UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

WAGNER SCHMITT

A New 3D Shape Descriptor based on Depth Complexity and Thickness Information

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. João Luiz Dihl Comba

Porto Alegre
Março 2015

CIP – CATALOGING-IN-PUBLICATION

Schmitt, Wagner

A New 3D Shape Descriptor based on Depth Complexity and Thickness Information / Wagner Schmitt. – Porto Alegre: PPGC da UFRGS, 2015.

68 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2015. Advisor: João Luiz Dihl Comba.

1. 3d shape retrieval. 2. CBR. 3. Depth complexity. 4. Thickness. 5. Descriptor. 6. Statistical. I. Comba, João Luiz Dihl. II. Título.

ACKNOWLEDGEMENTS

I would like to thank specially my master advisor, João Luiz Dihl Comba. I have learned many things since I became his student. He instructed me how to write a paper, how to search literature and how to build a presentation. I am also grateful to Alexandru Telea, for spending time reading my papers and providing useful suggestions about them.

I would like to thank also my professors from UFRGS that provide me fundamentals and instruments to develop this thesis. My thanks also to the National Council for Scientific and Technological Development (CNPq) for their financial support.

During the period of almost three years, many friends were essential to my growth. I am very thankful to all my colleagues and friends which shared my journey to become a master, specially to Vitor Reus, Leonardo Moura, Alessandro Vecchia, Guilherme Medeiros, Kassius Prestes, André Ferreira, Bruno Marques, Tomaz Rocha e Newton Barbosa.

Last but not the least important, I owe more than thanks to my girlfriend for the care and constant support and my family members for their unconditional love and support. My parents, Paulo Ricardo Schmitt and Nilsa Teresinha Schmitt, my brothers and sisters, Maurício Schmitt and Deise Schmitt. I would like to thanks also the support of those who contributed in some way during this period.

“Anything you’re good at contributes to happiness. ”

— BERTRAND RUSSELL

ABSTRACT

Geometric models play a vital role in several fields, from the entertainment industry to scientific applications. To reduce the high cost of model creation, reusing existing models is the solution of choice. Model reuse is supported by content-based shape retrieval (CBR) techniques that help finding the desired models in massive repositories, many publicly available on the Internet. Key to efficient and effective CBR techniques are shape descriptors that accurately capture the characteristics of a shape and are able to discriminate between different shapes. We present a descriptor based on the distribution of two global features measured on a 3D shape, depth complexity and thickness, which respectively capture aspects of the geometry and topology of 3D shapes. The final descriptor, called DCT (depth complexity and thickness histogram), is a 2D histogram that is invariant to the translation, rotation and scale of geometric shapes. We efficiently implement the DCT on the GPU, allowing its use in real-time queries of large model databases. We validate the DCT with the Princeton and Toyohashi Shape Benchmarks, containing 1815 and 10000 models respectively. Results show that DCT can discriminate meaningful classes of these benchmarks, and is fast to compute and robust against shape transformations and different levels of subdivision and smoothness.

Keywords: 3d shape retrieval. CBR. depth complexity. thickness. descriptor. statistical.

Um Novo Descritor de Formas 3D Baseado em Informações de *Depth Complexity* e *Thickness*

RESUMO

Modelos geométricos desempenham um papel fundamental em divérsas áreas, desde a indústria do entretenimento até aplicações científicas. Para reduzir o elevado custo de criação de um modelo 3D, a reutilização de modelos existentes é a solução ideal. Recuperação de modelos 3D utilizam técnicas baseadas em conteúdo (do inglês CBR) que auxiliam a busca de modelos desejados em repositórios massivos, muitos disponíveis publicamente na Internet. Pontos principais para técnicas CBR eficientes e eficazes são descritores de forma que capturam com precisão as características de uma forma 3D e são capazes de discriminar entre diferentes formas. Nós apresentamos um descritor com base na distribuição de duas características globais, extraídas de uma forma 3D, *depth complexity* e *thickness*, que, respectivamente, capturam aspectos da topologia e da geometria das formas 3D. O descritor final, chamado DCT (*depth complexity and thickness histogram*), é um histograma 2D invariante a translações, rotações e escalas das formas geométricas. Nós eficientemente implementamos o DCT na GPU, permitindo sua utilização em consultas em tempo real em grandes bases de dados de modelos 3D. Nós validamos o DCT com as Princeton e Toyohashi Forma Benchmarks, contendo 1815 e 10000 modelos respectivamente. Os resultados mostram que DCT pode discriminar classes significativas desses benchmarks, é rápido e robusto contra transformações de forma e diferentes níveis de subdivisão e suavidade dos modelos.

Palavras-chave: Reuperação de formas 3D, CBR, depth complexity, thickness, descritor, histograma.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

In the last three decades, researchers put a lot of effort on the analysis and retrieval of content-based data. The early years of this study focused mainly on the retrieval of text-based documents. These efforts led to the development of text search engines that changed the way we find and use information today, becoming part of our daily lives (*e.g.* Google[1]). They provide access to a vast amount of text on every conceivable topic by indexing large portions of the web.

Besides text, the web also provides a lot of multimedia data as well, such as images, sounds, videos, 3D models, and so on. These different kinds of data are also being targeted by content-based retrieval systems. Many specialized search engines are already providing front-end query interfaces for users to search in over hundreds of thousands indexed videos, images and other multimedia data directly.

Another non-textual data type that recently is becoming ubiquitous in the WEB is the 3D model. We suggest four main reason for this increase: powerful modeling tools have emerged along affordable laser scanners that can rapidly generate 3D mesh models of real-world objects; 3D printers reaching to the general public are creating a high demand for three-dimensional objects; general purpose graphic cards now have the power to render much more detailed 3D models and scenes, offering the possibility for even home users to create and visualize 3D content with ease; and the ease distribution via WEB due to fast connections and high storage servers available.

Such advances in creation, distribution and visualization of 3D model are essential nowadays, since 3D shapes are basic building blocks for most 3D computer graphics applications. Entertaining industry is creating a high demand for 3D content to be used in animation movies and games, Figure 1.1 shows images of recent games, movies and applications where 3D shapes are highly demanded. Also, many medical, biological, engineering and design applications require the use of 3D models as well, examples are applications for surgery simulations to help training surgeons (Figure 1.1(f)), Computer Aided Design (CAD) systems (Figure 1.1(e)), architectural or products designing tools, and so on.

Today, there are on-line databases with thousands of free 3D models[2]. Such advances in distributions of 3D models are changing the way we think about 3D data. In the near future, the main question will be how to find the model that is best suitable to integrate a scene, and less about how to create them. For example, in the building of a 3D virtual world, a person would

[1] www.google.com

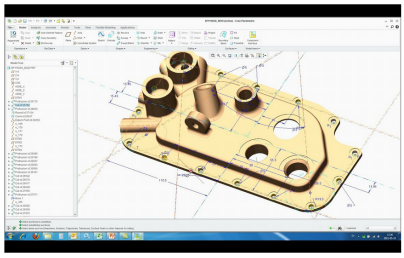
[2] Princeton 3D search engine (PRINCETON, 2001), Archive3d (ARCHIVE3D, 2014), Aim@Shape (Aim@Shape, 2014), and the McGill database (McGill, 2014)

(a) Animation movie.

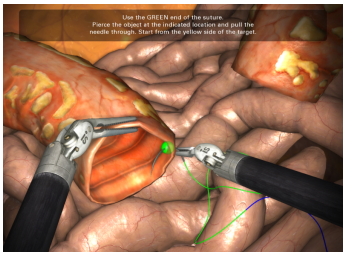
(b) Animation movie.

(c) Computer game.

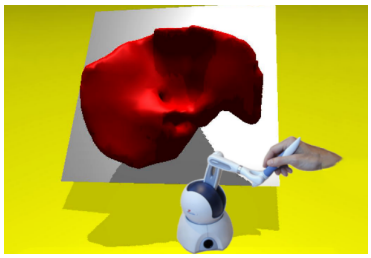
(d) Computer game.



(e) CAD Software.



(f) Surgery simulator.



(g) Organs simulation.

Figure 1.1: Examples of 3D model used in animation movies, games, CAD and medical simulations.

need several different models, such as buildings, cars, street lamps, and so on. To create all these models from scratch would be a laborious task that requires skill, time, and dedicated tools. A fast and less expensive solution would be to retrieve 3D models from such databases, increasing the reuse and interchange of 3D models. Moreover, models in many such databases have high mesh quality, sampling resolution, and are free of meshing defects, which only increases their reusability appeal.

Currently, the common way to search for models are via text queries and online catalogues.

The idea is to search for keywords in file names, captions or context. However, as seen in previous works (MIN; KAZHDAN; FUNKHOUSER, 2004), this method usually fail since most models available on the internet are poor annotated (*e.g., "P3322.3ds"*), annotated in foreign languages or containing misspelled labels, or, yet, annotated with unspecific or common keywords, which would flood the results with irrelevant models.

To overcome the aforementioned problems with text-based query techniques, Content-Based Retrieval (CBR) plays a key role in supporting such model reuse. Central to CBR is the usage of *shape descriptors* to capture the characteristics of 3D models. Given a descriptor, 'searching by example' accounts to comparing the example descriptor to descriptors of all models in the database (Fig. 1.2). Many such descriptors have been proposed in the last decade (OSADA et al., 2002; KAZHDAN; FUNKHOUSER; RUSINKIEWICZ, 2003; KAZHDAN; FUNKHOUSER; RUSINKIEWICZ, 2004; CHEN; ZHUANG; XIAO, 2010). However, no such proposal could present a definitive result to become a standard. One main reason for this is CBR is much harder for 3D shapes than for text or image retrieval (OSADA et al., 2002), due in turn to their non-trivial surface parametrization, large dimensionality and arbitrary topologies. For instance, 2D shapes with a 1D boundary contours have a natural arc length parametrization, on the other hand 3D surfaces of arbitrary genus do not. This makes much harder to extend common shape descriptor for 2D contours to be used for 3D surfaces. Moreover, the high dimensionality makes registration, feature correspondence and fitting model parameters much more expensive. As a result, such inherent 3D shape aspects prevent methods used to analyze other data types to be generalized to create good 3D shape descriptors.

Separately, the scalability and quality of many of the CBR techniques for 3D shapes proposed in the literature have not been conclusively tested with a massive number of models. This is due to the earlier lack of benchmarks, *i.e.* shape databases containing classification information to be used as ground truth for computing CBG precision and recall (TANGELDER; VELTKAMP, 2008). Until recent years, most such benchmarks contained less than 1000 shapes, such as the very popular Princeton shape database (SHILANE et al., 2004), created in 2004, which contains 907 models. Only more recently, in 2012, the large Toyohashi benchmark, containing 10000 models, was published (TATSUMA; KOYANAGI; AONO, 2012), the classification of the objects are shown in Figure 3.10.

In this thesis, we attempt to improve the state-of-the-art of CBR descriptors by proposing a novel 3D shape descriptor called the *depth complexity and thickness* (DCT) histogram. DCT uses a 2D histogram of depth complexity and thickness data measured from a 3D shape, which captures both topological (depth-complexity) and geometrical (thickness) shape characteristics.

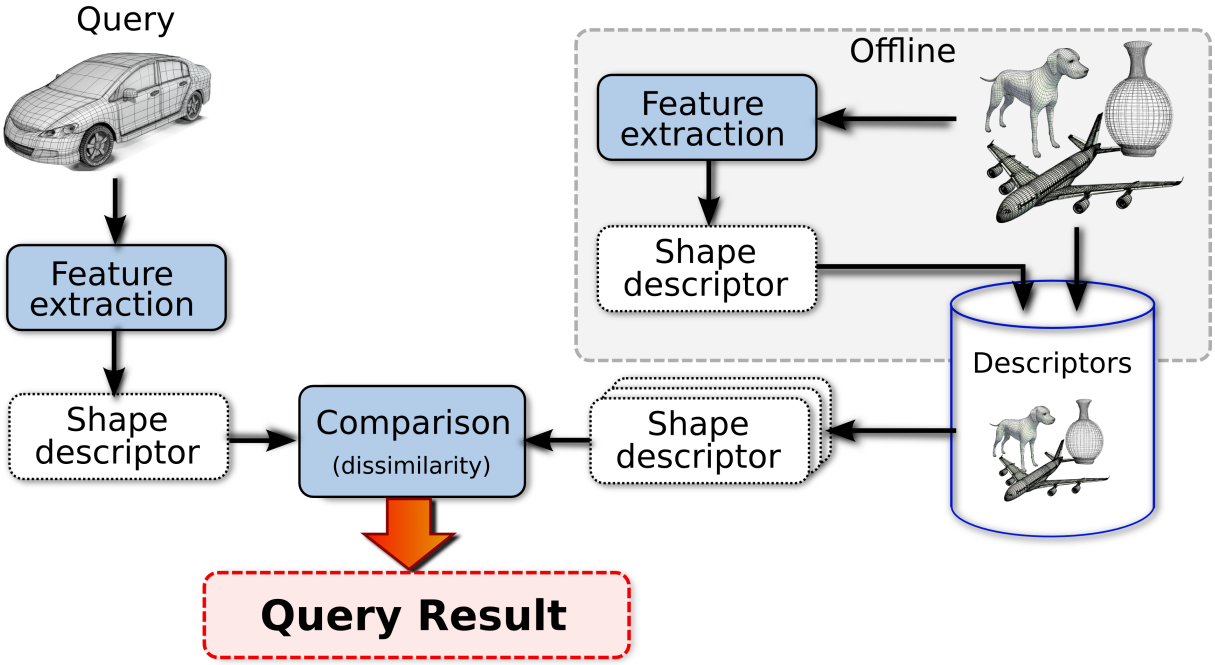


Figure 1.2: Conceptual diagram for a shape retrieval system.

DCT is rotation, translation, and scale invariant, and can be quickly computed on the GPU. We demonstrate the quality and scalability of our DCT descriptor by using both the Princeton and Toyohashi generic shape benchmarks. The experiments were performed on our implemented search engine, based on the provide Princeton 3D Search Engine (PRINCETON, 2001). We implemented it to validate and test our descriptor. In addition, a study on depth complexity data was performed, as it is a useful information for many other applications on shape analysis, such as computing transparency (LINDHOLM et al., 2014).

Our descriptor is mainly aimed to Generic 3D shape retrieval, which includes 3D objects that we often see in our common life. Examples of objects that fit in this classification are shown in Figure 3.10. Generic models are usually non-trivial to classify in classes or semantic groups due to their wide range of variation, for example, a table class can contain round or square tables; an insect class can have insects that fly and some that does not have wings; a vehicle class can contain trucks, cars, bulldozer and so on. We focused our research in this particular group of models because of its utmost importance in many applications, as they are often present objects in real life scenes.

1.1 Structure of this Thesis

The reminder of the thesis is organized as follow. Chapter 2 explains the fundamental concepts of 3D shape retrieval, such as distances and evaluation methods, necessary to understand the theory and implementation of our descriptor. Chapter 3 reference and discuss about CBR for 3D Shapes, descriptors and benchmarks proposed in the literature in the past years, and how they relate to ours. Chapter 4 describes our proposed DCT and explains how to efficient compute it relying on graphics hardware. Chapter 5 shows the results of DCT considering retrieval quality, robustness and efficiency. Finally, Chapter 6 we conclude our work and present some ideas for future work.

2 THEORETICAL BACKGROUND

In this Chapter, we explain the fundamental concepts of content-based 3D objects retrieval. How feature extraction and descriptors play a fundamental role in 3D object retrieval, and what are the desired properties of a good descriptor. In addition, we introduce metrics to compute distance between descriptors and tools to evaluate the acquired results. Finally, we make a gentle introduction on GPU computation and the programmable pipeline, which may be the future for fast descriptors.

2.1 Feature Extraction and Descriptors

A shape descriptor is a compact way of representing the main features of multimedia object without the necessity of human interaction. Depending on the type of multimedia, that can be sounds, images, videos, or 3D-models, the descriptor can be represented using different kinds of structures. Usually, for 3D objects, they are structured as a vector, a graph or a histogram. The main reasons for using a descriptor for content-retrieval, instead of using the object itself, is the possibility of building indices and to perform queries efficiently, besides the reduction of dimensionality.

Feature extraction is a fundamental part to construct a descriptor. The importance and how descriptive a feature is depends on the type of multimedia we are processing. The manual method of feature extraction is to search for textual data and annotations contained in a multimedia object. The automatic method for feature extraction consists in analyze the contents from the multimedia and extract informative and non redundant characteristics of each type. It is a more attractive area of research today, for being both promising and challenging. Figure 2.1 shows the concept of feature extraction for multimedia data.

A 3D-object is usually represented by a polygon mesh, which is a collection of vertices, edges and faces that define the shape. For simplicity and rendering purposes, a polygon mesh is usually converted into a triangle mesh by applying a triangulation algorithm. The formats used to distribute 3D-models, in most cases, contain only geometric and appearance information, usually devoid of structure or semantic information, Osada *et al.* (OSADA et al., 2002) defined this representation of 3D models as polygon soups - unorganized and degenerate sets of polygons. The most popular free formats are: VRML (Virtual Reality Modeling Language) (HARTMAN; WERNECKE, 1996), OFF (Object File Format) (WOTSIT, 2015), OBJ (Wavefront Object files) (WOTSIT, 2015), SMF (Simple Model Format) (FORMAT, 2015b), PLY (Polygon File Format) (FORMAT, 2015a).

Geometric and topological information are the most relevant characteristics of a 3D model. Secondary properties such as normals, color and texture can provide further information about the 3D object. However, almost all retrieval techniques consider only shape as the crucial property of a 3D object. Vranic *et al.* (VRANIC, 2003) classified features of a 3D object as:

- **image-based** features rely in 2D rasterized images of a 3D models, from different viewpoints;
- **3D geometry-based** features rely on the 3D quantities or attributes of the object. Examples would be volume, area, a curvature index of a parametric 3D surface, etc;
- **functions on a sphere** features rely on projecting the geometry of the object on an enclosing sphere. A point on the sphere is attributed a value, which can be a distance, surface area, information about shading, curvature index, etc;
- **statistical** features rely on measurements of local features, that are derived from several randomly selected points on the mesh, it can be curvature index, distance of the center of gravity, thickness, etc. These measurements are represented in a histogram and are purely statistical.
- **topological** features rely on skeletal structures, such as medial axis (JALBA; KUSTRA; TELEA, 2013b) and Reeb graphs (HILAGA et al., 2001).

Besides computation speed, good shape descriptors should satisfy additional application-specific features. For shape retrieval, the following features are considered desirable for a good descriptor (KAZMI; YOU; ZHANG, 2013; TANGELDER; VELTKAMP, 2008; MINGQIANG; IDIYO; JOSEPH, 2008; IYER et al., 2005; VRANIC, 2003):

- **non-restrictiveness** to enclosed or orientable polygonal meshes;
- **discriminative** accuracy, or the ability to capture subtle geometric and topological differences between shapes;
- **invariance** to translation, scaling, and rotation, collectively known as pose normalization;
- **robustness** against small-scale shape perturbations or noise and sampling resolution;
- **scalability** in terms of computational speed and memory.

For efficiency in large shape collections, comparing sequentially a query with all objects in the database is not a viable option. Retrieval should be fast, since it is a step computed online, efficient indexing search structures are needed to support efficient retrieval. It is reasonable to

Figure 2.1: Concept of multimedia retrieval.

require that the shape descriptor computation is fast enough for interactive querying.

Guided by these desirable features, we propose DCT, a statistical descriptor based on two measures: a depth complexity (DC) distribution and thickness (T) distribution, explained in Secs. 4.1 and 4.2 respectively. The two measures attempt to capture the geometric and topological features of a shape respectively, thereby increasing the descriptor's discriminative power. Both measures are robust in the presence of transformations, and can be efficiently implemented using GPU techniques. However, in our experimental search engine, we did not implement efficient indexing search structures, since our main concern was to provide an efficient and scalable descriptor, not focusing on the 3D search engine itself.

2.2 Similarity Search Metrics

In order to find how similar two objects are, it is necessary to compute the distance between pairs of descriptors. The term *distance* is used synonymously with the term *dissimilarity*. So, dissimilarity corresponds to the notion of how distant the objects are, small distances means small dissimilarity.

A dissimilarity measure is computed by a function of the form $d : X \times X \rightarrow \mathbb{R}^+ \cup \{0\}$, the result is a real value indicating the degree of resemblance between two descriptors. There are various properties of dissimilarity functions discussed in the literature (LUXBURG, 2004), consider the following ones:

- $\mathbf{P}_1$: $d(x, x) = 0$ (identity);
- $\mathbf{P}_2$: $d(x, y) \geq 0$ (non-negativity);
- $\mathbf{P}_3$: $d(x, y) = d(y, x)$ (symmetry);
- $\mathbf{P}_4$: $d(x, y) = 0 \Rightarrow x = y$ (definiteness);
- $\mathbf{P}_5$: $d(x, y) + d(y, z) \geq d(x, z)$ (triangle inequality).

A function that satisfies the conditions of $\mathbf{P}_1$ and $\mathbf{P}_2$ is called a dissimilarity function. Condition $\mathbf{P}_4$ also is very important for shape retrieval, since different descriptors that results in distance 0 from each other usually would not be discriminated in a meaningful way. The triangle inequality property is more desirable for partial matching, giving a small distance $d(x, y)$ if a part of x matches a part of y, it does not obey the triangle inequality. For a dissimilarity function to be called a *metric*, it must satisfies all the 5 axioms described. In case it satisfies only the axioms $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$ and $\mathbf{P}_5$ it is called *semi-metric*.

Most 3D shape descriptors proposed in the literature are represented as N-dimensional vectors of global features measurements, such as circularity, eccentricity, algebraic moments, etc. The most natural way of computing distances between vectors is to use a vector norm.

Let $\vec{v_1} = (v'_1, ..., v'_N)$ and $\vec{v_2} = (v''_1, ..., v''_N)$ be two feature vectors. the l_p norm between them is defined by:

$$d_p(\vec{v_1}, \vec{v_2}) = \| \vec{v_1} - \vec{v_2} \|_p = \left(\sum_{i=1}^{N} |\vec{v_{1i}} - \vec{v_{2i}}|^p \right)^{\frac{1}{p}} \quad \text{for } p = 1, 2, 3, ... \tag{2.1}$$

A variation of computing the l_p norm that allows to specify the importance and degree of correlation between the components is the quadratic form distance function, defined in (ANKERST

et al., 1999), and example of the quadratic form for l_2 norm is given by:

$$d_2^s(\vec{v_1}, \vec{v_2}) = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N} s_{ij}|\vec{v_{1i}} - \vec{v_{2i}}||\vec{v_{1j}} - \vec{v_{2j}}|} \tag{2.2}$$

This form take into account the similarities between components using a matrix S. Its elements can be computed using a relevance feedback (RUI et al., 1998). Also, a weight describing the importance of each component can be specified by populating the main diagonal $S = \mathit{diag}(\omega_1, ..., \omega_N)$. The flexibility of using a quadratic form increases the computational cost from $\mathcal{O}(N)$ for l_p norms to $\mathcal{O}(N^2)$.

Statistical descriptors – as the one we proposed in this thesis (see Chapter 4) –, are usually represented as histograms. There are many standard distances proposed in the literature to compare statistical distributions. The ones we used to experiment are the L_1 norm, Hellinger distance (a simple variation of Bhattacharyya distance), Correlation distance, Chi-square distance (PELE; WERMAN, 2010) and earth moverś distance (RUBNER; TOMASI; GUIBAS, 2000). Some others include Kolmogorov-Smirnov distance, Kullback-Leibler divergence distances (KULLBACK, 1968), match distances (SHEN; WONG, 1983a).

Given two DCT histograms H_1 and H_2, the distance is denoted by $d(H_1, H_2)$. The Hellinger distance is given by Equation 2.3. The correlation distance is given by Equation 2.4. The Chi-square distance is given by Equation 2.5. The L_1 distance is given by Equation 2.6. The Earth Mover's Distance (EMD) is more complex – for details we refer to (RUBNER; TOMASI; GUIBAS, 1998b). In all equations, $H(x)$ denotes the value of the x^{th} element of H, $\bar{H}$ denotes the average value of H, and N is the number of bins in H.

$$d(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\bar{H_1}\bar{H_2}N^2}} \cdot \sum_x \sqrt{H_1(x) \cdot H_2(x)}} \tag{2.3}$$

$$d(H_1, H_2) = \frac{\sum_x (H_1(x) - \bar{H_1}) \cdot (H_2(x) - \bar{H_2})}{\sqrt{\sum_x (H_1(x) - \bar{H_1})^2 \cdot \sum_x (H_2(x) - \bar{H_2})^2}} \tag{2.4}$$

$$d(H_1, H_2) = \sum_x \frac{(H_1(x) - H_2(x))^2}{H_1(x)} \tag{2.5}$$

$$d(H_1, H_2) = \sum_x |H_1(x) - H_2(x)| \tag{2.6}$$

2.3 Evaluation of Retrieval Effectiveness

The evaluation of descriptors is an important aspect to validate a proposal. Their retrieval performance are usually evaluated based on *precision* and *recall* values, which are commonly used in the field of information retrieval.

Given a query model M and a shape database D, we call *relevant* retrievals all models in D to which M should match. More specifically, for a given M, a perfect descriptor should return all shapes in D marked as being in the same class, or of the same type, as M. The definition of a relevant retrieval is subject to user choices or application scope. For example, in some cases a wide retrieval may lead to many matches of M in D, whereas narrow matches might be preferred in other cases.

2.3.1 Precision-Recall Curves

In information retrieval, precision P and recall R are defined in terms of a set of retrieved instances S and a set of relevant instances S_R, both belonging to a given database D. Precision is the fraction of retrieved instances that are relevant to the query, while recall is the fraction of relevant instances retrieved by the query.

Precision and recall are computed as

$$P = \frac{|S_R \bigcap S|}{|S|}, \qquad R = \frac{|S_R \bigcap S|}{|S_R|}. \tag{2.7}$$

The *order* in which objects are returned to the user in a query is also important. Ideally, more relevant objects (from the query perspective) should appear earlier in the query result S. The Average Precision (AP) metric is used for this purpose. Precision and recall are computed for each position of the ranked sequence S. The precision $P(R)$ is defined as a function of recall R, for all sequence positions. Typically, the function $P(R)$ varies over $R \in [0, 1]$. When the recall R is small, precision $P(R)$ is often large. When the recall R is large, precision $P(R)$ tends to be small. The AP of the query corresponds to the average of $P(R)$ over the entire range of R for that query. By plotting more than one precision-recall curves associated with different descriptors, it is possible to compare their performance intuitively.

The Mean Average Precision (MAP) evaluates the effectiveness of a set of Q queries, rather

than a single query. MAP is equal to the mean of the AP for each query:

$$MAP = \frac{1}{Q} \sum_{q=1}^{Q} AP_q \tag{2.8}$$

where AP_q is the AP of the q-th query in the sequence of Q queries. MAP is a standard metric for ranked results in text retrieval (TREC, 2014) and retrieval evaluation benchmarks (CLEF, 2014).

2.3.2 First Tier (FT) and Second Tier (ST)

Considering that the query itself is not presented in the ranked list, First Tier (FT) represents the percentage of success of the first C elements of the ranked list, where C is the quantity of objects of the same query class.

The Second Tier (ST) is very similar, except that it uses the size of $2 \times C$ for each element.

2.3.3 Nearest Neighbor (NN)

Nearest Neighbor (NN) in the search evaluation represents a measure if the top in the ranked list is relevant or not (belongs to the same class as the query). The final result is an average over all the objects in the database.

2.3.4 F-Measure

F-Measure (also know as F score or F_1 score) is often used in the field of information retrieval for measuring search, document classification, and query classification performance (BEITZEL, 2006). The idea of F-Measure is to combine precision and recall into a single value that indicates the accuracy of the whole system. F-measure ranges from $[0, 1]$, being 1 the best value.

F-Measure is computed as the harmonic mean of precision and recall and it is defined as:

$$F = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.9}$$

Since a user of a search engine is more interested in the first page of query results, for 3D shape search F-measure has been historically taken for the top $T = 32$ retrieved objects for every query (SHILANE et al., 2004).

2.3.5 Discounted Cumulative Gain(DCG)

Discounted Cumulative Gain (DCG) is based on the idea that the greater the ranked position of a relevant object the less valuable it is for the user, because less likely the user will examine the object due to time, effort, and cumulated information from the top ranked results. In other words it represents a measure that accounts for how many relevant objects are found around the top T ranked list.

First the Cumulated Gain (CG) is computed by replacing objects' IDs by their relevance values. The relevance values are 1 denoting a relevant object (from the same class of the query), 0 otherwise. GC is the predecessor of DCG and does not include the position of a result in the consideration of the usefulness of a result set. In this way, it is the sum of the graded relevance values of all results in a search result list.

CG is defined as follows. Denote the value of the i^{th} position in the gain vector G by G_i

$$CG_i = \begin{cases} G_1, & i = 1 \\ CG_{i-1} + G_i, & \text{otherwise} \end{cases} \tag{2.10}$$

The premise of DCG is that highly relevant documents appearing lower in a search result list should be penalized as the graded relevance value is reduced logarithmically proportional to the position of the result. The cumulated gain vector with a discount factor, DCG, is defined recursively by:

$$DCG_i = \begin{cases} G_1, & i = 1 \\ DCG_{i-1} + \frac{G_i}{\log_2 i}, & \text{otherwise} \end{cases} \tag{2.11}$$

The value of each correct results is estimated according to its position in the search. Objects located further down the list are less relevant.

2.4 Introduction to GPGPU Computation

Before the first programmable graphic cards, that was released in 2002 by NVIDIA, the graphics hardware were designed to be just accelerators, supporting only fixed-functions pipelines. Since them, not just game industries and artists start exploring this new technologies, but researches start to make use of the excellent floating point performance. This is when the General Purposed GPU computing (GPGPU) movement start to arise (RUEDA; ORTEGA, 2008). Todayś GPUs are general-purpose parallel processors with support for accessible programming

interfaces and industry-standard languages such as C. Developers who port their applications to GPUs often achieve speedups of orders of magnitude vs. optimized CPU implementations. Many works can be found in the literature in the fields of Signal Processing, Computer Vision, Computational Geometry and Scientific Computing that rely on GPUs to achieve more efficiency (OWENS et al., 2007).

After the graphics pipeline became programmable, many graphics programming languages (or shading language) start to appear. Due to the variety of target markets for 3D computer graphics, different shading languages have been developed. In addition, major companies such as OpenGL and Direct3D began to support them. Those languages were created mainly focusing on programming shader effects, offering access only to the most common structures such as vector, matrices and textures, fairly used in computer graphics. Even though, those languages were aimed to increase the visual realism of rendered images, and, therefore, widely used in video games and cinema. Some researchers start to using it to speed up many different kind of general purpose applications, such as (LUMSDAINE; GEORGIEV, 2012; HOFF III et al., 1999).

The most common shading languages today are Cg (MARK et al., 2003), GLSL (J. BALDWIN D., 2004) and HLSL(MICROSOFT, 2005), all abstract the capabilities of the underlying GPU and allow the programmer to write GPU programs in a more familiar C-like programming language. The graphics aimed nature of shading languages makes GPGPU programming more difficult than it needs to be. For example, initiating a GPGPU computation usually involves drawing a primitive. Looking up data from memory is done by issuing a texture fetch. Although, the program conceptually has nothing to do with drawing primitives and fetching textures.

In 2006, NVIDIA gave another step into the advance of general-computing. A team of researchers led by Ian Buck extended the C language with data-parallel constructs, where they introduced terms as stream and kernels. This parallel code ran over seven times faster than similar existing code. After that, NVIDIA applied Buckssólution to seamlessly run C on the GPU. Putting the software and hardware together. This technology was release as the name CUDA (NVIDIA, 2006) - the first solution for general-computing on GPUs.

2.5 Programmable Pipeline and GLSL

The emergence of general-purpose applications on graphics hardware has been driven by the rapid improvements in the programmability and performance of the underlying graphics

hardware. In this section we will outline the evolution of the GPU and describe its current hardware and software.

Interactive 3D graphics present many characteristics that differentiate it from more general computation domains. In particular, interactive 3D graphics applications require a high rate and are more prone to parallelism. Commodity GPUs structure their graphics computation in a similar organization called the graphics pipeline. This pipeline is designed to allow hardware implementations to maintain high computation rates through parallel execution (ROST, 2005). The pipeline is divided into several stages; all geometric primitives pass through every stage. In hardware, each stage is implemented as a separate piece of hardware on the GPU, allowing a task-parallel organization of the components. Figure 2.2 shows simplified pipelines stages of recent GPUs.

The pipeline receive as basic input a list of geometry, vertices and information about their connectivity. The output is an image in a frame-buffer. The stages of a simplified pipeline is as follow (ROST, 2005):

1. Geometry Stage: transforms each vertex from object space into screen space, assembles the vertices into triangles, and perform light calculations (known as per-vertex lighting).
2. Rasterization Stage: determine the screen position covered by each triangle and interpolates per-vertex parameters across the triangle.
3. Fragment Stage: computes the color for each fragment, using the interpolated values from the geometry stage. This stage can use values from global memory in the form of textures;

In the end, fragments are assembled into an image of pixels. Usually by choosing the closest fragment to the camera at each pixel location.

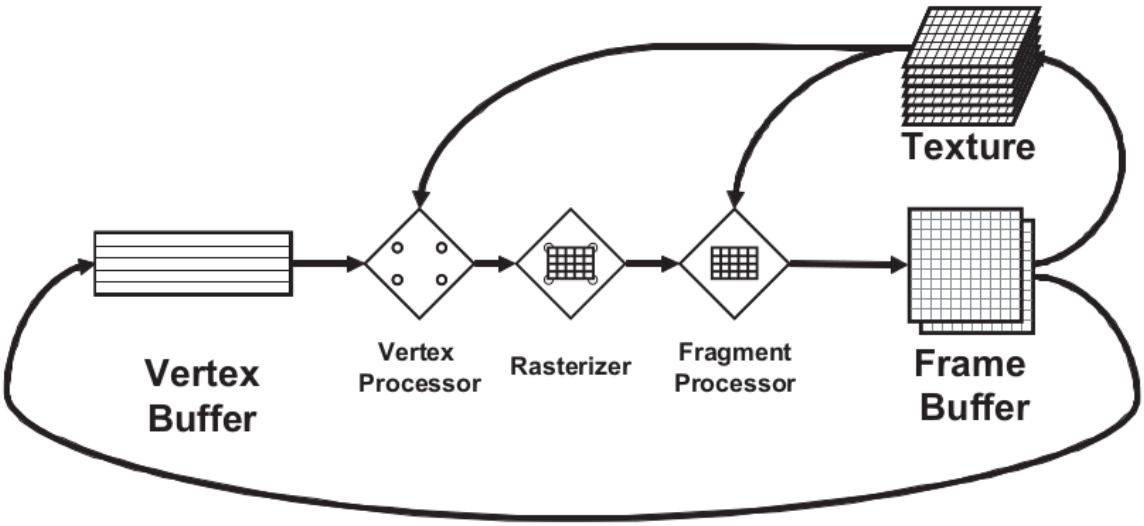


Figure 2.2: The modern graphics hardware pipeline. The vertex and fragment processor stages are both programmable by the user. SOURCE (OWENS et al., 2007)

3 RELATED WORKS

Data retrieval and analysis have been a very active area of research. The most obvious examples are text search engines. However, Content-Based Retrieval (CBR) and classification systems have also been developed for other multimedia data types, including audio (BHALKE; RAO; BORMANE, 2013; FU et al., 2011), images (YUE et al., 2011; AKGÜL et al., 2011), and video (HU et al., 2011). Retrieval of data based on shape has been studied in several fields, including computer vision, computational geometry, mechanical CAD, and molecular biology.

Content-Based Retrieval (CBR) of 3D shapes are a bit more recent subject than text retrieval, the idea of indexing 3D shapes emerged about two decades ago with the pioneer work of Paquet (PAQUET et al., 2000; PAQUET; RIOUX, 1997) on cords and moments-based descriptors. From this point, many different techniques and benchmarks have been developed. In this Chapter we will provide state-of-the-art information about CBR Systems, shape descriptors and benchmarks used for 3D shape retrieval.

To foment the research on 3D shape retrieval and analysis, there is an annual contest on this subject (SHREC – SHape REtrieval Contest). SHREC'14 is currently the latest one available and the ninth edition (PICKUP et al., 2014). They also provide in their website many specialized 3D shape objects including Generic Models, CAD Models, and 3D Face Models.

3.1 CBR Systems for 3D Shapes

Most sites that allow users to search for 3D models today are focused only in text-based queries or catalogues, such as Yobi3D [1], 3D Cafe [2], Arquive3D [3], among others. However, experimental sites with different query interface proposals are already available on the internet. This is the case of the Princeton 3D Model Search Engine [4]. Its interface provides three options for the user to perform a query: using a 2D or 3D sketch system, or uploading a 3D model. The search engine was used to validate the work proposals of Funkhouser *et al.* (FUNKHOUSER et al., 2003), Min *et al.* (MIN, 2004) and (VRANIC, 2003). Figure 3.1 shows the organization their 3D search engine composed of an online/offline system. This organization was used as reference to build our own experimental 3D search engine, used to test and validate our experiments.

In Funkhouser *et al.* (FUNKHOUSER et al., 2003), they investigate interfaces for untrained users to be able to search for 3D shapes. In case a user does not have a desired 3D model avail-

[1] https://www.yobi3d.com/
[2] http://www.3dcafe.com/
[3] http://archive3d.net/
[4] http://shape.cs.princeton.edu/search.html

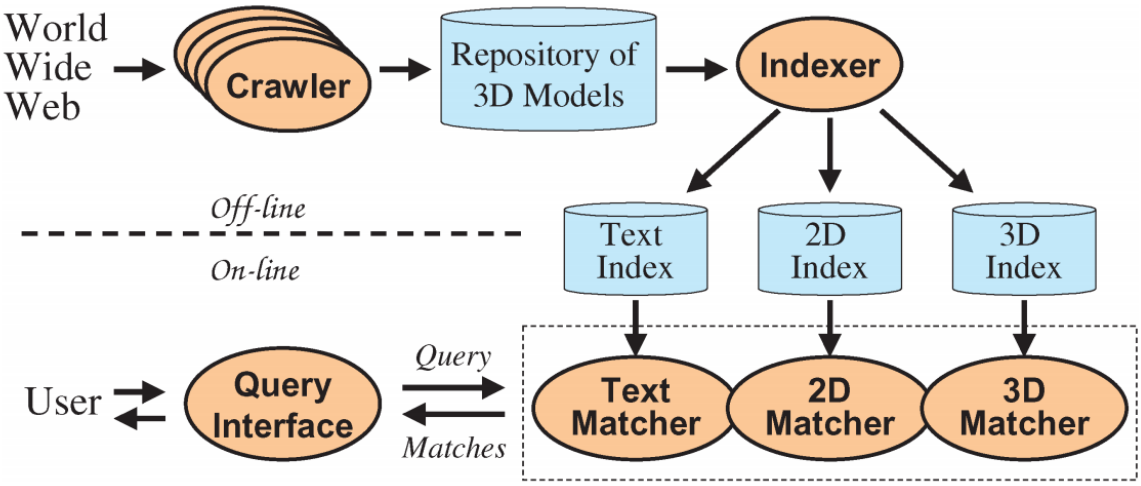


Figure 3.1: Princeton 3D Shape Search System organization.

able to be used as a query, it is provided a 2D or 3D sketch interface, shown in Figure 3.2(right) and 3.2(left), respectively. The 3D sketch system uses Teddy (IGARASHI; MATSUOKA; TANAKA, 1999), a tool to create 3D models from a 2D sketch. After the model is generated, it is used as query for the search engine. the main disadvantages are the difficulty for novice users to create models without a previous training – even being much simpler than professional modeling tools – and the limitation of shapes that can be created with it (round objects with topological genus zero). The alternative the authors found was to use 2D shapes created with a pixel paint program and match the results with 2D projections of 3D objects. They allowed the user to make up to three sketches: for the side, front and top perspectives, to compensate the lack of shape detail in a 2D sketch. The disadvantage of this technique is that it requires an algorithm to match a 2D sketch to a 3D object. The technique used was to match the drawings to projected images of 3D models rendered from different viewpoints and return the best matches, as in (MURASE; NAYAR, 1995).

3.2 Descriptors

Descriptors are the most fundamental part of CBR, good descriptors can greatly improve shape retrieval quality and efficiency. Therefore, a lot of different proposals for descriptors emerged in the Literature the last decade. A list of the most important descriptors and its classification according to Kazmi *et al.* (KAZMI; YOU; ZHANG, 2013) is presented in Table 3.1, for comprehensive surveys of 3D descriptors, see (KAZMI; YOU; ZHANG, 2013; TANGELDER; VELTKAMP, 2008; ZHANG; LU, 2004).

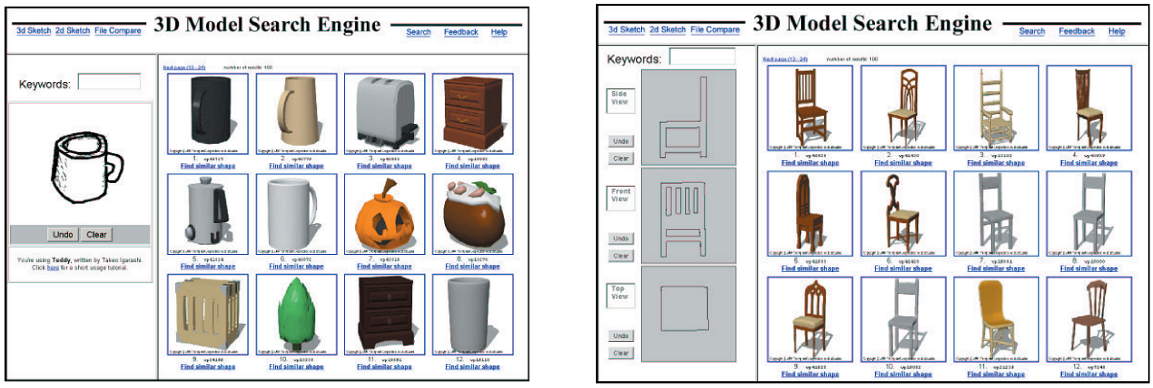


Figure 3.2: Princeton 3D Shape Search 3D and 2D sketch interfaces.

Classification	**Descriptor**
View Based	Adaptive Views Clustering Compact Multi-View Descriptor Lightfield Descriptor (LFD)
Histogram Based	Shape Spectrum Generalized shape distributions Bag-of-Features (BoF)
Transform Based	Spherical Harmonics Descriptor PCA Spherical harmonics Spherical Trace Transform
Graph Based	Skeletal Graph Based Reeb Graph Based
Hybrid 3D Descriptors	CMVD + STT Depth-Buffer + Silhouette + REXT SIFT + Bag of Features Depth-Buffer + Spherical Harmonics

Table 3.1: Classification of 3D shape descriptors. Extracted from (KAZMI; YOU; ZHANG, 2013)

Tangelder *et al.* (TANGELDER; VELTKAMP, 2008) proposed a simpler taxonomy of 3D descriptors, according to the authors, the descriptors can be organized in three large groups, they are: *Feature-based*, *Graph-based* and *Other*. There is an extensive list of descriptor proposed in each group, we will comment the ones we consider more relevant for this work.

All the descriptors mentioned in this Chapter focus on shape matching for generic 3D shapes. There are also other more specific techniques, that are concerned with partial matching (LIU et al., 2013), object poses (CHEN; ZHUANG; XIAO, 2010) or deformable 3D shapes (LI; HAMZA, 2013), among others.

3.2.1 Shape Distributions

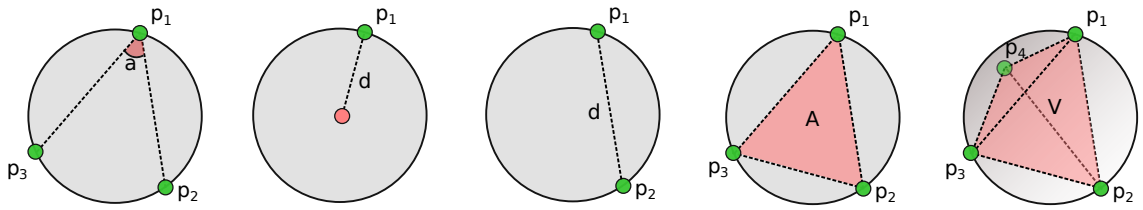
In 2002, Osada *et al.* (OSADA et al., 2002) improved the concept of using global features to describe an object by using a distribution of these features instead. This is a statistical approach that does not require any normalization of the model. The method requires the use of a shape function, which is a function that samples a particular geometric property of the model, since they claim it gives the better signatures for the shape. To obtain a distribution of the chosen feature, the shape function is randomly sampled and a histogram of this data is built. Their selected shape functions are described in Figure 3.3, measuring angles, euclidean distances, areas and volumes of the shape given random points on the surface of the object. It is worth mention that the probability of select a point on a mesh triangle is proportional to its area, this way any point on the object surface has the same chosen probability. To find a shape function that can result in good signatures is a challenging process, since they have to satisfy some desired properties, like the invariance to transformations, such as translation, rotation and scale, robustness, efficiency, among others.

After computing the distributions, the problem of shape matching is reduced to a simple comparison between them. There are many standard methods found in the Literature to compare distributions. In their work they experimented with eight simple dissimilarity measures, some methods include: the Minkowsky L_N distances, Bhattacharyya, Chi-Square and Earth mover's distance. They obtained the best results with the Minkowsky L_1 norm, which was followed closely by the Chi-square and Bhattacharyya distances.

The benchmark used by Osada *et al.* to test their descriptor are from models downloaded from the World Wide Web. Their similarity experiments used 133 models classified in 25 different classes.

The average extraction time of the shape distribution descriptor is around $1.12s$, using the $D2$ distance shape function on a PC with an 1.4 GHz AMD processor running Windows 2000 (OSADA et al., 2002). Also, the results were obtained based on a very small collection of models. The discriminant power of the shape distribution descriptor were not as good as the descriptors available at the time, such as the Spherical Harmonics Descriptor, refer 3.2.2.

An extension of the shape distributions proposed by Osada *et al.*, is the work of Ohbuchi *et al.* (OHBUCHI; MINAMITANI; TAKEI, 2003). In this work they proposed extended shape functions, where they convert a surface based model into an oriented point set model, and then measuring two features at the same time, distance and orientation of pair of points over the surface of the object. Their experiments showed an increased computational cost, but their



(a) Angle (A3) (b) distance (D1) (c) distance (D2) (d) area (D3) (e) volume (D4)

Figure 3.3: Five shape functions measuring geometric properties from a shape

retrieving performance was significantly better.

The performance of their technique was tested using three databases: one containing 215 VRML models; another consisting of 1213 models, collected from the internet, or modified by adding noise; the last one is the Princeton Shape Benchmark which contains 907 models for tests. We also used the Princeton Shape Benchmark in our experiments, except we merged the training and tests models to use all 1814 models for tests, since our algorithm does not need a training step.

Our technique also can be considered a Feature-based method that uses feature distributions to find dissimilarity values between shapes, the main difference is that we use two distributions at the same time that cover both geometric and topological characteristics of the shape.

3.2.2 Spherical Harmonics Descriptor

Also in the context of Feature-based descriptors, Kazhdan *et al.* (KAZHDAN; FUNKHOUSER; RUSINKIEWICZ, 2003) proposed the invariant spherical harmonics descriptor. They used a rotation invariant collections of spherical functions based on the original spherical harmonics descriptor proposed by (SAUPE; VRANIC, 2001). The main idea of this descriptor it to first voxelize the 3D model, where each voxel can have a value of 1 or 0. The binary voxel grid of the 3D model is then placed under concentric spheres and decomposed into spherical functions, Figure 3.4 shows this process.

This process starts with a normalization step. First, the 3D model is translated so that its center of gravity stay at the origin of the coordinate system, then the model is scaled by the average distance of a point on the surface of the model to the center of gravity. This step is represented by the following equation:

$$I' = \{v' | v' = (v - m_I)/d_{avg}, v \in I\}. \tag{3.1}$$

Where, v is the vertices of the model, m_I is the centroid distance from the origin and d_{avg} represents the average distance of a point on the surface of the model to the center of gravity.

The next step is to extract a cube region around the model on 3D-space $\mathbb{R}^3$ and rasterize it into a voxel cubic grid on the discrete 3D-space $\mathbb{Z}^3$. A voxel cell corresponds to a cuboid region of space, which is attributed a binary value 1 if there is a point on the surface of the object that is contained into the grid cell, or 0 otherwise. The voxel an airplane model is shown in Figure fig:spherical-harmonics-voxel(middle).

After the voxelization, the next step is to define binary functions on concentric spheres, Figure 3.4(right). The totally functions $f_r(\theta, \phi)(\theta \in [0, \pi], \phi \in [0, 2\pi])$ with center at origin, are applied to concentric spheres with radii ranging from $r = 1, \ldots, R$, where the recommended value is $R = 32$, by (KAZHDAN; FUNKHOUSER; RUSINKIEWICZ, 2003). Then the fast fourier transform is applied on the samples obtained for each sphere. The most important step, to ensure invariance with respect to rotation the functions are generated using the first L bands of spherical harmonics. The feature vector based on a binary voxel grid is formed by concatenating the signatures of spherical functions.

Vranic *et al.* (VRANIC, 2003) made some important observations in relation to this descriptor. He states that the discriminative power of descriptors based on voxel grids are limited by the low resolution of the grid itself, which loses too many fine details of the mesh. In addition, if the resolution of the voxel grid is increased, then the function on concentric spheres, corresponding to similar models whose scaling factors are slightly different, can be mismatched.

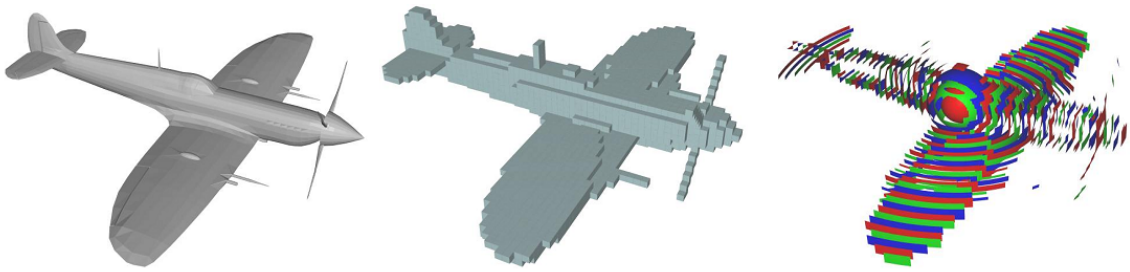


Figure 3.4: Spherical Harmonics Voxel Grid.

3.2.3 Reeb Graph based Descriptor

Another interesting descriptor was proposed by Hilaga *et al.* (HILAGA et al., 2001) into the context of Graph-based descriptors. It is a sophisticated technique that uses a skeletal structure

called *Multiresolutional Reeb Graph* (MRG) to describe the topology of a 3D object. The topology matching technique can be subdivided into two phases: generation of the representation, and computation of similarity between two graphs.

The main idea of this technique is to represent the topology information of the object by the MRG. To compute the MRG, it relies on the calculation of geodesic distances, which makes their descriptor invariant with respect to similarity transformation. The MRG is generated using a suitable function that approximates the integral of geodesic distances between a point p on the surface of the 3D model to all other points of the a point set defined on the surface of the model. Since the geodesic distance clearly changes with the scale of the object, it is required a normalization step.

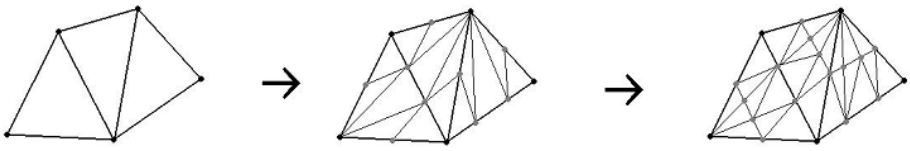


Figure 3.5: Reeb graph resampling vertices step.

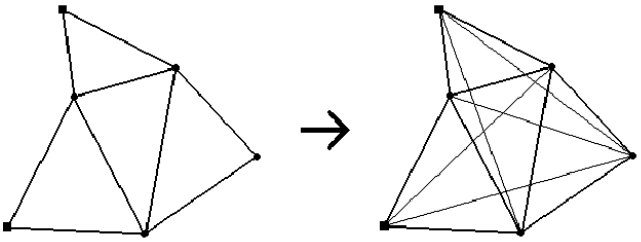


Figure 3.6: Reeb graph creating shortcut edges.

To avoid the complexity in computing the geodesic distances exactly, they use a technique that allows a trade-off between speed and precision. The technique is performed as pre-processing step, reorganizing the topology and geometry of the mesh by resampling its vertices until all edge lengths are less than a threshold θ and them adding shortcut edges. The resampling and Shortcut steps are shown in Figure 3.5 and Figure 3.6, respectively. After all new edges were generated, the model is treated as a weighted graph, where the weight of an edge is the geodesic distance between the endpoints of the edge. This way the problem of finding the geodesic distance can be interpreted as the problem of finding the shortest path from a

point in the graph (the source) to a destination.

After computing the distances, the construction of the *Multiresolution Reeb Graph* can be accomplished as seen in Figure 3.7. The main drawback, is that very similar shapes can be mapped to different MRBs, because of the sensibility of the μ_n-range boundaries. Figure 3.8 shown a case where this could happen.

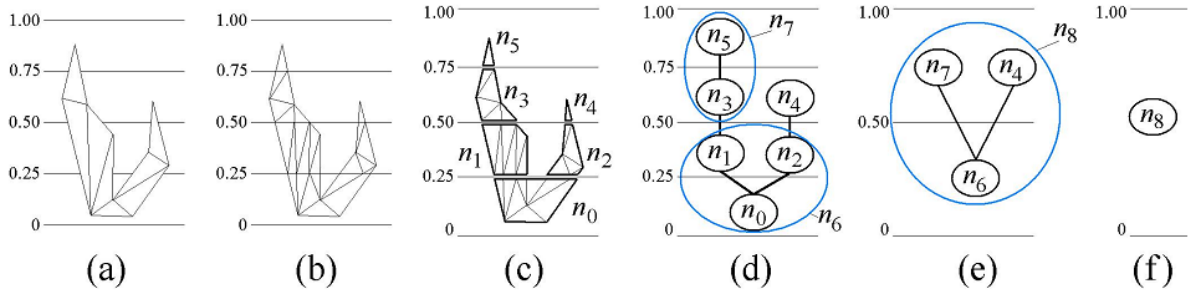


Figure 3.7: Reeb graph construction.

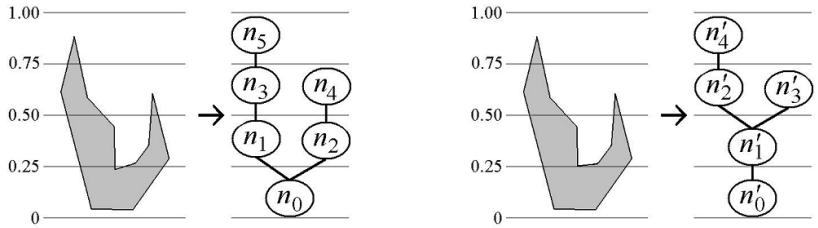


Figure 3.8: Reeb graph construction drawback.

The procedure to match two MRGs is based on a coarse-to-fine strategy. It starts by finding pairs of nodes whose similarity computation starts at the coarsest level. The similarity computation starts at the coarsest level and follow a set of rules, where the main one is that nodes are matching candidates only if they have the same μ_n-range. For the complete explanation of the rules, we refer to (KAZHDAN; FUNKHOUSER; RUSINKIEWICZ, 2003).

In the results presented in (KAZHDAN; FUNKHOUSER; RUSINKIEWICZ, 2003), the average time for constructing a MRG is $15s$, for a mesh with 10000 vertices. The results are obtained on a PC with an 400 MHz Pentium II processor running Linux. The average matching between two MRGs is approximately $0.05s$. We thought this is not a scalable technique, since in a retrieval system, for a given query, the procedure has to be performed 1000 times, then the retrieval system responds after 50 seconds, which is unacceptable for interactive applications.

3.2.4 Depth-Buffered Vector of Locally Aggregated Tensors (DB-VLAT) Descriptor

More recently, Tatsuma *et al.* (TATSUMA; KOYANAGI; AONO, 2012) proposed a new descriptor called DB-VLAT (Depth-Buffered Vector of Locally Aggregated Tensors). It is categorized as a feature-based descriptor, composed of Bag-of-Visual Words (BoVW) citeFuruya2009, a term most used on Computer Vision field. VoVW is regarded The main idea is to use a collection of depth-buffered images through rendering of a given 3D shape model.

Figure 3.9 illustrates, step-by-step, how DBVLAT's feature vector is generated. First, a pose normalization step is applied using a techniques proposed by (TATSUMA; AONO, 2009), that aligns the centroid and principal axes by generating random points on the surface of the 3D object, and a normal normalization that aligns the surface normals with respect to principal axes.

The next step is to enclose the model in a regular octahedron, and render the object placing the camera on each vertex of the octahedron, as well as on the midpoints of the edges, totalizing 18 viewpoints. The rendering is used to obtain a collection of depth-buffer images with a 256×256 resolution. From each depth-buffer image, a DIFT is performed, which is a feature vector similar to the SIFT image descriptor (LOWE, 2004), extracted from the patch with an equal interval and size.

To compare two DB-VLAT descriptors, the authors simply computed the Euclidean distance between them.

The DB-VLAT descriptor produced very good precision-recall plots testing with models from the Toyohashi Shape Benchmark. Their state-of-the-art technique outperformed established descriptors proposed in the literature, such as the D2 (OSADA et al., 2002) and the Spherical Harmonics (KAZHDAN; FUNKHOUSER; RUSINKIEWICZ, 2003). For the 6 classes tested by Tatsuma *et al.*, the only one that did not perform well was a vehicle-like 'tank' class. The authors stated that the geometry of this vehicle was harder to discriminate due to its oblong cannon. No information was given about the computational cost and scalability of their descriptor.

3.3 3D Shape Benchmarks

Benchmarking allows researchers to evaluate the quality results of different 3D shape retrieval approaches. Until recent years, available benchmarks were sparse, most small or specialized in a particular class of objects. For this reason, it was hard to evaluate through experiments

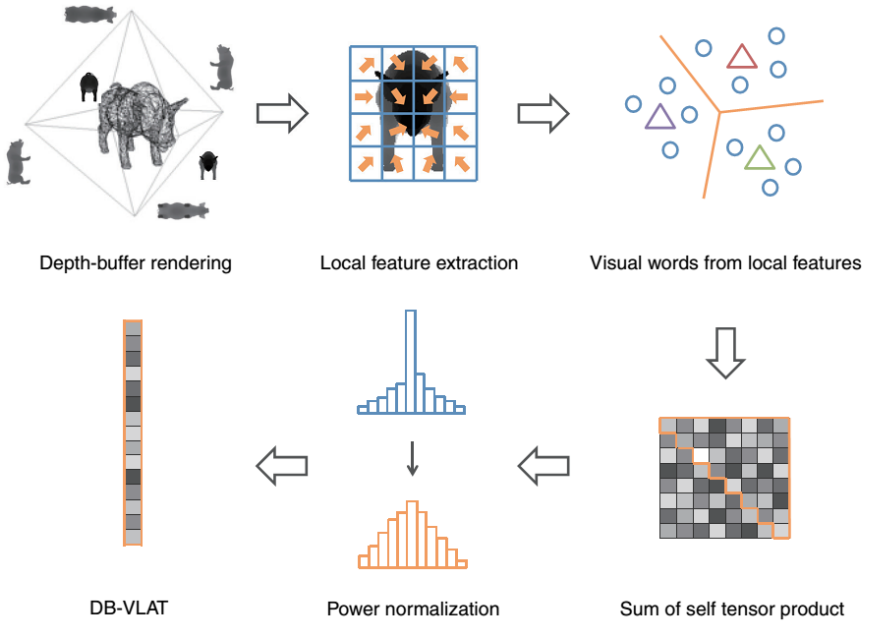


Figure 3.9: DB-VLAT pose extraction.

the quality and scalability of the proposed technique. Another issue, was that it was not possible to compare shape match results obtained by different researchers.

More recently, due to the explosion of models available on the WEB, larger and more diverse shape benchmarks start to emerge. Internet crawlers and specialized search engines help to find free models, then researchers classified and published them to the community. In Table 3.2 we list the benchmarks we found available on the WEB.

Benchmark	**Type**	**#M**	**#C**	**#Avg**
Princeton Shape Benchmark (PSB)	Generic	1814	90	10
Toyohashi Shape Benchmark (TSB)	Generic	10000	352	28
Konstanz 3D Model Benchmark (CCCC)	Generic	473	55	9
Generic Track Benchmark (SHREC12GTB)	Generic	1200	60	20
McGill Shape Benchmark (MSB)	Articulated	457	19	24
Watertight Shape Benchmark (WSB)	Articulated	400	20	20
Bonn's Architecture Benchmark (BAB)	Architecture	2257	363	25
Engineering Shape Benchmark (ESB)	CAD	867	45	19

Table 3.2: Benchmarks available in Literature. The columns #M corresponds to the number of models, #C the number of classes and #Avg is the average number of models per class. For the benchmarks listed, refer to PSB (SHILANE et al., 2004), TSB (TATSUMA; KOYANAGI; AONO, 2012), CCCC (BENCHMARK, 2008), SHREC12 GTB (SHREC, 2012), MSG (SIDDIQI et al., 2008), BAB (WESSEL; BLÜMEL; KLEIN, 2009).

Princeton Shape Benchmark and Toyohashi Shape Benchmark are more balanced benchmarks, covering a wide range of generic objects, which in turn makes them attractive for evaluating search performance in general. These benchmarks we chose to demonstrate the quality and scalability of our DCT descriptor. The PSB was used mostly due to its popularity, which allows comparison between descriptors found in the literature. The TSB was chosen due to its large number of models and classes, since we are most concerned with the scalability of our descriptor, for being a GPU accelerated algorithm it must support a large number of models at a interactive time. In addition, both benchmarks are suitable for general purpose 3D shape retrieval.

Class Name	#Objects	Class Name	#Objects	Class Name	#Objects	Class Name	#Objects	Class Name	#Objects
airship	33	ant	15	anvil	4	apple	13	aquarium	7
arrow	8	ashtray	6	automobile	151	axe	16	axel	12
ball	15	balloon	5	banana	11	barrel	17	barricade	6
bat	8	bathtub	9	batmobile	11	batwing	7	bear	11
bearing	4	bed	125	bed-lamp	117	bench	16	bicycle	75
big-small-d	5	billboard	12	biplane	69	block	7	boat	14
body	16	bone	18	book-set	55	bookend	4	boot	7
bottle	161	bowling	4	boxroom	10	brain	5	bridge	10
bucket	14	bug-feathered	10	bug-humanlike	13	building	35	bus	108
bus-shelter	4	bush	9	bust	18	butterfly	3	cage	4
camel	4	camera	7	can	10	candle	14	cannon	20
car-body	26	carousel	4	casket	4	cassette-tape	3	castle	18
cat-scan	3	cd-case	4	chain	4	character	248	chess	54
chess-set	5	chest	8	chip	29	church	10	cigarette	8
city	34	cleaver	17	clip-board	3	clock	43	closed-laptop	15
closed-piano	75	coffee-maker	6	column	27	commercial-plane	45	compass	12
component	14	computer-body	13	computer-keyboard	114	computer-monitor	12	container-truck	83
convertible	27	couch	40	covered-wagon	6	cow	8	crocodile	4
crypt	4	cube	6	cubes	10	cup	20	darts	10
deer	5	desk	27	desktop	16	dinosaur	35	diskette	14
dog	18	dolphin	22	door	45	door-knob	4	dragon	9
dragonfly	6	drawer	61	drill	7	driver	16	driver-head	9
drum-set	103	ear	3	earth	10	elephant	5	enterprise	80
extinguisher	7	eye-glasses	33	face	33	falcon	7	fan	37
felidae	26	fence	16	fighter-plane	234	fireplace	7	fish	143
flag	16	flip-phone	23	floor-lamp	101	flower	55	flying-balloon	36
flying-bird	42	flying-saucer	19	fork	7	four-legged-chair	186	frame	20
frog	5	game	6	gazebo	6	gear	16	geographic-map	55
glass-with-stem	40	glider	52	grape	5	grave	51	grid	9
griffin	3	guillotine	4	guitar	148	gun	120	hair	7
hammer	18	hand	36	hand-bell	4	hand-drill	4	hand-light	12
hang-glider	16	hat	15	head	116	heart	4	helicopter	98
helmet	30	hemisphere	5	horn	15	horse	78	horseshoe	3
hourglass	13	house	50	human	314	hydrant	18	iceberg	7
inkwell	4	iron	5	jellyfish	3	jewel	3	joystick	15
kangaroo	4	key	7	knife	120	ladder	7	ladybird	4
landscape	33	lathe	4	leaf	18	leg	4	lift-car	10
light	9	light-bulb	17	lighter	4	lighthouse	8	lock	5
long-chair	4	magnifier	6	mailbox	7	maze	20	meteor	10
microphone	10	microscope	7	mixer	4	modern-chair	13	modern-folly	4
molecule	10	monkey	4	motorbike	142	mushroom	9	nocontainer-truck	135
noleg-chair	17	normal-phone	120	nunchuck	3	one-legged-chair	21	opened-book	32
opened-laptop	181	opened-piano	90	organ	12	pear	11	pedal	3
piano-board	27	pig	3	pincerlike	8	pipe	5	planter	3
pod-racer	8	pole	64	pool	18	pool-table	12	pot	28
potted-plant	84	power-pole	5	printer	9	propeller-plane	257	pump	6
pumpkin	5	pushpin	3	pyramid	12	race-car	35	rack	4
rail	6	rake	4	range	8	recognizer	7	recorder	14
refrigerator	12	remote	7	ring	9	ring-toy	5	robot-animallike	11
robot-arm	10	robot-fourleg	7	robot-humanlike	35	robot-joint	30	robot-noleg	14
robot-pod	9	robot-twoleg	26	rocket	185	rocking-chair	5	rocking-horse	3
room	40	saddle	5	satellite	46	satellite-dish	10	scanner	4
scissors	3	scorpion	4	seashell	11	shade	10	shark	20
shelf	126	shield	15	ship	60	shoe	13	shovel	13
sign	24	single-book	79	single-drum	88	sink	9	skateboard	5
skull	15	slide	5	slider-phone	30	slot-machine	8	smoke	7
snake	5	spacefighter	4	spacepod	7	spaceship	43	spaceshuttle	23
spacestation	21	speaker	8	spear	7	sphere	21	spider	12
spoon	55	spray	4	spring	8	stadium	9	stake	4
standing-bird	76	stapler	3	stationery	31	stealth	47	steps	16
stool	12	store	6	stove	4	study-lamp	106	stuffed-specimen	7
submarine	32	suitcase	10	supercar	8	swing	7	switch	4
sword	166	syringe	5	table	115	tank	53	tape-deck	4
telephone	37	temple	3	three-wheeler	11	tiefighter	49	toilet	8
tooth	4	torch	3	tower	4	traffic-light	7	train	143
trash	11	tree	120	triangle	4	turntable	5	turtle	7
tv	16	typography	31	umbrella	8	unicycle	4	unicycle-fork	13
vase	55	video-camera	11	video-tape	4	violin	39	violin-case	3
warbird	13	watch	16	water-jug	6	weathercock	4	whale	4
wheel	78	wheeled-chair	97	wind-chime	6	windmill	4	window	18
wrench	10	x-wing	15						

Figure 3.10: Toyohashi benchmark classification. Contains 10000 3D shapes classified in 352 categories.

4 DCT DESCRIPTOR

DCT is a statistical descriptor we are proposing in this thesis, guided by the desirable characteristics listed in Chapter 2. The most important feature of our descriptor is the invariance to translation, scaling and rotation, not requiring a pose normalization step, such as the well known PCA method (JOLLIFFE, 1986), which may cause alignment problems when applied models with similar geometry, as seen in Figure 4.1. Another important characteristics are the speed – since it can be efficiently implemented using GPU techniques (see Chap. 4.5) –, and the lack of restriction concerning enclosing or orientation of the polygon mesh.

We based our descriptor on two measures: a depth complexity (DC) distribution and a thickness (T) distribution, explained in Secs. 4.1 and 4.2 respectively. These two measures attempt to capture the geometric and topological features of a shape respectively, thereby increasing the descriptor's discriminative power.

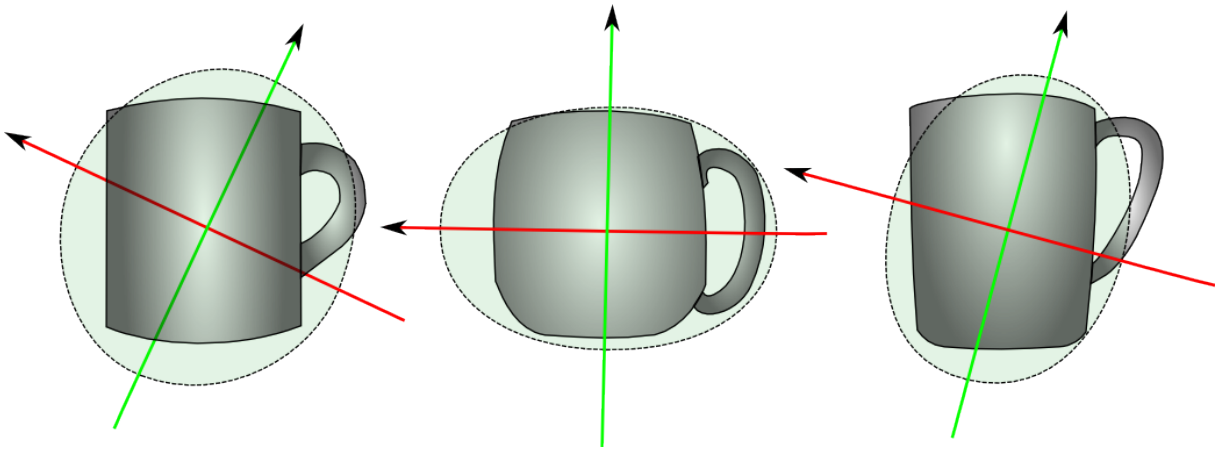


Figure 4.1: A collection of mugs drawn with their principal axes. Despite the similarity among the models, the principal axes orient the models in very different ways. Adjusted from (FUNKHOUSER et al., 2003)

4.1 Depth Complexity Distribution Descriptor

The first component of the DCT is a histogram of *depth complexity* (DC) information. Depth complexity is a term commonly used in the field of mesh transparency computation (MAULE et al., 2011), that originally represents the number of fragments processed for each pixel on the screen.

To explain the DC, consider a ray $\mathbf{r}$ in 3D space. The depth complexity of a 3D model with respect to $\mathbf{r}$ is defined as the number of intersections of $\mathbf{r}$ with the model. Now, con-

sider a bounding sphere around our given shape. Each ray that intersects the shape has also two intersection points with the sphere, described by their polar coordinates θ_1, γ_1 and θ_2, γ_2. Hence, the depth complexity for a given ray $\mathbf{r} = (\theta_1, \gamma_1, \theta_2, \gamma_2)$ can be encoded as a function $DC(\theta_1, \gamma_1, \theta_2, \gamma_2) \rightarrow \mathbb{N}$ of four variables. The 4D shape function DC, defined over the space of oriented rays, thus captures the depth complexity of the entire shape. Figure 4.3(a) shows the DC function values along three rays $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ for a simple shape, sketched on 2D for illustration purposes.

We next compute a depth complexity distribution by using an uniform sampling of the DC function over its ray space. To obtain an optimal balance between sampling density and computational speed, we tested this by computing distributions for varying sample counts on several models and using them further in performing CBR of 3D shapes. From our experiments, $500K$ ray samples provided sufficient resolution while being fast enough to compute (see Sec. 4.5). Figure 4.3(b) shows such a distribution for the shape in Fig. 4.3(a). Note that geometrically closed (watertight) shapes have only even values in their distributions, since intersections with the shape occur in pairs for all rays. Special cases such as tangent rays touching a vertice are infeasible due to the discrete nature of our algorithm.

Since each pixel represents a ray in the 3D camera space, we can efficiently compute DC in image space, by counting the fragments for each pixel during rasterization. Figure 4.2 shows this process. Section 4.5 explains in more detail how this process can be implemented efficiently relying on the GPU. Although, for collecting the DC information, the ray r does not have to be computed, but it can be calculated by unprojecting the pixel.

The DC distribution has several desirable properties. First, it is invariant to translation, rotation and scaling. Secondly, its computation does not impose any constraints on the mesh type used to represent the input shape (*e.g.* triangle, general- polygon, polygon soup) or mesh quality (clean structure *vs* having duplicate or T-vertices or very thin triangles). This allows its direct application on all models present in typical shape databases without requiring expensive clean-up operations. In contrast, shape descriptors that perform geometric computations on the mesh *surface*, such as (OSADA et al., 2002; KAZHDAN; FUNKHOUSER; RUSINKIEWICZ, 2003; HILAGA et al., 2001), may require preprocessing to enforce such mesh constraints.

4.2 Thickness Distribution Descriptor

The second component of the DCT descriptor is the thickness (T) distribution. Unlike the DC descriptor, thickness discriminates shapes according to their geometrical properties. As a

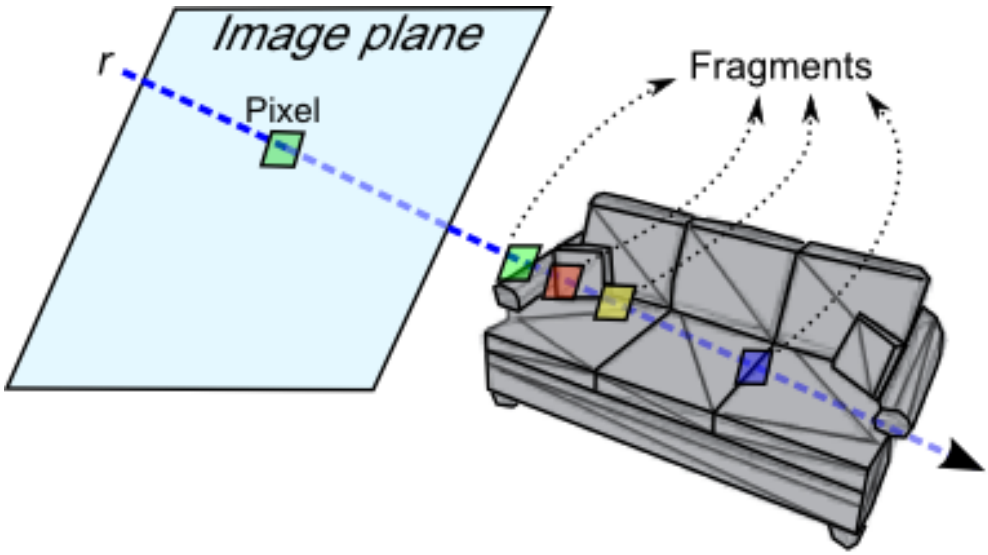
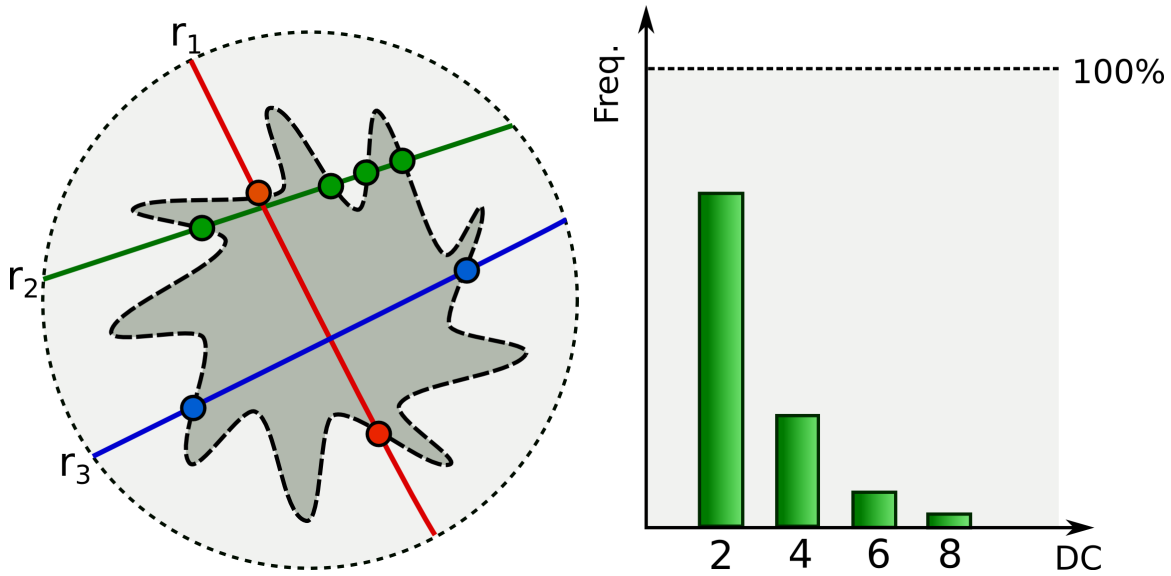


Figure 4.2: Example of how the DC can be computed for a ray r by counting the fragments in image space. In this example the ray r has a $DC = 4$.



(a) (b)

Figure 4.3: (a) DC function sampled for three rays $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ for a simple shape. For each ray, the number of colored shape-ray intersection dots gives the DC value. (b) A possible distribution of the DC in (a).

starting point, we use the definition of shape thickness along a given ray $\mathbf{r}$ given in (LIU et al., 2004): A ray $\mathbf{r}$ having $2n$ intersections $\mathbf{x}_i$, $1 \leq i \leq 2n$, with a shape generates n thickness values $t_i = \|\mathbf{x}_{2i+1} - \mathbf{x}_{2i}\|$, $1 \leq i \leq n$ as being the distances between consecutive intersection points on the ray. We stipulate that $n \leq 8$, so we compute only the values of thickness for $t_1, ..., t_8$.

This arguably does not decrease the description quality, since the average DC in a common model from PSB is about 4 intersections. This means 8 is a high enough value to capture any deeper details about the mesh. For detailed meshes with a much higher average DC, such as buildings with furniture inside or CAD models, for instance, inner parts of the model eventually would be reached by rays from another directions, avoiding losing information of deeper parts. Moreover, this restriction allows an efficient and simple GPU-based implementation for the thickness descriptor (see next Sec. 4.5).

Liu *et al.* proposed a related technique called Directional Histogram Model (DHM), where they define thickness as the distance between the first and last ray-shape intersection points (LIU et al., 2003). DHM can be easily computed in a two-pass OpenGL-based rendering process – front-facing polygons are rendered first, followed by back-facing polygons (for convex shapes). This can be easily implemented in the fixed OpenGL pipeline, by placing the camera at the desired position (sphere sample point) and orienting it towards the center of the bounding sphere. However, as (LIU et al., 2004) argues, the main disadvantage of DHM is the loss of internal geometrical shape properties, which occurs for complex models having concavities, such as buildings or CAD models. A similar GPU-based technique to the DHM was used to compute the medial axis, or curve skeleton, of 3D shapes (KUSTRA; JALBA; TELEA, 2013), with similar limitations for concave shapes. Separately, we note that more accurate and geometrically-motivated methods to estimate the local thickness of a shape exist, such as using the distance between a surface point and its closest medial surface point (TELEA; JALBA, 2011). However, such methods require the computation of the 3D medial surface, a process that is slow and delicate.

Similarly to the DC descriptor, we next convert the thickness (T) values to a distribution, using the same number and position of sample rays. Thickness is obtained during the same rendering pass that computes depth complexity, thus adding only a small overhead to the total computation (see Sec. 4.5). However, in contrast to the DC measurement, a scaling normalization must be used for the T measurement. For this, we proceed as follows. First, we define the distance between z_{near} and z_{far} of the OpenGL near and far clipping planes as being the diameter of the bounding sphere around the shape. This can be done by casting a ray along the viewing direction and retrieving the intersections against the bounding sphere. The two intersection points are then used to define the near and far clipping planes. Figure 4.4 shows this process.

This way, the z (depth) values, retrieved from the OpenGL Z-buffer, are all in the same range for different scales of the same shape. A second aspect to consider is the fact that the Z-buffer

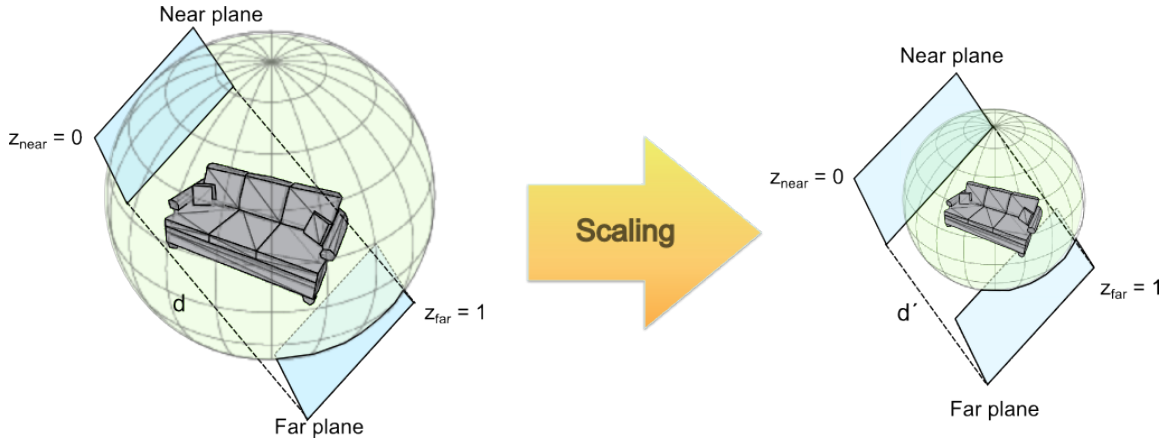


Figure 4.4: Example of how the scale normalization for Thickness (T) can be done only by carefully selecting the near and far planes of an orthographic camera in OpenGL, since Z values are given relatively by the distance between the Z_{near} and Z_{far} planes. However, the Z values grows by a nonlinear scale due to precision issues, which means a linearization process is required.

uses a nonlinear scale, which gives more precision objects near the near clipping plane. Hence, we find thickness values by converting 'raw' z values from the Z-buffer to linear depth values z_{linear} by

$$z_{linear} = \frac{(2 * z_{near})}{(z_{far} + z_{near} - z * (z_{far} - z_{near}))}. \tag{4.1}$$

4.3 DCT Descriptor

Our final DCT shape descriptor is a 2D histogram containing depth complexity and thickness information obtained by joining the 1D DC and T histograms in Secs. 4.1 and 4.2. The DCT histogram correlates frequency values of both histograms to obtain a better discrimination than using each histogram separately. Figure 4.5 (bottom row) shows the 2D DCT histogram for a typical model, with thickness mapped to the $x-$axis and depth complexity mapped to the y-axis respectively, and frequency (counts) mapped to color, using a blue-to-red color map.

4.3.1 1D Composed DCT Descriptor Variation

Besides using a 2D histogram structure to build our DCT descriptor, we also tried the approach of using two 1D histogram of depth complexity and thickness. The main idea is to obtain two dissimilarity values separately from each one of the distributions and compute the final dissimilarity value by a weighted average.

Advantages of representing the descriptor as two 1D histograms are that the distances be-

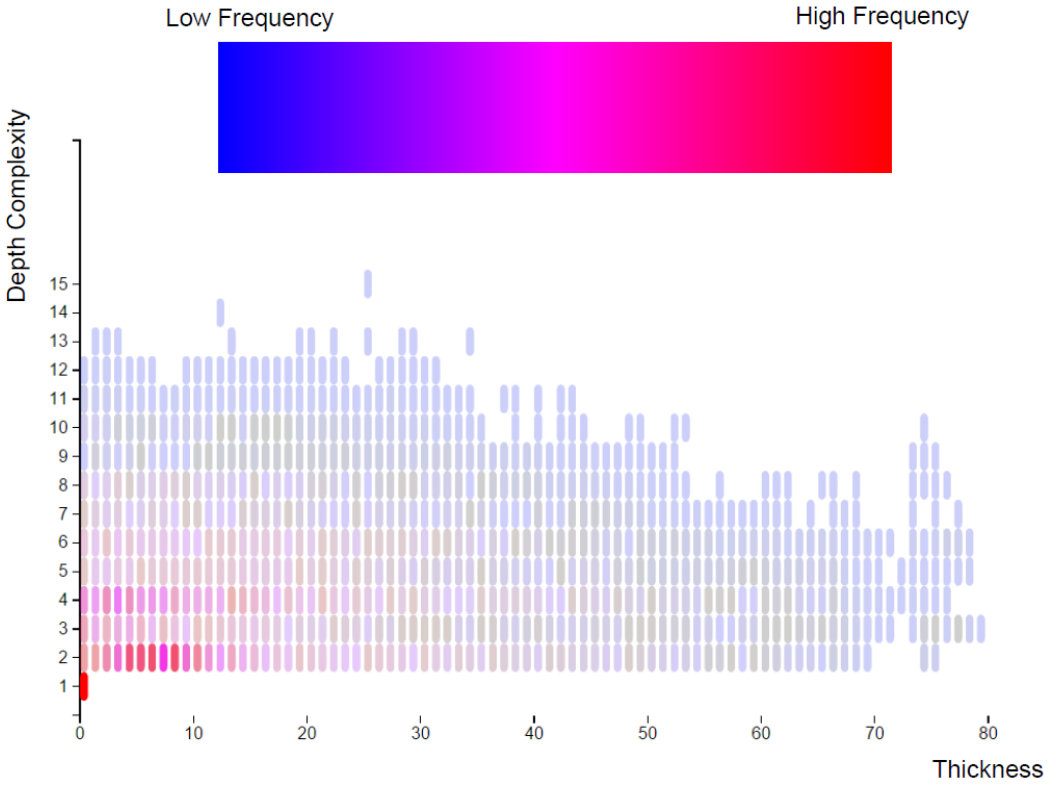


Figure 4.5: A plot of our 2D DCT Histogram, containing information of DC ($y-$axis), T ($x-$axis) and frequency (color scale from blue to red).

tween 1D distributions are easier and fast to compute; and the users can tweak the preliminarily weights from both 1D histograms to set the contribution of each measure (depth complexity or thickness). Figure 4.6 shows experimental result example considering 10 variations in weights of DC and T. However, the reason we chose the 2D histogram DCT, is because the 1D version loses the correlation between the data. This extra information allows the DCT to achieve a better discrimination performance.

4.4 Dissimilarity Measures

Once the DCT descriptor is computed for a set of 3D shapes, dissimilarities between shape-pairs can be computed by using various distance metrics based on the shapes' descriptors. Common distance metrics include the Minkowski L_N norms, match distances (SHEN; WONG, 1983b), the earth mover's distance (EMD) (RUBNER; TOMASI; GUIBAS, 1998a), and the Hellinger distance (HELLINGER, 1909), which is closely related to the Bhattacharyya distance (BHATTACHARYYA, 1943). Given two DCT histograms H_1 and H_2 and a distance metric d, we next assume that distance values are normalized, *i.e.* $d(H_1, H_2) \in [0, 1]$, with 0

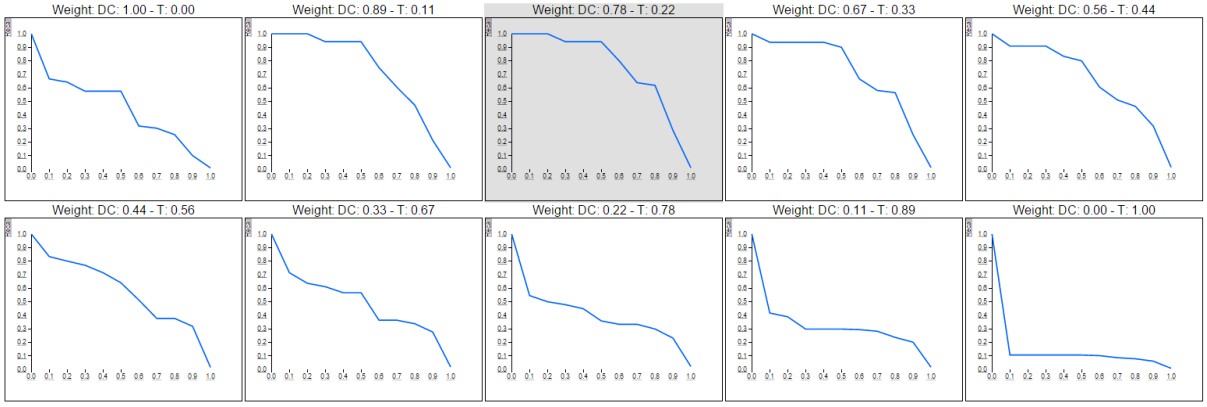


Figure 4.6: Example of 10 precision-recall plots considering 1D histogram of both depth complexity and thickness, generated by varying the weights of each component. DC weight decreases from right and bottom and Thickness decreases from left and top. Weights of DC: 0.78 and T:0.22 (greyed plot) achieved the best precision-recall plot.

implying perfect similarity and 1 maximal dissimilarity respectively.

For our CBR use-case, we tested the L_1, Hellinger, Chi-square, correlation, and EMD distance metrics, aiming to optimize the trade-off between computational performance and retrieval precision. Expressions for all tested distance functions are given in Chapter 2. The best results were given by the Hellinger distance (see Sec. 5 further).

4.5 Efficient GPU implementation

We next describe the efficient GPU-based computation of the DC and T distributions. The proposed algorithm makes the DCT descriptor scalable for large databases containing thousands of complex 3D shapes. Our approach has three steps: shape sampling (Sec. 4.5.1), data collection (Sec. 4.6), and normalized histogram construction (Sec. 4.6.1).

4.5.1 Shape sampling

Points are uniformly sampled over the bounding sphere around the shape. At each sample point, an OpenGL camera is pointed towards the sphere center (see sketch in Fig. 4.8 a). For each camera position, the 3D object is rendered, and the DC and T values are computed. Camera parameters are configured at each rendering pass as follows:

Projection: We use orthogonal rather than perspective projection to avoid precision loss due to perspective distortions that occur in distant parts of the scene, *e.g.*, triangles becoming very

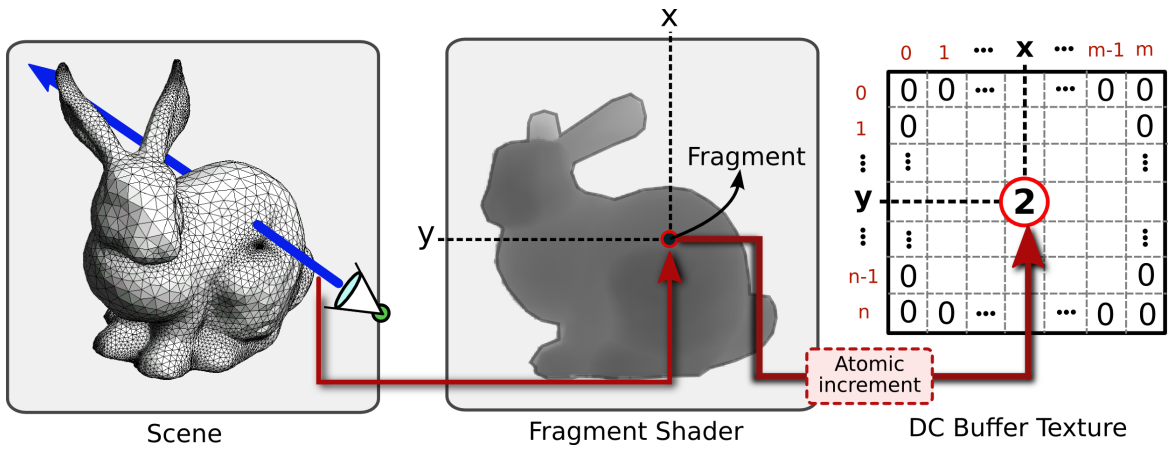
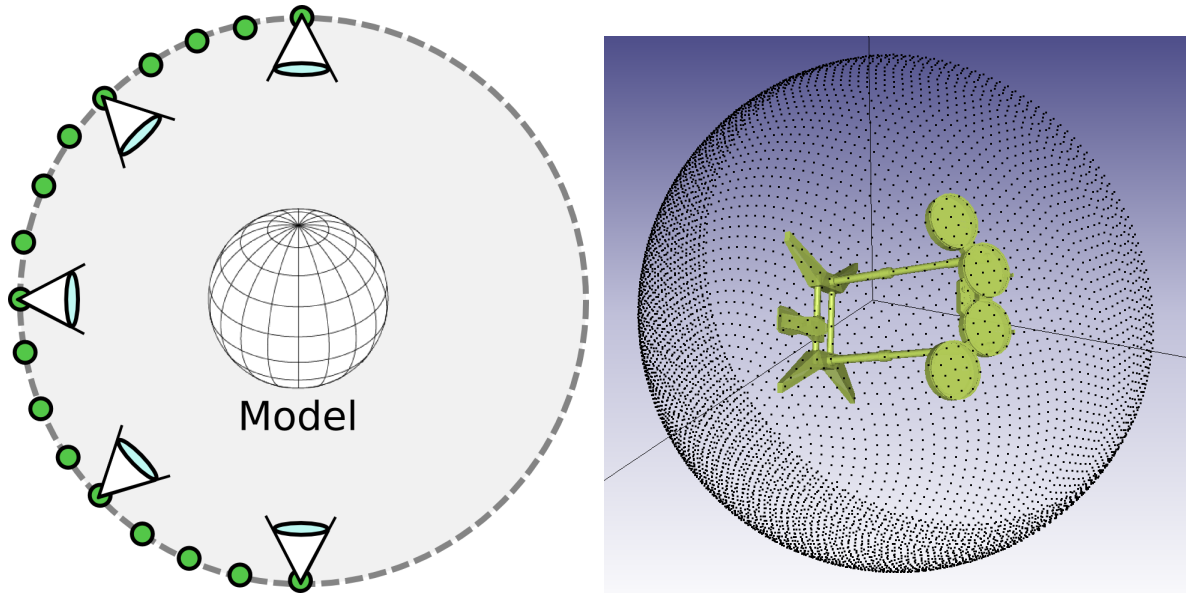


Figure 4.7: GPU computation of DC for a given view direction. The shape is rendered for each view direction. Using a fragment shader, the number of fragments for each pixel position (x, y) is counted and saved into a texture. The texture is bound to an OpenGL image unit, which allows to perform atomic increments inside the shader program.



(a) (b)

Figure 4.8: Computing the DC distribution. (a) Camera placement at sample locations. (b) Actual sample positions shown as black dots.

small. As such, each rendering pass samples rays that are parallel to the viewing direction.

Frustum size: We use a view frustum that best fits the shape to prevent losing depth precision during rendering. For this, we use a 3D bounding box of the input shape to define the camera coordinate system and also set the frustum size.

Resolution: The viewport resolution offers a trade-off between precision and speed. A reso-

lution of 256^2 pixels gave good results for all tested models. Larger resolutions are better for high-precision queries for a database having many complex objects with small detail polygons.

Clipping planes: As mentioned in Sec. 4.2, thickness values must be normalized for different shape scales. To obtain depth values within the same fixed range, we set the near and far clipping planes to (a) encompass the entire object and (b) have fixed values for all viewpoints. For this, we set the distance between the near and far planes to the diameter of the bounding sphere around the shape.
Finally, we note that the view up vector, which controls the camera rotation around the viewing direction, is irrelevant since both depth-complexity and thickness values for a 3D shape depend only on the viewing direction and distance to object, and not the camera rotation around this direction.

4.6 Data collection

In each rendering pass for a different viewpoint, OpenGL fragment shaders are used to compute the DC and T values for that viewpoint. Our shader uses a texture and an array of 2D textures. The first texture has only an integer channel to store the DC values. The array of 2D textures has 8 floating-point layers to store the z values of the 8 consecutive intersections, upcoming intersections are ignored. We found this being a reasonable solution since dynamic arrays can not be passed into a shader program. The texture and the array are bound to OpenGL image units, allowing us to perform safe atomic writes into the textures inside the shader program [1]. Figure 4.7 shows the procedure used to find the DC values for each ray by counting the number of fragments rendered at each pixel of the viewport. Similarly, for the thickness T, the depth value of each fragment is used to compute the distance between consecutive fragment-pairs. The only requirement is to convert the depth value to obtain a linear scale by the Equation 4.1.

4.6.1 Normalized histogram construction

In the last step, data collected from all sampled viewpoints is normalized by the total number of drawn rays, creating the DCT histogram, containing both T and DC values. These steps are performed after the renderization of each sampled viewpoint to avoid creating many simultaneous textures. This step is performed on the CPU. Apart from this step, the DCT is entirely

[1] https://www.opengl.org/wiki/Image_Load_Store

computed on the GPU. Since the texture resolution is relatively small (see Sec. 4.5.1), computing the final histogram normalization on the CPU does not incur significant penalties (see performance results in Chap. 5).

5 EXPERIMENTAL RESULTS

Several experiments were designed to evaluate the robustness, shape discrimination and efficiency of the DCT shape descriptor. The implementation was written in C++, using g++ version 4.7. Results were measured on an Intel Core i7 PC, with 12 GB RAM, running Linux Ubuntu 12.04.

Accepted input formats for 3D models include .obj or .off, which are common formats for publicly available 3D shape databases. For our tests, we used the Princeton and Toyohashi benchmarks. Princeton has 907 models for testing and 907 models for training for typical shape retrieval algorithms. Since the DCT algorithm does not need a training step, all 1814 models were used in our tests. These models are further divided into 80 different shape-classes. Toyohashi has 10000 models, organized in 352 classes. During testing, only classes containing at least 20 models were used to avoid a small number of samples in the MAP plots. Typical classes include humans, plants, airplanes, quadrupeds, chess pieces, among others. The list of distance metrics used to compare DCT descriptors is given in Tab. 5.4). All metrics, except the simple L_1 norm, were computed using the optimized implementations provided in the OpenCV library [1]. The evaluation of the descriptors uses three criteria: robustness (Sec. 5.2), performance (Sec. 5.3), and efficiency (Sec. 5.4).

5.1 3D Search Engine Experiment

Our experimental search engine includes model from both TSB and PSB databases. The models are listed by categories, allowing the user to pick any model from the databases to use as query. After the query is selected, the user can see a ranked list of models ordered by similarity. The categories and classification of the models are checked using the provided classification file from both Toyohashi and Princeton Benchmarks. Screenshots showing some of the main features can be seen in Figure 5.1.

The search engine also provides some options to tweak parameters for the DTC. It is possible to change the type of histograms it can use, a 2D histogram containing both DC and T data, or use the 1D version of the descriptor, allowing the user to specify a weight for the DC and for T.

By clicking on one model, the user can a plot of the histograms use for comparison, which allow further analysis.

The search engine does not use indices or any advanced structure to avoid the number of comparisons.

[1] http://opencv.org

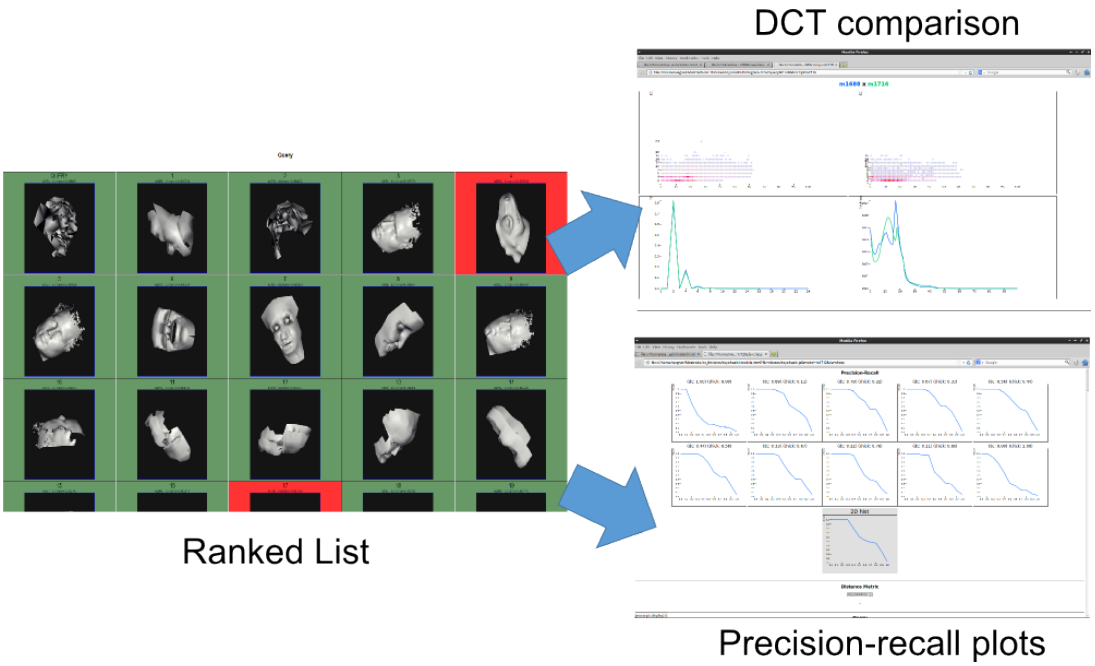


Figure 5.1: Our experimental 3D search engine. (left) A ranked list is presented, given a user query. The retrievals in green shows the relevant results, the red ones are the non-relevant. (right) Shows the possibility of analyze the results seeing side-by-side plots of the shape descriptors. Also precision-recall plots are constructed for each search, showing the performance of the descriptor. The results are given in real-time.

5.2 Robustness

The robustness of the DCT descriptor was measured by testing its invariance to affine transformations and mesh resolution issues. Three generic shapes from the Princeton database were selected and subjected to one of the following transformations:

- rotations of 45 degrees around the z, y and z axes respectively;
- anisotropic scaling by factors of 2.5 in the x-axis and 0.25 in the z-axis;
- mirroring against the $x-y$, $y-z$, and $x-z$ planes;
- surface subdivision using the Catmull-Clark algorithm (CATMULL; CLARK, 1998) with two iteration levels.

The 2D DCT histograms were computed after applying each transformation. As shown in Fig. 5.2, these histograms remain almost unchanged after each of the tested transformations, thus demonstrating the robustness of DCT.

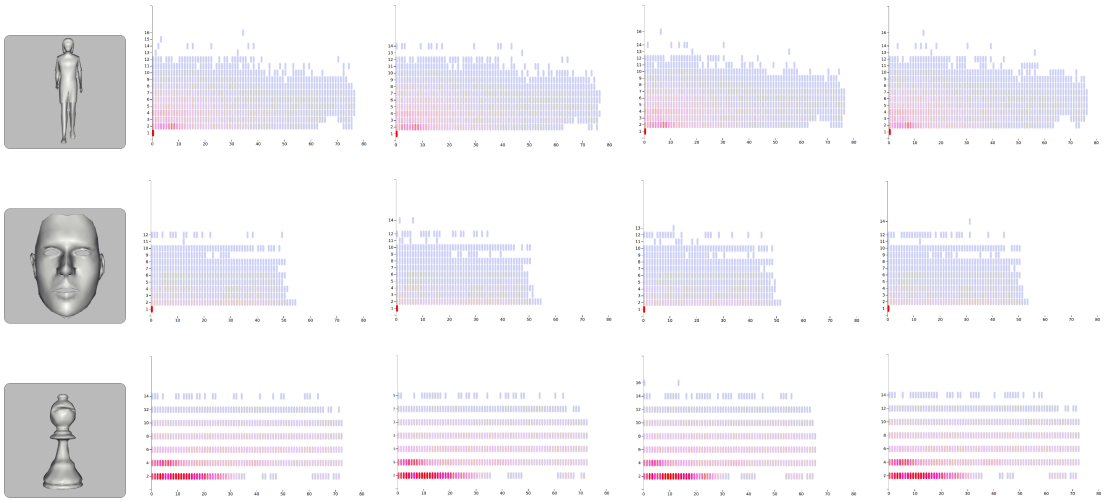


Figure 5.2: (a) Input test models. DCT histograms after applying rotation (b), anisotropic scaling(c), mirroring (d), and shape subdivision and smoothing (e). In each row the histograms are similar before and after transformations (Sec. 5.2).

5.3 Retrieval Performance

Measuring the retrieval performance of DCT requires querying the database and evaluating precision-recall plots. For this, we implemented a shape retrieval tool to generate the average precision-recall plots. As reference, we used the shape-class descriptions provided in the Princeton Shape Benchmark. Descriptors are compared using the Hellinger distance, which gave the best trade-off between quality and speed (see Sec. 4.4).

Plots of the separated components of the DCT descriptor can give insight on how the DCT changes according to a given shape. Figure 5.3 shows the DC and thickness (T) 1D histograms, as well as the combined 2D DCT histogram for three different models. The DC histogram is similar for the panther and the chess piece, as both shapes have few concavities and an average local thickness. The DC histogram for the tree is quite different, as this shape has a very different topology. In this example, the DC histogram is not enough to discriminate the three shapes. In contrast, the T histogram is more discriminative, revealing the fact that the geometric (local thickness) properties of these objects are quite different. The DCT histogram for the tree looks quite different from the DCT histograms of the other two shapes, which reflects its combined discriminatory power.

Figure 5.4 brings additional insights in the discriminatory power of DC and T histograms. Five objects belonging to the same shape-class were tested, for the classes *humans* and *chess pieces*. As can be seen, the same-class DC and T histograms are quite similar. Differences exist between the DC histograms and T histograms of different-class objects. However, if we

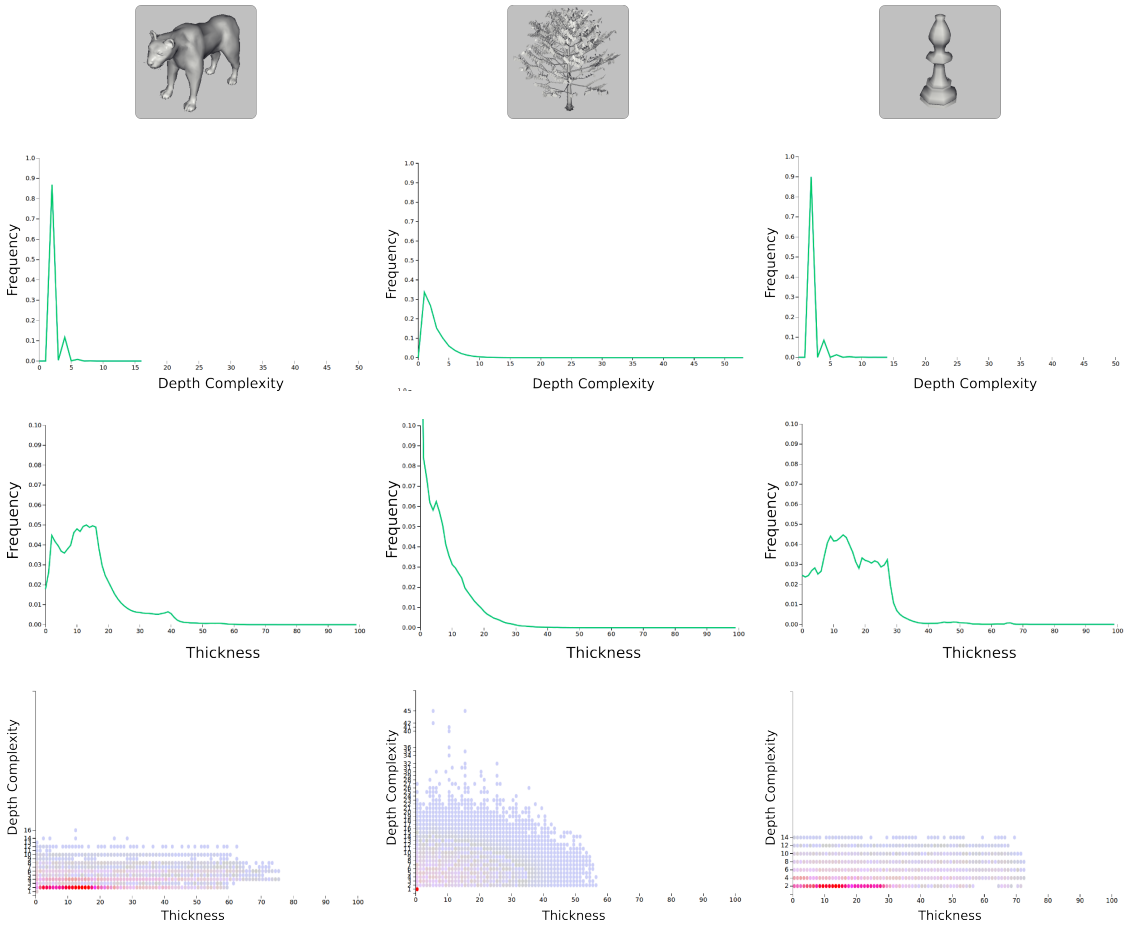


Figure 5.3: Comparison of DC, T, and DCT histograms (bottom three rows) for three objects of typical classes in the Princeton Shape Benchmark (shown in the top row). The DCT histogram combines the discriminative power of both DC and T histograms (Sec. 5.3).

combine the DC and T histograms in the DCT descriptor, an increased discriminatory power is obtained. Figure 5.5 shows how the DCT descriptor separates three objects in the *humans* and *chess pieces* classes.

Figure 5.6 shows the precision-recall plots for queries pertaining to objects located in six different shape classes. The best result was achieved for the *faces* class. One explanation for this result is the fact that face shapes are obtained from 3D scanners. As such, the shapes have a simple topology consisting of an open surface, and a DC histogram having a peak around the value 1 (many rays have a single intersection with the shape). The DC histograms for other shapes have quite different forms, since such shapes are typically closed (watertight). As such, the DC part of the DCT descriptor is very successful in separating faces from other shapes.

The query evaluation was more challenging for shapes in the *trees* and *quadrupeds* classes,. While most queries returned good results, queries for some more exotic shapes in these classes did not return many relevant results. Precision-recall plots have a high variance for these classes.

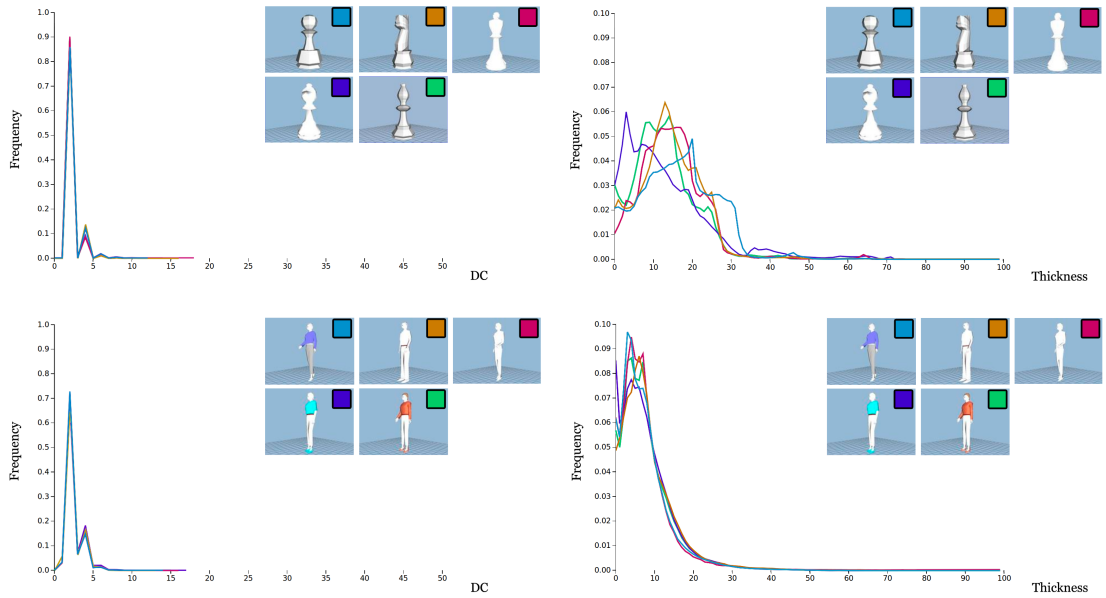


Figure 5.4: DC and T histograms for five shapes in the classes *humans* and *chess pieces*. Same-class objects have similar histograms (Sec. 5.3).

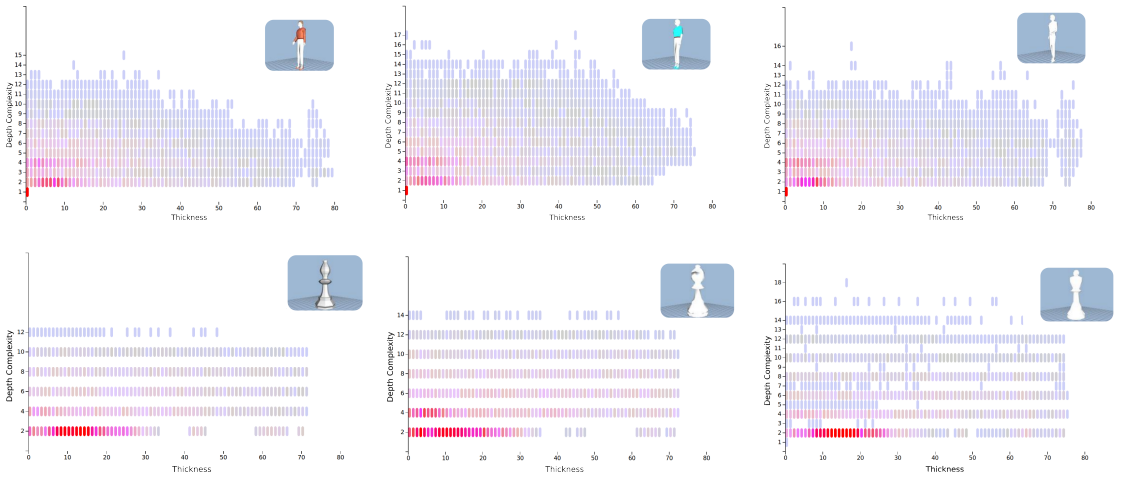
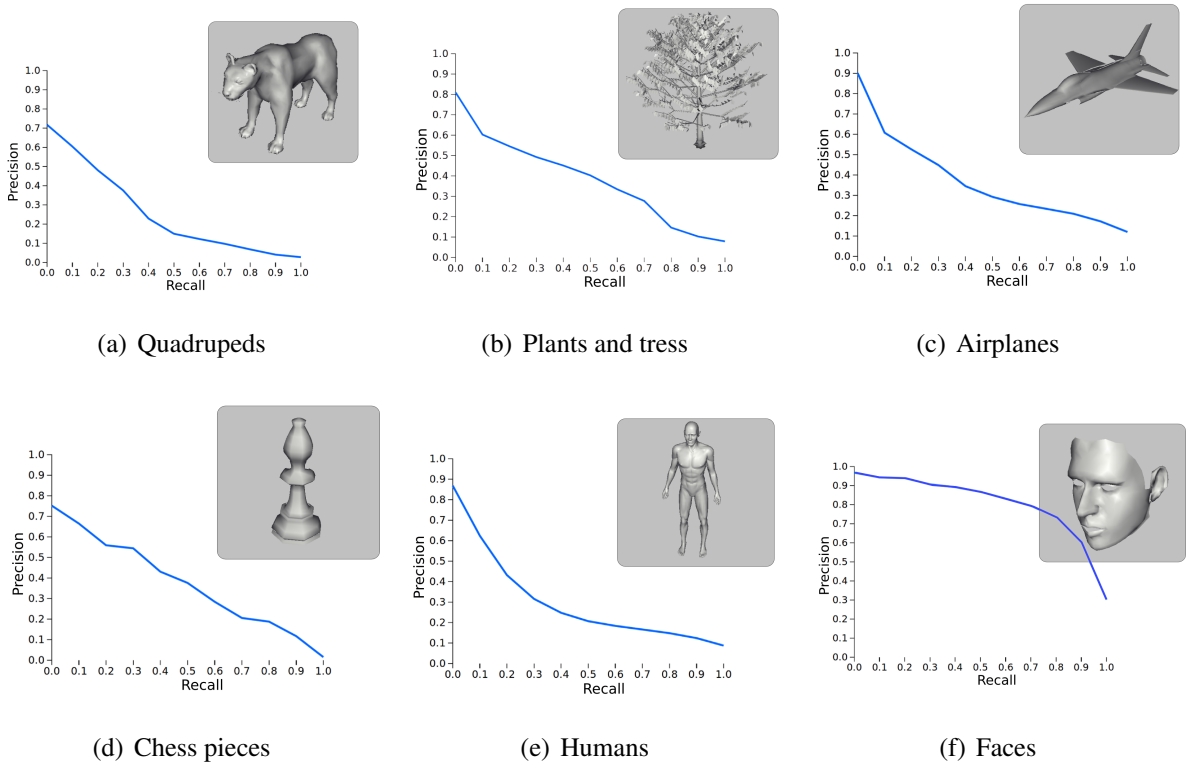


Figure 5.5: DCT histograms for objects of the *human* and *chess pieces* classes in the Princeton Shape Benchmark. The histograms clearly discriminate between the two classes. Although shapes in both classes have many rays with $DC = 2$, the chess pieces have a higher thickness T (Sec. 5.3).

Figure 5.7 shows queries that exemplify this behavior: Querying a typical plant gives quite good results. In contrast, querying an outlier such as a dead tree returns no relevant results at all. These examples show how subjective can be the classification of models, and how the classification can have a large influence on the final results.

Similar tests were performed on the Toyohashi shape benchmark, which contains 10000 shapes organized in 352 classes. Figure 5.8 shows the average precision-recall plots for the



(a) Quadrupeds (b) Plants and tress (c) Airplanes

(d) Chess pieces (e) Humans (f) Faces

Figure 5.6: Precision-recall plots of six shape-classes in the Princeton Shape Benchmark using the DCT descriptor. Descriptor dissimilarity is computed with the Hellinger distance (Sec. 5.3).

same shape-classes as in Fig. 5.6 (Princeton benchmark). Precision-recall results are worse for the Toyohashi benchmark than for the Princeton Benchmark. Upon closer analysis, it was observed that the Toyohashi benchmark contains degenerated models, and many models were disassembled or had many missing or duplicated triangles. Such models lead to spurious or missing ray-shape intersections, thereby introducing noise in the DC and T histograms.

Finally, for further analysis, we run our descriptor for the same classes tested by Tatsuma *et al.* (TATSUMA; KOYANAGI; AONO, 2012) to evaluate their DB-VLAT descriptor. In the specific classes discussed in their work DB-VLAT presented better precision-recall results than DCT. The main reason is the lack of robustness of our technique when 3D models present missing or duplicated triangles, which happens in the majority of the models of these classes. In addition, statistical descriptors do not have a great power of discrimination since they lose spatial information about the measures, as stated in (VRANIC, 2003).

In 2008, (TANGELDER; VELTKAMP, 2008) computed precision-recall plots for some of the most popular descriptors using the Princeton Shape Benchmark (PSB). The plots are presented in Figure 5.10a, they were averaged for every 3D shape object from the PSB. It is possible to note, intuitively, by analyzing the plots, that our descriptor performance is in be-

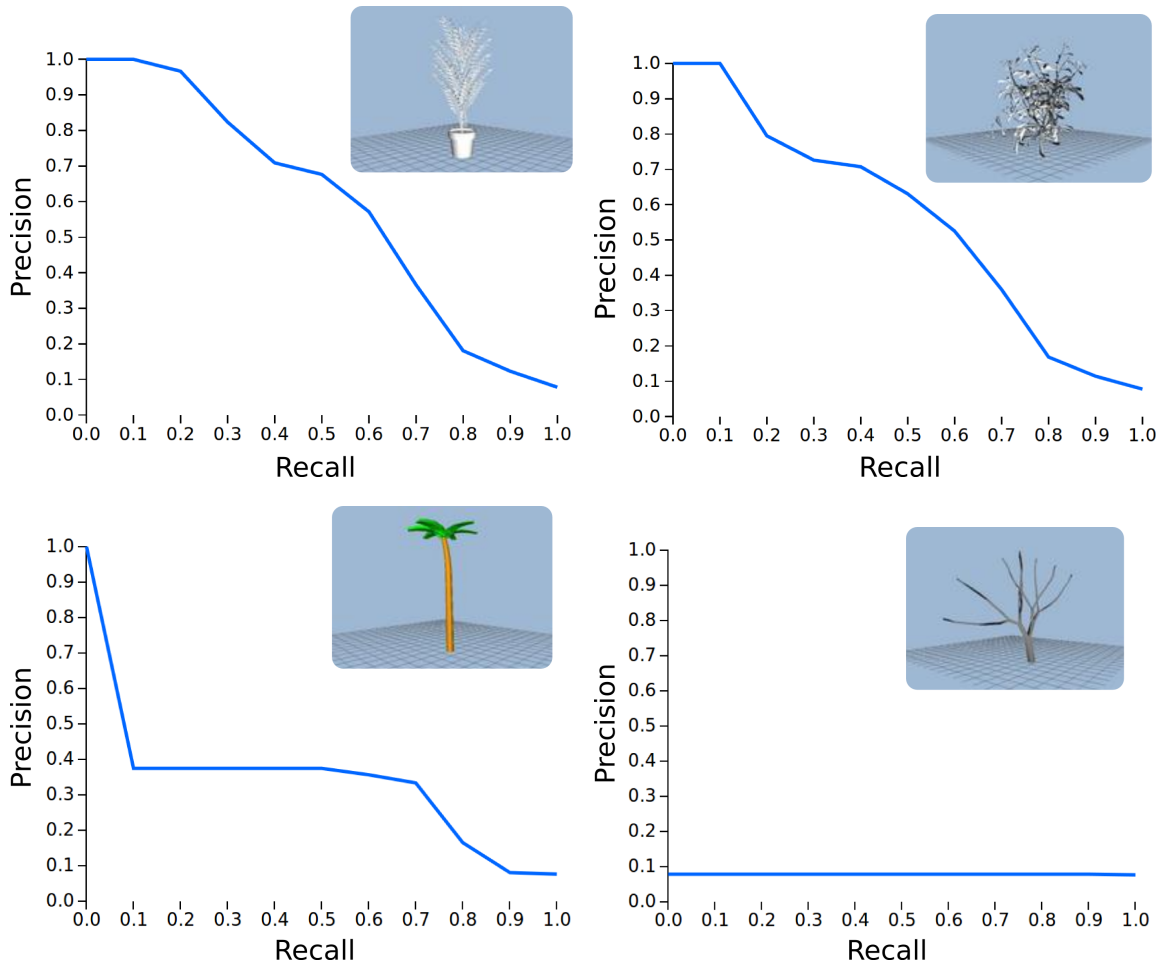
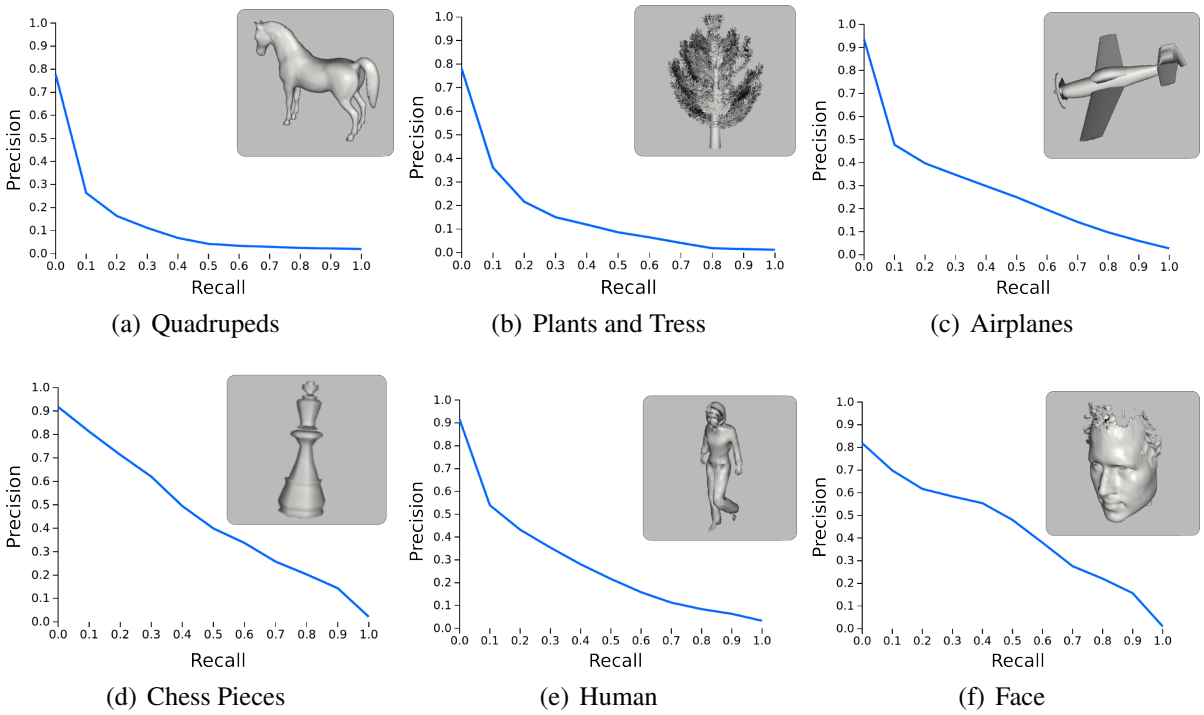


Figure 5.7: Precision-recall variance as a function on the query object. Selecting meaningful objects of the class lead to better results (Sec. 5.3).

tween the "CEGI" and "SECTORS" descriptors. We consider a good performance considering the discrimination restrictions of statistical descriptors.

Another measures of efficiency such as *Nearest Neighbor* (NN), *First Tier* (FT) and *Second Tier* (ST) are presented in Table 5.1. The values obtained also show a low performance compared with the state-of-the-art descriptor DB-VLAT. Their descriptor obtained a performance 41% better considering the NN measure, and 150% and 400% for the (FT) and (ST), respectively, which means most relevant results are shown next to the top. The discrimination of DB-VLAT outperforms our results for the selected classes of models, considering the discrimination factor.

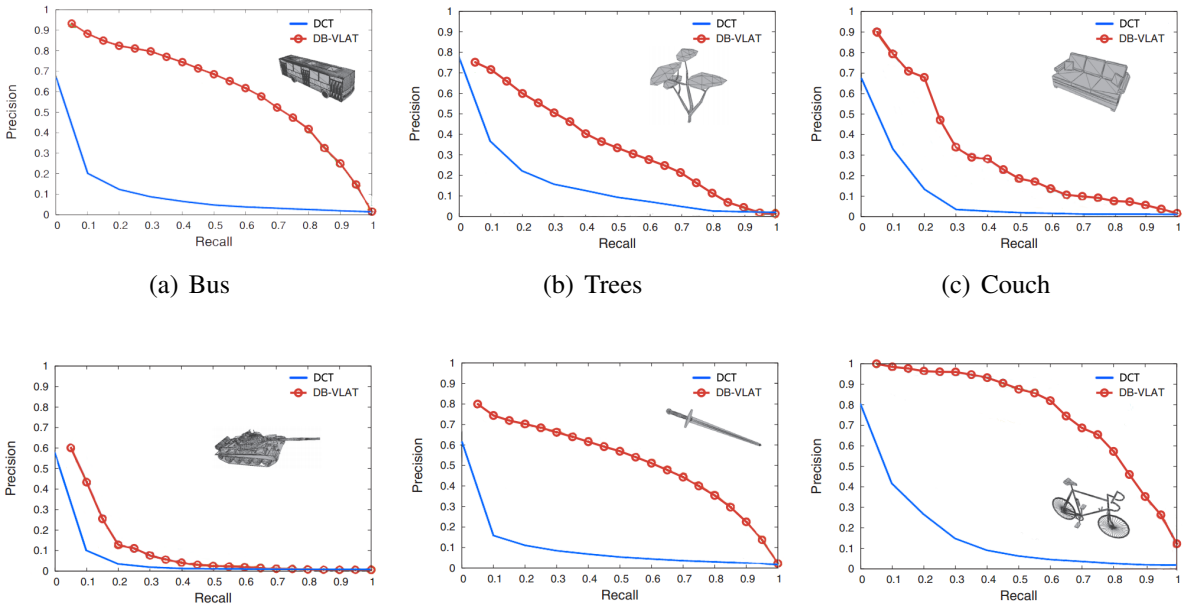
More conclusive comparisons could not be performed since we were not able to acquire an implementation of their descriptor. In addition, no information was given about the computational cost and scalability of the DB-VLAT descriptor. Although, retrieval quality is very important, factors such as speed and ease of implementation are also essential for a scalable



(a) Quadrupeds (b) Plants and Tress (c) Airplanes

(d) Chess Pieces (e) Human (f) Face

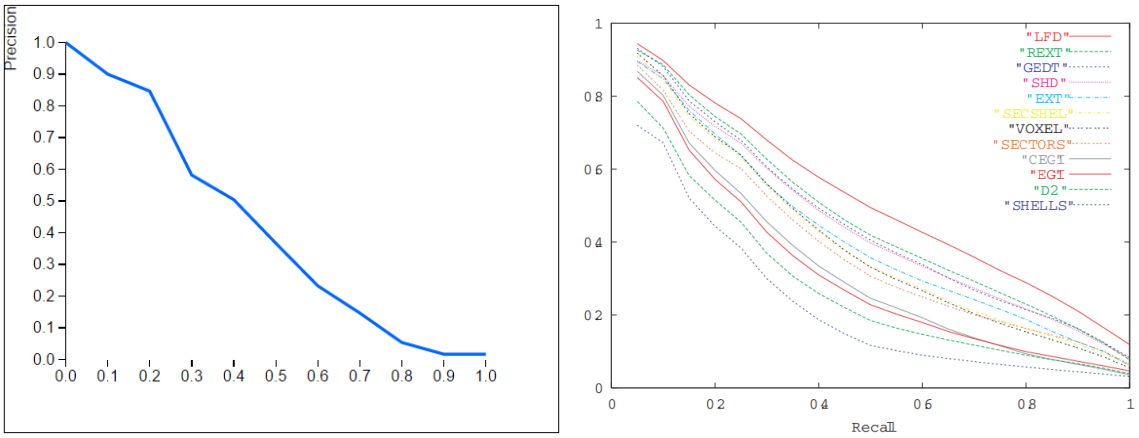
Figure 5.8: Precision-recall plots for six typical classes of the Toyohashi Shape Benchmark using the DCT descriptor. Descriptor dissimilarity is computed with the Hellinger distance (Sec. 5.3).



(a) Bus (b) Trees (c) Couch

(d) Tank (e) Swords (f) Bicycle

Figure 5.9: Precision-recall plots obtained by Tatsuma *et al.* using his DB-VLAT descriptor. The descriptor was tested for 6 different classes of shapes of the Toyohashi Benchmark. The plots were extracted from (TATSUMA; KOYANAGI; AONO, 2012).



(a) (b)

Figure 5.10: Precision-Recall curve performance comparison of shape descriptors with PSB. (a) Shows the performance of our DCT descriptor. (b) Shows the performance of popular descriptors in the literature (extracted from (TANGELDER; VELTKAMP, 2008)). The most important measured descriptors, from top to bottom are: (LFD) Light Field Descriptor (CHEN et al., 2003), (DEXT) Density-Based Shape Descriptor (AKGUL et al., 2006), (SHD) Spherical Harmonics Descriptor (KAZHDAN; FUNKHOUSER; RUSINKIEWICZ, 2003), (EGI) Extended Gaussian Images (KANG; IKEUCHI, 1993), (D2) Shape Distance Distribution (OSADA et al., 2002).

Descriptor	**Database**	**Nearest Neighbor**	**First Tier**	**Second Tier**
DCT	(PSB)	36%	14%	9%
DCT	(TSB)	59%	15%	10%
DB-VLAT	(TSB)	83%	38%	47%

Table 5.1: Measures of Nearest Neighbor, Frist Tier and Second Tier for the descriptors DCT and DB-VLAT. DCT measures were performed in both PSB and TSB. DB-VLAT values for the TSB where extracted from (TATSUMA; KOYANAGI; AONO, 2012).

approach. Results considering these factors are presented in (Sec. 5.4)

5.4 Computational Efficiency

A CBR system following the pipeline given in Fig. 1.2 was used to measure the efficiency of the DCT descriptor computation, using the Princeton and Toyohashi databases. DCT descriptors were generated for each database model and stored in its respective database. As a query, we used either a shape present in the database or a third-party shape. For each such query shape, we computed the DCT descriptor and compared it against all stored descriptors. Matches were returned in increasing order of dissimilarity. Clearly, this query process is suboptimal in terms of query speed as its complexity is linear in the number of database models. Hierarchical or

Princeton shape benchmark			**1815 models**
Model	Vertices	Faces	Time (ms)
Largest model	160940	316498	32884
Smallest model	31	35	206
Typical model	7691	15294	652
Average time per model			**0.968 sec**
Total running time			**1757 sec**

Table 5.2: Top: DCT descriptor computation time for the largest, smallest, and typical models in the Princeton database. Bottom: Average and total time required to build DCT descriptors.

indexing structures can be generated to further speed up the search by avoiding unnecessary descriptor comparisons. However, as the main goal in this work was the design of an efficient and computationally effective *descriptor*, rather than a computationally effective database *search structure*, we did not implement such structures.

Toyohashi shape benchmark			**10000 models**
Model	Vertices	Faces	Time (ms)
Biggest model	181912	362935	25377
Smallest model	6	8	122
Typical model	12858	24685	785
Average running time per model			**0.601 sec**
Total running time			**6013 sec**

Table 5.3: Top: DCT descriptor computation time for the largest, smallest, and typical models in the Toyohasi database. Bottom: Average and total time to build DCT descriptors.

	Princeton dataset	**Toyohashi dataset**
	Total time (ms)	Total time (ms)
L_1	165	722
Hellinger	**44**	**216**
Chi-square	36	196
Correlation	35	188
EMD	882090	4410000

Table 5.4: Time required to compare a query descriptor with all descriptors in the Princeton and Toyohashi databases (1815 and 10000 comparisons respectively), for different distance metrics. The Hellinger metric, which also delivers the best precision-recall curves, has also a favorable computational efficiency.

Tables 5.2 and 5.3 show the time taken to run the DCT computation for the Princeton and

Toyohashi benchmarks respectively. The parameters used were the same as those for the results thesis, *i.e.* $RS = 500K$ ray samples and $W = 256^2$ screen resolution. Our implementation took 1757 seconds to generate DCT descriptors for each of the 1815 models in the Princeton database and 6013 seconds for the DCT descriptors in the 10000-model Toyohashi database. As both databases have similar models with respect to the numbers of vertices and faces, a roughly linear performance in the model count can be observed. Table 5.4 shows the time required to compare a query descriptor with all descriptors stored in the two databases using 5 different distance metrics. The relative low speed of the L_1 metric is explained by the fact that no optimized data structures to store 2D sparse histograms was used. In contrast, the Hellinger, Chi-square and correlation metrics used such optimizations, provided by OpenCV, and showed a better (and comparatively similar) performance. Finally, the EMD distance was considerably slower than all other metrics, given its inefficiency when dealing with sparse histograms.

6 CONCLUSION

In this thesis, we presented DCT, a new 3D shape descriptor based on the distribution of depth complexity and thickness information. The result is a 2D depth-complexity-and-thickness (DCT) histogram, aimed at generic shape retrieval under translation, rotation and scale invariance. The descriptor has proven to be computationally efficient and robust. The DCT descriptor exhibits a promising retrieving power, in terms of capturing shape geometry and topology, as tested on the well-known Princeton and Toyohashi benchmarks. DCT can easily be implemented on the GPU and delivers performance rates compatible with online and interactive retrieval rates, even when using a linear search algorithm.

Although the discrimination power of DCT can be improved, this can also be said of other comparable distribution-based shape descriptors. DCT performs well on shape classes whose topology differs most from the topology of the average shapes present in a database. For such cases, the depth complexity histogram has a key contribution to the final dissimilarity measure. However, the thickness histogram is also important for further discrimination. As such, the combination of the two histograms provides a descriptor which is better than the two histograms taken separately.

One avenue for future work is to extend DCT to measure additional global shape features. Example features are the geodesic distance between ray-shape intersection points, which can be computed using recent GPU methods (JALBA; KUSTRA; TELEA, 2013a) and shape medial (skeleton) properties. Such features can lead to new high-dimensional shape descriptors that provide more effective comparison of complex 3D shapes. Separately, filters can be designed to improve the DCT retrieval quality by reducing the impact of missing or duplicated triangles. This would lead to more reliable and discriminative measurements of depth complexity and thickness.

REFERENCES

Aim@Shape. **3D Model Database**. 2014. <http://shapes.aimatshape.net>. [Online; accessed 09-Feb-2015].

AKGUL, C. et al. Density-based shape descriptors for 3d object retrieval. In: GUNSEL, B. et al. (Ed.). **Multimedia Content Representation, Classification and Security**. [S.l.]: Springer Berlin Heidelberg, 2006, (Lecture Notes in Computer Science, v. 4105). p. 322–329.

AKGÜL, C. B. et al. Content-based image retrieval in radiology: current status and future directions. **Journal of Digital Imaging**, Springer, v. 24, n. 2, p. 208–222, 2011.

ANKERST, M. et al. 3d shape histograms for similarity search and classification in spatial databases. In: **Proceedings of the 6th International Symposium on Advances in Spatial Databases**. London, UK: Springer-Verlag, 1999. (SSD '99), p. 207–226.

ARCHIVE3D. **Archive3D shape database**. 2014. <http://archive3d.net/>. [Online; accessed 09-Feb-2015].

BEITZEL, S. M. **On Understanding and Classifying Web Queries**. Thesis (PhD) — Illinois Institute of Technology, Chicago, IL, USA, 2006.

BENCHMARK, K. S. **3D CCCC Shape Benchmark**. 2008. <http://merkur01.inf.uni-konstanz.de/CCCC>. [Online; accessed 11-Jan-2015].

BHALKE, D.; RAO, C. R.; BORMANE, D. Musical instrument recognition using higher order moments. **Digital Signal Processing**, v. 5, n. 4, p. 133–138, 2013.

BHATTACHARYYA, A. On a measure of divergence between two statistical populations defined by their probability distributions. **Bulletin of the Calcutta Mathematical Society**, London, UK, v. 35, p. 99–109, 1943.

CATMULL, E.; CLARK, J. **Seminal Graphics**. New York, USA: ACM, 1998. 183–188 p.

CHEN, C.; ZHUANG, Y.; XIAO, J. Silhouette representation and matching for 3d pose discrimination: A comparative study. **Image and Vision Computing**, v. 28, n. 4, p. 654 – 667, 2010.

CHEN, D.-Y. et al. On visual similarity based 3d model retrieval. **Computer Graphics Forum**, Blackwell Publishing, Inc, v. 22, n. 3, p. 223–232, 2003.

CLEF. **The CLEF Initiative – Conference and Labs of the Evaluation Forum**. 2014. <http://www.clef-initiative.eu>. [Online; accessed 09-Feb-2015].

FORMAT, P. **PLY Format Specification**. 2015. <http://www.cs.virginia.edu/~gfx/Courses/2001/Advanced.spring.01/plylib/Ply.txt>. [Online; accessed 14-Jan-2015].

FORMAT, S. **SMF Format Specification**. 2015. <http://mgarland.org/archive/uiuc/class/realtime/dist/SMF.txt>. [Online; accessed 14-Jan-2015].

FU, Z. et al. A survey of audio-based music classification and annotation. **IEEE Transactions on Multimedia**, IEEE, v. 13, n. 2, p. 303–319, 2011.

FUNKHOUSER, T. et al. A search engine for 3d models. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 22, n. 1, p. 83–105, jan. 2003.

HARTMAN, J.; WERNECKE, J. **The VRML 2.0 Handbook: Building Moving Worlds on the Web**. [S.l.]: Addison-Wesley, 1996.

HELLINGER, E. **Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen**. [S.l.: s.n.], 1909.

HILAGA, M. et al. Topology matching for fully automatic similarity estimation of 3d shapes. In: **Proceedings of the 28th annual conference on Computer graphics and interactive techniques**. New York, NY, USA: ACM, 2001. (SIGGRAPH '01), p. 203–212.

HOFF III, K. E. et al. Fast computation of generalized voronoi diagrams using graphics hardware. In: **Proceedings of the 26th annual conference on Computer graphics and interactive techniques**. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999. (SIGGRAPH '99), p. 277–286.

HU, W. et al. A survey on visual content-based video indexing and retrieval. **Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on**, IEEE, v. 41, n. 6, p. 797–819, 2011.

IGARASHI, T.; MATSUOKA, S.; TANAKA, H. Teddy: A sketching interface for 3d freeform design. In: **Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques**. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999. (SIGGRAPH '99), p. 409–416.

IYER, N. et al. Three-dimensional shape searching: state-of-the-art review and future trends. **Computer-Aided Design**, v. 37, n. 5, p. 509 – 530, 2005.

J. BALDWIN D., R. R. K. E. **The OpenGL Shading Language version 1.10.59**. 2004. <http://www.opengl.org/>. [Online; accessed 11-Jan-2015].

JALBA, A.; KUSTRA, J.; TELEA, A. Surface and curve skeletonization of large 3D models on the GPU. **IEEE TPAMI**, v. 35, n. 6, p. 1495–1508, 2013.

JALBA, A. C.; KUSTRA, J.; TELEA, A. C. Surface and curve skeletonization of large 3d models on the gpu. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, v. 35, n. 6, p. 1495–1508, June 2013.

JOLLIFFE, I. **Principal component analysis**. [S.l.]: Springer-Verlang, 1986. (Springer series in statistics).

KANG, S. B.; IKEUCHI, K. The complex egi: A new representation for 3-d pose determination. **IEEE Trans. Pattern Anal. Mach. Intell.**, IEEE Computer Society, Washington, DC, USA, v. 15, n. 7, p. 707–721, jul. 1993.

KAZHDAN, M.; FUNKHOUSER, T.; RUSINKIEWICZ, S. Rotation invariant spherical harmonic representation of 3d shape descriptors. In: **Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing**. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003. (SGP '03), p. 156–164.

KAZHDAN, M.; FUNKHOUSER, T.; RUSINKIEWICZ, S. Symmetry descriptors and 3d shape matching. In: **Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing**. New York, NY, USA: ACM, 2004. (SGP '04), p. 115–123.

KAZMI, I.; YOU, L.; ZHANG, J. J. A survey of 2d and 3d shape descriptors. In: **Computer Graphics, Imaging and Visualization (CGIV), 2013 10th International Conference**. [S.l.: s.n.], 2013. p. 1–10.

KULLBACK, S. **Information Theory and Statistics**. 1968.

KUSTRA, J.; JALBA, A.; TELEA, A. Probabilistic view-based 3D curve skeleton computation on the GPU. In: **Proc. VISAPP**. [S.l.: s.n.], 2013. p. 137–145.

LI, C.; HAMZA, B. A multiresolution descriptor for deformable 3d shape retrieval. **The Visual Computer**, Springer-Verlag, v. 29, n. 6-8, p. 513–524, 2013.

LINDHOLM, S. et al. Hybrid data visualization based on depth complexity histogram analysis. **Computer Graphics Forum**, New York, NY, USA, p. 3–7, 2014.

LIU, X. et al. Directional histogram model for three-dimensional shape similarity. In: IEEE. **Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on**. [S.l.], 2003. v. 1, p. I–813.

LIU, Y. et al. 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**. [S.l.: s.n.], 2004. p. 896–903.

LIU, Z.-B. et al. A survey on partial retrieval of 3d shapes. **Journal of Computer Science and Technology**, Springer, v. 28, n. 5, p. 836–851, 2013.

LOWE, D. Distinctive image features from scale-invariant keypoints. **International Journal of Computer Vision**, Kluwer Academic Publishers, v. 60, n. 2, p. 91–110, 2004.

LUMSDAINE, A.; GEORGIEV, T. Plenoptic rendering with interactive performance using gpus. In: **Plenoptic Rendering With Interactive Performance Using GPUs**. [S.l.]: SPIE Electronic Imaging, 2012.

LUXBURG, U. von. **Statistical learning with similarity and dissimilarity functions**. [S.l.]: Logos-Verlag, 2004.

MARK, W. R. et al. Cg: a system for programming graphics hardware in a c-like language. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 22, n. 3, p. 896–907, jul. 2003.

MAULE, M. et al. Technical section: A survey of raster-based transparency techniques. **Comput. Graph.**, Pergamon Press, Inc., Elmsford, NY, USA, v. 35, n. 6, p. 1023–1034, dec. 2011.

McGill. **3D Shape Database**. 2014. <http://www.cim.mcgill.ca/~shape/benchMark>. [Online; accessed 09-Feb-2015].

MICROSOFT. **High-Level Shading Language**. 2005. <http://msdn.microsoft.com/en-us/library/windows/desktop/bb509638%28v=vs.85%29.aspx>. [Online; accessed 09-Feb-2015].

MIN, P. **A 3D Model Search Engine**. Thesis (PhD) — Princeton University, jan. 2004.

MIN, P.; KAZHDAN, M.; FUNKHOUSER, T. A comparison of text and shape matching for retrieval of online 3d models. In: **In Proc. European Conference on Digital Libraries**. London, UK: [s.n.], 2004. p. 209–220.

MINGQIANG, Y.; IDIYO, K. K.; JOSEPH, R. A Survey of Shape Feature Extraction Techniques. **Pattern Recognition, Peng-Yeng Yin (Ed.) (2008) 43-90**, IN-TECH, p. 43–90, Nov 2008.

MURASE, H.; NAYAR, S. K. Visual learning and recognition of 3-d objects from appearance. **International Journal of Computer Vision**, Kluwer Academic Publishers, Maryland, USA, v. 14, n. 1, p. 5–24, 1995.

NVIDIA. **CUDA - Compute Unified Device Architecture**. 2006. <https://developer.nvidia.com/category/zone/cuda-zone>. [Online; accessed 11-Jan-2015].

OHBUCHI, R.; MINAMITANI, T.; TAKEI, T. Shape-similarity search of 3d models by using enhanced shape functions. In: **Proceedings of the Theory and Practice of Computer Graphics 2003**. Washington, DC, USA: IEEE Computer Society, 2003. (TPCG '03), p. 97–.

OSADA, R. et al. Shape distributions. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 21, n. 4, p. 807–832, oct. 2002.

OWENS, J. D. et al. **A survey of general-purpose computation on graphics hardware**. 2007.

PAQUET, E.; RIOUX, M. Nefertiti: a query by content software for three-dimensional models databases management. In: **3-D Digital Imaging and Modeling, 1997. Proceedings., International Conference on Recent Advances in**. [S.l.: s.n.], 1997. p. 345–352.

PAQUET, E. et al. Description of shape information for 2-d and 3-d objects. **Signal Processing: Image Communication**, v. 16, n. 1-2, p. 103 – 122, 2000.

PELE, O.; WERMAN, M. The quadratic-chi histogram distance family. In: DANIILIDIS, K.; MARAGOS, P.; PARAGIOS, N. (Ed.). **Computer Vision - ECCV 2010**. London, UK: Springer Berlin Heidelberg, 2010, (Lecture Notes in Computer Science, v. 6312). p. 749–762.

PICKUP, D. et al. SHREC'14 track: Shape retrieval of non-rigid 3d human models. In: **Proceedings of the 7th Eurographics workshop on 3D Object Retrieval**. New York, NY, USA: Eurographics Association, 2014. (EG 3DOR'14).

PRINCETON. **3D Model Search Engine**. 2001. <http://shape.cs.princeton.edu/search.html>. [Online; accessed 11-Jan-2015].

ROST, R. J. **OpenGL(R) Shading Language (2nd Edition)**. [S.l.]: Addison-Wesley Professional, 2005.

RUBNER, Y.; TOMASI, C.; GUIBAS, L. A metric for distributions with applications to image databases. In: **Computer Vision, 1998. Sixth International Conference on**. New York, USA: [s.n.], 1998. p. 59–66.

RUBNER, Y.; TOMASI, C.; GUIBAS, L. J. A metric for distributions with applications to image databases. In: IEEE. **Computer Vision, 1998. Sixth International Conference on**. [S.l.], 1998. p. 59–66.

RUBNER, Y.; TOMASI, C.; GUIBAS, L. J. The earth mover's distance as a metric for image retrieval. **Int. J. Comput. Vision**, Kluwer Academic Publishers, Hingham, MA, USA, v. 40, n. 2, p. 99–121, nov. 2000.

RUEDA, A.; ORTEGA, L. Geometric algorithms on CUDA. In: **International Conference on Computer Graphics Theory and Applications**. [S.l.: s.n.], 2008. p. 107–112.

RUI, Y. et al. Relevance feedback: a power tool for interactive content-based image retrieval. **Circuits and Systems for Video Technology, IEEE Transactions on**, Chicago, IL, USA, v. 8, n. 5, p. 644–655, Sep 1998.

SAUPE, D.; VRANIC, D. 3d model retrieval with spherical harmonics and moments. In: RADIG, B.; FLORCZYK, S. (Ed.). **Pattern Recognition**. [S.l.]: Springer Berlin Heidelberg, 2001, (Lecture Notes in Computer Science, v. 2191). p. 392–397.

SHEN, H. C.; WONG, A. K. Generalized texture representation and metric. **Computer Vision, Graphics, and Image Processing**, v. 23, n. 2, p. 187 – 206, 1983.

SHEN, H. C.; WONG, A. K. Generalized texture representation and metric. **Computer Vision, Graphics, and Image Processing**, v. 23, n. 2, p. 187 – 206, 1983.

SHILANE, P. et al. The Princeton shape benchmark. In: **Shape Modeling International**. [S.l.: s.n.], 2004.

SHREC. **SHREC 2012 - Shape Retrieval Contest based on Generic 3D Dataset**. 2012. <http://www.itl.nist.gov/iad/vug/sharp/contest/2012/Generic3D/>. [Online; accessed 11-Jan-2015].

SIDDIQI, K. et al. Retrieving articulated 3-d models using medial surfaces. **Mach. Vision Appl.**, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 19, n. 4, p. 261–275, may 2008.

TANGELDER, J. W.; VELTKAMP, R. C. A survey of content based 3d shape retrieval methods. **Multimedia Tools Appl.**, Kluwer Academic Publishers, Hingham, MA, USA, v. 39, n. 3, p. 441–471, sep. 2008.

TATSUMA, A.; AONO, M. Multi-fourier spectra descriptor and augmentation with spectral clustering for 3d shape retrieval. **The Visual Computer**, Springer-Verlag, v. 25, n. 8, p. 785–804, 2009.

TATSUMA, A.; KOYANAGI, H.; AONO, M. A large-scale shape benchmark for 3d object retrieval: Toyohashi shape benchmark. In: **Signal Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific**. [S.l.: s.n.], 2012. p. 1–10.

TELEA, A.; JALBA, A. Voxel-based assessment of printability of 3D shapes. In: **Proc. ISMM**. [S.l.]: Springer LNCS 6671, 2011. p. 393–404.

TREC. **Text Retrieval Conference Site**. 2014. <http://trec.nist.gov>. [Online; accessed 09-Feb-2015].

VRANIC, D. V. **3D Model Retrieval**. Thesis (PhD) — University of Leipzig, Berlin, 2003.

WESSEL, R.; BLÜMEL, I.; KLEIN, R. A 3d shape benchmark for retrieval and automatic classification of architectural data. In: **Eurographics 2009 Workshop on 3D Object Retrieval**. [S.l.: s.n.], 2009. p. 53–56.

WOTSIT. **Format File Specification**. 2015. <http://www.wotsit.org/>. [Online; accessed 14-Jan-2015].

YUE, J. et al. Content-based image retrieval using color and texture fused features. **Mathematical and Computer Modelling**, Elsevier, v. 54, n. 3, p. 1121–1127, 2011.

ZHANG, D.; LU, G. Review of shape representation and description techniques. **Pattern Recognition**, v. 37, p. 1–19, 2004.