Schmitt. Similarity score fusion by ranking risk minimization for 3D object retrieval. In *Proceedings of the 2008 EUROGRAPHICS Workshop on 3D Object Retrieval*, pages 1–9, April 2008.

[AVMD04] Tarik Filali Ansary, Jean-Philippe Vandeborre, Said Mahmoudi, and Mohamed Daoudi. A Bayesian Framework for 3D Models Retrieval Based on Characteristic Views. In *Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium*, 3DPVT '04, pages 139–146, Washington, DC, USA, 2004. IEEE Computer Society.

[BA83] Horst Bunke and G. Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4):245–253, 1983.

[BA08] Silvia Biasotti and Marco Attene. SHape REtrieval Contest 2008: Stability Track on Watertight Models. Technical Report 01/2008, Institute of Applied Mathematics and Information Technologies of the CNR, 2008.

[BAB$^+$08] Silvia Biasotti, Dominique Attali, Jean-Daniel Boissonnat, Herbert Edelsbrunner, Gershon Elber, Michela Mortara, Gabriella Sanniti di Baja, Michela Spagnuolo, and Mirela Tanase. *Skeletal structures*. Springer, 2008. In Shape Analysis and Structuring.

[Bar12] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, New York, NY, USA, 2012.

[BBGO11] Alexander M. Bronstein, Michael M. Bronstein, Leonidas J. Guibas, and Maks Ovsjanikov. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics*, 30(1):1:1–1:20, February 2011.

[bDK05] Kai bo Duan and S. Sathiya Keerthi. Which is the best multiclass SVM method? An empirical study. In *Proceedings of the Sixth International Workshop on Multiple Classifier Systems*, pages 278–285, 2005.

[BHAT05] Eugen Barbu, Pierre Héroux, Sébastien Adam, and Éric Trupin. Using bags of symbols for automatic indexing of graphical document image databases. In *Graphics Recognition (GREC)*, pages 195–205, 2005.

[BI04] Angela Brennecke and Tobias Isenberg. 3D Shape Matching Using Skeleton Graphs. In *In Simulation and Visualization*, pages 299–310, 2004.

[BK10] Michael M. Bronstein and Iasonas Kokkinos. Scale-invariant heat kernel signatures for non-rigid shape recognition. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1704–1711, June 2010.

[BKS$^+$05] Benjamin Bustos, Daniel A. Keim, Dietmar Saupe, Tobias Schreck, and Dejan V. Vranić. Feature-based similarity search in 3D object databases. *ACM Computing Surveys*, 37(4):345–387, December 2005.

[Blü13] Ina Blümel. *Metadatenbasierte Kontextualisierung architektonischer 3D-Modelle*. PhD thesis, Humboldt-Universität zu Berlin, 2013.

[BLYL13] Xiang Bai, Chunyuan Li, Xingwei Yang, and Longin Jan Latecki. *Shape Retrieval and Classification Based on Geodesic Paths in Skeleton Graphs*. IGI Global, 2013. In Graph-Based Methods in Computer Vision: Developments and Applications.

[BMM$^+$03] Silvia Biasotti, Simone Marini, Michela Mortara, Giuseppe Patanè, Michela Spagnuolo, and Bianca Falcidieno. 3D Shape Matching through Topological Structures. In Ingela Nyström, Gabriella Sanniti di Baja, and Stina Svensson, editors, *DGCI*, volume 2886 of *Lecture Notes in Computer Science*, pages 194–203. Springer, 2003.

[BMP02] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[BMSF06] Silvia Biasotti, Simone Marini, Michela Spagnuolo, and Bianca Falcidieno. Sub-part correspondence by structural descriptors of 3D shapes. *Computer-Aided Design*, 38(9):1002 – 1019, 2006.

[Bor07] Karsten M. Borgwardt. *Graph Kernels*. PhD thesis, Ludwig-Maximilians-University Munich, 2007.

[BR08] Horst Bunke and Kaspar Riesen. Graph classification based on dissimilarity space embedding. In *Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, SSPR & SPR '08, pages 996–1007, Berlin, Heidelberg, 2008. Springer-Verlag.

[Bri01] British Standards Institution. *BS ISO 12006-2: Building construction - Organization of information about construction works - Part 2: Framework for classification of information*. Beuth-Verlag, November 2001.

[Bri07] British Standards Institution. *BS ISO 12006-3: Building construction - Organization of information about construction works - Part 3: Framework for object-oriented information*. Beuth-Verlag, August 2007.

[BRS03] Dmitriy Bespalov, William C. Regli, and Ali Shokoufandeh. Reeb Graph Based Shape Retrieval for CAD. *Proceedings of DETC'03*

2003 ASME Design Engineering Technical Conferences, September 2003.

[BYRN99] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[CDS$^+$05] Nicu D. Cornea, M. Fatih Demirci, Deborah Silver, Ali Shokoufandeh, Sven J. Dickinson, and Paul B. Kantor. 3D Object Retrieval using Many-to-many Matching of Curve Skeletons. In *Proceedings of the International Conference on Shape Modeling and Applications 2005*, SMI '05, pages 368–373, Washington, DC, USA, 2005. IEEE Computer Society.

[CL01] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm` [last accessed 7 February 2013].

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM Press.

[CR92] Nigel Cross and Norbert Roozenburg. Modelling the design process in engineering and in architecture. *Journal of Engineering Design*, 3(4):325–337, 1992.

[CR01] Vincent Cicirello and William C. Regli. Machining feature-based comparisons of mechanical parts. In *In International Conference on Shape Modeling and Applications, pages 176Ű187. ACM SIGGRAPH, the Computer Graphics Society and EUROGRAPHICS, IEEE Computer*, pages 176–185. Society Press, 2001.

[CRC$^+$02] Jonathan Corney, Heather Rea, Doug Clark, John Pritchard, Michael Breaks, and Roddy Macleod. Coarse filters for shape matching. *Computer Graphics and Applications, IEEE*, 22(3):65–74, May / June 2002.

[CSM07] Nicu D. Cornea, Deborah Silver, and Patrick Min. Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13:530–548, May 2007.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons Inc., second edition, 2006.

[CTSO03] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3D model retrieval. *Computer Graphics Forum*, 22(3):223–232, September 2003.

[DCG12] Helin Dutagaci, Chun Pan Cheung, and Afzal Godil. Evaluation of 3D interest point detection techniques via human-generated ground truth. *The Visual Computer*, 28(9):901–917, September 2012.

[DHS01] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, New York, NY, 2nd edition, 2001.

[DLL$^+$10] Tamal K. Dey, Kuiyu Li, Chuanjiang Luo, Pawas Ranjan, Issam Safa, and Yusu Wang. Persistent heat signature for pose-oblivious matching of incomplete models. *Computer Graphics Forum*, 29(5):1545–1554, 2010.

[DSS$^+$10] Mohamed Daoudi, Tobias Schreck, Michela Spagnuolo, Ioannis Pratikakis, Remco C. Veltkamp, and Theoharis Theoharis, editors. *Eurographics Workshop on 3D Object Retrieval, Norrköping, Sweden, May 2, 2010, Proceedings*. Eurographics Association, 2010.

[DvGNK99] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, January 1999.

[EF84] Mohamed A. Eshera and King-Sun Fu. A graph distance measure for image analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 14(3):398–408, May 1984.

[EMM03a] Mohamed El-Mehalawi and Allen Miller. A database system of mechanical components based on geometric and topological similarity. Part 1: representation. *Computer-Aided Design*, 35(1):83–94, January 2003.

[EMM03b] Mohamed El-Mehalawi and Allen Miller. A database system of mechanical components based on geometric and topological similarity. Part 2: indexing, retrieval, matching, and similarity assessment. *Computer-Aided Design*, 35(1):95–105, January 2003.

[ENR97] Alexei Elinson, Dana S. Nau, and William C. Regli. Feature-based similarity assessment of solid models. In *Proceedings of the fourth*

ACM symposium on Solid modeling and applications, SMA '97, pages 297–310, New York, NY, USA, 1997. ACM.

[ERB$^+$12] Mathias Eitz, Ronald Richter, Tamy Boubekeur, Kristian Hildebrand, and Marc Alexa. Sketch-based shape retrieval. *ACM Transactions on Graphics(Proceedings of SIGGRAPH)*, 31(4):31:1–31:10, 2012.

[ETA02] Michael Elad, Ayellet Tal, and Sigal Ar. Content based retrieval of vrml objects: an iterative and interactive approach. In *Proceedings of the sixth Eurographics workshop on Multimedia 2001*, pages 107–118, New York, NY, USA, 2002. Springer-Verlag New York, Inc.

[ETSL08] Chuck Eastman, Paul Teicholz, Rafael Sacks, and Kathleen Liston. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. Wiley Publishing, 2008.

[Fal98] Kristine K. Fallon. Early computer graphics developments in the architecture, engineering, and construction industry. *IEEE Annals of the History of Computing*, 20(2):20–29, April 1998.

[FGLW08] Rui Fang, Afzal Godil, Xiaolan Li, and Asim Wagan. A New Shape Benchmark for 3D Object Retrieval. In *Proceedings of the 4th International Symposium on Advances in Visual Computing*, ISVC '08, pages 381–392, Berlin, Heidelberg, 2008. Springer-Verlag.

[Fie73] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 98:298–305, 1973.

[FK04] Thomas Funkhouser and Michael Kazhdan. Shape-based retrieval and analysis of 3D models. In *ACM SIGGRAPH 2004 Course Notes*, SIGGRAPH '04, New York, NY, USA, 2004. ACM.

[FMA$^+$10] Alfredo Ferreira, Simone Marini, Marco Attene, Manuel J. Fonseca, Michela Spagnuolo, Joaquim A. Jorge, and Bianca Falcidieno. Thesaurus-based 3D object retrieval with part-in-whole matching. *International Journal of Computer Vision*, 89:327–347, September 2010.

[FMK$^+$03] Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs. A search engine for 3D models. *ACM Transactions on Graphics*, 22(1):83–105, 2003.

[FO09] Takahiko Furuya and Ryutarou Ohbuchi. Dense sampling and fast encoding for 3D model retrieval using bag-of-visual features. In *Proceeding of the ACM International Conference on Image and Video Retrieval*, CIVR '09, pages 26:1–26:8, New York, NY, USA, 2009. ACM.

[FS06] Thomas Funkhouser and Philip Shilane. Partial matching of 3D shapes with priority-driven search. In *Proceedings of the 2006 Symposium on Geometry Processing*, pages 131–142, Aire-la-Ville, Switzerland, Switzerland, 2006. EUROGRAPHICS Association.

[FSH11] Matthew Fisher, Manolis Savva, and Pat Hanrahan. Characterizing structural relationships in scenes using graph kernels. In *SIGGRAPH '11*. ACM, 2011. To appear.

[Gär03] Thomas Gärtner. A survey of kernels for structured data. *SIGKDD Explorations Newsletter*, 5(1):49–58, July 2003.

[Gär05] Thomas Gärtner. *Kernels for Structured Data*. PhD thesis, Universität Bonn, 2005.

[GB02] Simon Günter and Horst Bunke. Self-organizing map for clustering in the graph domain. *Pattern Recognition Letters*, 23(4):405 – 417, 2002.

[GCO06] Ran Gal and Daniel Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Transactions on Graphics*, 25(1):130–150, 2006.

[GJ90] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[Har54] Zellig Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[Hau99] David Haussler. Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA, 1999.

[HB07] Zaïd Harchaoui and Francis Bach. Image classification with segmentation graph kernels. In *Proceedings of the 2007IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '07, pages 1–8, 2007.

[Hen09] Armin Hentschel. *Nutzeransichten – Wohnarchitektur aus Sicht ihrer Nutzer*. PhD thesis, Humboldt-Universität zu Berlin, 2009.

[HH09] Jiaxi Hu and Jing Hua. Salient spectral geometric features for shape matching and retrieval. *The Visual Computer*, 25(5-7):667–675, 2009.

[HKSV02] Martin Heczko, Daniel Keim, Dietmar Saupe, and Dejan V. Vranic. Methods for similarity search on 3D databases. *Datenbank-Spektrum (in German)*, 2(2):54–63, 2002.

[HLR05] Suyu Hou, Kuiyang Lou, and Karthik Ramani. SVM-based semantic clustering and retrieval of a 3D model database. *Journal of Computer Aided Design and Application*, 2:155–164, 2005.

[Hol07] Klaus Holschemacher. *Entwurfs- und Berechnungstafeln*. Bauwerk Verlag GmbH, Berlin, 2007.

[Hol11] Dominik Holzer. BIM's Seven Deadly Sins. *International Journal of Architectural Computing*, 9:463–480, December 2011.

[HR07] Suyu Hou and Karthik Ramani. Calligraphic interfaces: Classifier combination for sketch-based 3D part retrieval. *Computers & Graphics*, 31(4):598–609, 2007.

[HRNS11] Tomoki Hayashi, Benjamin Raynal, Vincent Nozick, and Hideo Saito. Skeleton Features Distribution for 3D Object Retrieval. In *Proceedings of the 12th IAPR Machine Vision and Applications (MVA2011)*, pages 377–380, Nara, Japan, 13-15, June 2011.

[HSKK01] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 203–212, New York, NY, USA, 2001. ACM.

[HT98] Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. In *Proceedings of the 1997 conference on Advances in Neural Information Processing Systems 10*, NIPS '97, pages 507–513, Cambridge, MA, USA, 1998. MIT Press.

[IJL$^+$04] Natraj Iyer, Subramaniam Jayanti, Kuiyang Lou, Yagnanarayanan Kalyanaraman, and Karthik Ramani. A Multi-Scale Hierarchical 3D Shape Representation for Similar Shape Retrieval. In Horváth and Xirouchakis, editors, *Proceedings of the TMCE 2004, April 12-16, 2004, Lausanne, Switzerland, Edited by*. ASME, April 2004.

[IKH$^+$11] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 559–568, New York, NY, USA, 2011. ACM.

[ILSR02] Cheuk Yiu Ip, Daniel Lapadat, Leonard Sieger, and William C. Regli. Using shape distributions to compare solid models. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, SMA '02, pages 273–280, New York, NY, USA, 2002. ACM.

[IRSS03] Cheuk Yiu Ip, William C. Regli, Leonard Sieger, and Ali Shokoufandeh. Automated learning of model classifications. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, SM '03, pages 322–327, New York, NY, USA, 2003. ACM.

[IW02] Horace H. S. Ip and William Y. F. Wong. 3D head models retrieval based on Hierarchical facial region similarity. In *Proceedings of the 15th International Conference on Vision Interface*, pages 314–419, May 2002.

[JH99] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[JK00] Kalervo Järvelin and Jaana Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, pages 41–48, New York, NY, USA, 2000. ACM.

[JKIR06] Subramaniam Jayanti, Yagnanarayanan Kalyanaraman, Natraj Iyer, and Karthik Ramani. Developing an engineering shape benchmark for cad models. *Computer-Aided Design*, 38(9):939–953, 2006.

[Joa98] Thorsten Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of* *the 10th European Conference on Machine Learning*, ECML '98, pages 137–142, London, UK, UK, 1998. Springer-Verlag.

[Joh97] Andrew Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 1997.

[Kaz07] Michael Kazhdan. An Approximate and Efficient Method for Optimal Rotation Alignment of 3D Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1221–1229, July 2007.

[KCD$^+$03] Michael Kazhdan, Bernard Chazelle, David Dobkin, Thomas Funkhouser, and Szymon Rusinkiewicz. A reflective symmetry descriptor for 3D models. *Algorithmica*, 38(1), October 2003.

[Kei99] Daniel A. Keim. Efficient geometry-based similarity search of 3D spatial databases. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, SIGMOD '99, pages 419–430, New York, NY, USA, 1999. ACM.

[KFR03] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Proceedings of the 2003 Symposium on Geometry Processing*, pages 156–164, 2003.

[KMYG12] Young Min Kim, Niloy J. Mitra, Dong-Ming Yan, and Leonidas Guibas. Acquiring 3D indoor environments with variability and repetition. *ACM Transactions on Graphics*, 31(6):138:1–138:11, November 2012.

[KPNK03] Marcel Körtgen, Gil-Joo Park, Marcin Novotni, and Reinhard Klein. 3D Shape Matching with 3D Shape Contexts. In *The 7th Central European Seminar on Computer Graphics*, April 2003.

[Lau83] Jean-Paul Laumond. Model structuring and concept recognition: two aspects of learning for a mobile robot. In *Proceedings of the Eighth international joint conference on Artificial intelligence - Volume 2*, IJCAI'83, pages 839–841, San Francisco, CA, USA, 1983. Morgan Kaufmann Publishers Inc.

[LF06] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference* *on Knowledge discovery and data mining*, KDD '06, pages 631–636, New York, NY, USA, 2006. ACM.

[LGB$^+$12] Zhouhui Lian, Afzal Godil, Benjamin Bustos, Mohamed Daoudi, Jeroen Hermans, Shun Kawamura, Yukinori Kurita, Guillaume Lavoué, Hien Van Nguyen, Ryutarou Ohbuchi, Yuki Ohkita, Yuya Ohishi, Fatih Porikli, Martin Reuter, Ivan Sipiran, Dirk Smeets, Paul Suetens, Hedi Tabia, and Dirk Vandermeulen. A comparison of methods for non-rigid 3D shape retrieval. *Pattern Recognition*, 46(1):449 – 461, 2012.

[LGF$^+$10] Zhouhui Lian, Afzal Godil, Thomas Fabry, Takahiko Furuya, Jeroen Hermans, Ryutarou Ohbuchi, Chang Shu, Dirk Smeets, Paul Suetens, Dirk Vandermeulen, and Stefanie Wuhrer. SHREC'10 Track: Non-rigid 3D Shape Retrieval. In Mohamed Daoudi, Tobias Schreck, Michela Spagnuolo, Ioannis Pratikakis, Remco C. Veltkamp, and Theoharis Theoharis, editors, *3DOR*, pages 101–108. Eurographics Association, 2010.

[LGS10] Zhouhui Lian, Afzal Godil, and Xianfang Sun. Visual Similarity Based 3D Shape Retrieval Using Bag-of-Features. *Shape Modeling and Applications, International Conference on*, 0:25–36, 2010.

[LGW08] Xiaolan Li, Afzal Godil, and Asim Wagan. Spatially enhanced bags of words for 3D shape retrieval. In *Proceedings of the 2008 International Symposium on Advances in Visual Computing*, pages 349–358, 2008.

[Liu12] Qiong Liu. A Survey of Recent View-based 3D Model Retrieval Methods. *The Computing Research Repository (CoRR)*, abs/1208.3670, 2012.

[LJI$^+$03] Kuiyang Lou, Subramaniam Jayanti, Natraj Iyer, Yagnanarayanan Kalyanaraman, Sunil Prabhakar, and Karthik Ramani. A reconfigurable 3D engineering shape search system part II: database indexing, retrieval and clustering. In *Proceedings of the ASME 2003 International Design Engineering Technical Conferences*, pages 169–178, September 2003.

[LN07] Hamid Laga and Masayuki Nakajima. A boosting approach to content-based 3D model retrieval. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques* *in Australia and Southeast Asia*, pages 227–234, New York, NY, USA, 2007. ACM.

[Löf00] Jobst Löffler. Content-based Retrieval of 3D Models in Distributed Web Databases by Visual Shape Information. In *Proceedings of the International Conference on Information Visualisation*, IV '00, pages 82–87, Washington, DC, USA, 2000. IEEE Computer Society.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[LSAR10] Ce Liu, Lavanya Sharan, Edward H. Adelson, and Ruth Rosenholtz. Exploring features in a bayesian framework for material recognition. In *CVPR*, pages 239–246, 2010.

[LSF$^+$11] Hamid Laga, Tobias Schreck, Alfredo Ferreira, Afzal Godil, Ioannis Pratikakis, and Remco C. Veltkamp, editors. *Eurographics Workshop on 3D Object Retrieval 2011, Llandudno, UK, April 10, 2011. Proceedings*. Eurographics Association, 2011.

[LSG$^+$12] Bin Li, Tobias Schreck, Afzal Godil, Marc Alexa, Tamy Boubekeur, Benjamin Bustos, J. Chen, Mathias Eitz, Takahiko Furuya, Kristian Hildebrand, S. Huang, Henry Johan, Arjan Kuijper, Ryutarou Ohbuchi, Ronald Richter, Jose M. Saavedra, Maximilian Scherer, Tomohiro Yanagimachi, Gang-Joon Yoon, and Sang Min Yoon. SHREC'12 Track: Sketch-Based 3D Shape Retrieval. In Michela Spagnuolo, Michael M. Bronstein, Alexander M. Bronstein, and Alfredo Ferreira, editors, *3DOR*, pages 109–118. Eurographics Association, 2012.

[LSSK09] Bao Li, Ruwen Schnabel, Jin Shiyao, and Reinhard Klein. Variational surface approximation and model selection. *Computer Graphics Forum*, 28(7), October 2009.

[LVJ05] Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. Mesh saliency. *ACM Transactions on Graphics*, 24(3):659–666, 2005.

[LW88] Yehezkel Lamdan and Haim J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *Proceedings of the 2nd International Conference on Computer Vision*, ICCV '88, pages 238–249, 1988.

[LZL+12] Yong-Jin Liu, Yi-Fu Zheng, Lu Lv, Yu-Ming Xuan, and Xiao-Lan Fu. 3D model retrieval based on color + geometry signatures. *The Visual Computer*, 28(1):75–86, January 2012.

[LZQ06] Yi Liu, Hongbin Zha, and Hong Qin. Shape topics: A compact representation and new algorithms for 3D partial shape retrieval. In *Proceedings of the 2006 IEEE Conference on Computer Vision and Pattern Recognition*, volume 2 of *CVPR '06*, pages 2025–2032, 2006.

[MAD+06] Athanasios Mademlis, Apostolos Axenopoulos, Petros Daras, Dimitrios Tzovaras, and Michael G. Strintzis. 3D Content-Based Search Based on 3D Krawtchouk Moments. In *Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission*, 3DPVT '06, pages 743–749, Washington, DC, USA, 2006. IEEE Computer Society.

[MB97] Walter Meyer-Bohe. *Grundrisse öffentlicher Gebäude. Synoptische Gebäudetypologie*. Ernst & Sohn, 1997.

[MD06] Ameesh Makadia and Kostas Daniilidis. Light field similarity for model retrieval. In *SHREC2006 3D Shape Retrieval Contest*, pages 32–35, 2006. Technical Report UU-CS-2006-030, Department of Information and Computing Sciences, Utrecht University.

[MGGP06] Niloy J. Mitra, Leonidas Guibas, Joachim Giesen, and Mark Pauly. Probabilistic fingerprints for shapes. In *Proceedings of the 2006 Symposium on Geometry Processing*, pages 121–130, 2006.

[Mit90] William J. Mitchell. *The Logic of Architecture: Design, Computation, and Cognition*. MIT Press, Cambridge, MA, USA, 1st edition, 1990.

[MKF04] Patrick Min, Michael Kazhdan, and Thomas Funkhouser. A Comparison of Text and Shape Matching for Retrieval of Online 3D Models. In *In Proceedings of the European Conference on Digital Libraries*, pages 209–220, 2004.

[MLVT10] Sébastien Macé, Hervé Locteau, Ernest Valveny, and Salvatore Tabbone. A system to detect rooms in architectural floor plan images. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, DAS '10, pages 167–174, New York, NY, USA, 2010. ACM.

[MPSR01] David McWherter, Mitchell Peabody, Ali C. Shokoufandeh, and William Regli. Database techniques for archival of solid models. In *Proceedings of the 2001 ACM symposium on Solid modeling and applications*, SAM '01, pages 78–87, New York, NY, USA, 2001. ACM Press.

[Mun57] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5:32–38, 1957.

[NDK05] Marcin Novotni, Patrick Degener, and Reinhard Klein. Correspondence Generation and Matching of 3D Shape Subparts. Technical Report CG-2005-2, Universität Bonn, June 2005.

[Neu05] Ernst Neufert. *Bauentwurfslehre*. Vieweg, 38th edition, September 2005.

[NK01] Marcin Novotni and Reinhard Klein. A Geometric Approach to 3D Object Comparison. In *International Conference on Shape Modeling and Applications*, pages 167–175, May 2001.

[Nov03] Marcin Novotni. 3D Zernike Descriptors for Content Based Shape Retrieval. In *Proceedings of the 8th ACM Symposium on Solid Modeling and Applications*, SAM '03, pages 216–225. ACM Press, 2003.

[NPWK05] Marcin Novotni, Gil-Joo Park, Raoul Wessel, and Reinhard Klein. Evaluation of kernel based methods for relevance feedback in 3D shape retrieval. In *Proceedings of the Fourth International Workshop on Content-Based Multimedia Indexing*, CBMI '05, June 2005.

[NRH$^+$77] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometric Considerations and Nomenclature for Reflectance. *National Bureau of Standards*, 1977.

[NSB$^+$12] Matthias Nieuwenhuisen, Jörg Stückler, Alexander Berner, Reinhard Klein, and Sven Behnke. Shape-primitive based object recognition and grasping. In *Proceedings of the 7th German Conference on Robotics (ROBOTIK)*, May 2012.

[NXS12] Liangliang Nan, Ke Xie, and Andrei Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics*, 31(6):137:1–137:10, November 2012.

[OA10] Bahadir Özdemir and Selim Aksoy. Image classification using subgraph histogram representation. In *Proceedings of the 20th International Conference on Pattern Recognition*, ICPR '10, pages 1112–1115, 2010.

[OB07] Björn Ommer and Joachim M. Buhmann. Learning the compositional nature of visual objects. In *Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '07, pages 1–8, June 2007.

[OB09] Björn Ommer and Joachim M. Buhmann. Learning the compositional nature of visual object categories for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:501–516, 2009.

[OBBG09] Maks Ovsjanikov, Alexander M. Bronstein, Michael M. Bronstein, and Leonidas J. Guibas. Shape Google: a computer vision approach to isometry invariant shape retrieval. In *Proceedings of the 12th International Conference on on Computer Vision Workshops (ICCV Workshops)*, pages 320–327, September 2009.

[OF08] Ryutarou Ohbuchi and Takahiko Furuya. Accelerating bag-of-features SIFT algorithm for 3D model retrieval. In *Proceedings of the SAMT Workshop on Semantic 3D Media*, pages 28–30, 2008.

[OFCD02] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Transactions on Graphics*, 21(4):807–832, October 2002.

[OMMG10] Maks Ovsjanikov, Quentin Mérigot, Facundo Mémoli, and Leonidas J. Guibas. One point isometric matching with the heat kernel. *Computer Graphics Forum*, 29(5):1555–1564, 2010.

[OMT03] Ryutarou Ohbuchi, Takahiro Minamitani, and Tsuyoshi Takei. Shape-Similarity Search of 3D Models by using Enhanced Shape Functions. In *Proceedings of the Theory and Practice of Computer Graphics 2003*, TPCG '03, pages 97–, Washington, DC, USA, 2003. IEEE Computer Society.

[OOFB08] Ryutarou Ohbuchi, Kunio Osada, Takahiko Furuya, and Tomohisa Banno. Salient local visual featuers for shape-based 3D model retrieval. In *Proceedings of the 2008 IEEE International Conference on Shape Modeling and Applications*, SMI '08, pages 93–102, 2008.

[OOIT02] Ryutarou Ohbuchi, Tomo Otagiri, Masatoshi Ibato, and Tsuyoshi Takei. Shape-similarity search of three-dimensional models using parameterized statistics. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, PG '02, pages 265–, Washington, DC, USA, 2002. IEEE Computer Society.

[Pet94] Toni Petersen. *Art & Architecture Thesaurus*. ACM Press, 1994.

[Pho75] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, June 1975.

[Pla99] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.

[Pow11] David M. W. Powers. Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2011.

[PR00] Eric Paquet and Marc Rioux. Nefertiti: a tool for 3-d shape databases management. *Image and Vision Computing*, 2000.

[PRM$^+$00] Eric Paquet, Marc Rioux, Anil M. Murching, Thumpudi Naveen, and Ali J. Tabatabai. Description of shape information for 2-d and 3-d objects. *Signal Processing: Image Communication*, pages 103–122, 2000.

[PSG$^+$06] Joshua Podolak, Philip Shilane, Aleksey Golovinskiy, Szymon Rusinkiewicz, and Thomas Funkhouser. A Planar-Reflective Symmetry Transform for 3D Shapes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 25(3), July 2006.

[RB09a] Kaspar Riesen and Horst Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27(7):950–959, 2009.

[RB09b] Kaspar Riesen and Horst Bunke. Graph classification by means of lipschitz embedding. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39:1472–1483, December 2009.

[RLF09] Raif M. Rustamov, Yaron Lipman, and Thomas Funkhouser. Interior distance using barycentric coordinates. In *Proceedings of the Symposium on Geometry Processing*, SGP '09, pages 1279–1288, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association.

[Roo65] Daniel Roos. An integrated computer system for engineering problem solving. In *Proceedings of the November 30–December 1, 1965, fall joint computer conference, part I*, AFIPS '65 (Fall, part I), pages 423–433, New York, NY, USA, 1965. ACM.

[Roy68] Royal Institute of British Architects. *Construction indexing manual incorporating the authoritative United Kingdom version of the international SfB classification system and superseding the RIBA SfB/UDC Building Filing Manual 1961.* Royal Institute of British Architects, London, 1968.

[RS99] Volker Roth and Volker Steinhage. Nonlinear discriminant analysis using kernel functions. In *Advances in Neural Information Processing Systems*, pages 568–574. MIT Press, 1999.

[RT01] Volker Roth and Koji Tsuda. Pairwise coupling for machine recognition of hand-printed japanese characters. In *Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '01, pages 1120–1125, 2001.

[RTG98] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, pages 59–, Washington, DC, USA, 1998. IEEE Computer Society.

[Rus07a] Raif M. Rustamov. Augmented symmetry transforms. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications 2007*, SMI '07, pages 13–20, Washington, DC, USA, 2007. IEEE Computer Society.

[Rus07b] Raif M. Rustamov. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, SGP '07, pages 225–233, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[Rus10] Raif M. Rustamov. Robust volumetric shape descriptor. In *3DOR*, pages 1–5, 2010.

[RVRB10] Jonas Richiardi, Dimitri Van De Ville, Kaspar Riesen, and Horst Bunke. Vector space embedding of undirected graphs with fixed-cardinality vertex sequences for classification. In *Proceedings of the*

20th International Conference on Pattern Recognition, ICPR '10, pages 902–905, Los Alamitos, USA, 2010.

[RWP06] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. Laplace-beltrami spectra as "shape-dna" of surfaces and solids. *Computer-Aided Design*, 38(4):342–366, 2006.

[RWSN09] Martin Reuter, Franz-Erich Wolter, Martha Shenton, and Marc Niethammer. Laplace-beltrami eigenvalues and topological features of eigenfunctions for statistical shape analysis. *Computer-Aided Design*, 41(10):739–755, 2009.

[SBBF12] Michela Spagnuolo, Michael M. Bronstein, Alexander M. Bronstein, and Alfredo Ferreira, editors. *Eurographics Workshop on 3D Object Retrieval 2012, Cagliari, Italy, May 13, 2012. Proceedings*. Eurographics Association, 2012.

[Sch04] Friederike Schneider. *Floor Plan Manual Housing*. Birkhäuser, third edition, April 2004.

[SEL04] Rafael Sacks, Charles M. Eastman, and Ghang Lee. Parametric 3D modeling in building construction with examples from precast concrete. *Automation in Construction*, 13(3):291 – 312, 2004.

[SF83] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:353–362, 1983.

[SF06] Philip Shilane and Thomas Funkhouser. Selecting Distinctive 3D Shape Descriptors for Similarity Retrieval. In *Proceedings of the 2006 IEEE International Conference on Shape Modeling and Applications*, SMI '06, June 2006. Article No. 18.

[SF07] Philip Shilane and Thomas Funkhouser. Distinctive Regions of 3D Surfaces. *ACM Transactions on Graphics*, 26(2), June 2007. Article No. 7.

[SG02] Jeong-Jun Song and Forouzan Golshani. 3D Object Retrieval by Shape Similarity. In *Proceedings of the 13th International Conference on Database and Expert Systems Applications*, DEXA '02, pages 851–860, London, UK, UK, 2002. Springer-Verlag.

[SH95] David Slater and Glenn Healey. Combining color and geometric information for the illumination invariant recognition of 3-d objects.

In *Proceedings of the Fifth International Conference on Computer Vision*, ICCV '95, pages 563–, Washington, DC, USA, 1995. IEEE Computer Society.

[Shi08] Philip Shilane. *Shape Distinction for 3D Object Retrieval.* PhD thesis, Princeton University, April 2008.

[SKO00] Motofumi T. Suzuki, Toshikazu Kato, and Nobuyuki Otsu. A similarity retrieval of 3D polygonal models using rotation invariant shape descriptors. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC) 2000*, volume 4, pages 2946–2952, 2000.

[SMKF04] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The Princeton shape benchmark. In *Proceedings of the 2004 IEEE International Conference on Shape Modeling and Applications*, SMI '04, pages 167–176, June 2004.

[SMS$^+$04] Ying Shan, Bogdan Matei, Harpreet S. Sawhney, Rakesh Kumar, Daniel Huber, and Martial Hebert. Linear Model Hashing and Batch RANSAC for Rapid and Accurate Object Recognition. In *Proceedings of the 2004 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '04, pages 121–128, 2004.

[SS01] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA, 2001.

[SSGD03] Hari Sundar, Deborah Silver, Nikhil Gagvani, and Sven J. Dickinson. Skeleton based shape matching and retrieval. In *Proceedings of the 2003 IEEE International Conference on Shape Modeling and Applications*, SMI '03, pages 130–142, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[SSSCO08] Shy Shalom, Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. Part analogies in sets of objects. In *Proceedings of the 2008 EUROGRAPHICS Workshop on 3D Object Retrieval*, pages 33–40. EUROGRAPHICS Association, 2008.

[Sul96] Louis Sullivan. The tall office building artistically considered. *Lippincott's Magazine*, 57:403–409, March 1896.

[Sut64] Ivan E. Sutherland. Sketch pad a man-machine graphical communication system. In *Proceedings of the SHARE design automation workshop*, DAC '64, pages 6.329–6.346, New York, NY, USA, 1964. ACM.

[SV01] Dietmar Saupe and Dejan V. Vranic. 3D Model Retrieval with Spherical Harmonics and Moments. In *Proceedings of the 23rd DAGM-Symposium on Pattern Recognition*, pages 392–397, London, UK, UK, 2001. Springer-Verlag.

[SWK07] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, June 2007.

[SWWK08] Ruwen Schnabel, Raoul Wessel, Roland Wahl, and Reinhard Klein. Shape recognition in 3D point-clouds. In V. Skala, editor, *The 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, WSCG '08. UNION Agency-Science Press, February 2008.

[SZL92] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '92, pages 65–70, New York, NY, USA, 1992. ACM.

[SZM$^+$08] Kaleem Siddiqi, Juan Zhang, Diego Macrini, Ali Shokoufandeh, Sylvain Bouix, and Sven Dickinson. Retrieving articulated 3-d models using medial surfaces. *Machine Vision and Applications*, 19(4):261–275, May 2008.

[TS04] Tony Tung and Francis Schmitt. Augmented Reeb Graphs for Content-Based Retrieval of 3D Mesh Models. In *SMI*, pages 157–166, 2004.

[TV08] Johan W. H. Tangelder and Remco C. Veltkamp. A survey of content based 3D shape retrieval methods. *Multimedia Tools and Applications*, 39(3):441–471, 2008.

[TvBDK00] David M. J. Tax, Martijn van Breukelen, Robert P. W. Duin, and Josef Kittler. Combining multiple classifiers by averaging or by multiplying? *Pattern Recognition*, 33(9):1475–1485, September 2000.

[TVD09] Julien Tierny, Jean-Philippe Vandeborre, and Mohamed Daoudi. Partial 3D Shape Retrieval by Reeb Pattern Unfolding. *Computer Graphics Forum*, 9999(9999), 2009.

[Ued00] Naonori Ueda. Optimal linear combination of neural networks for improving classification performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(2):207–215, 2000.

[Vap98] Vladimir N. Vapnik. *Statistical learning theory*. Wiley, 1 edition, September 1998.

[VKH06] Vivek Verma, Rakesh Kumar, and Stephen Hsu. 3D building detection and modeling from aerial LIDAR data. In *Proceedings of the 2006 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '04, pages 2213–2220, Washington, DC, USA, 2006. IEEE Computer Society.

[Vra04] Dejan V. Vranic. *3D Model Retrieval*. PhD thesis, University of Leipzig, 2004.

[VRS$^+$06] Remco C. Veltkamp, Remco Ruijsenaars, Michela Spagnuolo, Roelof van Zwol, and Frank B. ter Haar. SHREC 2006 - shape retrieval contest. Technical Report UU-CS-2006-030, Department of Information and Computing Sciences, June 2006.

[VS01] Dejan V. Vranic and Dietmar Saupe. 3D Shape Descriptor Based on 3D Fourier Transform. In *Proceedings of the EURASIP Conference on Digital Signal Processing for Multimedia Communications and Services (ECMCS 2001) (editor K. Fazekas)*, pages 271–274, Budapest, Hungary, September 2001.

[VtH07] Remco C. Veltkamp and Frank B. ter Haar. SHREC 2007 - shape retrieval contest. Technical Report UU-CS-2007-015, Department of Information and Computing Sciences, June 2007.

[VtH08] Remco C. Veltkamp and Frank B. ter Haar. SHape REtrieval Contest (SHREC) 2008. In *Proceedings of the 2008 IEEE International Conference on Shape Modeling and Applications*, SMI '08, pages 215–216, 2008.

[VtH09] Remco C. Veltkamp and Frank B. ter Haar. Shrec 2009 - shape retrieval contest. In *3DOR*, pages 57–59, 2009.

[VZ09] Manik Varma and Andrew Zisserman. A statistical approach to material classification using image patch exemplars. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):2032–2047, November 2009.

[Wah99] Grace Wahba. Support vector machines, reproducing kernel hilbert spaces, and randomized gacv. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in kernel methods*, pages 69–88. MIT Press, Cambridge, MA, USA, 1999.

[WBK08a] Raoul Wessel, Rafael Baranowski, and Reinhard Klein. Learning distinctive local object characteristics for 3D shape retrieval. In *Proceedings of the 2008 International Workshop Vision, Modeling and Visualization*, VMV '08, pages 167–178. Akademische Verlagsgesellschaft Aka GmbH, Heidelberg, October 2008.

[WBK08b] Raoul Wessel, Ina Blümel, and Reinhard Klein. The room connectivity graph: Shape retrieval in the architectural domain. In V. Skala, editor, *The 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, WSCG '08. UNION Agency-Science Press, February 2008.

[WBK09] Raoul Wessel, Ina Blümel, and Reinhard Klein. A 3D shape benchmark for retrieval and automatic classification of architectural data. In *Proceedings of the 2009 EUROGRAPHICS Workshop on 3D Object Retrieval*, pages 53–56, March 2009.

[WCI04] Hau-San Wong, Kent K. T. Cheung, and Horace H. S. Ip. 3D head model classification by evolutionary optimization of the Extended Gaussian Image representation. *Pattern Recognition*, 37(12):2307–2322, December 2004.

[WK05] Jianhua Wu and Leif Kobbelt. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum*, 24(3):277–284, 2005.

[WK10] Raoul Wessel and Reinhard Klein. Learning the Compositional Structure of Man-Made Objects for 3D Shape Retrieval. In *EUROGRAPHICS 2010 Workshop on 3D Object Retrieval*, pages 39–46, May 2010.

[WLHZ10] Bin Wang, Dong Li, Kaimo Hu, and Hui Zhang. Shape similarity assessment approach for cad models based on graph edit distance. In

Short Paper Proceedings of the 2010 EUROGRAPHICS Workshop on 3D Object Retrieval, pages 13–16, May 2010.

[WLRB$^+$10] Markus Weber, Christoph Langenhan, Thomas Roth-Berghofer, Marcus Liwicki, Andreas Dengel, and Frank Petzold. a.scatch: Semantic structure for architectural floor plan retrieval. In *Proceedings of the 19th International Conference on Case-based Reasoning*, IC-CBR '10, pages 510–524. Springer Verlag, Heidelberg, 7 2010.

[WNK06] Raoul Wessel, Marcin Novotni, and Reinhard Klein. Correspondences between Salient Points on 3D Shapes. In L. Kobbelt, T. Kuhlen, T. Aach, and R. Westermann, editors, *Vision, Modeling, and Visualization 2006 (VMV 2006)*, pages 365–372. Akademische Verlagsgesellschaft Aka GmbH, Berlin, November 2006.

[WOV$^+$11a] Raoul Wessel, Sebastian Ochmann, Richard Vock, Ina Blümel, and Reinhard Klein. Efficient Retrieval of 3D Building Models Using Embeddings of Attributed Subgraphs. In *the 20th ACM Conference on Information and Knowledge Management (CIKM 2011) : Posters*, October 2011.

[WOV$^+$11b] Raoul Wessel, Sebastian Ochmann, Richard Vock, Ina Blümel, and Reinhard Klein. Efficient Retrieval of 3D Building Models Using Embeddings of Attributed Subgraphs. Technical Report CG-2011-2, University of Bonn, August 2011.

[WZC94] Jason T.L. Wang, Kaizhong Zhang, and Gung-Wei Chirn. The approximate graph matching problem. In *Proceedings of the 4th International Conference on Pattern Recognition*, ICPR '94, pages B:284–288, 1994.

[YP97] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[YSSK10] Sang Min Yoon, Maximilian Scherer, Tobias Schreck, and Arjan Kuijper. Sketch-based 3D model retrieval using diffusion tensor fields of suggestive contours. In *Proceedings of the international conference on Multimedia*, MM '10, pages 193–200, New York, NY, USA, 2010. ACM.

[YWC12] Tsz-Ho Yu, Oliver J. Woodford, and Roberto Cipolla. A Performance Evaluation of Volumetric 3D Interest Point Detectors. *International Journal of Computer Vision*, pages 1–18, September 2012.

[ZC01] Cha Zhang and Tsuhan Chen. Efficient feature extraction for 2D/3D objects in mesh representation. In *Proceedings of the 2001 Conference on Image Processing*, volume 3 of *ICIP 2001*, pages 935–938, October 2001.

[ZHDQ08] Guangyu Zou, Jing Hua, Ming Dong, and Hong Qin. Surface matching with salient keypoints in geodesic scale space. *Computer Animation and Virtual Worlds*, 19(3-4):399–410, 2008.

[ZP01] Titus B. Zaharia and Françoise J. Preteux. 3D shape-based retrieval within the MPEG-7 framework. In *Nonlinear Image Processing and Pattern Analysis*, volume 4304, pages 133–145, January 2001.

[ZP02] Titus B. Zaharia and Françoise J. Prêteux. Shape-based retrieval of 3D mesh models. In *ICME (1)*, pages 437–440. IEEE, 2002.

[ZSMS05] Juan Zhang, Kaleem Siddiqi, Diego Macrini, and Ali Shokouf. Retrieving articulated 3-D models using medial surfaces and their graph spectra. In *Proceedings of the 2005 International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 285–300, 2005.

[ZTS02] Emanuel Zuckerberger, Ayellet Tal, and Shymon Shlafman. Polyhedral surface decomposition with applications. *Computers and Graphics*, 26(5):733–743, October 2002.

[ZTW$^+$09] Zhiping Zeng, Anthony K. H. Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou. Comparing stars: on approximating graph edit distance. *Proceedings of the VLDB Endowment*, 2:25–36, August 2009.