Master Thesis

Content Based Retrieval of 3D Models using Generative 3D Models

Harald Grabner, BSc

Institute of Computer Graphics
and Knowledge Visualization

Graz University of Technology

September 2013

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

date	(signature)

Abstract

Improvements in techniques for digitizing, modeling and visualizing 3D shapes have led to an enormous amount of 3D models available on the internet. To explore such large databases, efficient retrieval methods are needed in order to obtain relevant results with respect to a given search query.

In this thesis a novel approach to retrieve 3D models, based on generative modeling techniques, is presented. The generative models span a shape space, of which a number of training samples is taken at random. The randomly generated training samples are used to train retrieval methods. In this way, no "real" training data is needed a priori and no additional meta data must be provided for the 3D models to be retrieved.

To train the retrieval methods, feature vectors are calculated for the randomly generated training samples. For this purpose, the training samples are voxelized, aligned and transformed using inverse distance transformation. The transformed training samples are split into a grid of regular cells. For each cell the histogram of inverted distances is calculated, which serves as the cells feature vector.

Two 3D model retrieval methods are presented. The first method learns a non-parametric density function for each cell, the second method uses a one-class support vector machine to estimate the distribution of training samples in the feature space. Furthermore, two approaches to improve retrieval effectiveness are presented. The first approach aims to improve retrieval results by calculating the joint probability on a Markov random field. The second approach uses diffusion processes to take the underlying structure of the data manifold into account. The effectiveness of the methods is demonstrated by testing the methods against Princeton shape benchmark using standard quality measures.

It is shown that it is possible to train retrieval methods using solely generative models. Furthermore it is shown, that the histogram of inverted distances can be used as a feature vector for spatial data.

Zusammenfassung

Methoden zur Digitalisierung, Modellierung und Visualisierung von 3D Objekten wurden in den letzten Jahren laufend verbessert. Die Verwendung dieser Methoden hat zu einer immensen Anzahl von 3D Modellen im Internet geführt. Um Datenbestände dieser Größenordnung zu durchsuchen werden Methoden benötigt, die auf Basis einer Suchanfrage relevante Resultate zurückliefern. Diese Methoden werden Retrieval-Methoden genannt.

In dieser Arbeit wird eine neuartige Retrieval-Methode für 3D Modelle präsentiert. Die Methode basiert auf Techniken der generativen Modellierung. Generative Modelle werden in dieser Arbeit benutzt um einen Raum von 3D Modellen aufzuspannen. Aus diesem Raum werden zufällig 3D Modelle ausgewählt, welche später zum Trainieren der Retrieval-Methoden benutzt werden. Auf diese Weise werden keine "echten" Trainingsdaten im Vorhinein benötigt und die zu suchenden Modelle müssen nicht mit Metadaten versorgt werden.

Um die Retrieval-Methoden zu trainieren werden Merkmalsvektoren für die zufällig ausgewählten 3D Modelle berechnet. Dazu werden die Modelle in das Voxel-Format konvertiert und anschließend ausgerichtet. Auf den ausgerichteten Modellen wird die inverse Distanztransformation berechnet. Die transformierten Modelle werden danach in ein regelmäßiges Gitter, bestehend aus Zellen, aufgeteilt. Für jede Zelle wird das Histogramm der inversen Distanzen berechnet. Dieses Histogramm stellt den Merkmalsvektor für die jeweilige Zelle dar.

In dieser Arbeit werden zwei Retrieval-Methoden präsentiert. In der ersten Methode wird pro Zelle eine parameterfreie Dichtefunktion gelernt und in der zweiten Methode wird eine Support Vector Machine benutzt um die Verteilung der Trainingsmodelle im Featureraum zu schätzen. Des Weiteren werden zwei Ansätze um die Retrieval-Ergebnisse zu verbessern vorgestellt. Im ersten Ansatz werden Markov-Netzwerke benutzt und im zweiten Ansatz werden Diffusionsprozesse benutzt. Die Effektivität der Methoden wird durch den Vergleich mit dem "Princeton Shape Benchmark" gezeigt. Dazu werden klassische Qualitätsmetriken verwendet.

In dieser Arbeit wird gezeigt, dass es möglich ist Retrieval-Methoden ausschließlich mit generativen Modellen zu trainieren. Des Weiteren wird gezeigt, dass das Histogramm der inversen Distanzen als Merkmalsvektor für 3D Modele verwendet werden kann.

Acknowledgements

This project would not have been possible without the support of many people. I would like to thank my supervisor Dr. Torsten Ullrich who supported and guided me throughout the project. Furthermore, I would like to thank all members of the Computer Graphics and Knowledge Visualization institute and especially my labmates Christian and Martin. I would also like to thank my family and friends for all the support they have provided. Special thanks goes to Patrick Min for making his tools "meshconv" and "binvox" available to the public and to Michael Donoser for publishing his diffusion code.

Contents

List of Figures

Chapter 1

Introduction

Improvements in techniques for digitizing 3D objects, modeling and visualization of 3D models have led to an enormous amount of 3D models available on the internet. Websites like "`archive3d.net`" host over 35.000 publicly available 3D models. Furthermore domain specific 3D part databases like "CADENAS Part Solutions" contain over 50.000 3D standard parts like screws, nuts or bolts. Another example of a large 3D model database is the RCSB protein database (`www.rcsb.org`). It hosts over 50.000 different proteins and their 3D representation. To explore such large databases, efficient retrieval methods are needed in order to obtain relevant results with respect to a given search query.

Retrieval of relevant 3D models is for instance needed when arranging 3D scenes of individual 3D models. A designer of a 3D scene could search for all chairs and tables in a 3D model database to arrange the interior of a particular room. Retrieval methods can also be used to facilitate reuse of 3D models. In mechanical engineering reusing and adapting parts of former projects can help to save design time. Therefore, a mechanical engineer designing a car engine could search for all pistons that fit the needs and adapt the best one, instead of designing the piston from ground up. Other application domains of 3D model retrieval include molecular biology, medicine or virtual reality.

The retrieval process can either perform on additionally attached *meta data*, or on the documents content. Meta data means "data about data". In the context of 3D models meta data could, for instance, be the type of object or a textual description of the object. Retrieval based on text-based methods is a well researched topic and text-based search functions can be found in many applications, such as websites or text processing software. However there are two disadvantages, when using text-based methods. First, meta data needs to be added and maintained manually and second, meta data might be inaccurate or subjective [1].

To overcome these problems content-based methods can be used. Content-based methods aim to find similar 3D models based on the content of a given query model. Those methods are necessary if no additional meta data are present.

An example of a meta data-based retrieval approach is illustrated in Figure 1.1. Based on tags added to each 3D model, models with the same tags can be retrieved. An example of a content-based retrieval approach is illustrated in Figure 1.2. Based on a set of 2D views, which are drawn by the user, similar models are retrieved. This method does not depend on any additional attached meta data.

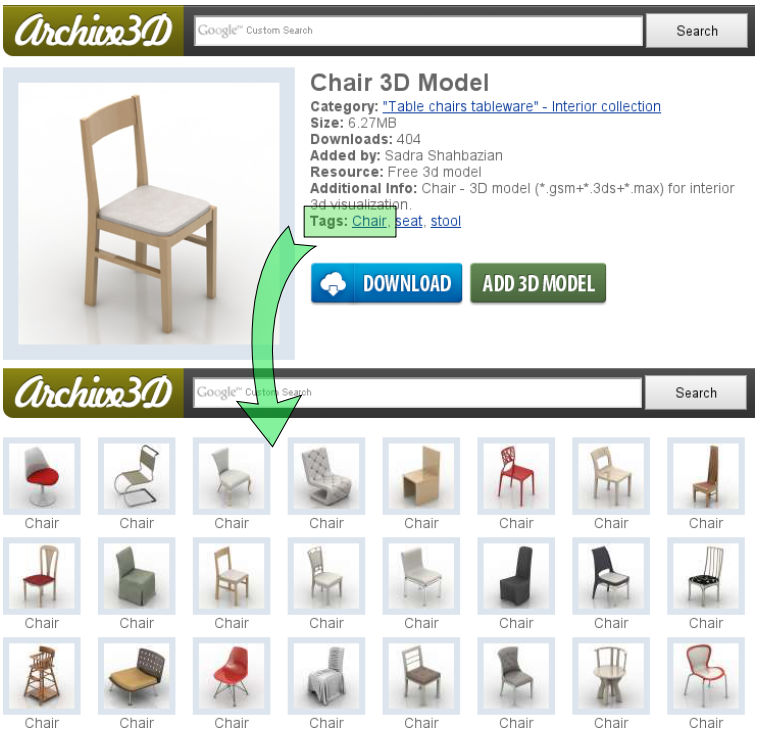

FIGURE 1.1: An example of a meta data-based retrieval approach. Based on tags added to a 3D model, models with the same tags can be retrieved (`archive3D.net`).

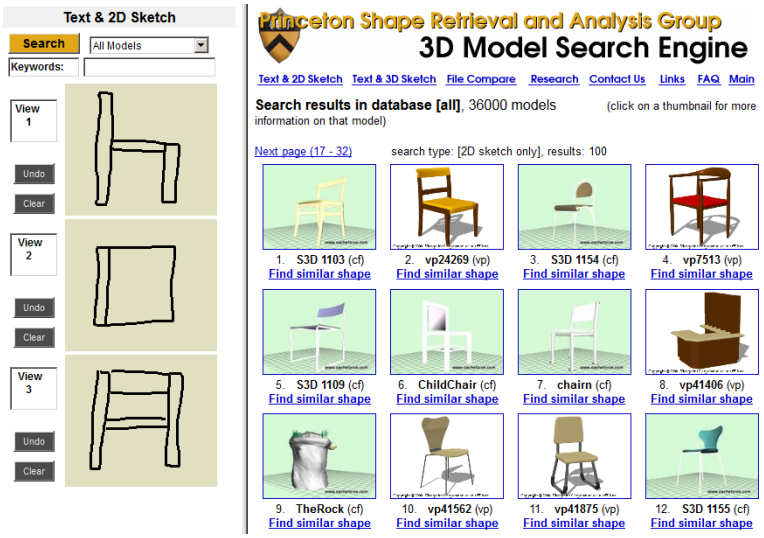

FIGURE 1.2: An example of a content-based retrieval approach. Based on a set of 2D views, which are drawn by the user, similar models are retrieved. This method does not depend on any additional attached meta data (`shape.cs.princeton.edu`).

In contrast to 3D models, text documents can be retrieved easily using content-based methods. For a collection of text documents an index of all words found in the documents can be built. Documents can simply be retrieved by returning all documents associated with a given word in the index. Unfortunately, 3D documents are not retrieved easily, because no such things as "words" are explicitly represented in the 3D model. To retrieve relevant 3D documents the similarity between two 3D models must be calculated.

In this thesis a novel approach to retrieve 3D models similar to *generative models* is presented. A generative 3D model, is an algorithm that takes a number of parameters and produces 3D geometry. By varying the values of the parameters and combining generative models, complex scenes can be modeled based on a set of formal construction rules. Figure 1.3 shows a generative model of a Greek temple with different parameter values. The parameters are number, height and distance of the temples pillars.

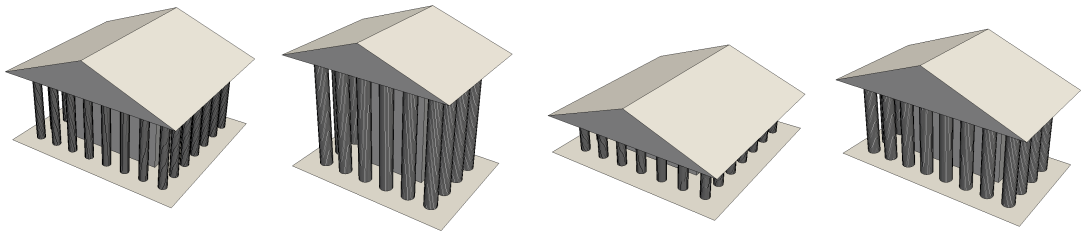

FIGURE 1.3: A generative model of a Greek temple. By varying the model parameters (number, height and distance of pillars), different 3D models are generated.

In this thesis generative models are used to describe 3D model classes. The generative models span a shape space, of which a number of training samples is taken at random. The randomly generated training samples are used to train the retrieval methods. In this way, no "real" training data is needed a priori. Retrieval is performed using the generative models as a query. This way no relevant models from the test set must be known to formulate a query, and no additional meta data must be provided for the 3D models to be retrieved.

The thesis is structured as follows: Chapter 2 presents the basics of information retrieval, probability and statistics and shape representations. Chapter 3 presents relevant related work in the field of content-based 3D model retrieval. In Chapter 4 two methods for retrieving 3D models similar to given generative models are presented. Furthermore two approaches to improve retrieval quality are presented. In Chapter 5 the retrieval methods are evaluated using standard benchmarks and quality measures. The thesis is concluded in Chapter 6.

Chapter 2

Background

2.1 Information Retrieval

Information retrieval is the process of finding documents that satisfy given search conditions within a collection of documents [2]. Well known examples of information retrieval systems are web search engines such as `www.google.com` or `www.yahoo.com`.

2.1.1 Structure of 3D Model Retrieval Systems

In the context of 3D model retrieval, the documents to be found are 3D models. The retrieval process can either perform on attached meta data, such as keywords or user comments, or on the document's content. This thesis will only deal with retrieval methods that operate solely on the document's content, so called content-based methods. Such methods are necessary if no additional meta data are present. Furthermore, meta data based retrieval systems can be improved by additionally incorporating content based methods [3, 4].

The overall retrieval process can be illustrated using the conceptual framework for shape retrieval by Tangelder et al. [5], which is depicted in Figure 2.1. As the first step in the framework a descriptor for all models from a 3D model database is extracted. The descriptor captures the models properties and features and is is later used for matching, by calculating a similarity measure on a pair of descriptors. To speed up the retrieval process an index can be built, based on the extracted descriptors. Well known indexing structures for special types of descriptors are for instance R-trees, X-trees or k-d trees. An overview of indexing structures for multimedia retrieval is given in the survey of Böhm and Berchtold [6]. Both tasks, the descriptor extraction as well as the creation of an index, can be done offline. Different types of descriptors and matching methods are covered in the related work chapter in Section 3.1.

After the generation of an index, users of 3D model retrieval systems can query the database using three different approaches:

Query by Example
The user of the retrieval system provides an existing 3D model. Based on the submitted 3D model the query descriptor is extracted and similar models are retrieved by comparing the query descriptor against the index.

Direct Querying
The user directly provides the query description of the wanted 3D model. This approach is only feasible under the condition, that the descriptor is provided in a human readable and editable form.

Browsing
The user can also browse through the database.

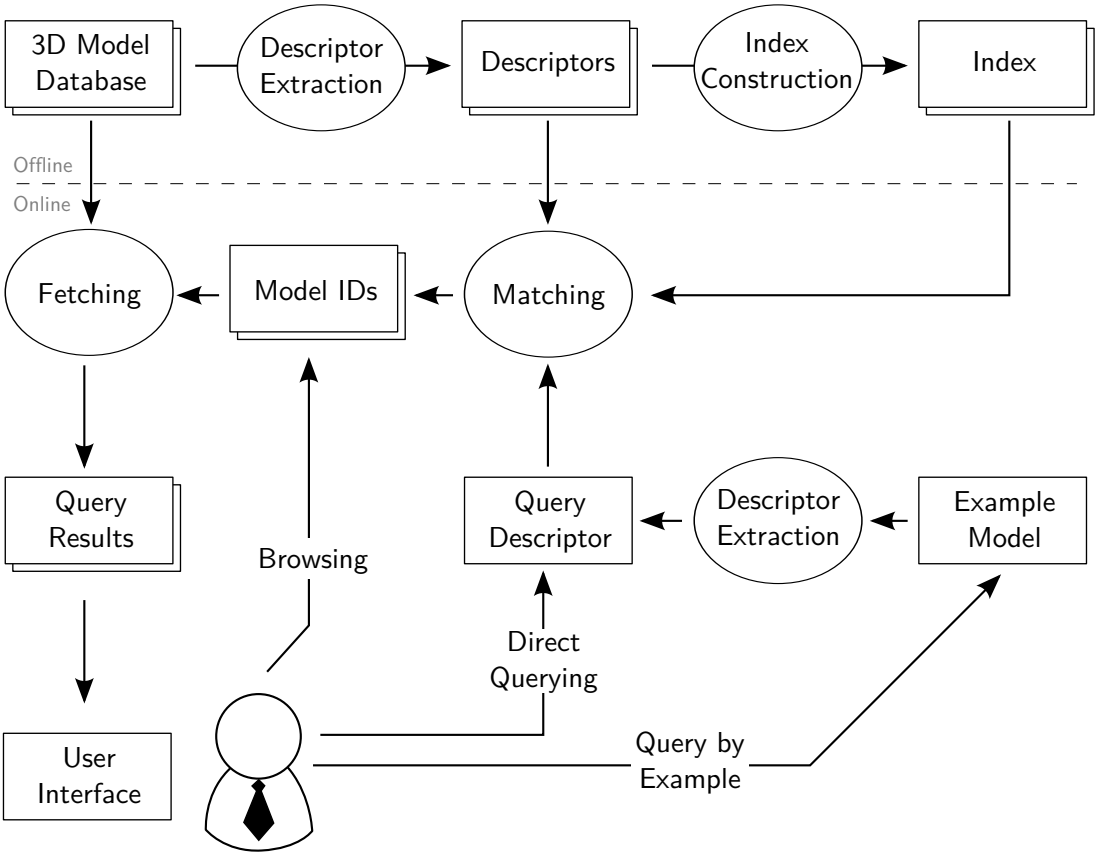

FIGURE 2.1: A conceptual framework for 3D content based retrieval systems. For all models in a 3D model database descriptors are calculated and an index is built based on the descriptors. A user can either search for 3D models by submitting an example model, by submitting a query descriptor or by simply browsing through the models. (Modified from [5])

2.1.2 Evaluation of Information Retrieval Methods

The effectiveness of information retrieval methods is evaluated by testing against standardized benchmarks. Those benchmarks consist of a collection of 3D models and gold standard results for each query [2]. For every possible query the gold standard marks each 3D model as either relevant or non-relevant, like depicted in Figure 2.2.

In the field of content based 3D model retrieval standardized benchmarks are for instance the Princeton shape benchmark [7] or the Purdue CAD shape benchmark [8]. A famous contest in the field of content based 3D model retrieval is the annual SHREC shape retrieval contest which is part of the Eurographics Workshop on 3D Object Retrieval [9].

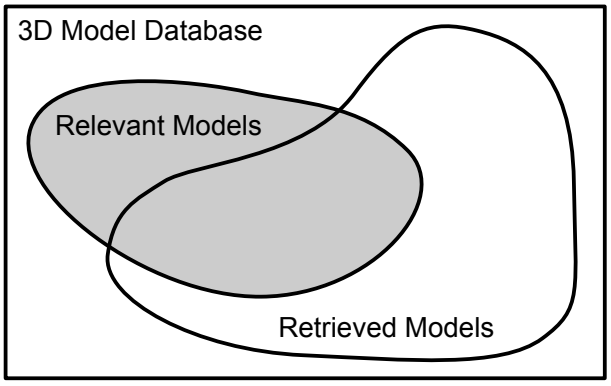

FIGURE 2.2: From a 3D model database a subset is retrieved by the retrieval method. Relevant models that have been retrieved are called true positives, whereas retrieved models that are not relevant are called false positives.

Combined with actual retrieval results, each retrieved model can be classified into one of four different sets: *true positives*, *false positives*, *true negatives* and *false negatives*.

Based on the retrieval results and the gold standard, the measures *precision* and *recall* can be calculated. Precision is defined as the fraction of retrieved models that are relevant and recall is defined as the fraction of relevant documents that have been retrieved [2]. The formulas for both measures are depicted in Equations 2.1 and 2.2.

$$\text{Precision} = \frac{\mid \text{relevant models retrieved} \mid}{\mid \text{retrieved models} \mid} = \frac{\mid \text{true positives} \mid}{\mid (\text{true positives}) \cup (\text{false positives}) \mid} \tag{2.1}$$

$$\text{Recall} = \frac{\mid \text{relevant models retrieved} \mid}{\mid \text{relevant models} \mid} = \frac{\mid \text{true positives} \mid}{\mid (\text{true positives}) \cup (\text{false negatives}) \mid} \tag{2.2}$$

If the retrieval process ranks the results according to their match, the precision and recall values can be calculated on the top k-Results, with increasing k. Those values are denoted Precision(k) and Recall(k) respectively. The precision and recall values can be plotted by displaying the precision on the abscissa and recall on the ordinate. This plot is called *precision-recall curve*.

An exemplary precision-recall curve is depicted as a solid line in Figure 2.3. Usually for small values of k the precision is high. At least, when k is greater than the number of relevant models, the precision decreases. The gold standard, would produce a horizontal line at precision = 1, which drops when k is greater than the number of relevant models.

The precision-recall graph exhibits a characteristic saw tooth shape. The spikes in the graph are often removed by calculating the interpolated precision [2]. The formula for the interpolated precision is given in Equation 2.3.

$$\text{Precision}_{\text{interp}}(k) = \max_{k' \geq k} \text{Precision}(k') \tag{2.3}$$

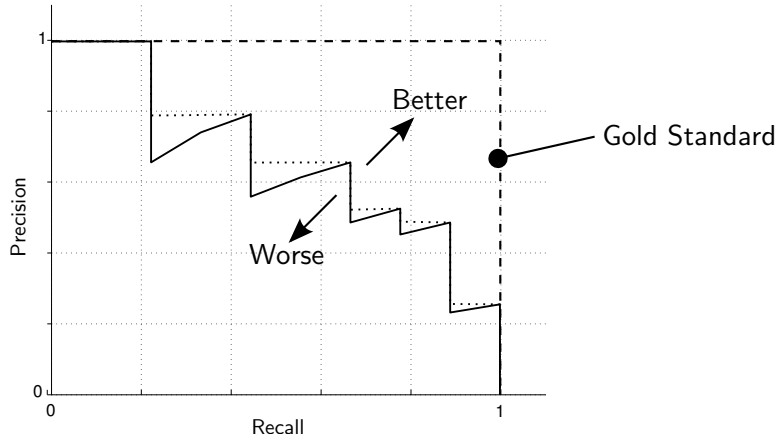

FIGURE 2.3: An exemplary precision-recall curve of a retrieval method. The gold standard would produce a horizontal line, which drops at recall = 1.

The justification for the interpolated precision is the assumption that users of a retrieval system would look at a few more documents, if the percentage of relevant documents would increase [2]. The interpolated precision-recall graph is depicted as a dotted line in Figure 2.3.

The precision-recall graph only illustrates the results of a single search query. To evaluate the results for a whole test collection, the averaged eleven point precision-recall graph can be used. Therefore, the precision is measured at eleven different recall levels of $(0, 0.1, 0.2, \ldots 1.0)$ for each query. The recall values are averaged over all queries and presented in a single graph [2]. An exemplary averaged eleven point precision-recall graph is depicted in Figure 2.4.

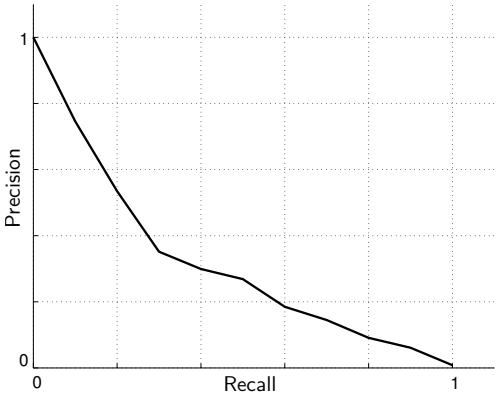

FIGURE 2.4: An exemplary averaged eleven point precision-recall graph. Using the averaged eleven point precision-recall graph, results for a whole test collection can be evaluated.

2.2 Probability and Statistics

In this Section, basics of probability and statistics are denoted. The notation is based on "Introduction to Probability" by Hoel [10] and "Pattern Recognition and Machine Learning" by Bishop [11].

2.2.1 Probability Spaces and Random Variables

A probability space is a triple $(\Omega, \mathscr{A}, \mathbf{P})$ consisting of a nonempty set of all possible outcomes of a random experiment Ω, a set of measurable events $\mathscr{A}$ and a probability measure $\mathbf{P}$. The set of all measurable events $\mathscr{A}$ consists of subsets of Ω and must adhere to the following properties:

- $(A \in \mathscr{A}) \Rightarrow (A^c \in \mathscr{A})$.
- $\big((A \in \mathscr{A}) \wedge (B \in \mathscr{A})\big) \Rightarrow \big((A \cup B) \in \mathscr{A} \wedge (A \cap B) \in \mathscr{A}\big)$.

The first property states, that the set $\mathscr{A}$ is closed under complementation, where A^c denotes the complement of A. The second property states that the set $\mathscr{A}$ is closed under union and intersection.

The probability measure $\mathbf{P}$ maps from elements in $\mathscr{A}$ to $\mathbb{R}$. The function $\mathbf{P}$ must adhere to the following properties:

- $\mathbf{P}(A) \geq 0 : \forall A \in \mathscr{A}$.
- $\mathbf{P}(\Omega) = 1$.
- $\big((A \in \mathscr{A}) \wedge (B \in \mathscr{A}) \wedge (A \cap B = \varnothing)\big) \Rightarrow \big(\mathbf{P}(A \cup B) = \mathbf{P}(A) + \mathbf{P}(B)\big)$.

The first property assures non-negativity of the probability measure. The second property states that the probability of the certain event is 1. The last property states that the probability of the union of two disjoint sets equals the sum of their probabilities.

On a probability space random variables are defined. Random variables denote the outcome of a random experiment as a numerical quantity.

A *discrete* real-valued random variable on a probability space $(\Omega, \mathscr{A}, \mathbf{P})$ is a function X, that maps from the set of all possible outcomes Ω to a finite or countably infinite set $\{x_1, x_2, \dots\} \subset \mathbb{R}$, such that $\{\omega \in \Omega : X(\omega) = x_i\}$ is a measurable event for all i. The probability for the event $\{\omega \in \Omega : X(\omega) = x_i\}$ is denoted as $\mathbf{P}(X = x_i)$. The function $\mathbf{f}$, defined by $\mathbf{f}(x) = \mathbf{P}(X = x_i)$, is called the discrete density function of X.

A *continuous* real-valued random variable on a probability space $(\Omega, \mathscr{A}, \mathbf{P})$ is a function X, that maps from the set of all possible outcomes Ω to $\mathbb{R}$, such that $\forall x \in (-\infty, \infty) : \{\omega \,|X(\omega) \leq x\}$ is an event and $\forall x \in (-\infty, \infty) : \mathbf{P}(X = x) = 0$. The function $\mathbf{F}$, defined by $\mathbf{F}(x) = \mathbf{P}(X \leq x)$, is called the distribution function of the random variable X. In practice distribution functions are usually defined using probability density functions.

A probability density function $\mathbf{f}$ is a non-negative function such that $\int_{-\infty}^{\infty} \mathbf{f}(x)\,\mathrm{d}x = 1$. The distribution function $\mathbf{F}$ defined by the probability density function $\mathbf{f}$ is given by

$$\mathbf{F}(x) = \int_{-\infty}^{x} \mathbf{f}(y)\,\mathrm{d}y. \tag{2.4}$$

The average of a random variable X weighted with its density function $\mathbf{f}$ is called the *expectation* of X, denoted $\mathbf{E}(X)$. For a given discrete random variable with a finite number of values n, its expectation is defined in Equation 2.5 and the expectation of continuous random variables is defined in Equation 2.6.

$$\mathbf{E}(X) = \sum_{i=1}^{n} x_i \mathbf{f}(x_i) \tag{2.5}$$

$$\mathbf{E}(X) = \int_{-\infty}^{\infty} x\mathbf{f}(x)\,\mathrm{d}x. \tag{2.6}$$

The variance of a random variable provides a measure of how much the random variable X spreads around its expectation is defined in Equation 2.7 and for two given random variables X and Y, the covariance measures how they vary together. The definition of covariance is given in Equation 2.8.

$$\mathbf{Var}(X) = \mathbf{E}\big((X - \mathbf{E}(X))^2\big) \tag{2.7}$$

$$\mathbf{Cov}(X, Y) = \mathbf{E}\big((X - \mathbf{E}(X))(Y - \mathbf{E}(Y))\big) \tag{2.8}$$

To further illustrate probability spaces and random variables, two simple random experiments will be presented.

Rolling a dodecahedral dice – The dodecahedron is a twelve sided polyhedron. It consists of twelve regular pentagonal surfaces and is one of the platonic solids [12]. Assuming the dice is fair, the chances for the dice landing on each side should be equal. As illustrated in Figure 2.5, the faces of the dodecahedron are colored differently and can therefore be distinguished. The set of all possible outcomes can be modelled using the set $\Omega = \{\text{yellow, green, gray, red, blue, black, white, purple, orange, pink, brown, cyan}\}$. We define the set of all measurable events as $\mathscr{A} = \mathcal{P}(\Omega)$, which denotes the power set of Ω. Obviously the $\mathscr{A}$ is closed under complementation, intersection and union as the power set of Ω contains all subsets of Ω. As the dice should be fair, the probability measure $\mathbf{P}$ is be modelled using $\mathbf{P}(A) = \frac{|A|}{12}$. Again the properties of valid probability distributions can easily be verified.

On the probability space $(\Omega, \mathscr{A}, \mathbf{P})$ a random variable X can be defined by a bijective mapping from Ω to the set $\{1, 2, \ldots, 12\}$. For the random variable the expectation is given by $\mathbf{E}(X) = \sum_{i=1}^{12} i/12 = 13/2$.

Continuous wheel of fortune – The continuous wheel of fortune is depicted in Figure 2.6. A random experiment can be created by rotating the wheel and recording the position of the pointer, when the wheel stops. The set of all possible outcomes Ω is the half-closed interval $[0, 2\pi)$ and the set of all measurable events $\mathscr{A}$ is the set of all subsets $A \subseteq [0, 2\pi)$. For symmetry reasons it should be assumed, that the probability density is equal everywhere. We therefore define the density function of the random variable X to be $\mathbf{f}(x) = \frac{1}{2\pi}$ if $x \in [0, 2\pi)$ and zero elsewhere. The distribution function $\mathbf{F} = \int_{-\infty}^{x} \mathbf{f}(x)\,\mathrm{d}x$ is a valid distribution function and can for instance be used to calculate

the probability of the random variable taking a value in the interval $I = [a, b] \subset [0, 2\pi)$ using $\mathbf{P}(a < X \leq b) = \mathbf{F}(b) - \mathbf{F}(a)$.

Given the random variable X and the density function $\mathbf{f}$ the expectation can be calculated. The expectation is given by $\mathbf{E}(X) = \int_0^{2\pi} x/2\pi \, \mathrm{d}x = \pi$.

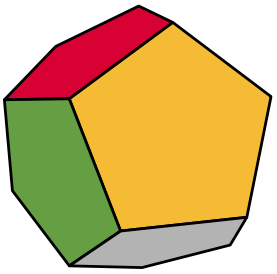

FIGURE 2.5: A dodecahedron with differently colored faces.

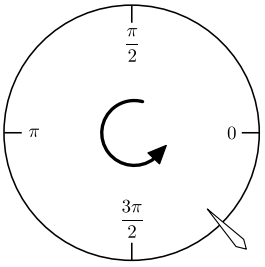

FIGURE 2.6: A continuous version of the wheel of fortune.

2.2.2 Gaussian Distribution

The Gaussian distribution (also normal distribution), is a common model for the distribution of continuous variables. Its density function in the univariate case is depicted in Equation 2.9, where μ denotes the mean of the curve and σ denotes the standard deviation. In the multivariate case the density function for an n-dimensional vector x is depicted in Equation 2.10, where μ again denotes the mean and Σ denotes the covariance matrix.

$$\mathbf{N}(x|\mu, \sigma) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\Big(- \frac{(x-\mu)^2}{2\sigma^2}\Big) \tag{2.9}$$

$$\mathbf{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{n/2}} \frac{1}{(|\Sigma|)^{1/2}} \exp\Big(- \frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\Big) \tag{2.10}$$

The expectation of a random variable X under the univariate Gaussian distribution is μ, whereas the variance is given by σ^2. In the multivariate case, the expectation is μ, whereas Σ describes the covariances. In this thesis the Gaussian distribution is used for estimating a non-parametric density, as explained in Subsection 2.2.3.

2.2.3 Kernel Density Estimation

Kernel density estimation is a non-parametric method for estimating the density of a random variable X, based on observations $(x_1, x_2, \ldots, x_N)$. Non-parametric, in this context, means that the estimation method makes only few assumptions about the form of the distribution. The general form of the kernel density estimation is depicted in Equation 2.11, where K_σ denotes a smoothing kernel with smoothing parameter σ. A kernel is a real valued function that maps to values greater or equal to zero and the integral over the function must be one. A valid kernel function is for instance the truncated uniform distribution, which maps to $1/2$ if $|u| \leq 1$ and zero elsewhere. However this kernel might introduce a discontinuous density estimation function.
Smoother results can be achieved using Gaussian kernel density estimation, where the

density is estimated as the average of a number of Gaussian densities. The Gaussian density estimation function is depicted in Equation 2.12.

$$\mathbf{f}_K(x) = \frac{1}{N}\sum_{i=1}^{N} K_\sigma(\|x - x_i\|) \tag{2.11}$$

$$\mathbf{f}_G(x) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right) \tag{2.12}$$

As the estimated function is a valid density function and as it is a sum of continuous functions, the function itself is continuous everywhere. The only free parameter is the standard deviation of the Gaussian kernels σ, which acts as a smoothing parameter. The effect of the smoothing parameter is visualized in Figure 2.7. If the smoothing parameter is set to low ($\sigma = 0.05$), the estimated function oscillates in the vicinity of sample points and if the smoothing parameter is set to high ($\sigma = 0.6$), the estimated function only poorly approximates the original function.

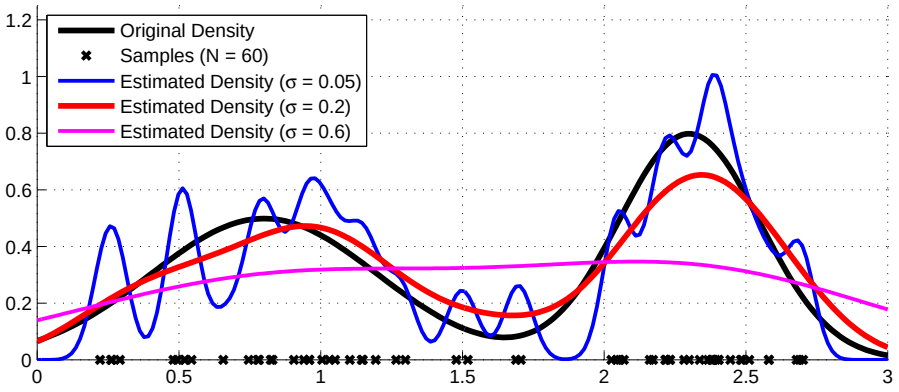

FIGURE 2.7: The figure shows the effect of the smoothing parameter. If the smoothing parameter is set to low ($\sigma = 0.05$), the estimated function oscillates in the vicinity of sample points and if the smoothing parameter is set to high ($\sigma = 0.6$), the estimated function only poorly approximates the original function.

2.2.4 Histogram Density Estimation

The Histogram density estimation is another common non-parametric density estimation method. Using the univariate histogram method, the image space of the random variable is partitioned into k equi-sized bins of length Δ_i. The probability for X falling into bin i is given by $p_i = {}^{n_i}/_{N\Delta_i}$, where n_i denotes the number of observations of X falling into bin i and N denotes the overall number of observations. The histogram method estimates a valid density function and its only parameter is the number of bins k. However, the histogram method introduces artificial discontinuities in the density function at the border of the bins.

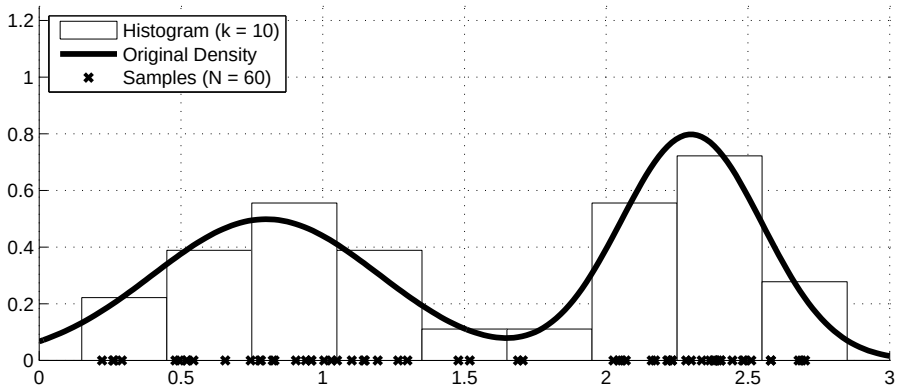

FIGURE 2.8: The figure shows an estimated density function created with the histogram density estimation method. The only free parameter is the number of bins, which was set to 11 in that case.

2.2.5 Support Vector Machines

The support vector machine is a method for classifying a given set. Classification is the problem of assigning a class label c to an input vector x. In this section support vector machines are introduced for the linear separable case with two classes.

A support vector machine performs classification based on a given set of training samples $x_1, \ldots, x_n$ and their classifications $t_1, \ldots, t_n$, where $t_i \in \{-1, 1\}$ in the case of two classes. We assume, that the training samples can be separated linearly in a feature space. The classification function is given by: $f(x) = \text{sign}(w^T\theta(x) + b)$, where w and b describe the classification hyperplane to be learned and θ is a function that maps from the sample space to the feature space. For simplicity of the formulas, we assume that w is of unit length. Figure 2.9 depicts the classification hyperplane and two linearly separable sets.

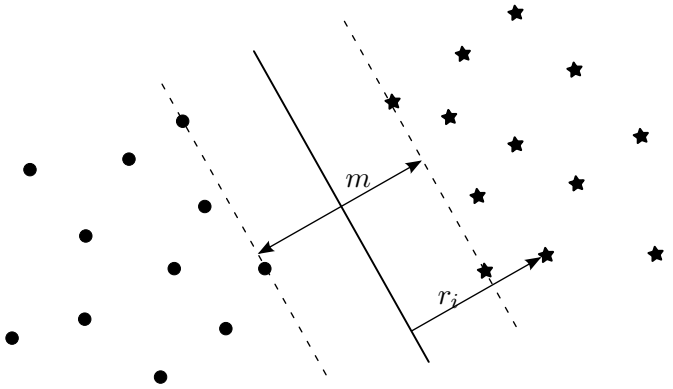

FIGURE 2.9: The figure shows the functional margin r_i of feature point x_i and the margin m of the separating hyperplane. The depicted hyperplane already has maximum margin.

The functional margin of training samples x_i in the feature space is their distance to the classification hyperplane and is given by $r_i = t_i(w^T\theta(x_i) + b)$. The training samples with the least distance to the classification hyperplane are called support vectors. The margin of the classification hyperplane is given by $2 \cdot min\{r_i\}$. Maximizing the margin of the

classification hyperplane leads to a particular solution of the classification hyperplane. The optimal parameters for the hyperplane are denoted in Equation 2.13. For the derivation and the solution of the optimization problem the reader of this thesis is referred to Bishop's "Pattern Recognition and Machine Learning" [11].

$$\arg\max_{w,b} \left\{ \min_{i} \left\{ t_i \big(w^T \theta(x_i) + b \big) \right\} \right\} \tag{2.13}$$

2.2.6 Principal Component Analysis

Principal component analysis is a method to calculate an orthogonal projection that maximizes the variance of the projected data. Let the unit-length vector u_1^T denote a linear mapping from the sample space to a one dimensional Euclidean space. The variance of the projected data is given by $\frac{1}{n}\sum_{i=1}^{n}\left(u_1^T x_i - u_1^T \bar{x}\right)^2 = u_1^T S u_1$, where $\bar{x}$ denotes the mean of the sample data and S is the covariance matrix given by: $S = \frac{1}{n}\sum_{i=1}^{n}\big(x_i - \bar{x}\big)\big(x_i - \bar{x}\big)$. Maximizing the projected variance under the side constraint $|u_1| = 1$ leads to the expression: $Su_1 = \lambda u_1$, which implies that the principal components are eigenvectors of the covariance matrix.

The principal components analysis is used in Chapter 4 to align 3D models along their principal axes. Figure 2.10 shows the principal components of a random two-dimensional point set.

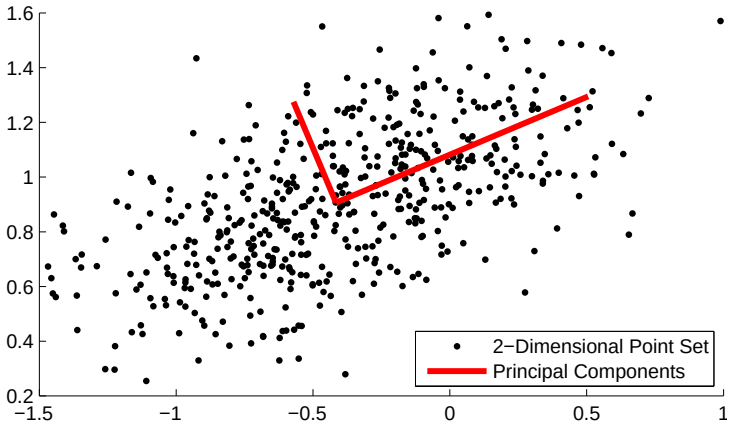

FIGURE 2.10: The figure shows a random 2-dimensional point set and its principal components.

2.2.7 Markov Random Fields

A Markov random field is an undirected probabilistic graphical model. It is described by a graph $\mathcal{G}$, whose vertices $x_i \in (x_1, \ldots, x_n)$ are random variables. The vertices are connected with edges, that describe probabilistic dependencies between two vertices. Figure 2.11 depicts a Markov random field composed of four random variables.

For a given Markov random field, the joint probability $\mathbf{p}(x)$ can be written as a product of the potential functions of their maximal cliques $\mathcal{C}_i$. A clique is a subset of a graph's

vertices, such that all vertices are connected to each other. A maximal clique is a clique which is not a strict subset of a larger clique [13]. The Markov random field in Figure 2.11 consists of two maximal cliques. A potential function is a positive function, that expresses the affinity of a given clique. Common choices for potential functions are exponential functions, such as the exponential distribution, which is defined by the density function $\mathbf{f}(x) = \lambda e^{(-\lambda x)}$. If the set of vertices within a clique is denoted as $x_{\mathcal{C}}$, the joint probability for the whole Markov random field is given by the formula depicted in Equation 2.14

$$\mathbf{p}(x) = \frac{1}{Z} \prod_{\mathcal{C}} \psi_{\mathcal{C}}(x_{\mathcal{C}}) \tag{2.14}$$

$$Z = \sum_{x} \prod_{\mathcal{C}} \psi_{\mathcal{C}}(x_{\mathcal{C}}) \tag{2.15}$$

where $\psi_{\mathcal{C}}(x_{\mathcal{C}})$ denotes the potential of clique $x_{\mathcal{C}}$. The function Z is called partition function and ensures, that the joint probability is a valid probability distribution.

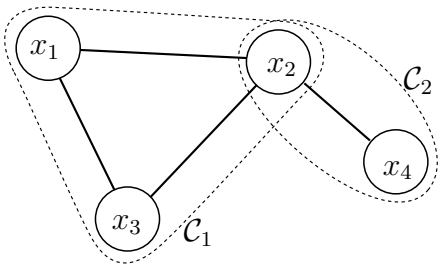

FIGURE 2.11: A Markov random field with two maximal cliques.

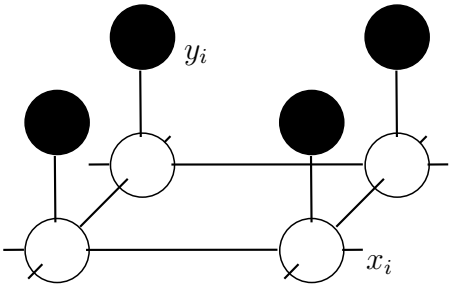

FIGURE 2.12: The derived Markov random field for image denoising.

To illustrate the applicability of Markov random fields, an example based on Bishop's example [11, pp. 387–389] will be illustrated here. Given a noisy binary image, whose pixels are denoted by y_i, a de-noised image composed of pixels x_i can be derived by maximizing the joint probability of a Markov random field. The output pixels x_i are connected with the according input pixels y_i by edges $\mathcal{E}$, as the value of the output pixel x_i should depend on the input pixel y_i. Furthermore the output pixels are connected with their four neighborhood by edges $\mathcal{F}$, as the output intensity should be consistent at neighborhoods. The derived Markov random field is depicted in Figure 2.12. The black vertices represent the intensity values of the input image and the white vertices represent the intensity values of the output image. To establish a joint probability for the given Markov random field, a potential function for all maximal cliques is needed. Obviously all maximal cliques in the example are of size two and can be divided in two different types: Edges between input and output pixels and edges between output pixels. Assuming black pixels are denoted by -1 and white pixels are denoted by $+1$ a valid joint probability function can be modeled using:

$$\mathbf{p}(x) = \frac{1}{Z} \prod_{(x_i,y_i)\in\mathcal{E}} \exp(\alpha x_i y_i) \prod_{(x_i,x_j)\in\mathcal{F}} \exp(\beta x_i x_j), \tag{2.16}$$

where α and β are free parameters. By optimizing the joint probability function the most likely output image can be calculated.

The input image is shown in Figure 2.13. Based on the joint probability function an optimization problem can be formulated, that finds values for the pixels x_i such that the probability $\mathbf{p}(x)$ is maximal. A solution for the optimization problem can be approximated using the iterated conditional models (ICM) algorithm. This algorithm simply performs a greedy optimization step for every coordinate until convergence. By choosing appropriate values for α and β and optimizing the joint probability function with the ICM algorithm the de-noised output image in Figure 2.14 is obtained.

FIGURE 2.13: A noisy black and white image.

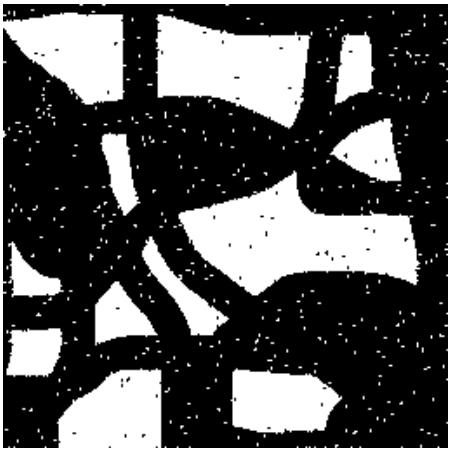

FIGURE 2.14: Image de-noised by optimizing a joint probability function.

2.3 Shape Representations

Many different shape representations exist. The choice of the "right" shape representation depends on the problem domain. For example 3D models in a computer game should be rendered quickly and look acceptably realistic, whereas in product design surfaces should look smooth and exhibit certain geometric properties while rendering time is not of utmost importance. In this section the mainstream shape representation models from Alan Watts "3D Computer Graphics" [14] will be presented.

It should be noted here, that the terms "object", "3D model" and "shape" are often used as synonyms. In this thesis the convention of Alan Watt will be used: A 3D model is a description of the shape of an object.

2.3.1 Polygonal Model Representation

Using a polygonal model representation the 3D shape is approximated with a mesh of planar faces. The accuracy of the representation can be controlled by the number of faces. A triangle mesh representation of a sphere is depicted in Figure 2.15. However it is obvious, that models composed of planar faces are not smooth. The polygonal model can be stored in different types of mesh data structures. The simplest form is a list of triangles, where every triangle knows coordinates of its vertices. This representation is, for instance, used in the stereo lithography file format (STL), which serves as a low level file exchange format for polygonal models.

However in a simple triangle list edges and vertices are not explicitly represented, which makes traversing and editing of the mesh a cumbersome job. That is why, more elaborate data structures have evolved like the halfedge data structure [15]. The halfedge data structure splits an edge into two opposite halfedges. Every halfedge knows its opposite halfedge, its incident face, its incident vertex and the next halfedge to traverse along its face. This data structure is depicted in Figure 2.16.

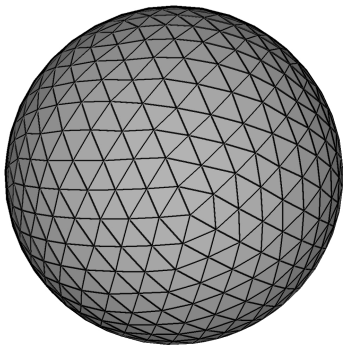

FIGURE 2.15: A sphere represented by a triangle mesh. The accuracy of the representation can be controlled by the number of faces. However, models composed of planar faces are not smooth.

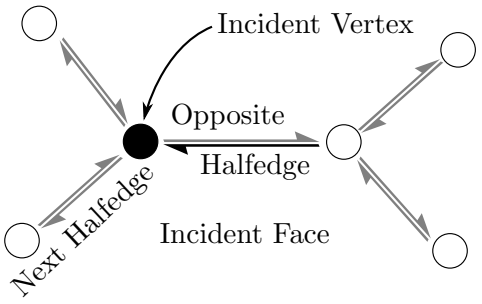

FIGURE 2.16: The halfedge data structure. Every halfedge knows its opposite halfedge, its incident face, its incident vertex and the next halfedge to traverse along its face (Modified from [16]).

2.3.2 Parametric Patches

One disadvantage of polygonal model representations is, that they cannot represent smooth objects exactly. However smooth surfaces are of utmost importance in some industries, for instance in the car industry. That is why parametric patches were developed. Parametric patches are represented explicitly by mathematical functions. The most common parametric patch representation is the non-uniform rational B-spline (NURBS) representation [17]. A NURBS Surface with random parameters is depicted in Figure 2.17.

To represent a whole object using parametric patches a boundary representation (B-rep) is needed. A B-rep represents a solid object by its "skin" which divides the outer from the inner space. The B-rep combines topological and geometrical aspects. On the topological side a 3D object is represented by a set of faces, which are bounded by edges. The edges are delimited by vertices. The topological entities have a geometric description. Faces are described as parts of surfaces, which can for instance be parametric patches or a plane surface or a cylindrical surface. The edges, that bound the faces are described as parts of curves, which can for instance be parametric curves, circles or straight lines. The vertices delimiting the edges are described as points [18]. The B-rep data structure is depicted in Figure 2.18.

2.3.3 Constructive Solid Geometry

Constructive Solid Geometry (CSG) is a high level shape representation, where models are described by the application of Boolean set operations on elementary objects, so

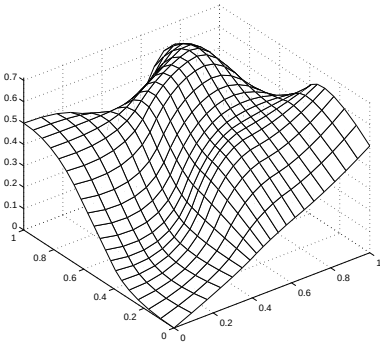

FIGURE 2.17: An example of a NURBS surface. The parameters were set to randomly chosen values.

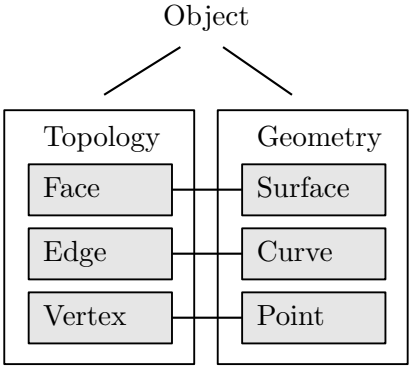

FIGURE 2.18: The B-rep data-structure combines topological and geometrical aspects. Topologically an object is represented using faces, edges and vertices. Geometrically faces are described as parts of surfaces, edges are described as parts of curves and vertices are described as points.

called primitives. Primitives are for instance: spheres, cylinders or cuboids, whereas the Boolean operations are subtraction, union and intersection. Figure 2.19 shows how cylinders can be combined to form a pipe with a flange using CSG.

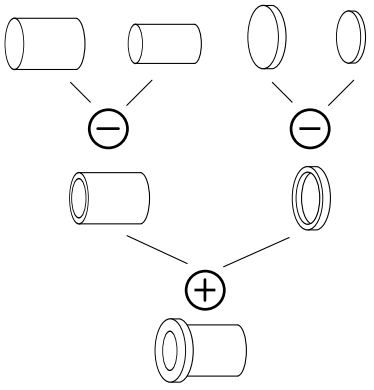

FIGURE 2.19: The figure shows how cylinders can be combined to form a pipe with a flange using Constructive Solid Geometry.

2.3.4 Spatial Subdivision Techniques

Spatial subdivision methods divide the object space into parts and label each part as either empty or filled. Voxel representations divide the object space into a uniform grid of small cubes, so called voxels. The voxel representation in 3D is analogous to the 2D bitmaps. As an example for voxel models, the Stanford bunny is depicted in Figure 2.20. A more elaborate subdivision method is the octree data structure, where the object space is divided recursively until the required accuracy is achieved. The octree version of the Stanford bunny is depicted in Figure 2.21.

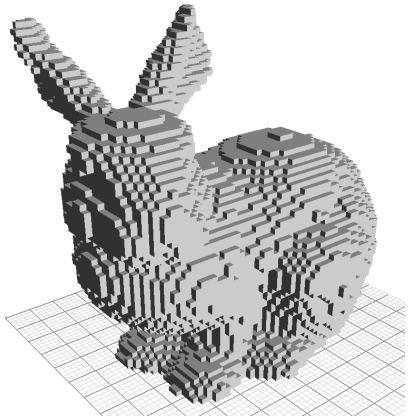

FIGURE 2.20: A voxelized version of the Stanford bunny.

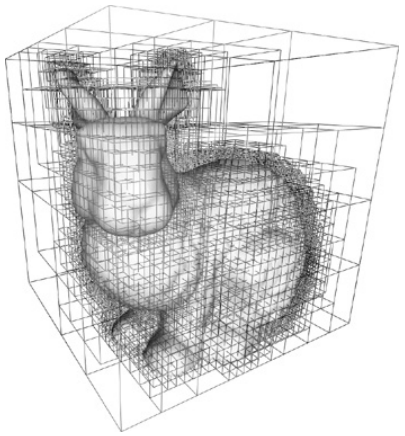

FIGURE 2.21: The Stanford bunny represented in an octree. [19]

2.3.5 Implicit Representation

3D models can be represented implicitly by the solution of an equation $\mathbf{f}(x) = 0$. The domain of the function $\mathbf{f}(x)$ is the 3-dimensional Euclidean space $\mathbb{E}^3$ and it maps to $\mathbb{R}$. The shape of the 3D model is given by the set of points, that satisfy the equation.

For example a sphere can be defined implicitly by the equation $0 = x_1^2 + x_2^2 + x_3^2 - 1$. The representation of 3D objects is restricted to very special objects. Implicitly represented objects can for instance be rendered using a raytracer or by converting the implicit function to a polygonal representation.

2.3.6 Procedural Models

Manual modeling of complex scenes, such as cities, can become infeasible due to the number of 3D objects and their level of detail. As a consequence, procedural modeling techniques have been developed. Procedural modeling techniques employ formal construction rules to create 3D Models [20]. A list of notable procedural modeling techniques can be found in Müller et al. [20] and Vanegas et al. [21]. Common procedural modeling techniques include: Lindenmayer systems, shape grammars and generative modeling.

Lindenmayer systems are a type of grammar. They are commonly used for modeling biological structures such as trees or ferns. The grammar is a three-tuple consisting of an alphabet, a set of productions and a start symbol. A production defines how a symbol can be replaced by a word. The grammar defines a language, whose words can be interpreted graphically [22]. Prusinkiewicz [22] used turtle graphics to interpret the words graphically.

Shape grammars are a formal system, where derivation rules are applied directly on labeled lines and points. Shape grammars were introduced by Stiny [23] and they have successfully been used for the construction and analysis of buildings.

Generative modeling is a modeling paradigm, where the process of the model generation is described and not only its end product [24]. Havemann et al. [24] have introduced the “Generative Modeling Language”, which is a postscript-like language to create

3D-models. Another approach to generative modeling called “Euclides” has been introduced by Schinko et al. [25]. This approach uses JavaScript to describe the modeling operations. Euclides is used in this thesis and further described in Section 4.1.

Chapter 3

Related Work

Many content-based retrieval methods for 3D models have been proposed recently. Tangelder et al. [5] and Bustos et al. [26] have both surveyed literature on content-based retrieval methods.

Bustos et al. as well as Tangelder et al. list important properties of retrieval methods. Both list efficiency, effectiveness as well as robustness. Effectiveness refers to the quality of the retrieval results, it can, for instance, be measured using precision-recall graphs as described in Subsection 2.1.2. Efficiency refers to the resources needed for retrieval. Efficiency is usually measured by the response time of a retrieval method and the memory used for storing the search index. Robustness means that the retrieval method is insensitive to small variations, such as noise and topological degeneracies [26].

Tangelder et al. further specify the need for pose normalization, the ability of partial model matching and the need for a certain shape representation as attributes of retrieval methods.

If the similarity measure used in the matching phase is not invariant under scale, translation or rotation, the models need to be normalized first. Scale and transition can be normalized by scaling the object to the unit bounding box and centering at its center of gravity. The rotation can for instance be normalized by applying principal component analysis, as described in Subsection 2.2.6. However more elaborated alignment techniques exist for special types of 3D models. An extreme example for such a technique is "Upright Orientation of Man-Made Objects" by Fu et al. [27], where a machine learning algorithm is trained to predict the correct orientation, based on a feature vector. Normalization may be a necessary preprocessing step for some matching methods, but it should be noted that information may be lost through normalizing 3D models. As an example, by normalizing models to a common size, toy cars may become very similar to real cars.

The ability of partial matching is important for matching incomplete models and for finding parts in a larger scene. Biasotti et al. [28] argue that partial matching is a desired feature for engineering applications, as partly similar models can be reused and adapted to fit new needs. Retrieval methods are depended on a certain type of shape representation. The most prominent shape representations have been covered in Section 2.3. Furthermore for some retrieval methods the models must be manifold [5]. An intuitive definition for manifoldness of a surface at a point is that the surface patch

that lies within a sufficiently small ϵ-sphere around that point is topologically equivalent to a disk. A surface is called manifold, if it is manifold at each point [15].

Tangelder et al. divide shape matching methods into three categories: feature-based, graph-based and geometry-based methods. Feature-based methods operate on global (e.g. volume, area) or local (e.g. curvature) properties. Graph-based methods calculate a graph, such as a skeleton, based on the 3D model and perform matching based on graphs. Geometry-based methods operate directly on the models geometric representation [5].

Matching methods cannot always be classified clearly into one of the categories, as many methods combine two or more different matching paradigms. Representative methods of each category will be presented in the following subsections.

3.1 Feature Based Retrieval Methods

Feature based methods can be further divided, based on whether global features, global feature distributions, local features or spatial maps of features are used [5].

3.1.1 Global Feature Based Retrieval Methods

Global feature methods calculate a single feature vector for the whole 3D model. Examples for global features are the scale, the size of the bounding box or the density of the bounding box used by Paquet et al. [29]. Kolonias et al. [30] incorporate the aspect ratio of the objects bounding box, which is also a global feature, into their similarity measure.

It is obvious that the aforementioned global features do not discriminate the object very well, however they can be calculated quickly are easy to implement and can for instance be used to filter the set of relevant 3D models before searching with other retrieval methods to reduce the overall run time. Tangelder et al. [5] argue that global feature methods are not very sensitive to variations in object details as they describe the overall object.

3.1.2 Global Feature Distribution Based Retrieval Methods

3D Models can also be described by the global distribution of their features. Osada et al. [31] have introduced a descriptor based on the probability distribution of geometric properties. They call their descriptors “Shape distributions”. They compare five different geometric properties, namely the angle between three points, the distance between the center of gravity and a point on the surface, the distance between two points on the surface, the square root of the area of a triangle of three points on the surface and the cube root of the volume of a tetrahedron between four points on the surface. They found out that the probability distribution of the distance between two random points on the surface led to the best results. Figure 3.1 shows the shape distributions of a mug and a sphere, which act as the shape descriptor for retrieval.

The shape distribution has for instance been extended by Hazma et al. [32] to work with geodesic distances and by Liu et al. [33] to work with the thickness of the 3D model.

Tangelder et al. [5] observed that shape distribution methods are able to distinguish broad model categories very well, however they are insensitive to model details.

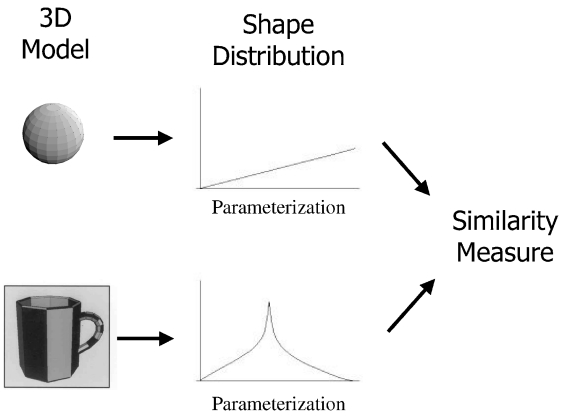

FIGURE 3.1: The figure shows the shape distributions of a mug and a sphere. The distributions act as the shape descriptor for retrieval [31].

3.1.3 Spatial Maps of Features

Instead of calculating a single global feature vector or the probability distribution of features, features can also be calculated for sections of the 3D model. Such methods are called spatial maps by Tangelder et al. [5], they capture the relative spatial location of the features of an object.

Ankerst et al. [34] have introduced spherical shape histograms. Shape histograms decompose the space into disjoint cells, which correspond to the bins of a histogram. As depicted in Figure 3.2 in 2D, they have proposed three different types of histograms: shell bins, sector bins and combined bins. The spherical cells are centered at the center of gravity and the corresponding histograms are built from uniformly distributed points on the objects surface. They have evaluated their method for the retrieval and classification of molecules.

Kriegel et al. [35] also divide the space into shape cells, however they use uniform cubes. They first convert the model to a voxel representation, divide the model into cells and then calculate features on each cell. A schematic 2D example of their approach is shown in Figure 3.3. They have analyzed the features: cell density, solid angle and the eigenvalues of the principal components of a cell. The cell density is given by the number of voxels inside a cell divided by the cells volume, the solid angle measures the concavity and the convexity of surfaces.

3.1.4 Local Feature Based Methods

Local feature based methods build a feature vector, based on the neighborhood of a point on the models surface [5].

Johnson et al. [36] introduced spin images as a local descriptor for 3D models. For that, they require the model to be represented in an oriented polygonal mesh. At every mesh vertex the normal vector and the coordinates of the vertex define a plane going

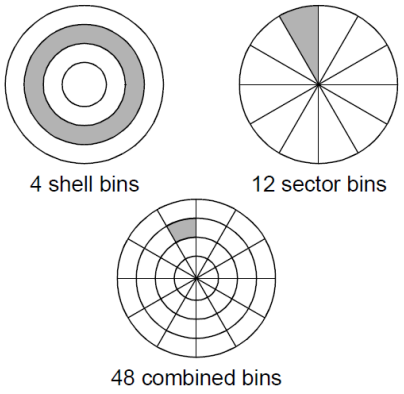

FIGURE 3.2: A two-dimensional version of the shape histograms introduced by Ankerst et al. (modified from [34])

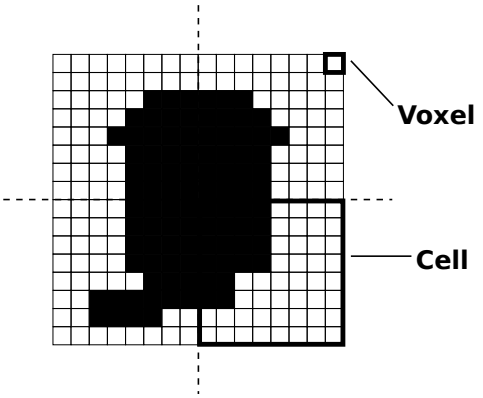

FIGURE 3.3: A two-dimensional version of the shape histogram introduced by Kriegel et al. (modified from [35])

through that vertex and a ray, starting at that vertex and pointing into the normal direction. For every neighboring vertex the perpendicular distance to the ray α and the signed perpendicular distance to the plane β can be calculated. Using those distances of the neighboring vertices, a histogram can be built for every vertex. This histogram is called the spin image. Spin images of a scene and a model are depicted in Figure 3.4. Dark areas of the spin images indicate, that many neighboring points were projected to that bin of the histogram. The parameters of the spin image are the size of the image, the number of bins and the size of the neighborhood. Two models are matched by calculating the correlation coefficient of their spin images. The models are said to be similar if many points on the surface are similar. This approach does not depend on normalized poses and also allows for partial matching as indicated in Figure 3.4.

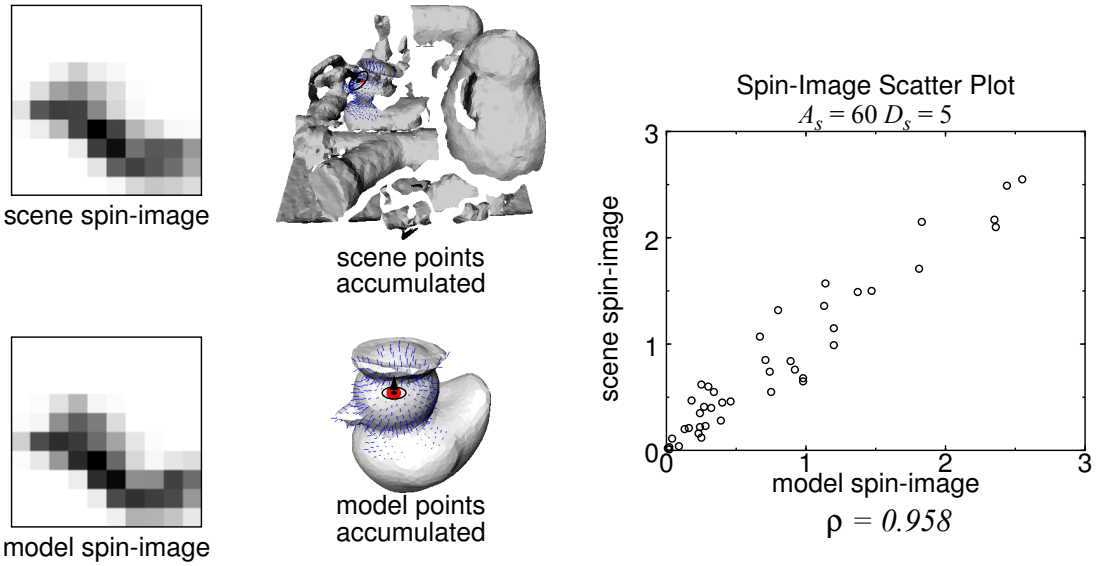

FIGURE 3.4: The image shows the application of spin images for partial matching. The middle column shows two vertices with similar neighborhood in two distinct models. On the left side the corresponding spin images are shown. The scatter plot on the right side indicates, that both spin images linearly correlate. (modified from [36])

Another local feature based method is "3D Shape Contexts" by Körtgen et al. [37]. Körtgen et al. calculate a shape histogram for uniformly sampled points on the surface of a 3D model. The shape histogram is built based on the relative position of uniformly sampled points on the surface. This histogram is called the shape context for a point. The shape histograms used are based on those depicted in Figure 3.2. Körtgen et al. compare shape contexts using the χ^2 distance on normalized histograms.

Wessel and Klein [38] have introduced a method to learn the compositional structure of man-made objects. Therefore the object must be represented as a 3D point cloud. The point cloud is split into primitives, such as planes, spheres, cylinders, cones and tori using a point cloud segmentation algorithm [39]. Spin images are used to describe the primitives. A probabilistic framework is used to learn the spatial arrangement of the detected shape primitives.

3.2 Graph Based Retrieval Methods

Graph based retrieval methods operate on a graph that represents the 3D model. The graph describes how components are connected and it is used for matching. Even though graphs can be seen as special types of features, they will be treated in an own section in this thesis, as there are many retrieval methods based on graphs.

Graphs can be compared by computing the editing distance between two graphs or by calculating the maximal common subgraph. The edit distance between graph_1 and graph_2 is the shortest sequence of edit operations that transform graph_1 into graph_2. A maximum common subgraph of two graphs graph_1 and graph_2 is a subgraph of both graphs such that there is no larger subgraph of graph_1 and graph_2. Both problems are NP-complete problems, that is why heuristics are mostly used to approximate their solutions [40].

Schnabel and Klein [41] have introduced a graph based method for the recognition of objects in 3D point clouds. A point cloud segmentation algorithm [39] is used to split the input model, represented as a point cloud, into planes, spheres, cylinders, cones and tori. Based on the primitives the topology graph is constructed, where a vertex represents a primitive and an edge represents the adjacency of two primitives. Using subgraph matching, 3D objects can be found in the topology graph of the point cloud.

3.2.1 Model Graph Based Methods

Model graph based methods directly operate on a graph based representation of a model. Examples for graph based representations are CSG-trees or the B-Rep data structure. Such methods are especially relevant for the retrieval of CAD models [5].

Cicirello and Regli [42] presented a method to calculate the similarity of solid models based on their machining features. Machining features, in their case, are extrusions, ribs, chamfers or holes. They start by converting the tree of machining features, which is actually a special type of CSG representation, into the model dependency graph. The model dependency graph is an acyclic graph where the vertices represent features and the edges represent some sort of spatial dependence of the features. The similarity of

two model dependency graphs is calculated by approximating the size of the greatest common subgraph.

McWherter [43] proposed a method to index solid objects represented as B-Reps. For that they convert the models from B-Rep to model signature graph. The model signature graph is a graph, where the vertices represent the faces of the B-Rep and edges represent the edges of the B-Rep. For every vertex the following four attributes are stored: face type, geometric representation properties, the face area and a set of surface normals. For every edge the edge type, a convexity identifier, geometric representation parameters and the length of the edge are stored. The affinity between two solid models is calculated by comparing the spectrum of their model signature graphs, which is the sorted set of eigenvalues of the graphs adjacency matrix. The spectrum is treated as a vector in metric space. The vectors in the metric space can be indexed to improve performance. The overall solid model indexing process is depicted in Figure 3.5.

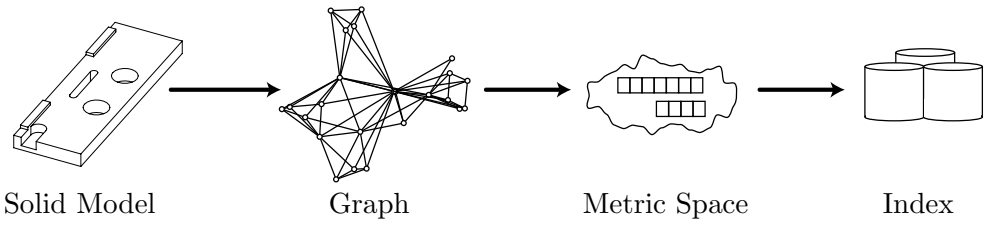

FIGURE 3.5: Overview of the McWherter's approach. Solid models are first converted to the model signature representation, over which a metric space is constructed. The vectors in the metric space can then be indexed. (modified from [43])

3.2.2 Skeleton Based Methods

Skeleton based retrieval methods operate on the skeleton of a 3D model. The skeleton can either be defined manually or it can be automatically extracted. Skeleton extraction algorithms are for instance the volume thinning algorithm by Gagvani et al. [44] or "Skeleton Extraction by Mesh Contraction" by Au et al. [45]. Gagvani et al. perform iterative thinning on a voxel representation of the 3D model whereas Au et al. iteratively contract a 3D mesh using Laplacian smoothing, as illustrated in Figure 3.6.

Sundar et al. [46] have developed a matching method based on skeletons, created by the method of Gagvani et al. [44]. They call their skeletons shape graphs. Each node of the shape graph stores a geometric signature vector which encodes the geometric properties of the neighborhood and a topological signature vector which encodes the topology of the subtrees rooted at that vertex. Two shape graphs are matched by defining a cost function between edges, and approximating the bipartite graph with maximum cardinality and minimum weight. Figure 3.7 shows two different 3D models with similar skeletons matched with their skeleton matching method.

Special types of skeletons are Reeb graphs. A Reeb graph is defined as the quotient space of a shape and a quotient function. An exact definition is given in Hilaga et al. [47]. Hilaga et al. use the integral of geodesic distance as the quotient function and calculate the Reeb graph for multiple resolutions. Matching is performed iteratively from the coarsest Reeb graph to the finest Reeb graph.

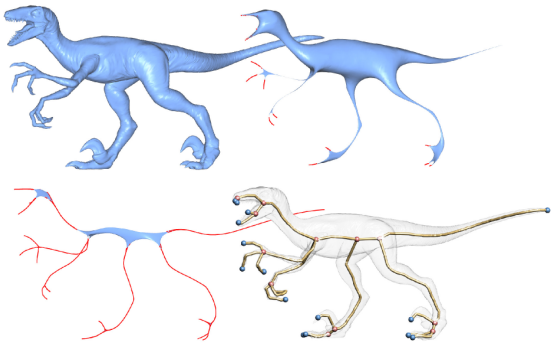

FIGURE 3.6: The Figure shows iterative skeleton extraction by mesh contraction by Au et al. (modified from [45])

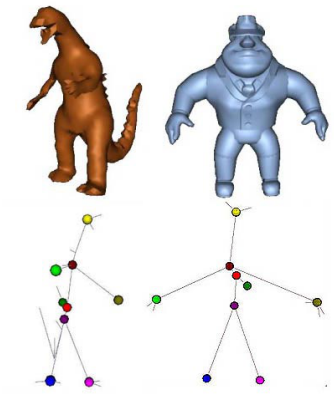

FIGURE 3.7: Two different 3D models with similar skeletons are matched with the skeleton matching method by Sundar et al. [46]

3.3 Geometry Based Retrieval Methods

Geometry based retrieval methods operate solely on a geometric representation of a 3D model. Those methods can be divided in to view based retrieval methods, volumetric error based retrieval methods, and weighted point set based retrieval methods.

3.3.1 View Based Retrieval Methods

View based similarity methods match models by matching views from various viewpoints. The main idea is that, 3D models are similar if they look similar from various viewpoints [48].

Chen et al. [48] presented a retrieval method, that matches models based on their light field. A light field of a 3D model in this context describes the radiometric properties of light in a space. A light field is a four dimensional function, which is represented by Chen et al. as collection of 2D views around the model. The lightfield is created by rendering the 3D model from the 20 vertices of a dodecahedron containing the model. Two 3D models and their lightfields are represented in Figure 3.8. As the silhouettes of the model are identical from opposite vertices, only 10 views are necessary. The dissimilarity of the 3D model is given by the sum of the view's dissimilarity. The dissimilarity of two images is calculated using the method presented in "An Integrated Approach to Shape Based Image Retrieval" by Zhang et al. [49]. To achieve rotation invariance, all 60 possible rotations of the dodecahedron are examined and the minimum dissimilarity is taken to be the dissimilarity of the two 3D models.

3.3.2 Volumetric Error Based Retrieval Methods

Novotni et al. [50] approximate the similarity of two objects by calculating their volumetric error. Therefore they calculate the distance field of one object and calculate the overlap of the other object. A measure on the histogram of distances of the overlap

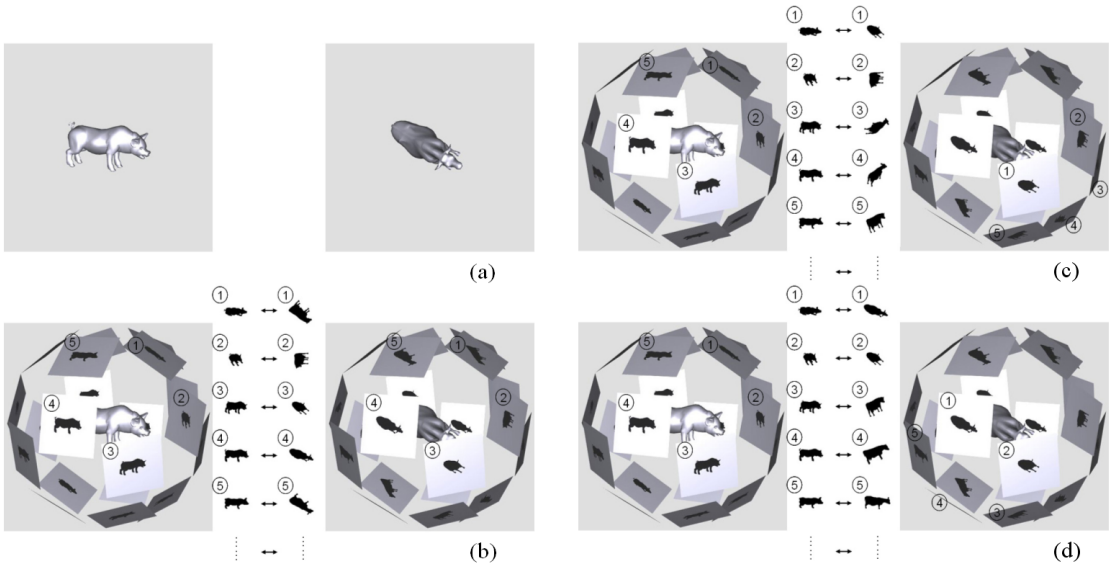

FIGURE 3.8: The Figure shows two similar 3D models and their corresponding light fields. The dissimilarity of two lightfields is given by the dissimilarity of the views. Three of the 60 different rotations of the second model (a), which influence the similarity, are shown (b,c,d) [48].

is used to approximate the similarity. The method of Novotni et al. depends on pose normalization and the similarity measure is not symmetric.

Sánchez-Cruz et al. [51] have presented a method that calculates the voxels to move and how far they have to move to map one voxel model to another. Their method operates on two voxelized models, called model A and model B. First they calculate the voxels, that are not in the intersection of the voxelized models A and B and formulate an optimal bipartite graph matching problem between the difference set of A and B and the difference set of B and A. The weight function for the edges is given by the euclidean distance between the voxels. Figure 3.9 shows two volcanoes (a, b) and their difference sets (c, d).

3.3.3 Weighted Point Set Based Retrieval Methods

Weighted point set based retrieval methods describe the geometry of an object using a set of salient points and their weights. Tangelder et al. [52] choose salient points that have a high magnitude of Gaussian curvature. Two models are compared by comparing the set of salient points and their weights. Tangelder et al. compare two models by introducing a variant of the earth movers distance, that obeys the triangle inequality. Figure 3.10 shows four airplanes and a visualization of their weighted points beneath them. The model on the left is used to query by example and its weighted point set is laid over the other point sets in the visualization.

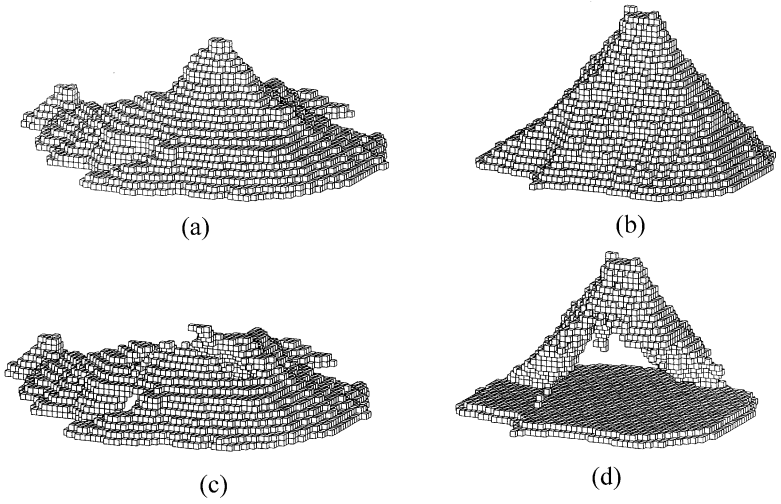

FIGURE 3.9: The Figure shows two volcanoes (a,b) and their difference sets (c,d). By finding the minimum costs of moving the voxels shown in (c) to the voxels shown in (d) a similarity measure is obtained (modified from [51]).

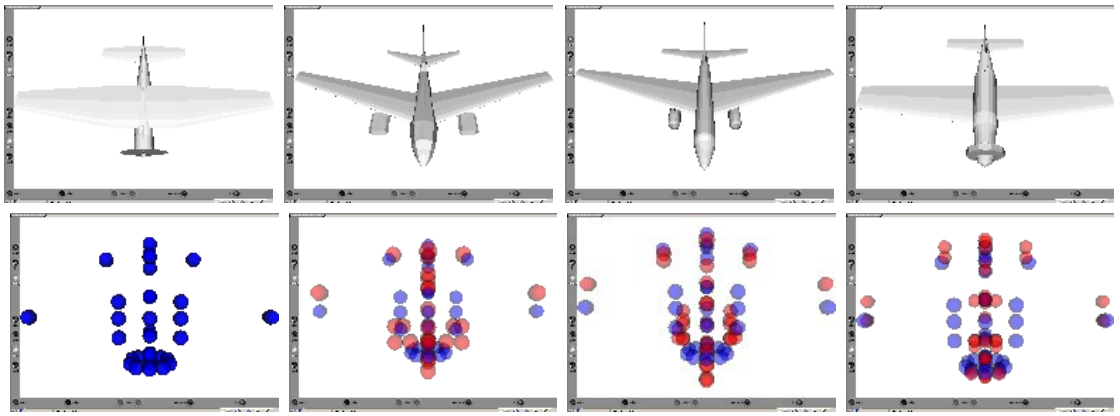

FIGURE 3.10: The Figure shows four airplanes and a visualization of their weighted points beneath them. The model on the left is used to query by example and its weighted point set is laid over the other point sets in the visualization. (modified from [52])

Chapter 4

Methodology

In this thesis I present two methods to retrieve models similar to given generative models. The first method is based on kernel density estimation, it is presented in Section 4.4. The second method uses support vector machines to match 3D models. The method is presented in Section 4.5.

Furthermore I present two attempts to improve retrieval results. By modeling a Markov random field long range conformity of the 3D models is rewarded. This approach is presented in Section 4.6. Another attempt to improve the retrieval results is using diffusion processes, it is described in Section 4.7.

In my approach generative models are used to describe 3D model classes. In the training phase, the generative models span a shape space, of which a number of training samples is taken at random. In this way, no "real" training data is needed a priori. The generative modeling system and the generation of training examples is presented in Section 4.1.

In a preprocessing step, training samples are converted to a voxel representation and their principal axes are aligned to the canonical Euclidean basis vectors. The voxelization and pose normalization step is described in Section 4.2.

When the training models are voxelized and normalized, they are transformed using inverse distance transformation. The volume model is then split into a regular grid of cubes, so called cells. For each cell the histogram of inverted distances is determined, which acts as a feature vector for the cell. The process of feature vector calculation is described in Section 4.3.

In the recognition phase we need to find the most similar models of a test set for a given object class. The object class is represented by its generative model and its learned density function. The first step of the recognition phase is to voxelize and align the models from the test set, calculate the distance transformations and split the models into cells. For each cell the similarity is estimated using kernel density estimation (Section 4.4) or support vector machines (Section 4.5).

4.1 Generation of Training Examples

In my approach 3D model classes are represented by generative models. A generative 3D model M, is an algorithm that takes an argument vector x and produces 3D geometry $M(x)$. Each generative 3D model is used to generate training samples for the class it represents. Without loss of generality, the model's parameter domain $D(M)$ has a multidimensional, rectangular structure; i.e. the Cartesian product of closed intervals. By randomly sampling a number of argument vectors x_i from $D(M)$, a representative subset of the shape space is generated in form of a simple list of random models $M(x_i)$, which are used in the training phase. Figure 4.1 depicts the generation of training examples.

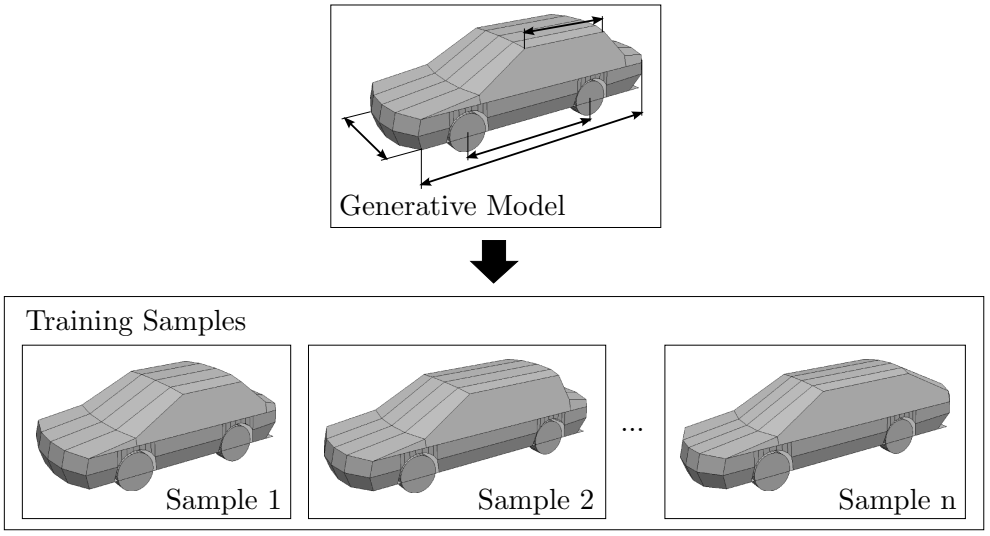

FIGURE 4.1: Training samples can be generated by randomly sampling the parameter domain $D(M)$ of a generative model M. In the bottom of the Figure random samples are depicted.

In this thesis generative models are created using the generative modeling framework Euclides, introduced by Schinko et al. [25]. The Euclides framework translates generative models written in JavaScript to other platforms, like Java or HTML. JavaScript [53] is a well known scripting language, which is easy to read and already used by non-computer scientists [25]. The structure of the Euclides framework is depicted in Figure 4.2. Generative knowledge, depicted on the left side, is provided in JavaScript files. Euclides scans and parses the input and translates it to different platforms. In our case the target platform is Java. The resulting Java code is then compiled and executed. As a result the Java code returns a generated 3D model.

The JavaScript scripts, describing the model classes, must, at least, provide the functions `shapeName()`, `shapeDomain()` and `shape()`. The `shapeName()` function simply returns the name of the model, for example "car". The `shapeDomain()` function returns an array that contains the lower and the upper bounds for every model parameter. The `shape()` function takes an array containing numerical values and returns concrete 3D geometry represented by an indexed face set. An arbitrary number of training models can be created by calling the `shape()` function with random samples from the parameter space.

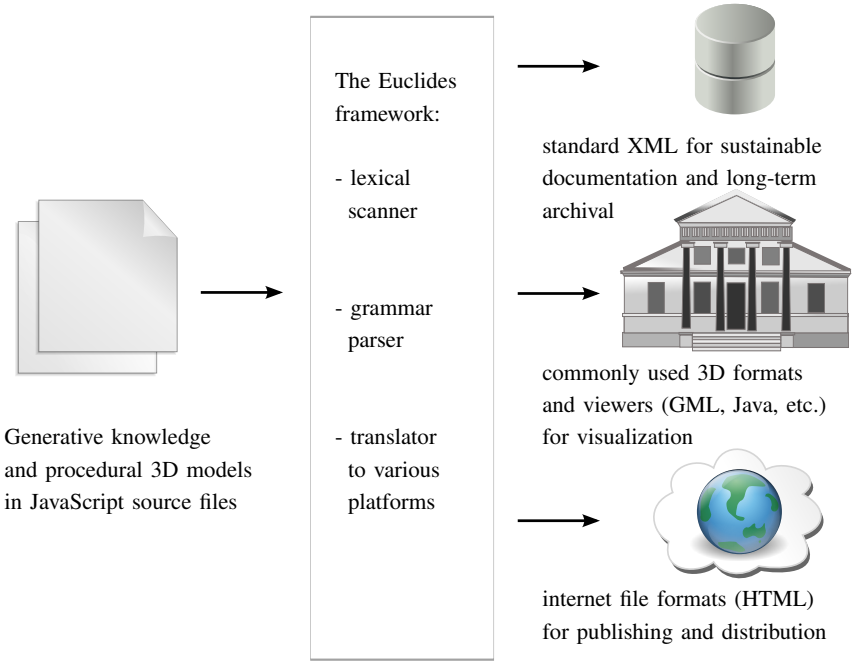

FIGURE 4.2: Generative knowledge, depicted on the left side, is provided in JavaScript files. Euclides scans and parses the input and translates it to different platforms (modified from [25]).

The models are stored in the (.off)-file format, which is also used by the Princeton shape benchmark. The (.off)-file format is a simple serialization of the indexed face set. Listing 4.1 illustrates a generative model of a cube. The cube's only parameter is its edge length. The generative model shown in Figure 4.1 represents the object class "sedan car". The model takes six parameters (wheel base, axle track, car height, car length, top length, and trunk length). It generates and returns a 3D model with a fixed topology (consisting of 672 polygons) and varying geometry. The model is implemented in about 350 lines of code.

```
function shapeDomain() { return [ 5, 8]; }
function shapeName()   { return "cube";  }

function shape(parameters) {
  var myObject = geometricObject();
  var a = parameters[0];

  myObject.addPoly([[0, 0, 0], [a, 0, 0], [a, a, 0], [0, a, 0]]);
  myObject.addPoly([[0, 0, a], [a, 0, a], [a, a, a], [0, a, a]]);
  myObject.addPoly([[0, 0, 0], [a, 0, 0], [a, 0, a], [0, 0, a]]);
  myObject.addPoly([[0, a, 0], [a, a, 0], [a, a, a], [0, a, a]]);
  myObject.addPoly([[0, 0, 0], [0, a, 0], [0, a, a], [0, 0, a]]);
  myObject.addPoly([[a, 0, 0], [a, a, 0], [a, a, a], [a, 0, a]]);

  return myObject;
}
```

LISTING 4.1: A simple example of a generative 3D model. The generative model generates a cube. The cube's only parameter is its edge length.

4.2 Voxelization and Pose Normalization

After the generation of training examples the training models are converted to a voxel format. Patrick Mins [54] voxelizer "binvox" is used to convert the training models to a grid size of $R \times R \times R$. The voxelized training models are then aligned using Principal Component Analysis (PCA). After the application of PCA, the principal axes of the training models are aligned to the canonical basis vectors and the center of gravity of all models is centered at $\left({}^{R}/_{2}, {}^{R}/_{2}, {}^{R}/_{2}\right)$. The aligned and unaligned version of a model are depicted in Figure 4.3. If n is the number of training models, the family of aligned training models is noted $(T_i)_{\{i \in 1,\ldots,n\}}$. After alignment, the volume $\mathbf{v}(T_i, x, y, z)$ of a given element in the voxel grid is either 1, if the voxel contains a part of the surface of the model or zero, otherwise.

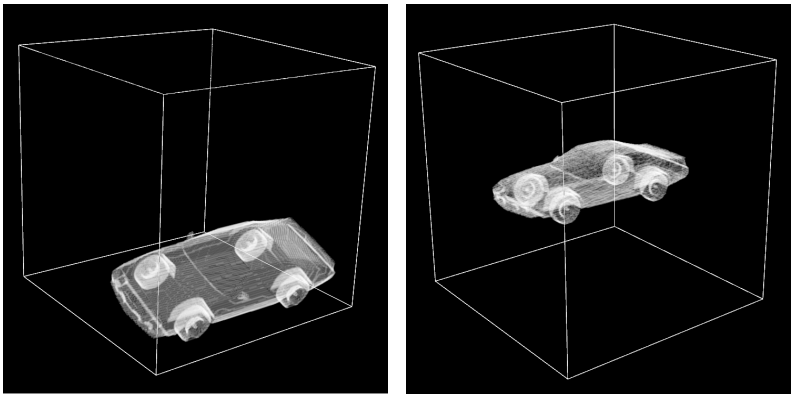

FIGURE 4.3: Unaligned and aligned version of a voxelized car model. The principal axes of the aligned model are aligned to the canonical basis vectors and the center of gravity of all model is centered at $\left({}^{R}/_{2}, {}^{R}/_{2}, {}^{R}/_{2}\right)$.

However, aligning the model using PCA does not guarantee a unique alignment. One reason is that the normalized principal axes are only unique except for the sign. First, if v is a unit length eigenvector, then obviously $-v$ is also a unit length eigenvector. The two airplanes on the left side in Figure 4.4 illustrate this problem.

Second, the axes are sorted according to their eigenvalues. If the eigenvalues for a model class are similar to each other, then the orientation of the principal axes depends on numerically unstable values. Figure 4.4 illustrates this problem. It shows two models of the same class, which are aligned differently: the eigenvector with largest eigenvalue of the model on the left-hand side points along the fuselage, whereas the eigenvector with largest eigenvalue of the model on the right-hand side points along the wings.

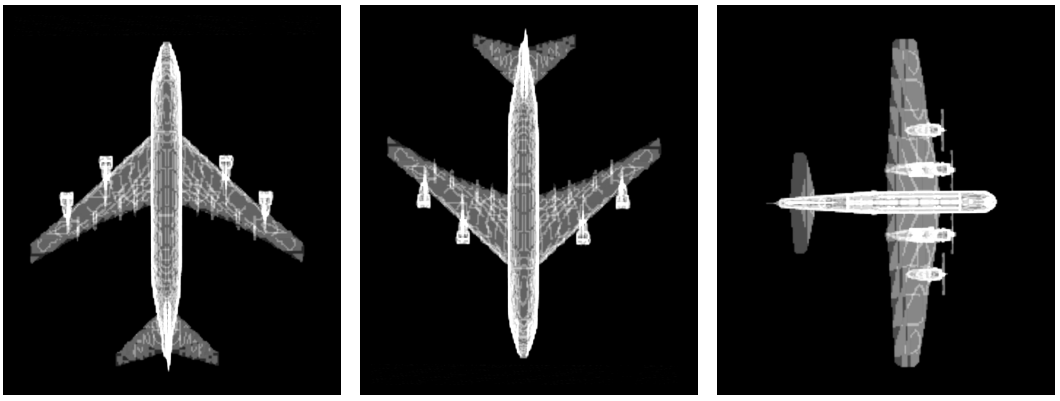

FIGURE 4.4: Two models of the same class which are aligned differently using standard Principal Component Analysis (PCA). As aligning the model using PCA does not guarantee a unique transformation, the axes of the aligned coordinate system may have to be swapped.

To circumvent this problem we transform the model to additional coordinate systems by swapping principal axes if their associated eigenvalues are similar. Furthermore, we transform the model to coordinate systems with inverted axis directions. By considering only valid transformations that preserve the coordinate system's orientation, the algorithm uses

- 4 coordinate systems, if all eigenvalues are sufficiently dissimilar,
- 8 coordinate systems, if two values are similar, and
- 24 coordinate systems, if all eigenvalues are similar.

The model is transformed to all potential coordinate systems. The unique, aligned test model X is the one with the greatest overlap with the average density of the training models.

4.3 Feature Vector Calculation

Based on the aligned training models the inverse distance models are computed. The volume of the distance transformed training samples $(D_i)_{\{i\in 1,\ldots,n\}}$ is defined by

$$\mathbf{v}(D_i, x, y, z) = \max\left\{\frac{cut - d(T_i, x, y, z)}{cut}, 0\right\}, \tag{4.1}$$

where $d(T_i, x, y, z)$ denotes the Euclidean distance of point (x, y, z) to the models surface. The value *cut* adjusts the rate of diffusion of the inverse distance transformation.

For the calculation of the inverted distance neither manifoldness nor watertightness is necessary. In fact, the inverse distance transformation could also be calculated for point sets. Figure 4.5 illustrates an aligned voxeled training model as well as its corresponding inverse distance transformed version.

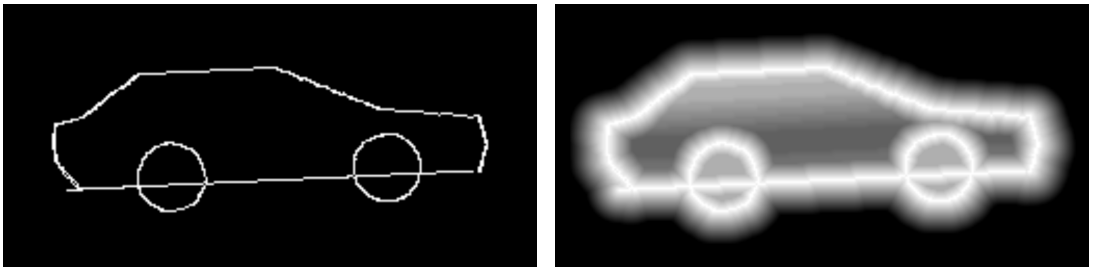

FIGURE 4.5: Visualization of the cross section of the aligned voxel model and the inverse distance model of a car. The resolution of the voxel model is $256 \times 256 \times 256$ and the cutoff value for the distance calculation is 16.

After the calculation of the inverse distance model, the model is split into a regular grid of cubic cells. The cross section of a inverse distance model is depicted in Figure 4.6. Let p denote the number of cells along one axis, then the total number of cells is p^3. Each cell has a side length s with $s = \frac{R}{p}$. If

$$\left(C_{(i,a,b,c)}\right)_{(i,a,b,c)\in(1\ldots n)\times(1\ldots p)^3} \tag{4.2}$$

denotes the family of the cells for model i, then the volume of the cell $C_{(i,a,b,c)}$ at position $(x, y, z) \in (1 \ldots s)^3$ is

$$\mathbf{v}(D_i, (a-1)\cdot s + x, (b-1)\cdot s + y, (c-1)\cdot s + z). \tag{4.3}$$

For each cell $C_{(i,a,b,c)}$ the histogram of inverse distances with k bins is calculated. The histogram is normalized by dividing through its largest member and serves as a k-dimensional feature vector of the cell. This feature vector of cell $C_{(i,a,b,c)}$ is denoted as $h_{(i,a,b,c)} \in [0, 1]^k$.

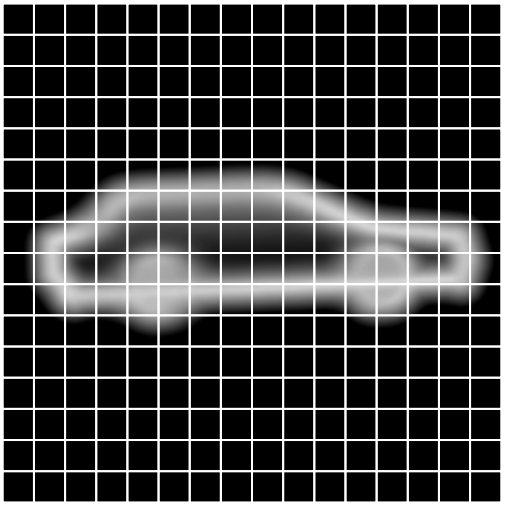

FIGURE 4.6: Cross section of an inverse distance model split into a grid of cubes. For each cube the histogram of inverse distances is calculated.

4.4 Matching using Kernel Density Estimation

The first matching method presented uses kernel density estimation. Based on the feature vectors $h_{(i,a,b,c)}$ a non-parametric density function for each cell position is estimated (a, b, c) using Gaussian kernel density estimation [11]. The density function for a cell at position (a, b, c) is

$$P(h', a, b, c) = \frac{1}{N} \sum_{i=1}^{n} \frac{\exp\left(-\frac{\|h' - h_{(i,a,b,c)}\|^2}{2\sigma^2}\right)}{(2\pi\sigma^2)^{b/2}}, \tag{4.4}$$

where σ represents the standard deviation of the Gaussian kernel. The standard deviation acts as a smoothing factor.

Usually the standard deviation can be estimated easily using appropriate estimation methods [55]. However, in this case, at some positions all the features are exactly the same. This situation often occurs at border cells, where the inverse distance to the surface is zero everywhere. To solve this problem, the standard deviation σ has been set to an empirically determined value.

Matching is performed on an aligned test model X. Therefore the inverse distance transformation is calculated. Like in the training phase, the inverse distance transformed model is partitioned into cells and the corresponding histograms $(h'_{(a,b,c)})_{(a,b,c)\in(1\ldots p)^3}$ are calculated. Using the density functions for each cell, the probability of a sample object belonging to a learned class can be approximated:

Let X be a test model and $(h'_{(a,b,c)})_{(a,b,c)\in(1\ldots p)^3}$ denote the feature vectors of the test model, then the joint probability of model X belonging to the learned class is

$$\prod_{(a,b,c)\in 1\ldots p^3} P(h'_{(a,b,c)}, a, b, c). \tag{4.5}$$

We call this matching method the "histogram of inverted distances - kernel density estimation" (HID-KDE) method.

4.5 Matching using Support Vector Machines

The second matching method uses support vector machines to estimate the distribution of the training models in the feature space. The idea for this algorithm is based on the method introduced by Chen et al. [56]. For this method a single feature vector for each training model is shaped by concatenating the cell feature vectors $h_{(i,a,b,c)}$. The new feature vector is called f_i and is in $[0, 1]^{(kp^3)}$.

The feature vectors are used to train a one-class support vector machine. This special type of vector machine has been introduced by Schölkopf et al. [57] and has the advantage that only positive examples are needed in the training phase.

Let X be a test model and $f' \in [0, 1]^{(kp^3)}$ denote the feature vector of the test model, then the one-class SVM model returns a similarity value that indicates the likelihood of the test model belonging to the learned class. This matching method is called the "histogram of inverted distances - support vector machine" (HID-SVM) method.

4.6 An Approach to Improve Retrieval Results using Markov Random Fields

To further improve the retrieval quality a Markov random field is constructed and the joint probability is calculated on the Markov random field. The assumption is, that by incorporating pairwise potentials long range consistency of the retrieved model can be rewarded. The structure of the 3D Markov random field is depicted in Figure 4.7. The cells are represented by the vertices V in the Markov random field and cell neighborhoods are represented through edges E. The letter 'X' in the figure represents a sample 3D model, for which we like to calculate the probability of belonging to the learned class.

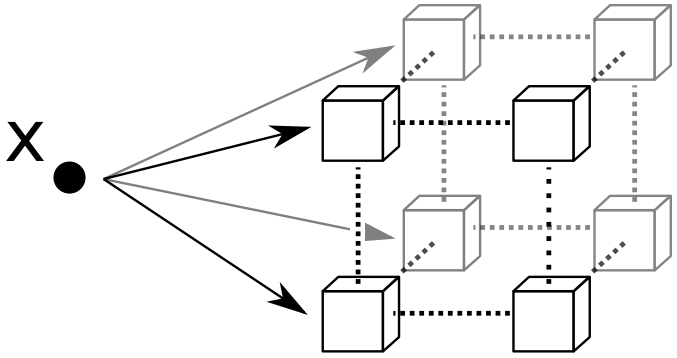

FIGURE 4.7: The structure of the 3D Markov random field. The cells of the 3D model are represented as vertices. Neighbouring vertices are connected with an edge.

We define the model for our Markov random field as follows:

$$p(x) = \frac{1}{Z(x)} \prod_{v \in V} \psi^1(v) \prod_{(v,w) \in E} \psi^2(v,w) \tag{4.6}$$

where $Z(x)$ represents the partition function which assures, that $P(x)$ is a valid density function. The instance potentials are modeled using the function $\psi^1(x)$. The pairwise potential functions are modeled using $\psi^2(v,w)$.

Let us assume that $(h'_{(a,b,c)})_{(a,b,c)\in(1\ldots p)^3}$ denotes the calculated cell histograms of the 3D model X. For the instance potentials, the learned cell histograms are used. The instance potentials is defined as $\psi^1(v) = \exp(P(h'_{(a,b,c)}, a, b, c))$, where $h'_{(a,b,c)}$ denotes the cell histogram for the vertex v. The pairwise potentials are defined as $\psi^2(v,w) = \exp(|\psi^1(v) - \psi^1(w)|)$.

4.7 An Approach to Improve Retrieval using Diffusion Processes

The second approach to improve retrieval results is using diffusion processes. Using matching methods for 3D models, the similarity or affinity between two models can be calculated. Assuming that N is the number of 3D models, a square $N \times N$ matrix of all 3D models can be calculated. This matrix will be called the affinity matrix A here.

Retrieving 3D models similar to the i-th model using the affinity matrix A only, is done by extracting the i-th row in the affinity matrix and sorting this row by its affinity

values. However, doing so ignores the structure of the underlying data manifold [58]. Diffusion processes re-evaluate the affinities of all models in the context of all other elements. This is done by diffusing the affinity values through the graph described by the affinity matrix [58]. Donoser et al. [58] have recently surveyed diffusion processes for retrieval. They mention that diffusion processes have the ability to significantly improve applications like retrieval. Figure 4.8 shows the affinity values before (left) and after (right) applying a diffusion process.

FIGURE 4.8: The figure shows the affinity values before (left) and after (right) applying a diffusion process. Using only pairwise affinities is not sufficient to capture the intrinsic structure of the data manifold. Applying a diffusion process spreads affinities through the graph described by the affinity matrix [58].

One of the best known diffusion algorithm is the Google page rank algorithm [59]. For the page rank algorithm an affinity matrix A is needed, which contains all pairwise affinity values. Each row of the affinity matrix A is divided by its sum, which gives the row stochastic matrix P. The matrix P can be interpreted as a transition matrix for randomly walking on a complete graph of size N.

Assuming we want to retrieve 3D models similar to the model with id j, the N-dimensional probability vector f_0 must be initialized with:

$$f_0(i) = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \tag{4.7}$$

The probability vector, which contains the final affinities for all other 3D models can be iterative calculated using:

$$f_{(t+1)} = f_t P, \tag{4.8}$$

This update operation is performed iteratively until convergence. In this thesis a slightly modified diffusion algorithm will be used, where diffusion is constrained to the k-nearest neighbors in the affinity matrix. The update step changes to $W_{(t+1)} = PW_tP^T$, where W denotes the new affinity matrix, which is initialized with P. This approach is called "Locally constrained diffusion process" (LCDP), it has been introduced by Yang et al. [60].

Chapter 5

Evaluation

5.1 Evaluation Setup

The matching methods were tested against the Princeton shape benchmark [7] at the base classification setting. The base classification setting is the most fine-grained classification, it divides the 907 models into 90 classes. For the evaluation the generative models "commercial airplane" and "sedan car" were modeled. The sedan car test case has 10 positive examples and the commercial airplane test case has 11 positive examples.

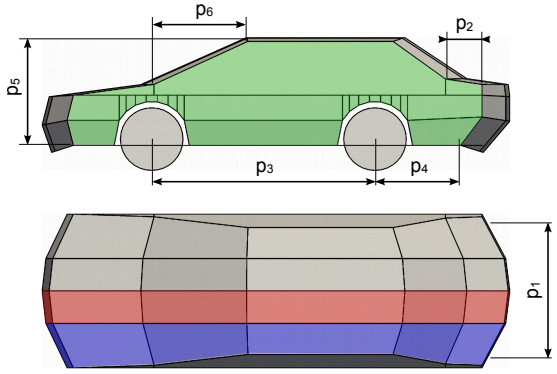

FIGURE 5.1: The generative description of the "sedan car" model has six degrees of freedom. These parameters $(p_1, \ldots, p_6)$ define the car's outer shape.

5.1.1 Sedan Car

The sedan car class was modeled using a JavaScript file with ≈ 350 lines of code (LOC). The script has six parameters, that control the dimensions of the car. The effect of the parameters on the main dimensions of the car is visualized in Figure 5.1.

Other dimensions, such as the length of the top of the car are either derived by combining multiple parameters or set to a constant value. The car is modeled by first defining only the side of the car, which is colored in green in Figure 5.1. The side is then extruded twice using tapered extrusions with different angles. The first extrusion creates the blue parts and the second one creates the red parts in Figure 5.1. By mirroring the side and the two extrusions around its center plane the whole car is created. The tires are modeled using cylinders. The extrusion as well as the creation of a cylinder are

implemented as JavaScript functions. Using functions allows simple reuse of code parts in other generative models.

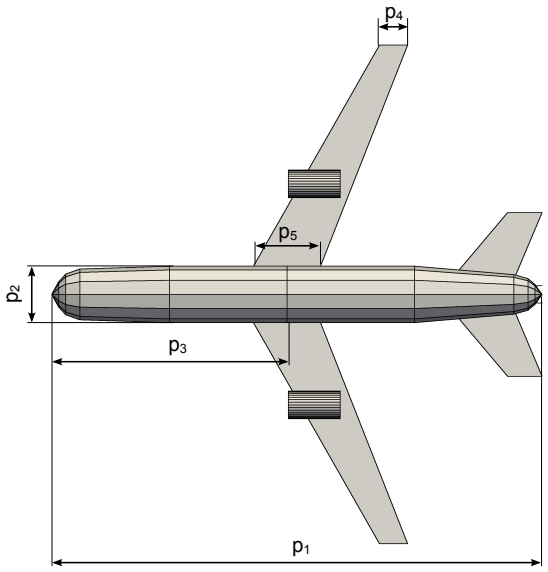

FIGURE 5.2: The class of commercial airplanes is defined by a generative model that takes five parameters. The effect of its parameters is visualized in this construction plan.

5.1.2 Commercial Airplane

The commercial airplane class was modeled in JavaScript as well ($\approx$ 250 LOC). The script has five parameters, that control the dimensions of the airplane. The effect of the parameters on the main dimensions of the airplane is visualized in Figure 5.2. Other dimensions, such as the angle or the length of the wings are either derived by combining multiple parameters or set to a constant value. The fuselage of the airplane is created using a rotational surface which is then slightly deformed to model the cockpit and the aft fuselage. The jet engines are modeled using cylinders. Again the rotational surface is implemented as a JavaScript function, which could be reused in other models.

5.2 Evaluation of the Histogram of Inverted Distances - Kernel Density Estimation Algorithm

This method relies on six parameters. The values for the retrieval parameters were evaluated empirically. All parameters and their corresponding values are displayed in Table 5.1.

The benchmark was executed on a modern computer with an Intel *i*7 950 CPU and 12 GB Ram. The generation and voxelization of 64 models for both model classes took 103 seconds. The learning phase took 292 seconds. The actual retrieval process for both classes took about 4.75 hours. That is 9.41 seconds in average per test model. However most of the time is spent for converting and aligning the models and calculating the descriptors, the actual calculation of the similarity takes 0.3 seconds on average. The effectiveness of a retrieval method is evaluated using the precision-recall graph, like

Parameter		Value
n	Number of training samples	64
R	Resolution of the voxel models	256
p	Number of cells in one direction	16
σ	Standard deviation for the Gaussian kernel density estimation	$^1/_4$
cut	Cutoff value of the inverted distance calculation	16
k	Number of histogram bins	8

TABLE 5.1: The parameters and their corresponding values listed in this table have been used for benchmarking our method.

presented in Section 2.1.2. Figure 5.3 shows the precision-recall graphs for the classes "sedan car" and "commercial airplane".

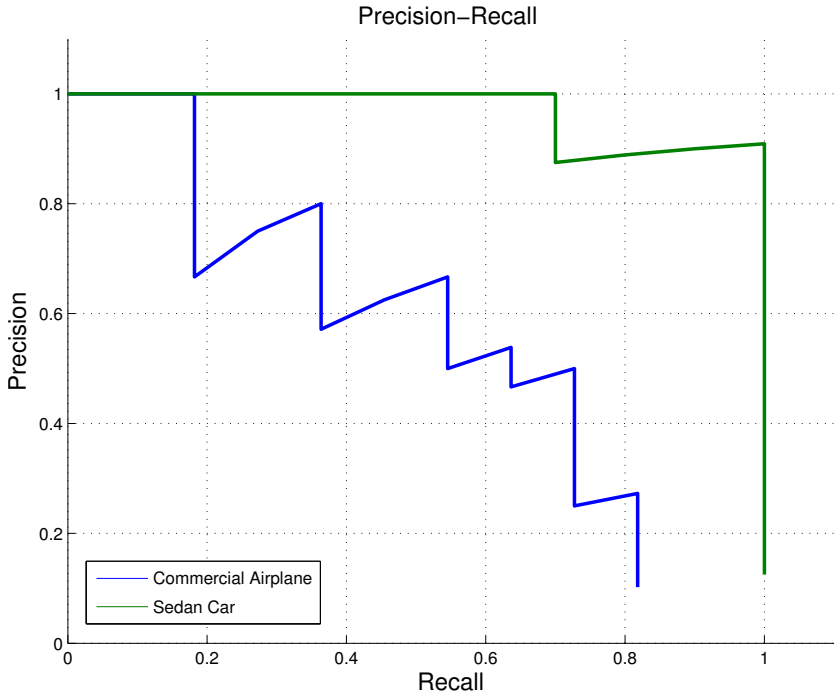

FIGURE 5.3: The precision-recall graph shows the benchmark results of the classes "sedan car" and "commercial airplane" using the histogram of inverted distances algorithm.

The graph indicates, that the method is able to find similar objects to the given generative models. The retrieval results for the class sedan car are almost perfect. However the commercial airplane class performs significantly worse. This is because the generative model is not detailed enough to differentiate between different airplane classes (commercial, fighter, biplane etc.) as can be seen in Figure 5.5. Figure 5.4 shows the top 16 retrieval results for the class sedan car.

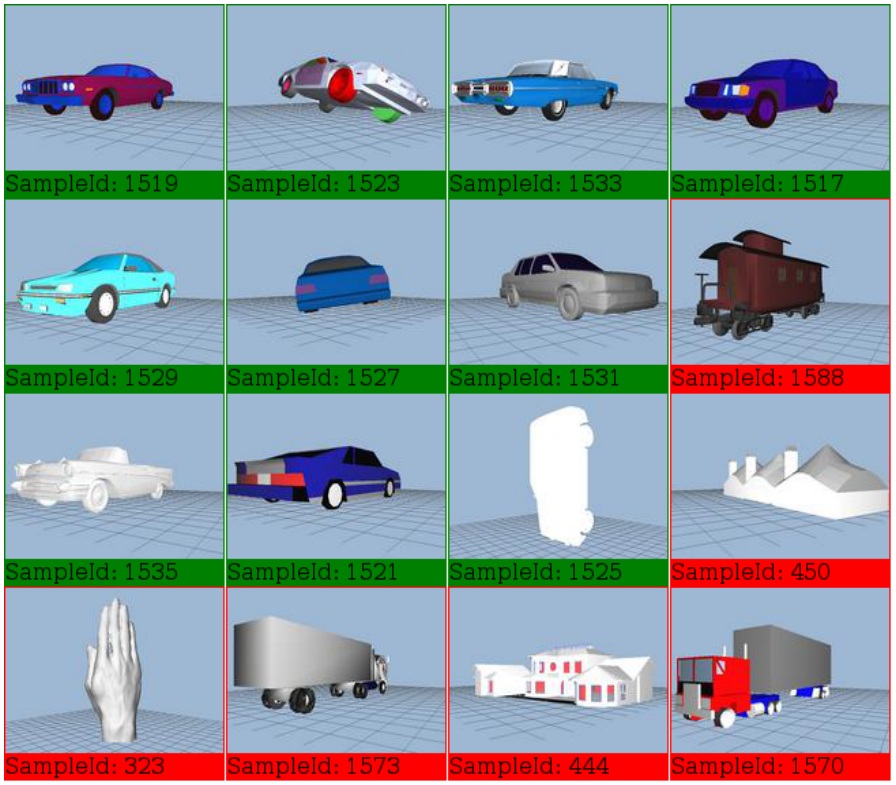

FIGURE 5.4: The top 16 retrieval results for the class sedan car.

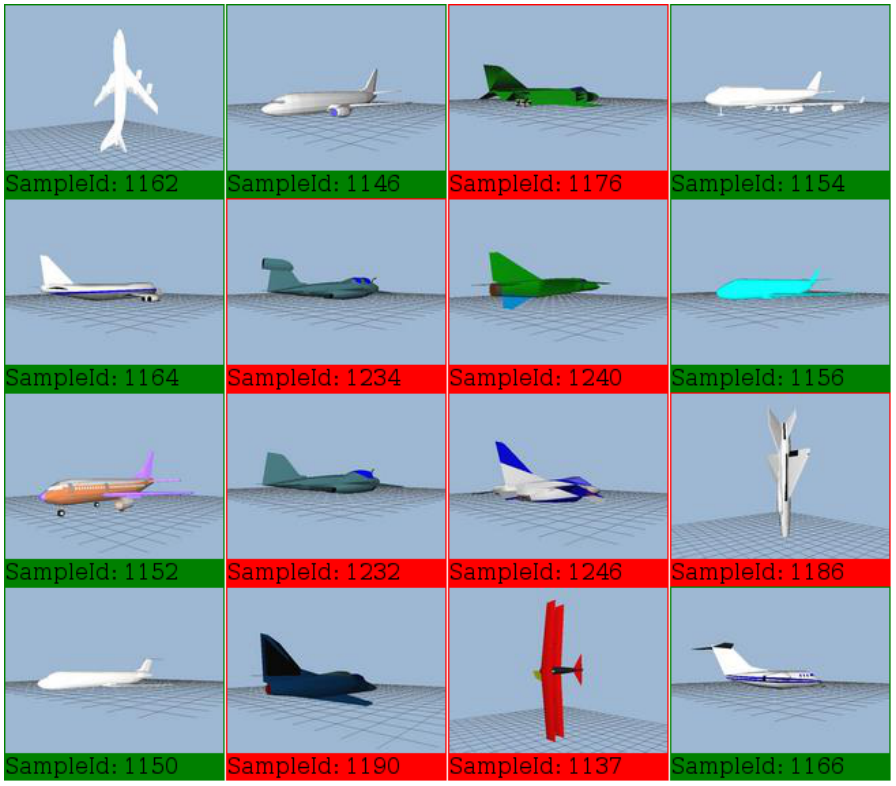

FIGURE 5.5: The top 16 retrieval results for the class commercial airplane. Even though all models in the top results are airplanes, there are many false positives. That is because the Princeton benchmark distinguishes between many different airplane classes (commercial, fighter, biplane etc.), and we only search for commercial airplanes.

5.3 Evaluation of the Histogram of Inverted Distances - Support Vector Machine Algorithm

The parameters for the evaluation were the same as for the histogram of inverted distances algorithm, presented in Table 5.1. During the training phase, training the support vector machines adds another 0.1 seconds per class, which is a negligible amount of time. In the recognition phase the calculation of the similarity per model declines from 0.3 to 0.006 seconds when using support vector machines instead of kernel density estimation.

The effectiveness of a retrieval method is evaluated using the precision-recall graph, like presented in Section 2.1.2. Figure 5.6 shows the precision-recall graphs for the classes "sedan car" and "commercial airplane".

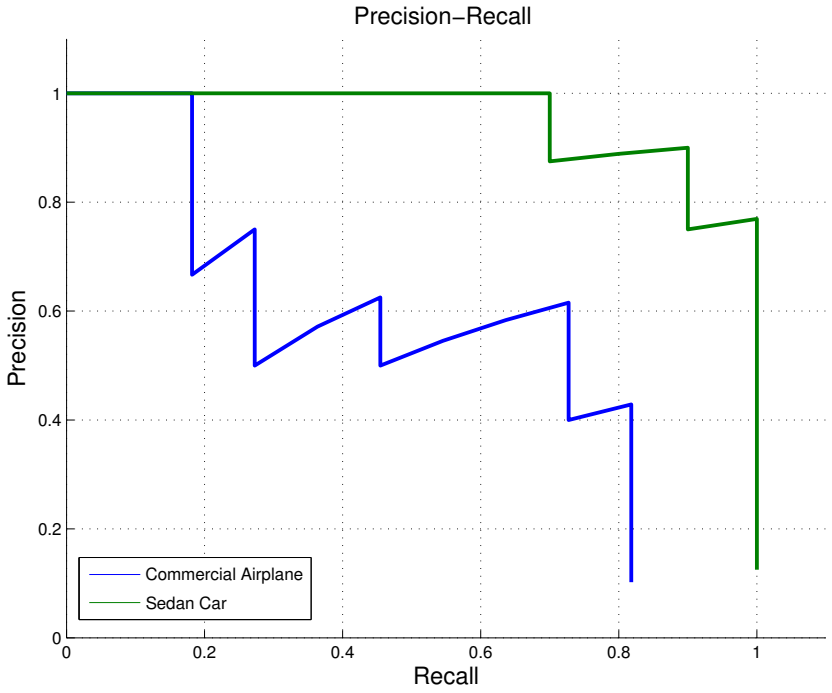

FIGURE 5.6: The precision-recall graph shows the benchmark results of the classes "sedan car" and "commercial airplane" using the histogram of inverted distances - support vector machine algorithm.

The results are very similar to the kernel density estimation algorithm. Again the results for class sedan car are almost perfect. The results for the class airplane are however better compared to the kernel density estimation algorithm. The top 16 results for the sedan car class and the commercial airplane class are shown in Figures 5.8 and 5.7 respectively.

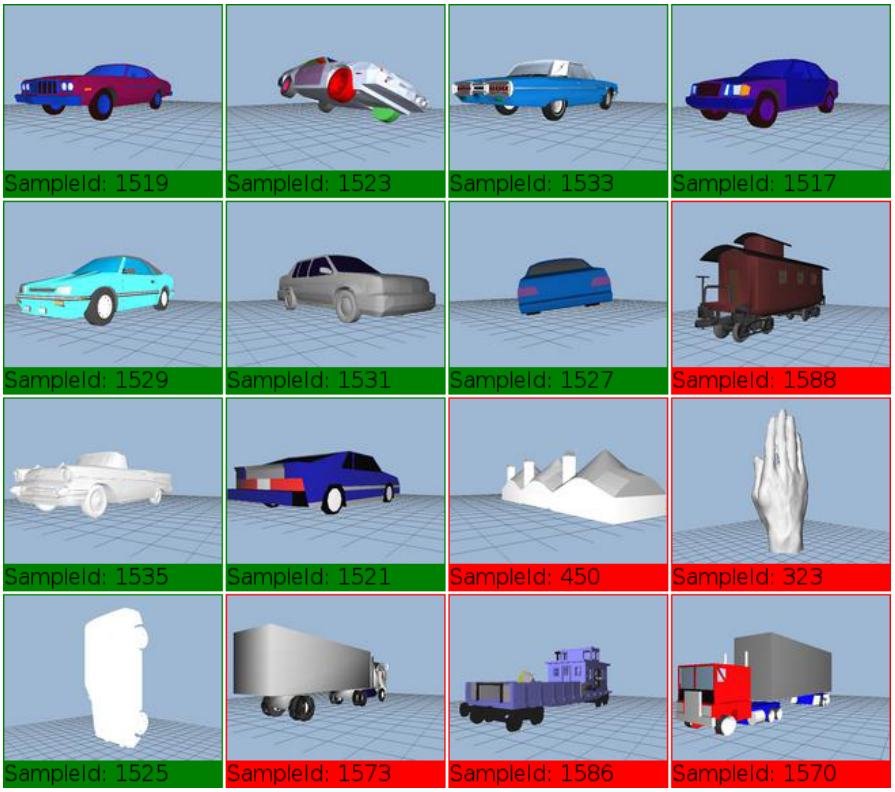

FIGURE 5.7: The top 16 retrieval results for the class sedan car.

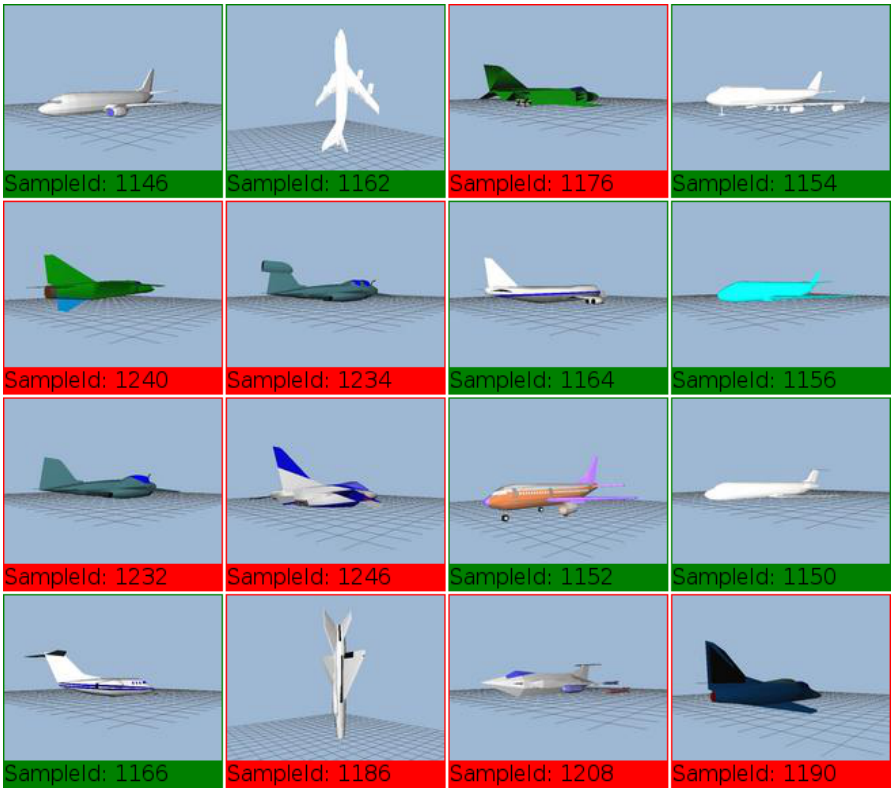

FIGURE 5.8: The top 16 retrieval results for the class commercial airplane. Even though all models in the top results are airplanes, there are many false positives. That is because the Princeton benchmark distinguishes between many different airplane classes (commercial, fighter, biplane etc.), and we only search for commercial airplanes.

5.4 Evaluation of the Markov Random Field Approach

The Markov random field approach additionally requires the calculation of the joint probability on the Markov random field, as described in section 4.6. This adds another 0.0005 seconds, which is again negligible.

The effectiveness of a retrieval method is evaluated using the precision-recall graph, like presented in Section 2.1.2. Figure 5.9 shows the precision-recall graphs for the classes "sedan car" and "commercial airplane".

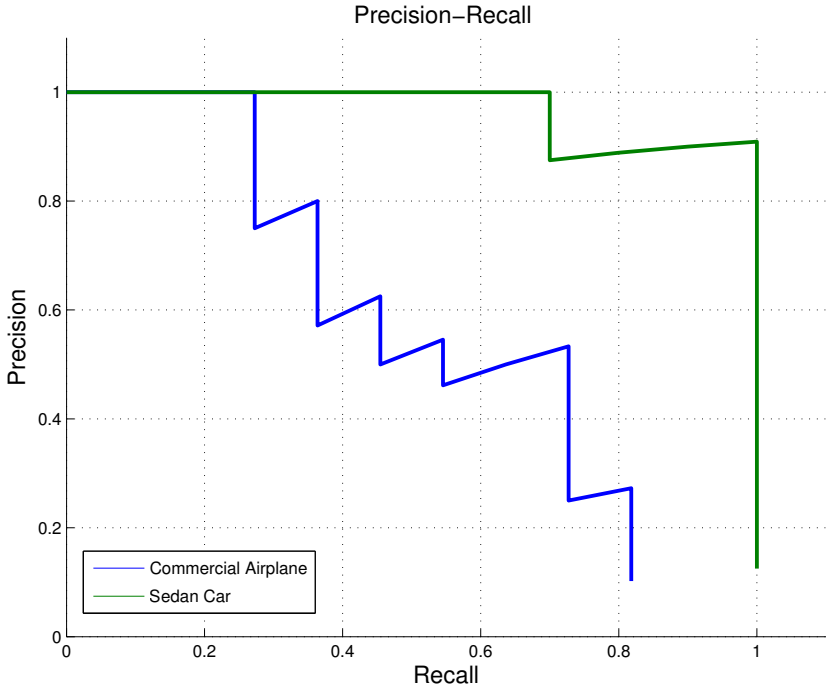

FIGURE 5.9: The precision-recall graph shows the benchmark results of the classes "sedan car" and "commercial airplane" using the histogram of inverted distances - Markov random field algorithm.

The results are slightly better, compared to the kernel density estimation approach, as can be seen in Figure 5.9. The top 16 results for the sedan car class and the commercial airplane class are shown in Figures 5.11 and 5.10 respectively.

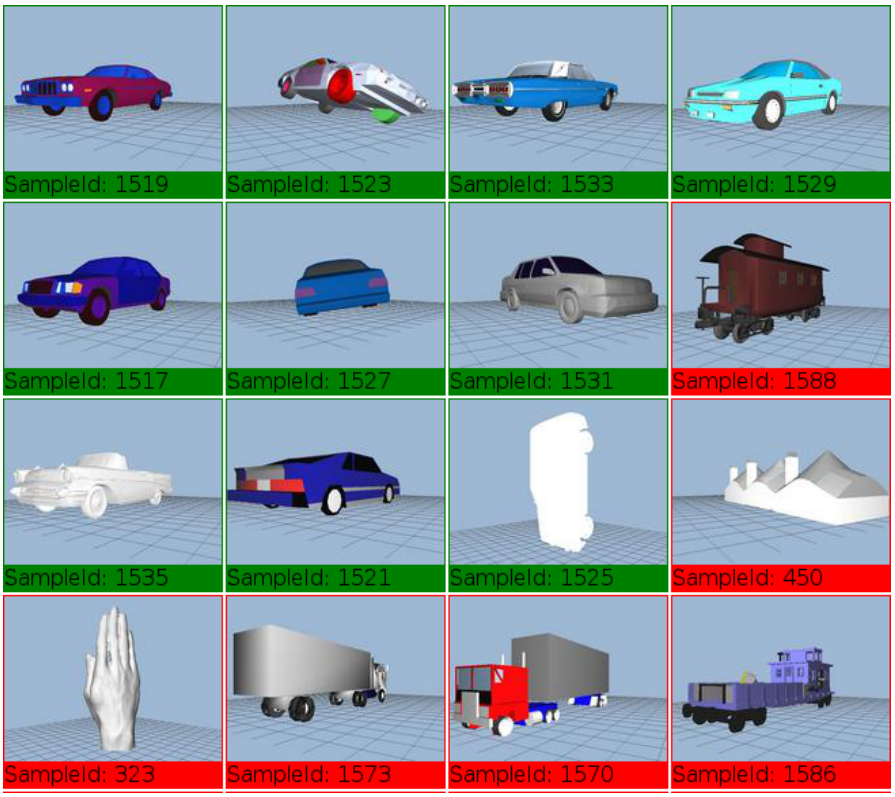

FIGURE 5.10: The top 16 retrieval results for the class sedan car.

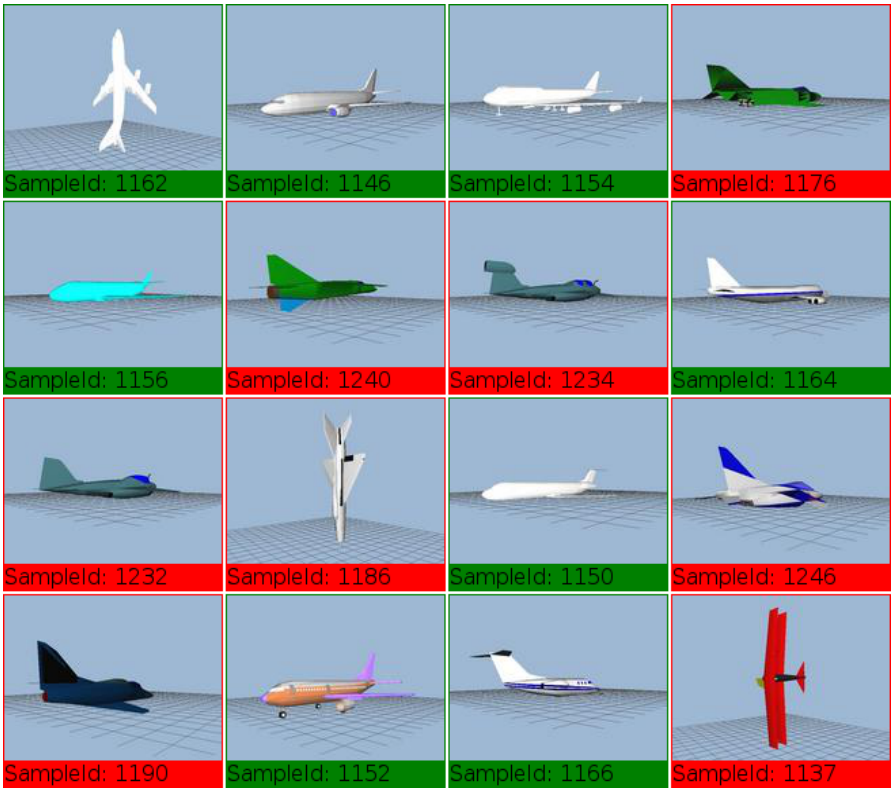

FIGURE 5.11: The top 16 retrieval results for the class commercial airplane. Even though all models in the top results are airplanes, there are many false positives. That is because the Princeton benchmark distinguishes between many different airplane classes (commercial, fighter, biplane etc.), and we only search for commercial airplanes.

5.5 Evaluation of the Diffusion Process Approach

To apply diffusion processes in the field of retrieval, a $N \times N$ matrix containing all pairwise similarities is needed. This matrix is called the affinity matrix. The pairwise similarities are calculated by comparing the cell histograms of the models. Like in Johnson et al. [36], the similarity between two cells is given by the correlation coefficient. The similarity between two models is given by the sum of the cells similarity.

The construction of the affinity matrix for the whole benchmark requires 1814^2 model comparisons. One comparison of two models took 0.34 seconds on average, which results in 310.77 hours for the whole benchmark. To speed up the calculation of the affinity matrix, the workload was distributed to seven similar computers. Thereby calculation time was reduced to two days. From the affinity matrix for the whole benchmark, the affinity matrix for the test set was extracted. This affinity matrix is depicted in Figure 5.13 using a the color map depicted in Figure 5.12. Black color indicates low similarity, whereas white color indicates high similarity.

FIGURE 5.12: The Figure shows the color map used to visualize the affinity matrix. Black color indicates low similarity, whereas white color indicates high similarity.

The only parameter for applying the locally constrained diffusion process, is the number of nearest neighbors k to consider. The value k was set to 4, as this led to the best results. The application of the diffusion process takes 1.1 seconds until convergence. The resulting affinity affinity matrix is depicted in Figure 5.14.

FIGURE 5.13: Affinity matrix before the application of the diffusion process.

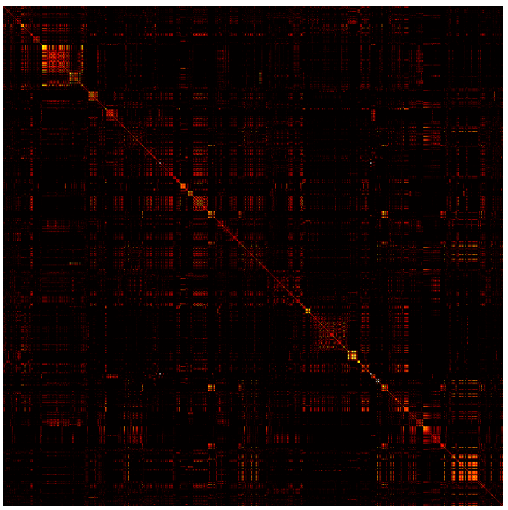

FIGURE 5.14: Affinity matrix after the application of the diffusion process.

The effect of the diffusion process on the retrieval effectiveness is visualized with the averaged eleven point precision-recall graph in Figure 5.15. The green precision-recall graph shows the results before the application of the diffusion process and the blue graph shows the results after the application of the diffusion process. It can be seen that the precision slightly decreases at lower recall values, but increases at recall values of about 0.4.

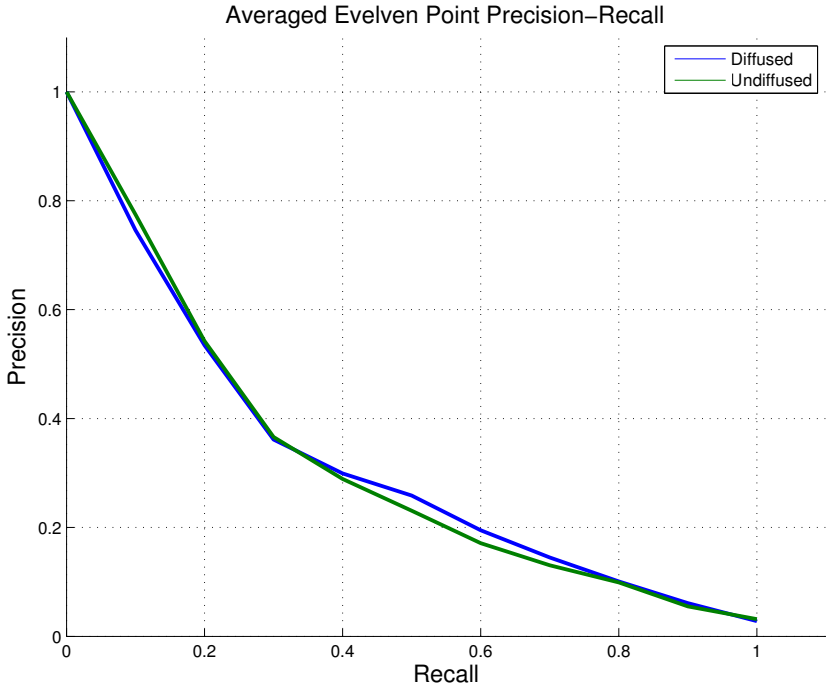

FIGURE 5.15: Eleven point precision-recall plot. The green precision-recall graph shows the results before the application of the diffusion process and the blue graph shows the results after the application of the diffusion process.

Figure 5.16 and Figure 5.17 show how the diffusion process can improve retrieval performance. Before the application of the diffusion process (Figure 5.16) only four faces are found in the top 16 results. By apply the locally constrained diffusion process eleven faces can be found in the top 16 results.

However this process might also reduce the retrieval performance as shown in Figure 5.18 and in Figure 5.19. Figure 5.18 shows that seven chessboards can be found in the top 16 retrieval results. However in the three nearest neighbors two false positive models are found. Those objects have the shape of a circular disc. After diffusion the retrieval performance reduces, because many objects similar to circular discs are present in the top 16 retrieval results.

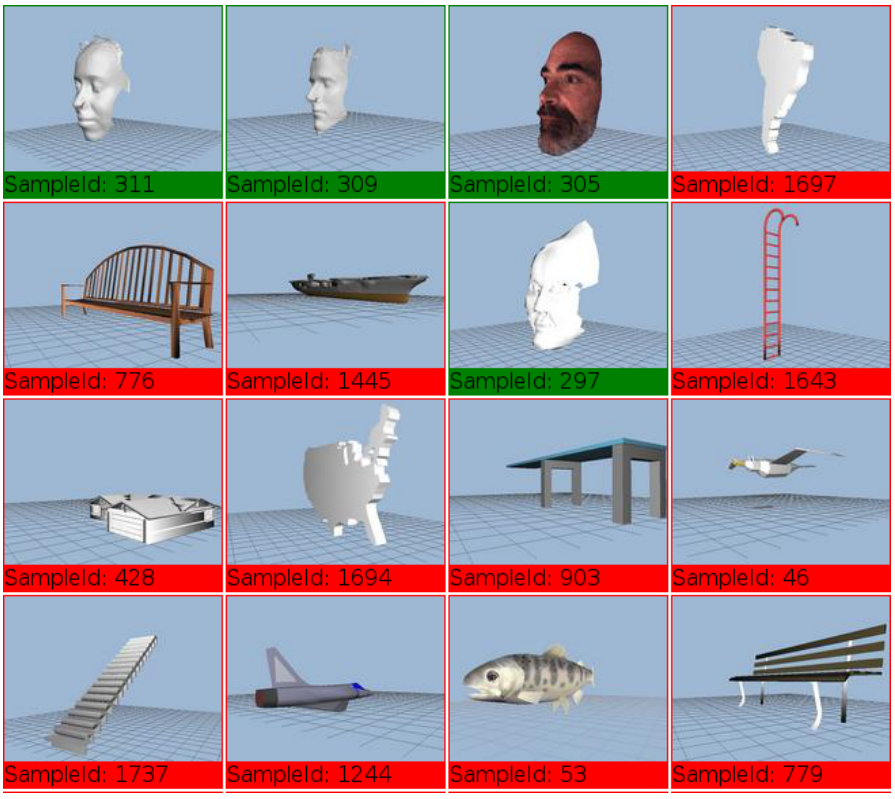

FIGURE 5.16: The Figure shows the top 16 retrieval results before applying diffusion. Only four faces can be found in the top 16 retrieval results.

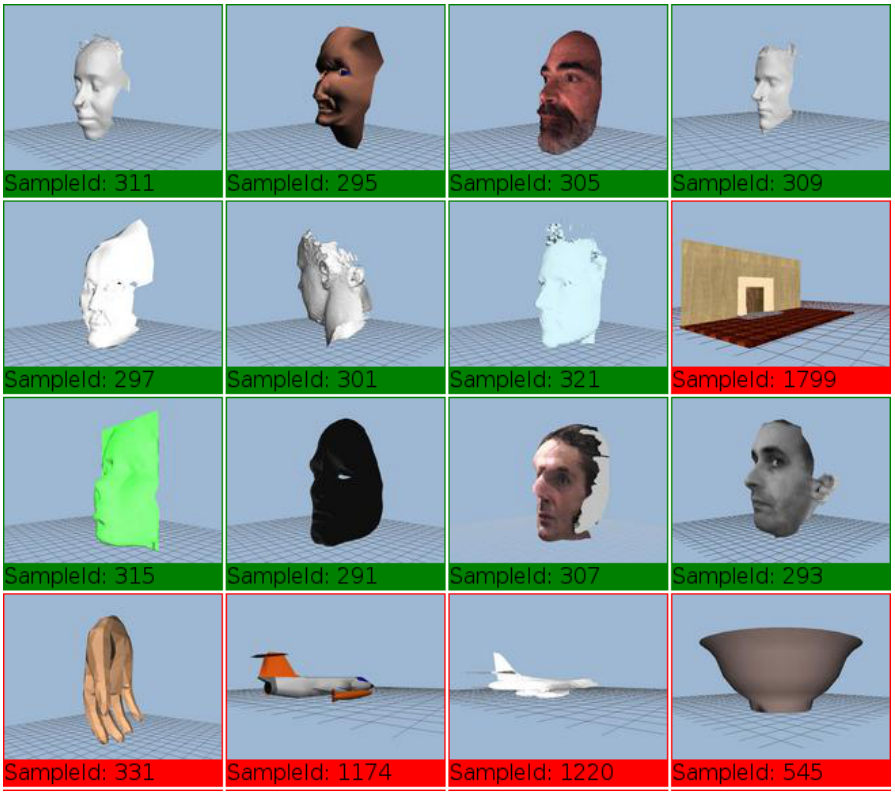

FIGURE 5.17: The Figure shows the top 16 retrieval results after applying diffusion. Eleven faces can be found in the top 16 retrieval results.

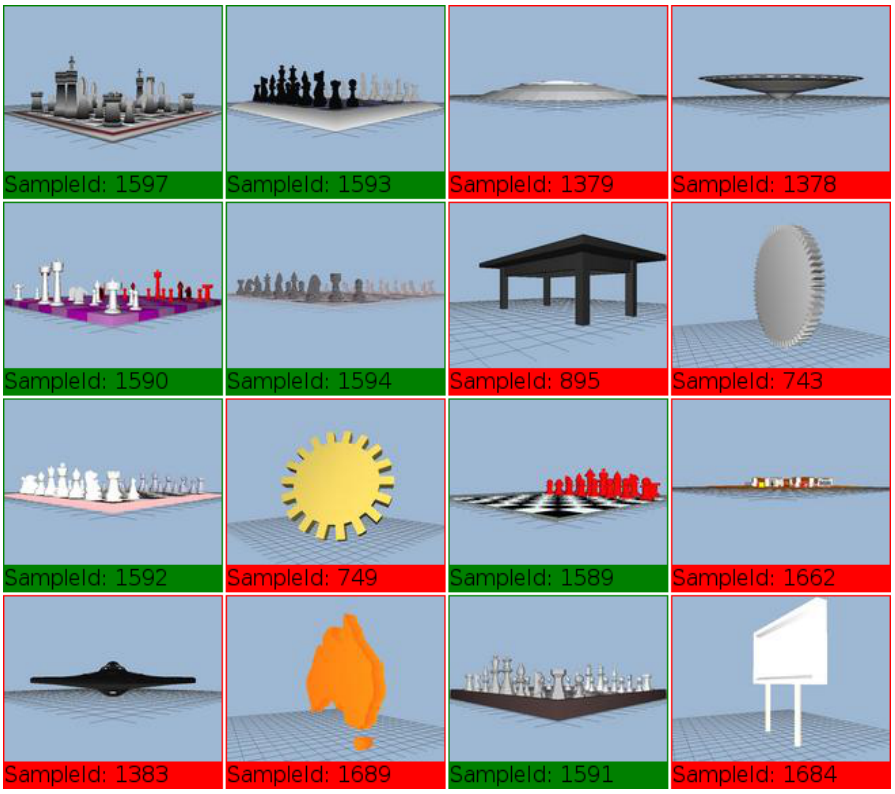

FIGURE 5.18: The Figure shows the top 16 retrieval results before applying diffusion. Seven chessboards are found in the top 16 retrieval results.

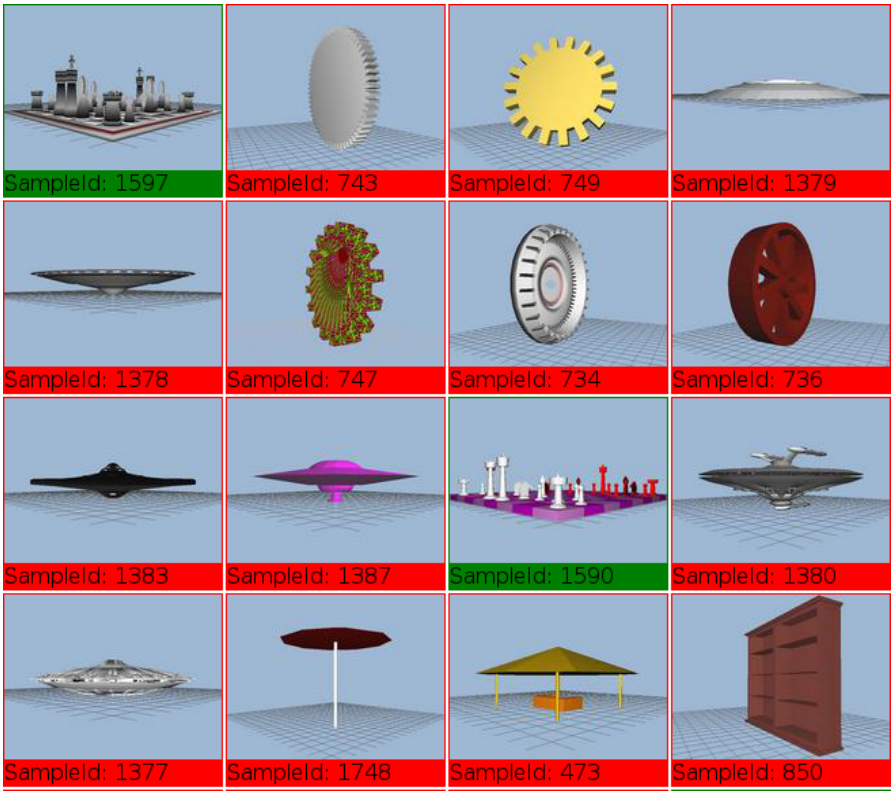

FIGURE 5.19: The Figure shows the top 16 retrieval results after applying diffusion. The retrieval quality reduces, because many objects similar to circular discs are present in the top 16 retrieval results.

Chapter 6

Conclusion

In this thesis I have presented a new approach to perform content-based retrieval of 3D shapes based on generative modeling techniques. The generative models are used to describe 3D model classes, respectively, 3D shape spaces. In the training phase, the shape spaces are sampled randomly. In this way, no "real" training data is needed a priori.

Two retrieval algorithms, the "histogram of inverted distances - kernel density estimation" method and the "histogram of inverted distances - support vector machine" method, have been developed. Both methods use a voxel representation, PCA alignment and inverse distance transformations on a grid. For each cell in the grid the histogram of inverted distances is calculated. Furthermore two approaches to improve retrieval quality have been developed. The first approach uses Markov random fields and the second one uses diffusion processes.

Kernel Density Estimation The "histogram of inverted distances - kernel density estimation" method uses kernel density estimation to learn a non-parametric density function for each cell. In the recognition phase, the cell histograms for the test model are calculated. The similarity between a cell of a test model and a learned cell is estimated using the corresponding learned density function. The similarity of the whole model is given by the product of all cell similarities.

Support Vector Machine The second algorithm is called "histogram of inverted distances - support vector machine". It trains a one-class support vector machine to estimate the distribution of training samples in the feature space. For that the cell histograms are concatenated to a global feature vector for each training model. In the recognition phase, the global feature vector is calculated for each test model and the support vector machine is used to estimate, whether the test samples belong to the distribution of training samples.

Markov Random Field In the first approach to improve retrieval results, a Markov random field is constructed and the joint probability is calculated. It has been shown, that the Markov random field approach could slightly improve retrieval performance.

Diffusion Processes The second approach uses diffusion processes, to take the underlying structure of the data manifold into account, similar to the Google page rank algorithm. It has been shown that retrieval performance slightly decreased at low recall rates, but increased at recall rates of about 0.4.

As a consequence, the contribution to 3D shape retrieval is a novel indexing and retrieval method based on generative models and histograms of inverted distances. I have shown that it is possible to train the retrieval methods using solely generative models, by evaluating the method using the Princeton benchmark.

As a benefit, this technique eliminates the cold start problem in the training phase. A generative description implemented in a few lines of code is sufficient to generate a reasonable training set. Furthermore, it has been shown that the histogram of inverted distances can be used as a feature vector for spatial data.

Bibliography

[1] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262–282, 2007.

[2] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, July 2008. ISBN 9780521865715.

[3] Michael S. Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2(1):1–19, February 2006. ISSN 1551-6857. doi: 10.1145/1126004.1126005. URL `http://doi.acm.org/10.1145/1126004.1126005`.

[4] Patrick Min, Michael Kazhdan, and Thomas Funkhouser. A comparison of text and shape matching for retrieval of online 3D models. In Rachel Heery and Liz Lyon, editors, *Research and Advanced Technology for Digital Libraries*, number 3232 in Lecture Notes in Computer Science, pages 209–220. Springer Berlin Heidelberg, January 2004. ISBN 978-3-540-23013-7, 978-3-540-30230-8. URL `http://link.springer.com/chapter/10.1007/978-3-540-30230-8_20`.

[5] J. W. H. Tangelder and R. C. Veltkamp. A survey of content based 3D shape retrieval methods. *Multimedia Tools and Applications*, 39(3):441–471, 2008. URL `http://www.springerlink.com/index/988121hl24227239.pdf`.

[6] Christian Böhm, Stefan Berchtold, and Daniel A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys (CSUR)*, 33(3):322–373, 2001. URL `http://dl.acm.org/citation.cfm?id=502809`.

[7] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. *Shape Modeling Applications, 2004. Proceedings*, pages 167–178, 2004. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1314504`.

[8] 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. URL `http://www.sciencedirect.com/science/article/pii/S001044850600100X`.

[9] Bo Li, Afzal Godil, Masaki Aono, X. Bai, Takahiko Furuya, L. Li, R. Lpez-Sastre, Henry Johan, Ryutarou Ohbuchi, and Carolina Redondo-Cabrera. SHREC'12 track: Generic 3D shape retrieval. In *Proceedings of the 5th Eurographics conference on*

3D Object Retrieval, pages 119–126, 2012. URL `http://dl.acm.org/citation.cfm?id=2381218`.

[10] Paul Gerhard Hoel and Stone . *Introduction to probability theory*. Houghton Mifflin, Boston, 1971. ISBN 039504636X 9780395046364.

[11] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, October 2007. ISBN 0387310738.

[12] Philip Schneider and David H. Eberly. *Geometric Tools for Computer Graphics*. Morgan Kaufmann, October 2002. ISBN 9780080478029.

[13] Eric W. Weisstein. Maximal clique – from wolfram MathWorld. URL `http://mathworld.wolfram.com/MaximalClique.html`.

[14] Alan H Watt. *3D computer graphics*. Addison-Wesley, Harlow, England; Reading, Mass., 2000. ISBN 0201398559 9780201398557.

[15] Mario Botsch. *Polygon mesh processing*. A K Peters, Natick, Mass., 2010. ISBN 9781568814261 1568814267.

[16] Lutz Kettner. Designing a data structure for polyhedral surfaces. In *Proceedings of the fourteenth annual symposium on Computational geometry*, SCG '98, pages 146–154, New York, NY, USA, 1998. ACM. ISBN 0-89791-973-4. doi: 10.1145/276884.276901. URL `http://doi.acm.org/10.1145/276884.276901`.

[17] Les A. Piegl and Wayne Tiller. *The Nurbs Book*. Springer, January 1997. ISBN 9783540615453.

[18] Ian Stroud. *Boundary Representation Modelling Techniques*. Springer, December 2006. ISBN 9781846286162.

[19] Matt Pharr and Randima Fernando. *GPU gems 2: programming techniques for high-performance graphics and general-purpose computation*. Addison-Wesley, Upper Saddle River, NJ, 2005. ISBN 0321335597 9780321335593.

[20] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. *ACM Trans. Graph.*, 25(3):614–623, July 2006. ISSN 0730-0301. doi: 10.1145/1141911.1141931. URL `http://doi.acm.org/10.1145/1141911.1141931`.

[21] Carlos A. Vanegas, Daniel G. Aliaga, Peter Wonka, Pascal Müller, Paul Waddell, and Benjamin Watson. Modelling the appearance and behaviour of urban spaces. In *Computer Graphics Forum*, volume 29, pages 25–42, 2010.

[22] Przemyslaw Prusinkiewicz. Graphical applications of l-systems. In *Proceedings of graphics interface*, volume 86, pages 247–253, 1986.

[23] George Stiny. Introduction to shape and shape grammars. *Environment and planning B*, 7(3):343–351, 1980.

[24] Sven Havemann and Dieter W Fellner. *Generative mesh modeling*. PhD thesis, University of Braunschweig-Institute of Technology, 2005.

[25] Christoph Schinko, Martin Strobl, Torsten Ullrich, and Dieter W. Fellner. Scripting technology for generative modeling. *International Journal On Advances in Software*, 4(3 and 4):308–326, 2012. URL `http://www.thinkmind.org/index.php?view=article&articleid=soft_v4_n34_2011_6`.

[26] B. Bustos, D. Keim, D. Saupe, and T. Schreck. Content-based 3D object retrieval. *IEEE Computer Graphics and Applications*, 27(4):22–27, 2007. ISSN 0272-1716. doi: 10.1109/MCG.2007.80.

[27] Hongbo Fu, Daniel Cohen-Or, Gideon Dror, and Alla Sheffer. Upright orientation of man-made objects. *ACM Trans. Graph.*, 27(3):42:1–42:7, August 2008. ISSN 0730-0301. doi: 10.1145/1360612.1360641. URL `http://doi.acm.org/10.1145/1360612.1360641`.

[28] 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.

[29] Eric Paquet, Marc Rioux, Anil Murching, Thumpudi Naveen, and Ali Tabatabai. Description of shape information for 2-d and 3-d objects. *Signal Processing: Image Communication*, 16(1):103–122, 2000.

[30] Ilias Kolonias, Dimitrios Tzovaras, Sotiris Malassiotis, and Michael G Strintzis. Fast content-based search of vrml models based on shape descriptors. *Multimedia, IEEE Transactions on*, 7(1):114–126, 2005.

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

[32] A Ben Hamza and Hamid Krim. Geodesic matching of triangulated surfaces. *Image Processing, IEEE Transactions on*, 15(8):2249–2258, 2006.

[33] Yi Liu, Jiantao Pu, Hongbin Zha, Weibin Liu, and Yusuke Uehara. Thickness histogram and statistical harmonic representation for 3d model retrieval. In *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*, pages 896–903. IEEE, 2004.

[34] Mihael Ankerst, Gabi Kastenmüller, Hans-Peter Kriegel, and Thomas Seidl. 3d shape histograms for similarity search and classification in spatial databases. In *Advances in Spatial Databases*, pages 207–226. Springer, 1999.

[35] H-P Kriegel, P Kroger, Zahi Mashael, Martin Pfeifle, Marco Pötke, and Thomas Seidl. Effective similarity search on voxelized cad objects. In *Database Systems for Advanced Applications, 2003.(DASFAA 2003). Proceedings. Eighth International Conference on*, pages 27–36. IEEE, 2003.

[36] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433–449, 1999. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=765655`.

[37] 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*, volume 3, pages 5–17, 2003. URL `http://cg.tuwien.ac.at/hostings/cescg/CESCG-2003/MKoertgen/paper.pdf`.

[38] 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.

[39] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, June 2007.

[40] Horst Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.

[41] Ruwen Schnabel, Raoul Wessel, Roland Wahl, and Reinhard Klein. Shape recognition in 3d point-clouds. In *Proc. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision*, volume 2, 2008. URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.2344&rep=rep1&type=pdf`.

[42] Vincent Cicirello and William C. Regli. Machining feature-based comparisons of mechanical parts. In *Shape Modeling and Applications, SMI 2001 International Conference on.*, pages 176–185, 2001. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=923388`.

[43] David McWherter, Mitchell Peabody, Ali C. Shokoufandeh, and William Regli. Database techniques for archival of solid models. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 78–87, 2001. URL `http://dl.acm.org/citation.cfm?id=376968`.

[44] Nikhil Gagvani and Deborah Silver. Parameter-controlled volume thinning. *Graphical Models and Image Processing*, 61(3):149–164, 1999. URL `http://www.sciencedirect.com/science/article/pii/S1077316999904951`.

[45] Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. Skeleton extraction by mesh contraction. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pages 44:1–44:10, New York, NY, USA, 2008. ACM. ISBN 978-1-4503-0112-1. doi: 10.1145/1399504.1360643. URL `http://doi.acm.org/10.1145/1399504.1360643`.

[46] Hari Sundar, Deborah Silver, Nikhil Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. In *Shape Modeling International, 2003*, pages 130–139, 2003. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1199609`.

[47] 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*, pages 203–212, 2001. URL `http://dl.acm.org/citation.cfm?id=383282`.

[48] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3D model retrieval. In *Computer graphics forum*, volume 22, pages 223–232, 2003. URL `http://onlinelibrary.wiley.com/doi/10.1111/1467-8659.00669/full`.

[49] Dengsheng Zhang and Guojun Lu. An integrated approach to shape based image retrieval. In *Proc. of 5th Asian conference on computer vision (ACCV)*, pages 652–657, 2002. URL `http://lightfieldretrieval.googlecode.com/svn/trunk/Report/accv_integrate.pdf`.

[50] M. Novotni and R. Klein. A geometric approach to 3D object comparison. In *Shape Modeling and Applications, SMI 2001 International Conference on.*, pages 167–175, 2001. doi: 10.1109/SMA.2001.923387.

[51] Hermilo Snchez-Cruz and Ernesto Bribiesca. A method of optimum transformation of 3D objects used as a measure of shape dissimilarity. *Image and Vision Computing*, 21(12):1027–1036, November 2003. ISSN 0262-8856. doi: 10.1016/S0262-8856(03)00119-7. URL `http://www.sciencedirect.com/science/article/pii/S0262885603001197`.

[52] Johan WH Tangelder and Remco C. Veltkamp. Polyhedral model retrieval using weighted point sets. *International journal of image and graphics*, 3 (01):209–229, 2003. URL `http://www.worldscientific.com/doi/abs/10.1142/S021946780300097X`.

[53] David Flanagan. *JavaScript: The Definitive Guide*. O'Reilly Media, Inc., April 2011. ISBN 9781449308162.

[54] Patrick Min. binvox 3d mesh voxelizer, keywords: voxelization, voxelisation, 3D model. URL `http://www.cs.princeton.edu/~min/binvox/`.

[55] M. C. Jones, J. S. Marron, and S. J. Sheather. A brief survey of bandwidth selection for density estimation. *Journal of the American Statistical Association*, 91(433): 401–407, March 1996. ISSN 0162-1459. doi: 10.2307/2291420. URL `http://www.jstor.org/stable/2291420`.

[56] Yunqiang Chen, Xiang Sean Zhou, and Thomas S. Huang. One-class SVM for learning in image retrieval. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 1, pages 34–37, 2001. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=958946`.

[57] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001. URL `http://www.mitpressjournals.org/doi/abs/10.1162/089976601750264965`.

[58] Michael Donoser and Horst Bischof. Diffusion processes for retrieval revisited. URL `http://vh.icg.tugraz.at/publications/CVPR_2013_Diffusion.pdf`.

[59] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: bringing order to the web. 1999. URL `http://ilpubs.stanford.edu:8090/422`.

[60] Xingwei Yang, Suzan Koknar-Tezel, and Longin Jan Latecki. Locally constrained diffusion process on locally densified distance spaces with applications to shape retrieval. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 357–364. IEEE, 2009.