

Dipartimento di Informatica, Bioingegneria,  
Robotica ed Ingegneria dei Sistemi

---

**Similarity reasoning for local  
surface analysis and recognition**

by

Elia Moscoso Thompson

Theses Series

**DIBRIS-TH-2021-XXXIII**

---

DIBRIS, Università di Genova

Via Opera Pia, 13 16145 Genova, Italy

<http://www.dibris.unige.it/>

**Università degli Studi di Genova**

**Dipartimento di Informatica, Bioingegneria,**

**Robotica ed Ingegneria dei Sistemi**

**Ph.D. Thesis in Computer Science and Systems Engineering**

**Computer Science Curriculum**

**Similarity reasoning for local  
surface analysis and recognition**

by

Elia Moscoso Thompson

March, 2021

**Dottorato di Ricerca in Informatica ed Ingegneria dei Sistemi**  
**Indirizzo Informatica**  
**Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi**  
**Università degli Studi di Genova**

DIBRIS, Univ. di Genova  
Via Opera Pia, 13  
I-16145 Genova, Italy  
<http://www.dibris.unige.it/>

**Ph.D. Thesis in Computer Science and Systems Engineering**  
**Computer Science Curriculum**  
(S.S.D. INF/01)

Submitted by Elia Moscoso Thompson  
DIBRIS, Univ. di Genova

• • • •

Date of submission: March 2021

Title: Similarity reasoning for local surface analysis and recognition

Advisor: Dr. Silvia Biasotti  
Istituto di Matematica Applicata e Tecnologie Informatiche (IMATI) 'E. Magenes' - CNR,  
Genova, Italy

• • •

Ext. Reviewers:  
Dr. Daniela Giorgi  
Prof. Marco Tarini  
Prof. Remco C. Veltkamp

## Abstract

*This thesis addresses the similarity assessment of digital shapes, contributing to the analysis of surface characteristics that are independent of the global shape but are crucial to identify a model as belonging to the same manufacture, the same origin/culture or the same typology (color, common decorations, common feature elements, compatible style elements, etc.). To face this problem, the interpretation of the local surface properties is crucial.*

*We go beyond the retrieval of models or surface patches in a collection of models, facing the recognition of geometric patterns across digital models with different overall shape. To address this challenging problem, the use of both engineered and learning-based descriptions are investigated, building one of the first contributions towards the localization and identification of geometric patterns on digital surfaces. Finally, the recognition of patterns adds a further perspective in the exploration of (large) 3D data collections, especially in the cultural heritage domain.*

*Our work contributes to the definition of methods able to locally characterize the geometric and colorimetric surface decorations. Moreover, we showcase our benchmarking activity carried out in recent years on the identification of geometric features and the retrieval of digital models completely characterized by geometric or colorimetric patterns.*

# Acknowledgements

First and foremost, a HUGE thank you to my tutor Dott. Silvia Biasotti: she guided and supported me throughout my PhD as only the best tutors do and without her this thesis would not exist. Me being the sole author of this work does not make justice to all the thing she helped me with.

A special thank goes to Dott. Michela Spagnuolo, the director of CNR-IMATI: she encouraged me to address this research topic and connected me to other international research groups. I also thank everyone at CNR-IMATI E. Magenes: each one of you contributed directly or indirectly to my work and to keep going through this journey.

Another huge thank you goes to Dr. Julie Digne and Prof. Raphaelle Chaine from CNRS-LIRIS, for mentoring me in one of the most crucial aspects of my thesis and for working together all these years.

Then, I thank all the people at DIBRIS involved in my thesis, Prof. Enrico Puppo, Prof. Manuela Chessa and Prof. Paola Magillo, as well as the PhD course coordinator Prof. Giorgio Delzanno.

Thanks to the reviewers of this manuscript, Dott. Daniela Giorgi, Prof. Marco Tarini and Prof. Remco Veltkamp, for their corrections and precious suggestions.

Thank to every author and co-author that worked with me on my publications: these collaborations means a lot to me and gave me valuable experience in communicating my researches to others.

Finally, I would like to thank the project GRAVITATE, that partially founded my research during all these years.

And thank YOU, for having even the slightest look at my thesis! It means a lot.

## Way less formal acknowledgments [ITA-ENG]

[ITA] Ciao! Grazie per star leggendo queste parole. Ovviamente, devo un sacco di ringraziamenti a Voi, dove Voi è un sacco di persone. Tuttavia, invece di partire con il classico ‘Ringrazio la mia famiglia -lista e aneddoto-, amici -lista e aneddoto-’, voglio provare qualcosa di diverso è più breve. In altre parole, è tempo di partire per una tangente infinitamente poco coerente col contesto in cui si trova. Se questa cosa vi fa strano, piacere sono Elia, spero che diventeremo amici!

Essendo i ringraziamenti di una tesi, partiamo da quello. Occorre premettere una cosa: come molti di voi saprete, io sono molto pigro e umile (e bello, ovviamente). La combo delle prime due proprietà mi ha messo veramente i bastoni tra le ruote durante tutti questi anni di tesi (e anche prima). Tuttavia, l'ammontare di supporto e aiuto che ho avuto da Voi è stato (e continua ad essere) infinito. Ogni membro della mia famiglia, ogni partner, ogni amico andato e venuto mi ha dato un aiuto che va da una piccola spintarella a una spinta che 'ahia-ma-che-male-fai-piano' per farmi andare avanti. Spesso, non vi meritavo. Altre volte vi meritavo, ma vi ho deluso. Qualunque fosse l'esito della vostra spinta sappiate che, anche se vale veramente poco, non smetterò mai di sentirmi in debito con ognuno di Voi.

Non intendo ringraziare tutti uno ad uno, in questa parte. Siete tantissimi, cari e care (o car\*, come si dice ma mi sapeva di automobile) e non bastano il quintuplo delle pagine di questa tesi per menzionare tutti (la verità: ci ho provato due volte e non ne sono capace, qualcuno mi insegni pls). Tuttavia, tutte le persone in questa infinita lista si meritano un grazie per, alla fine, lo stesso motivo. Infatti, crescendo, mi sono reso conto di quanto sia difficile essere pazienti e gentili col prossimo, specialmente se questo prossimo è un po' un culo pesante. Voi, tuttavia, lo siete stati con me, in qualsivoglia forma e misura, e questa è una cosa che tendo spesso a dimenticare. Mettere questo fatto nero su bianco mi ricorda che la vita è più bella si ha un 'Voi' da ringraziare.

Grazie per avermi sopportato. Grazie a chi è stato, e chi è ancora, parte di me.

Troppo melenso? Delusi di non aver letto il vostro nome da nessuna parte? Voi mi avete nominato nei vostri ringraziamenti e vi sentite presi in giro? Volete sapere i vostri meriti nel dettaglio? Mi hai cresciuto nella pancia per 9 mesi e volevi di più? (questo si applica a un numero molto ristretto di persone) Allora scrivimi o chiamami, sarò felice di parlarti in modo meno melenso di tutto quello che avresti voluto sentire davanti ad una birra (o qualunque cosa beviate per essere felici). Questo vale oggi e per sempre (e se non sono povero ve la offro pure, va la).

Questo è tutto! Ora per cortesia fate finta di leggere la tesi e ditemi che è bella così il mio ego si gonfia un pochino.

---

[ENG] Hello there! Thank you for reading these words. Of course, I own many thanks to You, where You is a very large number of people. However, instead of the classic 'I thank my family, friends, etc', I would like to try something different and shorter. In other words, it is time to begin a speech about something out of context. If this surprises you, well, nice to meet you, I'm Elia, I hope we'll become friends!

Being thesis acknowledgments, let's start from that. One thing beforehand: as You probably know, I'm lazy and humble (and beautiful, of course). This combo of the first two properties really put a spanner in the works for all these years (and even before that). However, the amount of help and support You gave me (and that you keep giving me) was immense. Every member of my family, every partner, friend who has come and gone has given me help ranging from a little

push to a 'ohi-wait-calm-down-it-hurts' push to keep me going. Often, I didn't deserve you. Or I did, but I disappointed you. Whatever the result of your push, know that I will never stop feeling indebted to You, even if this doesn't mean much to you.

I don't plan to thank you one by one, as I said. There are not enough pages in the world to properly thank you (the real reason: I tried two times and both drafts suck, please teach me how to do it). However, I must thank all the people in this infinite list for, fundamentally, the same reason. Indeed, growing up I realized how hard it is to be kind and patient to others. However, You have been with me, in whatever shape and amount, which is something that is easy to forget. Even now, despite all these years my English is still terrible, how did you manage to reach this point of the acknowledgments without giving up? It is amazing! Putting this pen to paper makes me realize how better life is if one has a 'You' to thank.

Thank you for putting up with me. Thank you for being a part of me.

Is this too silly for you? Are you disappointed? Did you mention me in your acknowledgments and now you regret it? Do you want more details on your merits? Did you grow me in your belly for 9 month and you wanted more from me? (this may relate to a very small number of people) Then contact me! We can talk about whatever you want in front of a beer (or the drink of your choice that makes you happy). This is true today and forever (and, if I'm not poor, I'll offer the drinks).

That's all folks! Now please pretend to read my thesis and tell me that's good so I can inflate my ego a tiny bit.

# Table of Contents

<b>Introduction</b>	<b>4</b>
<b>I Preliminary concepts</b>	<b>10</b>
<b>Chapter 1 State of the art</b>	<b>11</b>
1.1 Pattern Retrieval . . . . .	13
1.2 Pattern Recognition . . . . .	20
<b>Chapter 2 Pattern-related surface properties</b>	<b>26</b>
2.1 Color properties on surfaces . . . . .	26
2.2 Geometric surface characterization . . . . .	28
2.3 Comparative analysis of curvature evaluation methods . . . . .	31
<b>II Pattern Descriptors</b>	<b>37</b>
<b>Chapter 3 Edge Local Binary Pattern</b>	<b>38</b>
3.1 Method description . . . . .	40
3.1.1 The edgeLBP description . . . . .	40
3.1.2 Parameter settings . . . . .	45
3.1.3 Computational cost . . . . .	47
3.2 Experimental results . . . . .	47



<b>Chapter 4</b>	<b>Mean Point Local Binary Pattern</b>	<b>54</b>
4.1	Method description . . . . .	54
4.1.1	The mpLBP descriptor . . . . .	55
4.1.2	Parameter settings . . . . .	58
4.2	Experimental results . . . . .	59
4.2.1	Different choices of the ring sampling scheme . . . . .	66
4.2.2	Computational cost . . . . .	67
<b>Chapter 5</b>	<b>Benchmarking activities</b>	<b>70</b>
5.1	Performance measures for pattern retrieval methods . . . . .	70
5.2	Retrieval of gray patterns depicted on 3D models . . . . .	72
5.2.1	Dataset . . . . .	72
5.2.2	Results . . . . .	73
5.3	Feature Curve Extraction on Triangle Meshes . . . . .	79
5.3.1	Evaluation of feature curve characterization methods . . . . .	79
5.4	Retrieval of surface patches with similar geometric reliefs . . . . .	91
5.4.1	Dataset . . . . .	92
5.4.2	Results . . . . .	93
5.5	River gravel characterization . . . . .	98
5.5.1	Dataset . . . . .	99
5.5.2	Performances and results . . . . .	100
<b>III</b>	<b>Pattern Recognition</b>	<b>106</b>
<b>Chapter 6</b>	<b>The surface pattern recognition problem</b>	<b>107</b>
6.1	The dataset . . . . .	107
6.2	Geometric pattern recognition . . . . .	109
6.3	Open challenges . . . . .	112

<b>Chapter 7</b>	<b>Patch Characterization via Energy Optimization and Local Similarity</b>	<b>115</b>
7.1	Preliminaries on the Graph-cut definition and its application to 3D models . . . .	116
7.2	Query point sampling . . . . .	118
7.3	Signals extraction . . . . .	119
7.4	Graph-cut setup . . . . .	121
7.5	Surface segmentation . . . . .	123
7.6	Examples . . . . .	124
<b>Chapter 8</b>	<b>Learning-based approaches</b>	<b>133</b>
8.1	Signal aggregation based on dictionary learning . . . . .	134
8.1.1	Surface sampling and signal extraction . . . . .	135
8.1.2	The case of sparsity equal to 1 . . . . .	136
8.1.3	The case of sparsity greater than 1 . . . . .	138
8.2	Multi-view RESNET50: a description based on CNN . . . . .	140
8.2.1	Local feature characterization . . . . .	142
8.2.2	Training of the models . . . . .	144
8.2.3	Tests on the Model set . . . . .	146
<b>IV</b>	<b>Future works and conclusions</b>	<b>151</b>
<b>Chapter 9</b>	<b>A search engine for 3D shape collections</b>	<b>152</b>
9.1	Conceptual model . . . . .	153
9.2	Similarity assets . . . . .	155
9.3	User-driven dataset navigation . . . . .	160
<b>Chapter 10</b>	<b>Concluding remarks</b>	<b>170</b>
<b>Bibliography</b>		<b>174</b>

# Introduction

The digital revolution is impacting most aspects of our society and its success largely depends on the capability of digital sciences to answer the many challenges posed by the vast amount of digital data available in all fields. Extracting knowledge out of big raw data is among the predominant goals, together with making algorithms smarter and faster. This is particularly challenging for digital content that is highly structured, as in the case of 3D digital representations of real or designed objects (e.g., archaeological artifacts). In recent years, there is a wealth of joint collaborations between Cultural Heritage and Computer Graphics experts. On the one hand, archaeologists recognize that the development of quantitative methods for the analysis and description of archaeological fragments would contribute to many tasks, both in terms of efficiency and effectiveness. On the other hand, the research questions posed by archaeologists pose new challenges to computer scientists on the characteristics that the methods developed must possess and the problems to be addressed. As an example of fruitful collaboration between Computer Graphic and Cultural Heritage experts, we mention the GRAVITATE project [UUTCIC<sup>+</sup>] that aimed at facilitating the analysis and retrieval of fragments that possibly relate to each other and draw archaeological hypotheses previously impossible to make.

## Archaeological motivation

In archaeological excavations, Cultural Heritage artifacts are often found in various stages of incompleteness, with parts either totally missing or components of their overall structure being fragmented in various pieces. A tedious task for conservators is to re-compose these fragmented objects into their original shape. The reasons for such an investment are various: the more complete the shape of an object the better its typology-based classification. A complete shape of an object is more indicative of its function in the past and finally a restored object has a higher aesthetic value than its fragmented parts and thus more appealing for its musealisation. The restoration process involves many aspects such as the detection of common edges, the overall morphology or the continuation of patterns along their external face (such as decorations, either painted or incised, etc.). Since archaeological excavations often yield thousands of artifacts, the restoration work occurs once these objects are already stored in deposits. Moreover, it may

occur that fragments belonging to an object are stored separately and thus the restorer must rely on drawings or digital acquisitions of the fragments in order to propose possible joining of fragments, often supported by similarities in the decorations along their external face or the recognition of common details. In case fragments are stored in different locations and/or under separate administrations, a physical restoration is an almost impossible task.

As in many other challenging scenarios, these difficulties may be simplified with the aid of automatic tools that act on digitalised versions of the archaeological artifacts the experts need to work on. Indeed, a restoration process based on digital quantitative methods rather than visual observation may optimize the entire restoration process, making it faster and more accurate. In algorithmic terms, this analysis requires a translation of the needs of Cultural Heritage experts into quantifiable descriptions and methods that suit their interest. Aspects of interest for Cultural Heritage experts and that need to be taken into account by the algorithms are, for instance, the color, the size, the overall shape, the thickness of the object, the object decorations, the way certain details are depicted (e.g.: the eyes of a statue can be represented differently from one civilization to another), etc.. On one hand, some of these measurements are easily converted into quantities and compared: for example, the size of a model can be estimated by considering the size of the minimal bounding box. Thus, if we consider two objects, their encumbrance can be easily compared. On the other hand, many aspects of interest have no easy description. Decorations, for example, include a wide range of elements, from the corrugations that shape the eye of a statue to the repeated curls on the head of a bas-relief depiction of a man representing his hair. Indeed, decorations cannot be easily converted into a quantifiable descriptor that allows for an easy local comparison between models.

## **Problem statement**

Within this scenario, this thesis focuses on decorations characterized by small variations of geometry or color, like a circle or something slightly more complex, repeated over the surface of an object *without altering the overall surface shape*. In this work, we refer to this kind of decoration with the term of *surface patterns* (or just *patterns*). This kind of decorations was relevant in the use-case of the GRAVITATE project, as well as a characteristic of many other collections. A correspondence of these surface decorations can be easily found in images, under the name of *image textures*. While the analysis of image textures is largely studied and counts different methods for both their formalization and characterization, the same does not happen for surface patterns. This is partially justified also by the kind of data: indeed, working with images it is different than with meshes or point clouds, as the last two structures are representations less structured, not unique and are, in general, heavier to process. Still, the similarities between surface patterns and image textures characterization problems lead us to wonder if it is possible to deal with surface patterns with approaches similar to those used in image texture analysis. Starting from these considerations, the scientific question that this thesis aims to answer, at least partially, is: *How to extend*

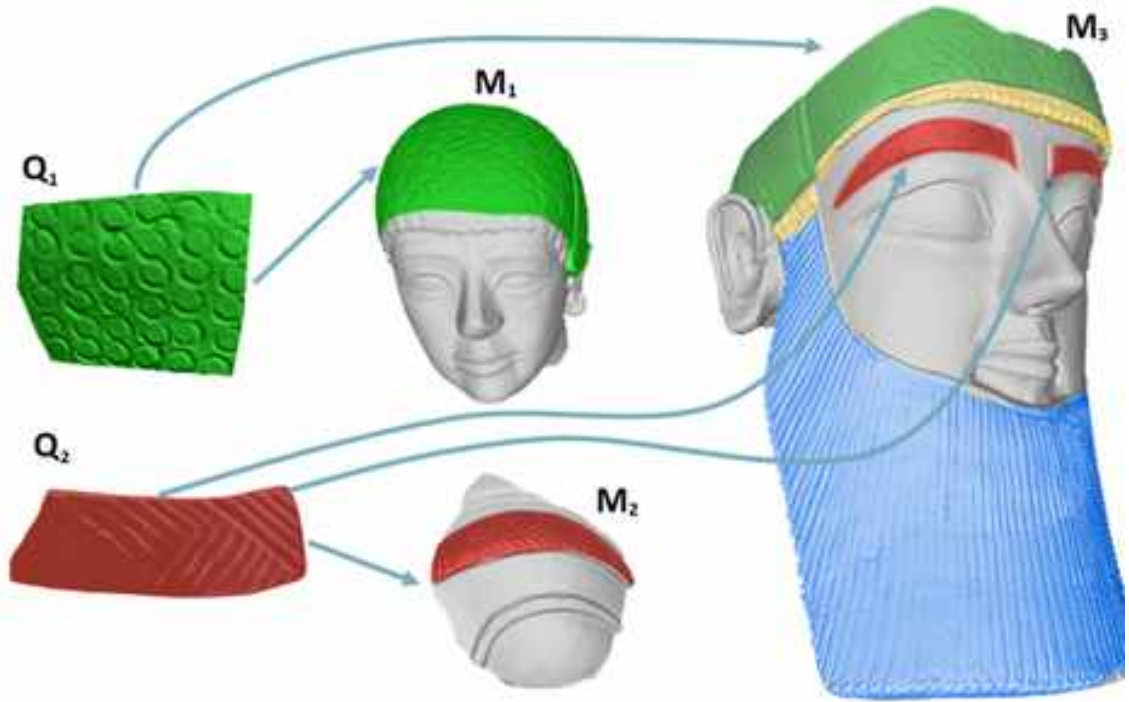


Figure 1: A visual representation of the pattern recognition problem as intended in this work. Two query models  $Q_*$  are covered by a pattern. Such a pattern is then identified (both in terms of presence and location) on the models  $M_*$ .

*to digital surfaces what has been done for image textures?* In practice, this means addressing the problem of pattern recognition on surfaces in the same terms as for images.

Our research focuses on the characterization of the patterns available on a given model and on their identification/recognition on other models. We call this task the *pattern recognition* problem. More formally, we define a *surface pattern* (or a *pattern*) a repeated, oriented basis structure element that is present on a surface  $S$ ; such an element that can be described by one or more surface properties. The base structure element can represent a variation in color or geometry of the surface and can assume different shapes, like a circle, a segment, and so on. The size of this detail is small enough to not affect the overall bending of the surface even when it is geometric. In this manuscript we deal with two kinds of patterns: geometric patterns (defined by surface corrugations) and colorimetric patterns (defined by the color distribution on the surface). We formalize the pattern recognition problem as follows (we refer to Figure 1 for a visual representation). Let us consider a set of query pattern samples  $Q_i$ , i.e., a set of (small) sample surface fully characterized by a single pattern, and a set of models  $M_j$ . where a model  $M_j$  may have one, multiple or none pattern located somewhere on its surface. For each  $Q_i$ , the goal is to understand if the pattern represented on  $Q_i$  is also present on  $M_j$  (for all the  $M_j$ ) and, in case of a positive match, to locate it on the surface. In the current literature, the pattern recognition



Figure 2: A visual representation of the pattern retrieval problem. A query model  $Q$  with a bark-like pattern impressed on its surface is selected. In the ideal case, models with bark-like patterns are retrieved before then models with different reliefs, independently of the global geometry of the models. The "check" and "cross" marks highlight models that are relevant or non-relevant to the query.

(as intended in this manuscript) is faced in very specific cases (i.e.: on data with significant constraints on the models characteristics) and to the best of our knowledge no one has proposed a general pattern description method able to deal with multiple patterns on a single model. We also explore a strictly related problem, called *pattern retrieval* problem. This problem (refer to Figure 2 for a visual overview) consists in assessing the similarity among 3D models on the basis of the patterns that they are fully covered by. In other words, the characterization is strictly based on the pattern and does not take in consideration the size of the overall model, its shape, etc..

Parallel to our research, we looked for way to provoke interest in our research field. We did it by working on a series of contest called SHape REtrieval Contest (SHREC) [VRS<sup>+</sup>06]. Such contests provides many resources to compare and evaluate 3D retrieval methods. The tracks organized during the years have covered different tasks with respect to different aspects (metric learning, outlier detection, correspondence, robustness, stability, registration, classification, recognition, pose estimation, machine learning, etc.) on different kind of 3D data (rigid or non-rigid models, partial or complete models, sketch-based 3D retrieval) and in generic or domain specific models (CAD, biometrics, architectural, protein, etc.).

## Contribution

Overall, this thesis deals with the automatic *characterization of digital surfaces based on their patterns*. Preliminary, to address the geometric pattern characterization problem, we need to better understand which method better characterizes small reliefs and features on a surface. For this reason we have created two benchmarks: one for evaluating algorithms for curvature estimation and the other for addressing feature curve recognition. Indeed, feature recognition is more addressed in the literature than pattern recognition but roots on the strictly related concept of analyzing well-distinctive geometric variations.

The pattern retrieval allows us to start facing the problem of characterizing patterns without bothering about its position on the surface. As a solution for this problem, we proposed two methods called *edge Local Binary Pattern* and *mean point Local Binary Pattern*. Both methods achieved state of the art performances and are validated on many datasets. We supported the pattern retrieval research with an extensive benchmarking activity that counts three benchmarks on three different kinds of pattern: colorimetric patterns, geometric patterns and river gravel bed scans.

The main contribution of my thesis lies in the solutions proposed to the pattern recognition problem. To meet this challenge, we faced several challenges and had to carefully understand and analyze the task, both in terms of which data representation use and which methods for local analysis have been previously explored in this direction. Our efforts took the shape of two promising research paths, one based on an engineered descriptor and another on learned ones. In the engineered method we mix local similarity with an energy minimization problem faced with the graph-cut method. In the learning-based approaches we investigated two methods that exploit the potential of the dictionary learning and of the convolutional neural network infrastructure to characterize patterns.

Simultaneously, we created datasets and metrics for evaluated methods for retrieval of geometric and colorimetric patterns on surfaces, the recognition of curve features and geometric patterns and the classification of river bed gravels. In particular, we have actively contributed to the identification of strategies and trends in these topics that are strictly related to this thesis topic.

Finally, we foresee dataset exploration as a possible future application context for our pattern recognition solutions, indeed, we propose a search engine design in which our pattern recognition solutions could easily fit.

## **Organization of the thesis**

This thesis is divided in four parts. The first three parts are further divided in three chapters, while the last part is made by two chapters.

Part I introduces the preliminary concepts required to frame our work including an overview of the current literature and concept used. Chapter 1 overviews the current literature regarding the contributions to the pattern retrieval and recognition problems. The most pertinent contributions to each problem are briefly described in each section. Since the number of methods and algorithms able to deal with pattern recognition on surfaces is very limited, we focused on two approaches that are interesting but address the problem of recognizing one pattern per model. Chapter 2 is an introduction to the surface properties we use to characterize colorimetric and geometric patterns. While such an introduction is straightforward for the first ones, geometric variations require an in-depth analysis of the current literature, as many different approaches are

available, with different pros and cons. Our measure of choice will be the surface curvature and, to understand which approximation of this concept better suits our purposes, we make a comparative analysis of different curvature estimation methods.

Part II examines the pattern descriptor methods we developed for the pattern retrieval problem and our benchmarking activity. Chapter 3 examines the design of the edge Local Binary Pattern, a pattern descriptor we developed initially for geometric patterns and further extended on colorimetric patterns and tested on multiple datasets. Results achieved state of the art performances and are still competitive to the most recent contributions. Chapter 4 presents a second pattern descriptor that achieves performances similar to the edge Local Binary Pattern, with a lower computational cost. In this chapter we describe how the method is defined and how it performs on different datasets, comparing the results with other methods. Chapter 5 describes the three benchmarks we worked on regarding the pattern retrieval problem. The main difference between them is the kind of models and patterns considered: the first is on gray-scale patterns, the second is on geometric reliefs and the last is on river gravel bed scans.

Part III showcases our contribution to the pattern recognition problem. Chapter 6 deepens the challenges of the pattern recognition problem. Indeed, several significant challenges are discussed in their aspects, starting from the type of data involved to the need of dealing with large models. Chapter 7 presents our method for characterizing patches of a model via energy optimization and local similarity. The energy minimization aspect is faced using the graph-cut method. Patches with patterns are compared to each other using a pattern retrieval method. In Chapter 8 we face the pattern recognition problem using two learning approaches. The first approach uses a signal aggregation method based on the well known sparse decomposition method, while the second is based on a multi-view approach of a surface patches combined with a convolutional neural network.

Part IV focuses on future developments and applications of the research done in Part II and Part III. Chapter 9 proposed the dataset exploration as one of the research axes of a search engine built on a dataset of 3D models. We propose the design for a search engine defined for datasets of different nature and that can easily adapt to new datasets and new similarity criteria with respect to those already implemented. In this regard, the research proposed in this thesis can become new axes of this search engine. Finally, the summary of what we achieved in this work and the conclusive discussions can be found in Chapter 10.



# **Part I**

## **Preliminary concepts**

# Chapter 1

## State of the art

The goal of this chapter is to overview how pattern characterization stands in the Computer Vision literature. This work distinguishes mainly between geometric patterns (characterized by small surface variations) and colorimetric patterns (characterized by variations of the colors of the object scans).

Methods in the current literature for shape retrieval and matching can be classified according to their type of input, their local or global nature, their robustness to noise, their model representations, their invariance to shape transformations and their suitability to partial matching[TV08, TCL<sup>+</sup>13, BCBB16].

A quantitative analysis of the feature matching performance of local shape descriptors over standard datasets has been recently proposed in [YZC17]. Most of these methods analyze the surface on the basis of its global appearance, discarding surface details and local shape variations. For instance, the SHOT descriptor [TSDS11] is meant to solve point-to-point correspondences among sets of feature points. Similarly, the Fast Point Feature Histograms [RBB09] (or FPFH) are defined to align overlapping point clouds. An example of the potential of the FPFH is shown in Figure 1.1.

Since SHOT and FPFH can be seen as implicit solvers of local matching for 3D data, they were adopted, in general, to address the similarity by local characterization. However, this approach results weak when working with patterns, since the global shape of the surface is relevant in the use cases of these methods, when in pattern characterization the description must be only local (or rather, independent from the global surface).

On a similar note, the concept of self similarity is related to the pattern analysis as well. By assuming that a surface has details that repeat multiple times with different quality and embeddings, one can use the combined data given by all the iterations of that level of detail and convert it into useful data. For example, in [HCDC17] the self similarity of 3D point clouds are used to

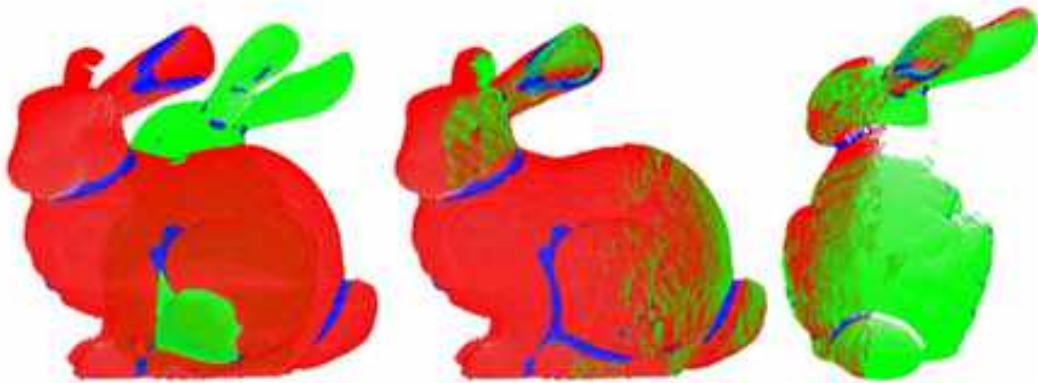


Figure 1.1: The alignment obtained by the FPFH. Left: two point clouds of the same object in different positions. Center and right: two views of the alignment computed by the FPFH. Image from [RBB09].

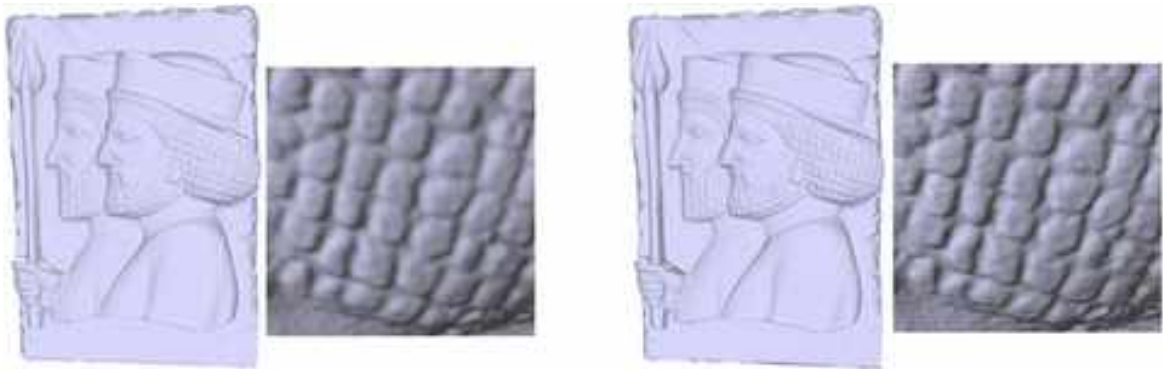


Figure 1.2: Detail enhancement as proposed in [HCDC17] on an archaeological model. Left: the initial scan and one detail. Right: the same model and detail after the detail enhancement. Image from [HCDC17].

enhance the details of the surface, tackling a problem called *super resolution*. Figure 1.2 shows the detail enhancement on a Persepolis slab.

Overall, self-similarity and partial similarity are the key concepts currently referred to detect repeated, local features over a surface [LBZ<sup>+</sup>13, YZC17]. For instance, the method [GCO06] uses surface curvatures for recognizing salient shape features. Once these features are computed, they are mapped using a geometric hashing mechanism that determines the best transformation among these regions by means of a voting scheme. Such a technique is able to recognize repeated surface features (circles or stars) over a surface but, being based on geometry hashing, it is scale dependent and suffers of the local definition of “curvature” that could become insufficient when dealing with highly eroded or perturbed surfaces. Similarly, [IT11] observed that

though isolated feature points often do not suffice, their aggregation provides adequate information regarding similarity. Then, the combination of segmentation techniques with the neighbor description of the feature points yields the detection of similar parts in bas-reliefs and archaeological artifacts. However, every surface part was considered as stand alone and no particular attention was allocated to the detection of repeated patterns. Moving further in this direction, the method proposed in [TBF18] adopts the Hough transform to fit aggregated sets of feature points into template curves: while this approach naturally overcomes the problem of finding multiple instances of the same curve, it requires the surface can be locally projected on a plane and the vocabulary of possible local shape elements is limited to those that have an algebraic expression.

## 1.1 Pattern Retrieval

The pattern retrieval counts several approaches. We recognize two strategies that tackle this problem. On one hand, one could reduce the data dimension, i.e., to project the 3D data into an opportune plane (image) and apply an image pattern recognition algorithm to the projected data. On the other hand, one can work on the definition of a pattern description directly on the surface, which is not straightforward as it involves the treatment of three-dimensional data.

The generalization of several image descriptions to (even textured) surfaces has been explored in several works, e.g. the meshHOG [ZBH12] and the meshSIFT [SKVS13]. However, most of these methods mainly addressed the shape matching problem without focusing on the local variations.

A detailed evaluation and comparison of methods for 3D texture retrieval and comparison can be found in [BCA<sup>+</sup>16] and several SHREC contests [CBA<sup>+</sup>13, BCA<sup>+</sup>14, GFF<sup>+</sup>15]. Focusing on the colorimetric information, the descriptors adopted so far, include: feature-vectors, where the color is treated as a general property of the overall shape, [Suz01], or its subparts in [GG16]; local or global views of the objects [WCL<sup>+</sup>08, PZC13]; correspondences among feature points (e.g., the CSHOT descriptor [TSDS11]); the tracking of the evolution of the color channels sub-level sets [BCGS13].

As a reference to the first typology of methods, I mention the method in [OVSP13] for tree species classification. There, the geometric variations of the tree trunk models are represented with a 3D deviation map over a cylinder that is flattened on a plane using the Principal Component Analysis (PCA) technique. Then, the geometric textures are compared using variations of the complex wavelet transform [KBC06]; see [OVSP13] for a detailed implementation analysis. Similarly, [ZPS<sup>+</sup>16] adopts a deviation map and projects the reliefs and engravings of rock artifacts into an image and classifies them (see Figure 1.3).

Moreover, due to the success of deep neural networks in different application domains, deep learning-based 3D shape features have been proposed for 3D shape analysis. In [WSK<sup>+</sup>15] au-

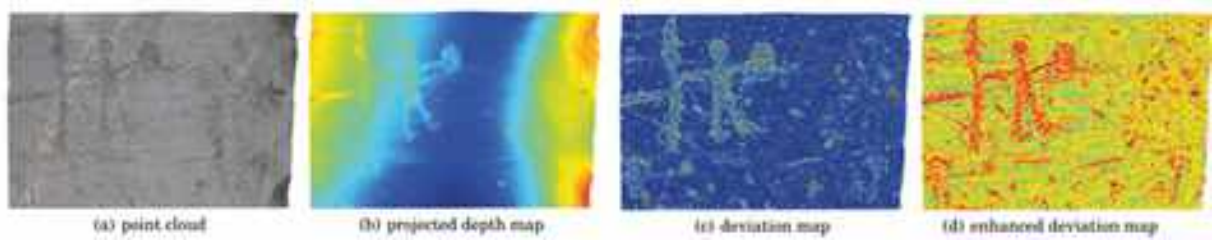


Figure 1.3: Deviation map computation in [ZPS<sup>+</sup>16]: (a) the scanned point cloud viewed from the projection direction; (b) the projected depth map; (c) the deviation map; (d) the deviation map with enhanced topographic structure emphasizes peck-marks that make up the engraving. Image from [ZPS<sup>+</sup>16].

thors proposed to represent 3D shapes as a probability distribution of binary variables on a 3D voxel grid. Boscaini et al. [BMM<sup>+</sup>15] employed the windowed Fourier transform to points on the meshed surface to form a local frequency representation. These local frequency representations are then passed through a bank of filters to form a deep representation for 3D shapes. By constructing a geodesic convolution operator, Masci et al. [MBBV15] generalized the convolutional neural network to non-euclidean manifolds for 3D shape retrieval and correspondence.

In the remainder of this section we overview the methods developed in the last years for the pattern retrieval problem. In particular, we focus our attention on the mesh Local Binary Pattern [WTBB16, WTdB15, WBB15], the Tutte/meanC/SIFT/FV [Gia18] and the most performing learning-based methods proposed in the last SHREC contest on the subject [MBG<sup>+</sup>20].

## Mesh Local Binary Pattern

The Mesh Local Binary Pattern (meshLBP) extends the LBP [OPH96] to triangle meshes, addressing 3D pattern classification and retrieval directly on the surface mesh. The main idea behind the meshLBP is that triangles play the role of pixels; there, the 8-neighbor connectivity of images is ideally substituted by a 6-neighbor connectivity around triangles. Rings of pixels became rings of faces, built with a sort-of front weave expansion from each face. Figure 1.4 shows how a face is created on a uniform mesh. Working on (non-textured) meshes, the role of the gray-scale color is replaced by the mean curvature but it is also possible to use color information stored in the texture. The meshLBP provides an efficient coding of a 3D pattern, providing a compact representation of the pattern. This has been shown by testing variations of this method as in [TWB19a, TWB19b], from the same authors, on recent benchmarks [BMTA<sup>+</sup>17, BMTB<sup>+</sup>18]. Moreover, meshes with patterns usually have a high resolution, in order to represent the patterns of the surface decently enough to be characterized. This usually leads to mesh simplifications (in terms of number of faces and vertices), that almost always create irregularities, either in terms of connectivity or non-uniform vertex distribution. These facts jeopardize the efficacy

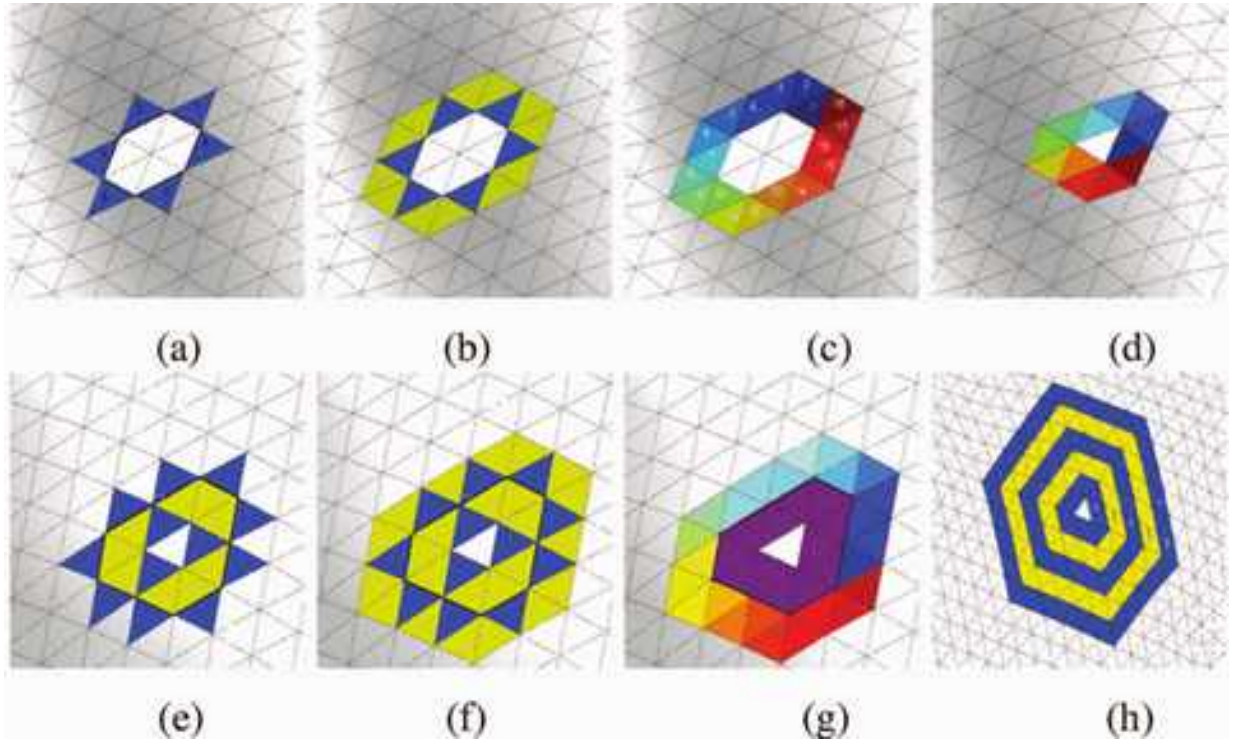


Figure 1.4: Row 1- Construction of an ordered ring of the meshLBP: (a) Initial Four facets (blue); (b) Bridging the gap between the pairs of consecutive Fout facets with the Fgap facets (yellow); (c) The obtained ordered ring; (d) Ordered ring constructed around a central facet. Row 2 - Multiresolution mesh-LBP construction: (e) Extraction of the next sequence of Fout facets, as the facets adjacent to Fgap and which are not part of the current ring; (f) Extracting the Fgap facets; (g) The second ordered ring extracted; (h) Five concentric ordered rings. Image from [WTBB16].

of an expansion strategy based exclusively on elements of the mesh for detecting and coding representation-independent features. This aspect is deepened in Chapter 3.

## Improved Fisher Vector for pattern characterization

In [Gia18], the main idea is to tackle the pattern retrieval problem by considering images that encode the patterns on the surface. The proposed approach exploits the pooling of local invariant features based on the Improved Fisher Vector [PSM10] (IFV) for relief pattern characterization. This is inspired by the great success of applying the same method in the context of image textures. On one hand, classical fisher vector (FV) store the mean and the covariance deviation vectors per component of a Gaussian Mixture Model (GMM) fitted to the unlabeled data. On the other hand,



Figure 1.5: Images obtained mapping the mean curvature evaluated on patch vertices onto the parametrization coordinates. Top row, left: mesh renderings. Top row, right: images obtained with the best fitting plane. Bottom row: images obtained on boundary constrained parametrization (after geodesic circle cropping). Image from [Gia18].

IFV improves the performance of the representation by applying a nonlinear kernel to the vector components and applying  $l - 2$  norm normalization. To apply the IFV to the dataset as done by the author, it is necessary first to extract an image from each model. To do it, the mesh is uniformly remeshed to simplify the estimates and regularize the meshes. Then, the mesh must be projected on a plane, from which the image is sampled. The best method proposed in that work is based on the Tutte embedding [FH05], mapping the patch boundary onto a circle on the plane. The author then crops a large geodesic area that should include all the pattern elements characterizing the texture class. By fixing the pixel/patch size ratio, images are sampled from the projected patches. An example of the passage from 3D model to image is shown in Figure 1.5. The pixel color depends on the mean curvature values of the vertices that fell in the respective square of the pixel grid. The final result is an 8-bit depth image. Local features are then estimated on the resulting raster images using the SIFT transform. The SIFT descriptor is estimated in all the peaks detected by a multiscale Difference of Gaussian detector. A Gaussian Mixture model is then fitted to the estimated set of features, which is then encoded by the IFV encoding. The latter is the descriptor of the pattern on the patch, that is used to estimate the similarity between patches based on their pattern. The results are validated on a recent benchmark [BMTA<sup>+</sup>17]. However, being limited to just an image of a fixed size per model is limiting in case of patches with odd bendings. Moreover, the images can be flowed in the latter case. The third model in Figure 1.5 is an example of a strong surface bending that was not resolved by the Tutte embedding.

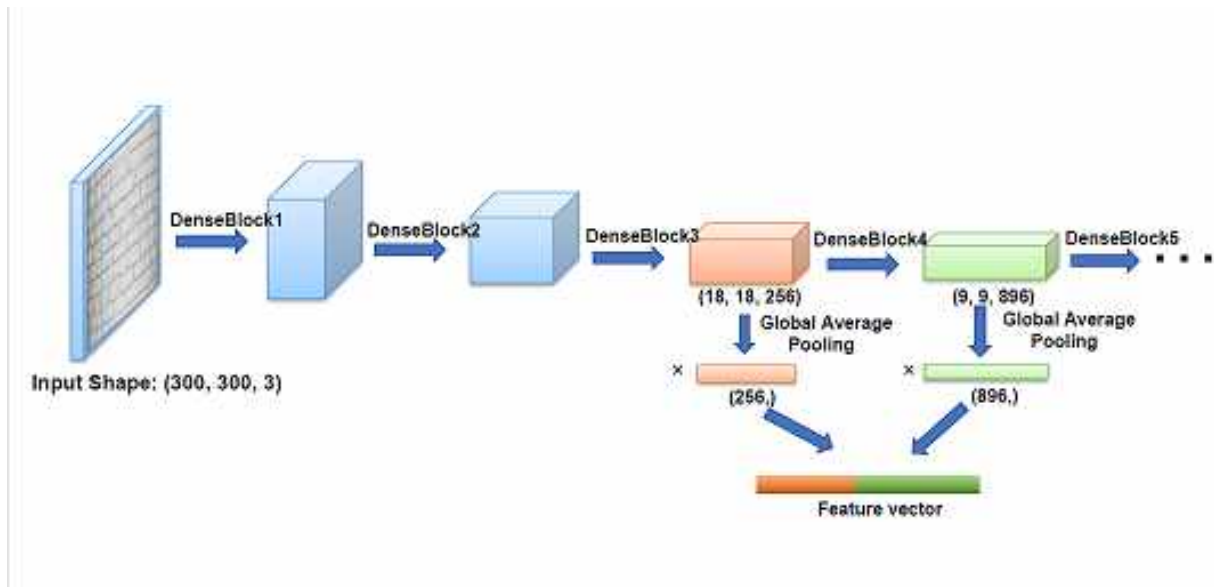


Figure 1.6: Illustration for step three "Extracting Feature by concatenating feature vectors from different layers of pre-trained models" of the DFE method (the figure illustrates the step when using Dense-Net-201). Image from [MBG<sup>+</sup>20].

## Deep Feature Ensemble

In [MBG<sup>+</sup>20], Nguyen-Dinh et al. proposed a method called Deep Feature Ensemble, which classifies patterns based on multiple pre-trained nets. The first step of this method is to upright the 3D object by transforming the object into a new 3D coordinate system so that the object stands vertically across the y-axis for the ease of rendering. Then, an image is extracted from the model so that it has the most relief pattern selected. Such images are used as representatives of the models. The authors propose using these pre-trained models to extract features of each image (or pattern). Many high-performance models such as ResNet [HZRS15], DenseNet [HLP<sup>+</sup>19], VGG [SZ14], and Efficient-Net [TL19] suit this purpose. The authors propose to synergize the information extracted from different intermediate layers of different pre-trained networks by assembling feature vectors. First, a square image containing patterns is given to a pre-trained neural network. Then, the output tensor of a chosen intermediate layer of that network with the shape of  $(h, w, channelsize)$  passes through a Global Average Pooling Layer to create a vector with a length of  $(channelsize,)$  used as a feature vector. The use of the Global Average Pooling makes for more robust results with respect to spatial translations in the images. Finally, the Deep Feature Ensemble descriptor is obtained by concatenating all the output of the pre-trained nets. A visual representation of the way authors ensemble the feature vectors from different layers in different models is shown in Figure 1.6. Descriptors are compared with metrics such as cosine similarity,  $L^1$  or  $L^2$  distance and so on. The authors combine Average Query Expansion



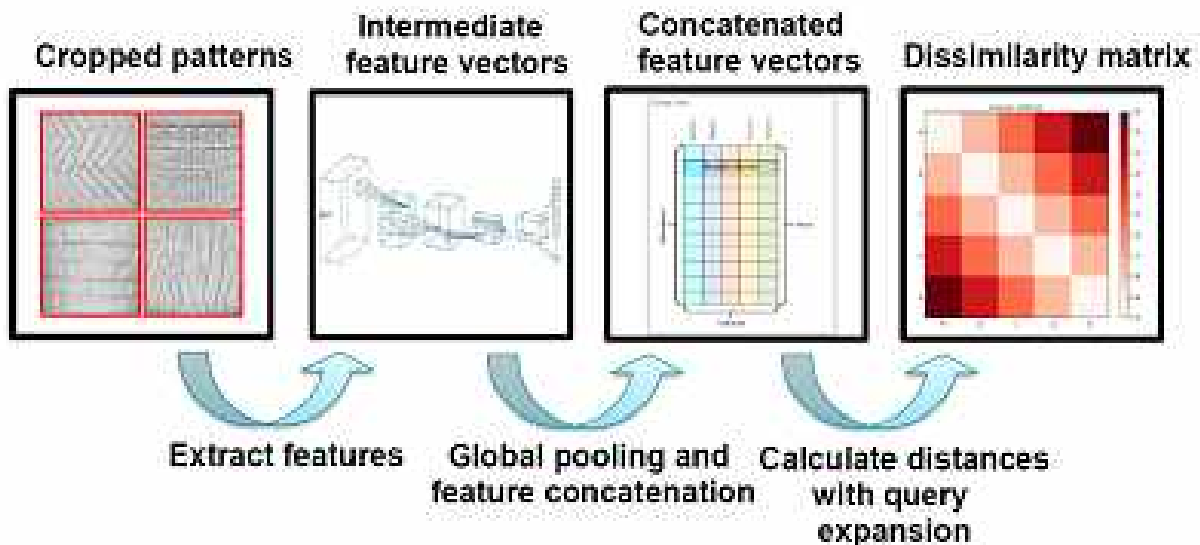


Figure 1.7: Overview of the third and fourth steps of the Deep Feature Ensemble method (DFE). Image from [MBG<sup>+</sup>20].

(AQE)[AD17] with a view to helping our model to remove noises. This method is summarized in Figure 1.7

In the same context, Gigli et al. proposed the DPML method, which starts from the extraction of patch images from the mesh surfaces and uses them to train a Siamese Neural Network [CHL05] to learn a distance function between each pair of images. More precisely, the first part of this method is dedicated to obtain multiple images of the object that contain only local textures. The goal is to project the local neighborhood over a plane and obtain an elevation image. For this reason, only the neighborhoods that are as flat as possible are selected. To estimate such a property, the covariance based features are used. Validated neighborhoods are projected over the tangent space of the surface. A regular grid is defined over the tangent space, and each element of the grid corresponds to a pixel of the image. The intensity values of the image correspond to the distance between the points projected over the element and the tangent plane. In order to obtain a uniform sized patch the method crops them to obtain images of size  $231 \times 231$  (equal to the smallest image extracted with this process). For each patch, crops are computed so that there is the minimum number of void pixels in each image. An overview of this process is reported in Figure 1.8. The second part of the DPML method consists in selecting the images that are used to train a Siamese neural network with the Triplet Loss. The architecture is composed of three CNNs sharing the same weights. In this case the VGG16 [SZ14], without fully connected layers, is chosen as CNN. The CNNs work in parallel taking as input a triplet of images and generating a comparable feature vector, as shown in Figure 1.9. The Triplet Loss minimizes the distance between an anchor and a positive, both of which have the same identity, and maximizes

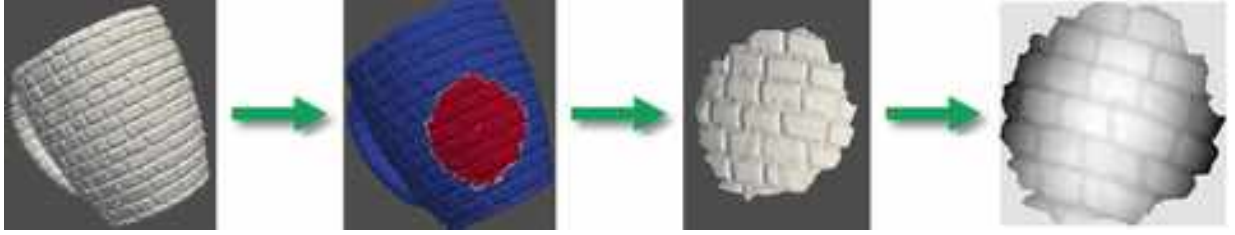


Figure 1.8: Pipeline of the first steps of the DPML method. A local neighborhood is projected over a plane and an elevation image is derived.

the distance between the anchor and a negative of a different identity, i.e. an image from a different object [CSSB10]. Finally, the distance  $\Delta$  between two objects  $\mathcal{S}_i$  and  $\mathcal{S}_j$  is defined as the minimum distance between any couple of images belonging to the two surfaces:

$$\Delta(\mathcal{S}_i, \mathcal{S}_j) = \min_{h,k \in \{1, \dots, m_i\} \times \{1, \dots, m_j\}} \delta(I_h, I_k),$$

where  $\delta(I_h, I_k)$  is the similarity function learned by the Siamese neural network.

## Histogram of Double Distance Transform

For colorimetric patterns, during the SHREC' 18 is on colorimetric patterns retrieval [MTW<sup>+</sup>18], Velasco-Forero considers the texture on the mesh as binary ones, i.e., only containing white and black vertices. A function that maps each interior vertex (exterior) on the mesh with the distance to the nearest vertex on the exterior (interior) set of vertices is defined. A color shape is defined as  $\mathcal{S} = (\mathbb{V}, \mathbb{S}, \mathbf{F})$ , where  $\mathbb{S}$  is a closed triangles mesh in  $\mathbb{R}^3$  and  $\mathbf{F} : \mathbb{S} \rightarrow \mathbb{R}^3$  represents the mapping from each vertex in the mesh to a given color space. The descriptor is computed by these four steps. For a given mesh, vertices are divided in two sets: the set of white vertices ( $\mathbb{V}_w$ ) and that of black vertices ( $\mathbb{V}_b$ ). Then, the  $k$ -nearest neighbor graph ( $k$ -NNG) of  $\mathbb{V}_w$  on  $\mathbb{V}_b$  and the  $k$ -NNG  $\mathbb{V}_b$  on  $\mathbb{V}_w$  are computed. Finally, histograms are computed on the following quantities:

$$NormD_{k, \mathbb{V}_w}(\mathbf{v}) = Dist_1(\mathbf{v}, \mathbb{V}_w) / Dist_k(\mathbf{v}, \mathbb{V}_w),$$

$$NormD_{k, \mathbb{V}_b}(\mathbf{v}) = Dist_1(\mathbf{v}, \mathbb{V}_b) / Dist_k(\mathbf{v}, \mathbb{V}_b);$$

respectively  $\forall \mathbf{v} \in \mathbb{V}_w$  and  $\forall \mathbf{v} \in \mathbb{V}_b$ . The concatenation of the computed histograms is the pattern descriptor. A comparative analysis of the performance of these methods for geometric and colorimetric pattern retrieval is exhibited in Chapter 5.

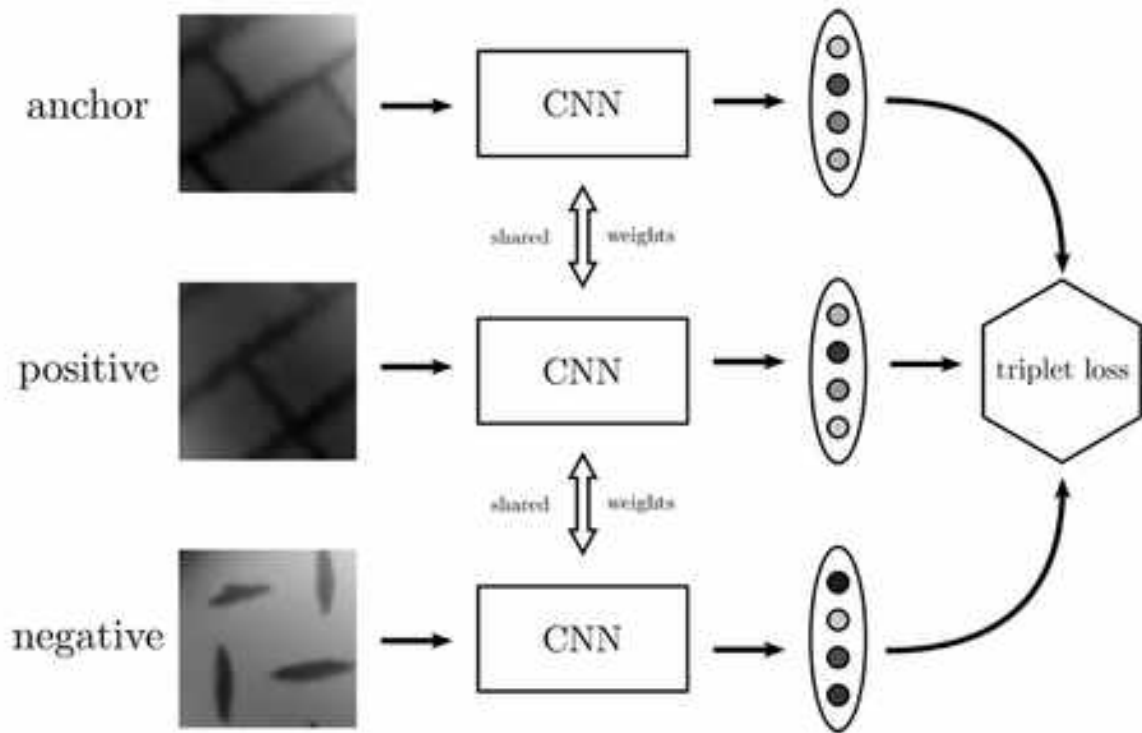


Figure 1.9: Overview of the network of the DPML method. Such a network consists of a batch input layer and a deep CNN. It defines the image embedding by using a triplet loss during training. Image from [MBG<sup>+</sup>20].

## 1.2 Pattern Recognition

The problem of pattern recognition is crucial for applications such as the classification and reconstruction of fragmented artifacts of archaeological dataset based on the representations of local features (like how hair, beards or fabrics are represented on statues and similar artifacts). Such a task, that is still open and only a few methods have been proposed. In particular, there are two contributions that go in the direction of the pattern recognition problem as we intended in this work.

### Automatic shard classification

The first one is from Debrouette et al. [DTC<sup>+</sup>17], which defines a method for the automatic classification of 3D scans of shards with respect to their decoration. In particular the five most characteristic decorations, represented in Figure 1.10, define five different classes that are used



Figure 1.10: The 5 classes of the shard collection. Image from [DTC<sup>+</sup>17].



Figure 1.11: Highlighting the relief patterns: (a) depth map projected from the 3D points, (b) improved depth map by inpainting step, and (c) local variance map. Image from [DTC<sup>+</sup>17].

to locally characterize the surface.

First, the 3D point clouds are converted to depth maps by projecting them onto planes. This allows for a projection that does not cause any undesired distortion of the scanned object once processed. The projection is done using the PCA. To avoid issues with uneven samples (which may lead to black pixels that contain no info from the points), these ‘black’ areas are automatically detected and filled using a smoothing estimator propagating along the image gradient. The estimator is a weighted average of the neighborhood and the fast marching level-set method was used to fill missing image information. Then, a histogram normalization is applied to the depth map. Finally, a local variance estimator was applied on the depth map to highlight shallow relief information. Figure 1.11 shows an example of this process in action.

The local variance operator acts as a high pass filter to remove the curvature of the shard. Next a saliency map is extracted. The saliency region is assumed to be the highest textured region in the local variance map. As the elements making up the pattern have geometrical shapes, authors apply a detector of corner points and a density-based spatial clustering to extract the saliency region. Authors use the FAST method [RD06] for finding the saliency value. Examples of the final saliency map and all the previous one obtained for all five classes are shown in Figure 1.12.

Then, the authors binarize the saliency maps by thresholding the map values, using the Niblack’s algorithm [Nib85] plus an optimization that reduced the number of artifacts left by the Niblack’s algorithm. An example of the final binary maps are shown in Figure 1.13. The classification of the models is divided in two steps: the characterization of feature based on the binary maps





















	Class A (Diamonds)	Class B (Ovals)	Class C (Sticks)	Class D (Squares 3 rows)	Class E (Chevrons)
RGB Picture					
Depth Map					
Variance Map					
Saliency Region					

Figure 1.12: A selection of shards of the five most representative classes and their respective maps versions. Image from [DTC<sup>+</sup>17].

and the use of the extracted feature as input to train a multiclass support vector machine (SVM). First, author apply the *pyramid histogram of visual words* (PHOW) [BZM07] as a descriptor in the saliency region, which is a variant of the dense SIFT descriptor [Low04], extracted at multiple scales through a spatial pyramid scheme. A series of SIFT descriptors was computed on a regular grid with a spacing of the map pixels and different circular supports for scale variation (radius of 4, 6, 8, and 10 pixels). This sampling with overlapping patches at multiple scales was considered to efficiently characterize the repeating frieze. Then, the K-means clustering was used to build a codebook of visual words compiled from tens or hundreds of thousands of SIFT descriptors. The final classification is done with a C-SVM, with the one-against-all strategy. Different kernels are considered by the authors, with the best results obtained by the Gaussian kernel with a  $\chi$ -square distance [DD09]. While the work above is in spirit close to the goal of this manuscript, there are multiple limitations that block it from being a method that can be generalized to a more broad

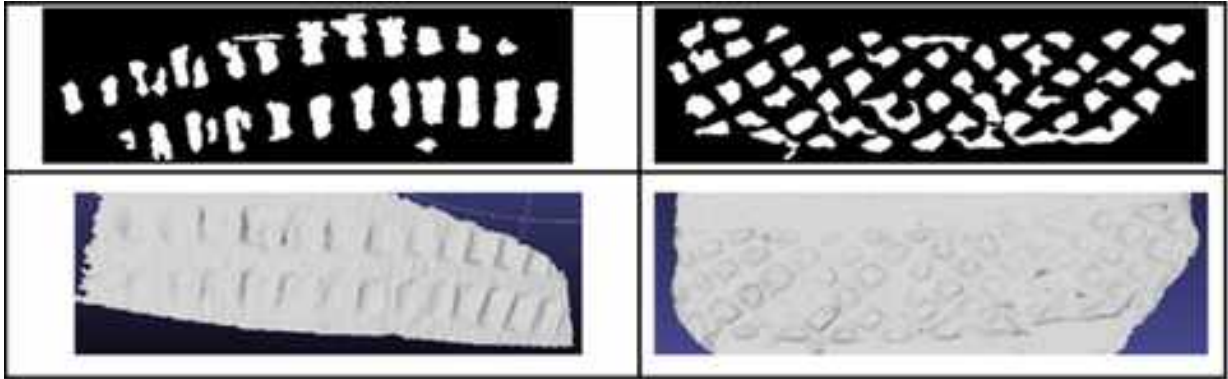


Figure 1.13: Top row: the final binary maps. Bottom row: the respective 3D point cloud. Image from [DTC<sup>+</sup>17].

use. Mainly, the fact that the shape of the scanned objects is basically flat or close to flat avoids entirely the problem of distortions when converting 3D data into 2D data.

### **Tree specimens classification based on bark reliefs**

The second contribution is the one presented by [OVSP13]. The goal of that work is to classify tree specimens from terrestrial laser scans. To solve the problem, the idea is to classify the trees based on their bark texture, which in the context of this work can be identified as a pattern. In Figure 1.14 we report the scans of the barks of the most common tree specimens, according to the authors.

The similarity is mainly driven by a patch in correspondence of the trunk at a fixed height and size. This allows patches roughly similar from tree to tree, which simplifies the characterization of the bark relief. Indeed, the relief is extracted by considering the difference between the original patch and its smoothed version. This leads to a 3D deviation map, which then is converted into a 2D deviation map via the Maximum Variance Unfolding [WPS05] (MVU) dimensionality reduction algorithm. The latter step basically transforms the 3D deviation map into a height map. Figure 1.15(Top) shows the 2D deviation map for the five most common tree specimens.

The main idea is to cluster the points over a certain height in order to define clusters or regions whose shape allows the authors to classify the different specimens. This is achieved by a thresholding process first (empirically set equal to the median value of the deviation values), followed by the DBSCAN [EKSX96]. Results of this process are shown in Figure 1.15(Bottom).

From the deviation map and its segments, the authors compute four characteristics that are used to classify the scans: the number of clusters per segmented 2D deviation map, the roughness, the principal component analysis (PCA) features and the shape and intensity features. In particular,



Figure 1.14: 3D point clouds of the five tree species. Image from [OVSP13].

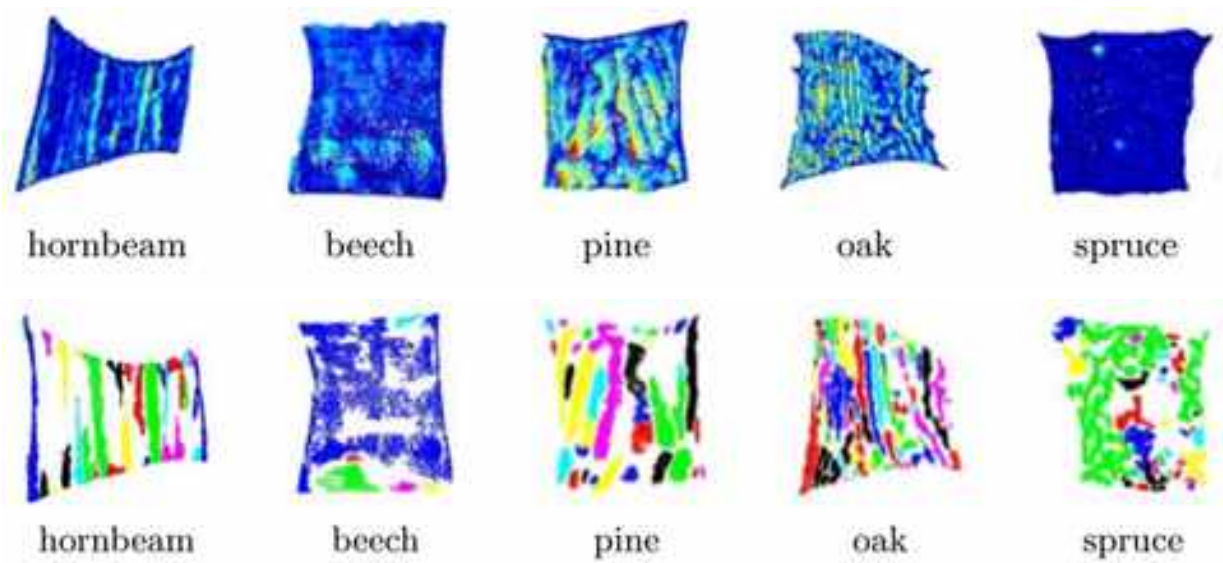


Figure 1.15: Top row: an example of 2D deviation map for each of the five species. Bottom row: an example of segmented 2D deviation map for each of the five species. Image from [OVSP13].

PCA features are a set of values that can be computed from an application of the PCA to the clusters of points, like the percentage of the total variance explained by each principal component, the maximum and the median distances between the observations and the center of the data set, the longest and shortest diameter, etc.. The minimum, maximum, mean, median and standard deviation values of these PCA features for all the clusters of the segmented 2D deviation map are used as for classification. The shape and intensity features are obtained from each cluster by considering a set of shape and intensity estimations. Again, authors use the median, the mean, the standard deviation, the minimum and the maximum of the intensity and the shape features of all the clusters and use them as classification attributes. This makes for a total of 128 features that can be extracted from each bark scan. Authors used the Random Forest [Bre01] classifier to select the 30 most relevant features, using datasets of trees with ground truth. Overall, this method classifies a 3D object based on its pattern and it is further extended in [SZ19] to almost

planar surfaces for the characterization of pecking decorations in rocky arts. However, such an height map analysis cannot be trivially extended to a general surface, due to the assumptions on the overall shape of the latter. Indeed, models with non-trivial topology make the local analysis required in the pattern recognition problem much more complex (i.e.: it is much harder to distinguish local surface variation from overall variations).



# Chapter 2

## Pattern-related surface properties

Patterns characterization depends on the study of surface properties that can define a pattern, being the characterization of a pattern modeled as a (small) property variation. This thesis deals with geometric patterns and colorimetric patterns. This chapter focuses on the properties used to characterize these two kinds of patterns. It is worth mentioning that this chapter is mainly devoted to the geometric properties of the surface. Indeed, we observed very minor changes in performances when considering properties similar to the gray-scale defined in different color spaces. Moreover, an in-depth analysis of all the existing ones is out of the scope of our work. However, for completeness, in the following section we briefly introduce the color space and we consider in our tests.

### 2.1 Color properties on surfaces

Colorimetric patterns are encoded by analyzing the variation of the color channels of the embedding one finds more appropriate for a given dataset. For example, a dataset with colorimetric patterns that depend strongly on the contrast can be described by the variation in the gray-scale embedding, or the HSV. On the other hand, if patterns are mainly of one color (e.g.: red), it could be useful to embed the colors in a color space that has one channel focused on that hue (e.g.: RGB, CieLab, CMYK).

We tested multiple color embeddings and, for our tests, the CieLab color space resulted in the best choice, for multiple reasons. Mainly, since our main application is assisting archaeologists in their reasoning, we look for patterns that they can spot with their eyes. This makes the CieLab color distribution very appealing, since it correlates the Euclidean distance between the colors (in that color space) with the color difference experienced by the human visual system.

The CieLab is a color-opposing embedding of the visible spectrum in which the difference be-

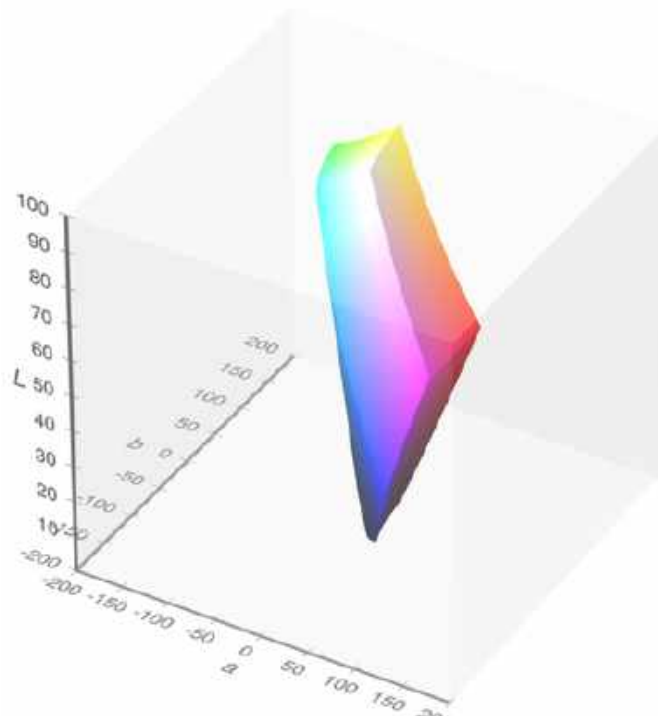


Figure 2.1: Representation of the color distribution in the CieLab color space. Source: [https://en.wikipedia.org/wiki/CIELAB\\_color\\_space](https://en.wikipedia.org/wiki/CIELAB_color_space).

tween colors with opposite tones (formally, complementary colors) is reflected in the position of the colors in the color space. In other words, assuming that the color space is a flat disk, opposite colors are placed in opposite poles. In particular, the way complementary colors are placed in the CieLab color space is similar to the one humans have, this does not happen in the other commonly adopted color spaces, such as the RGB. Figure 2.1 represents the way colors are distributed based in the CieLab color space. It has three color channels that, when combined, are able to represent the visible spectrum of light. These channels are the **L**-channel, which is related to the human concept of luminosity, the **a**-channel, which represents the red/green balance, and the **b**-channel, which represents the yellow/blue balance. As a side note, despite the appearance, the L-channel is not a translation, stretching or compression of the commonly adopted gray-scale of the RGB color space.

For our tests, the L-channel resulted enough to characterize the colorimetric pattern of our datasets. While some artifacts have decorations with a lot of different colors and variations, for ancient ones many colors are faded and the ones that are still visible are mainly defined by the contrast of two colors instead of the variations in the single color tones.

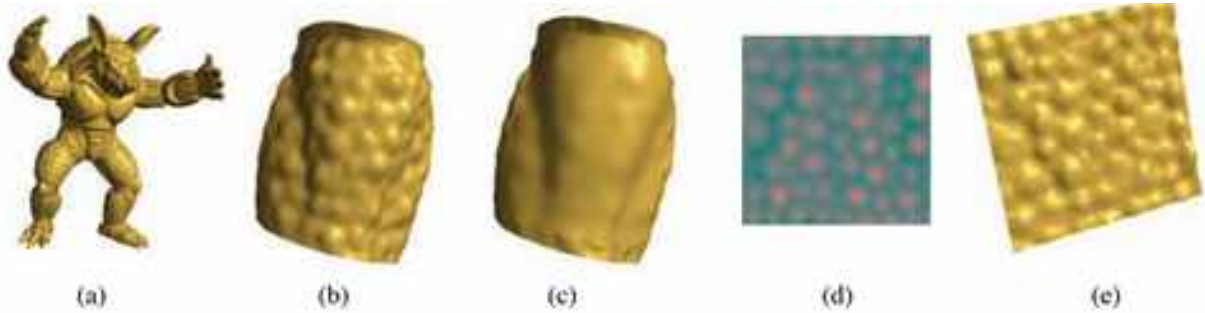


Figure 2.2: Geometric texture extraction based on height functions: (a) the armadillo model; (b) a curved sample patch; (c) the smoothed surface; (d) the extracted geometric texture image; (e) the reconstructed texture on a plane. Source: [LHGM05].

## 2.2 Geometric surface characterization

A completely different and more in-depth analysis is required for the geometric patterns. Several properties could describe and characterize local variations in the geometry of the surface, like the well known height functions (or height maps), the Laplace Beltrami operator and the curvature.

For example, in a local characterization via elevation (or height function) a pattern is defined as a vector of displacements with respect to the underlying shape [LHGM05] or in terms of rotation and displacement lengths from the normal vector [ADBA09]. The underlying shape is usually obtained via smoothing. A technique based on this approach is used, for example, to face the problem of texture synthesis and transfer [LHGM05], as shown in Figure 2.2.

On a similar note, starting from the theory, the Laplace operator generalizes the second order derivative to higher dimensions and characterizes the variation of a function. Indeed, the solution to the Laplace eigenvalue problem  $\Delta f = -\lambda f$  is an orthonormal eigensystem (with an infinite number of base elements). Thinking of the surface as a vibrating membrane, the higher is the eigenvalue, the higher is the corresponding frequency. In other words, smaller details of the surface can be found at higher frequencies. The discretization of the Laplace operator, which is called Laplace-Beltrami operator, requires complex calculations, thus it is usually approximated by different formulations. For example, a well known formulation [MDSB03] of this operator is the following:

$$\Delta v_i = \frac{1}{2} \sum_{j \in \mathcal{N}(v_i)} (\cotan(\alpha_{ij}) + \cotan(\beta_{ij})), \quad (2.1)$$

where  $v_i$  is a vertex of the mesh,  $\mathcal{N}(v_i)$  is its one ring neighborhood,  $\alpha_{ij}$  and  $\beta_{ij}$  are the angle opposite to the edge  $v_i v_j$  in the two triangles adjacent to  $v_i v_j$ . The applications of this operator are numerous, such as shape analysis (global and local), synthesis and shape correspondence.

Last but not least, the curvature is the geometric property most used to describe the concept of

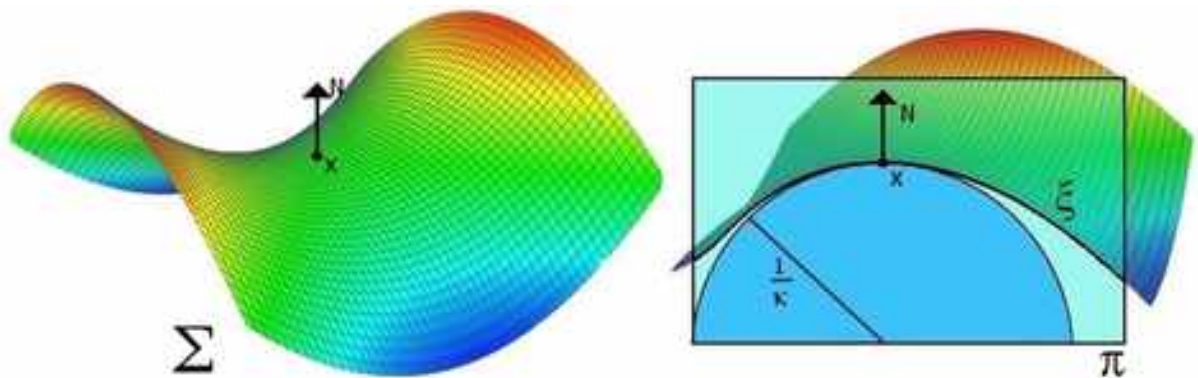


Figure 2.3: Representation of the normal curvature to the surface  $\Sigma$  in a point  $x$ . The oriented vector  $N$  represents the normal vector in  $x$ . On the right, the surface is sectioned along the plane  $\pi$  that generates the curve  $\xi$ . The radius of the osculating circle in  $x$  relative to  $\pi$  determines the normal curvature.

geometric surface variations. Its definition starts from the concept of the *principal curvatures*, which measure the maximum and minimum bending in different directions of a regular surface at each point  $x$ . The latter must be a differentiable point (in the 3D Euclidean space) of the surface (let us call it  $\Sigma$ ). To compute the bendings in all the possible directions in  $x$ , one has to define the intersection between the surface and a plane that passes through  $x$  and that contains the normal vector in  $x$ . Because of the regularity hypothesis on the surface, the intersection will always result in a curve, on which the classic definition of curvature can be applied (i.e.: the reciprocal of the radius of the osculating circle). The extreme curvature values assume the name of *maximal curvature*  $k_1$  and *minimal curvature*  $k_2$  and, together they are called *principal curvatures of the surface at  $x$* . The values are considered positive if the curve turns in the same direction of the normal, and the negative otherwise. An example of the plane/surface intersection is depicted in Figure 2.3. While  $k_1$  and  $k_2$  already contain local surface information, usually a combination of the two is used for more synthetic surface analysis. For example, the *Gaussian curvature*  $K$  is an intrinsic property defined as  $k_1 k_2$ . Another example is the *mean curvature*  $H$ , an extrinsic property defined as  $\frac{1}{2}(k_1 + k_2)$ . Similarly, the *shape index*  $SI$  is another extrinsic property defined as  $\frac{2}{\pi} \arctan \frac{k_2 + k_1}{k_2 - k_1}$ . All these possible combinations can serve different scopes (e.g.:  $K$  could be used to classify a point as elliptic, hyperbolic or parabolic) with their pros and cons.

Among elevation, Laplace-Beltrami operator and curvature, we focus our attention on the latter. To our experience, in our use cases, the height functions *as they are* are still too sensible to the deformations of the surface. Also, in some models the pattern is lightly chiseled on the surface, which makes the height map less significant. Notice that we use a variation of the elevation in Chapter 7 and Chapter 8: we thus do not discard this option entirely, still its performance with its classic definition are far lower with respect to those of the property we opted for (see later). Different is the problem with the Laplace Beltrami operator: to the best of our knowledge,

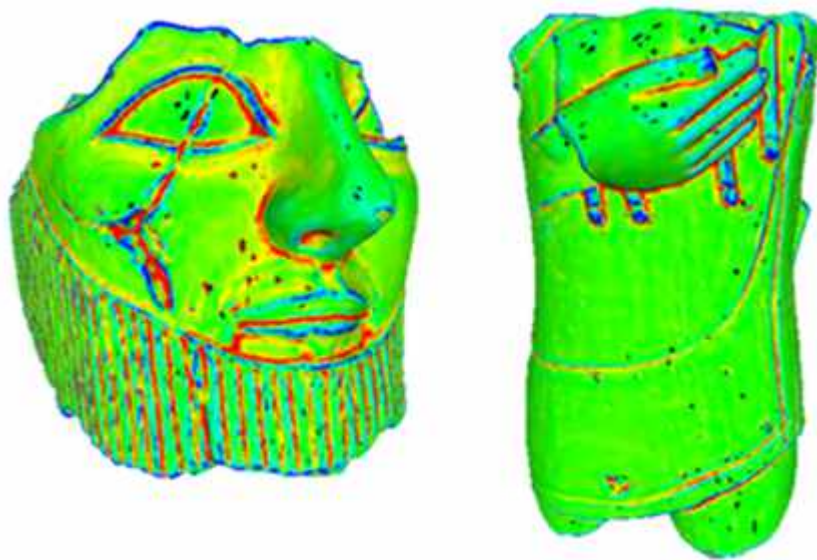


Figure 2.4: Representation of the mean curvature on some real artifacts. Colors scales with the values of the curvature. Notice how it is almost impossible to distinguish between patterns and non patterns just based on the colors and how easier the task became once configurations of values are taken into count (e.g.: a beard is characterized by a ‘line’ of vertices that have the same curvature value).

computing higher frequencies of its solution is highly demanding in terms of computational cost, especially with meshes with a very high number of vertices. Indeed, our use cases have very high resolution models, which usually makes the use of this operator prohibitive.

For these reasons, we focus our attention on curvature and the different ways to approximate it on surfaces to promote the identification of well-detailed and isolated geometric variations of the surface while discarding the whole shape structure.

Indeed, it we will use the curvature variations between different points of the mesh as starting data for our pattern descriptors. That is because the curvature, *in its continue definition*, is sensitive enough to vary on any pattern, assuming configurations of curvature values that repeats over the pattern area (see Figure 2.4).

While the curvature punctual value is not enough for a classification by itself, an effective and compact description of these configurations will lead to a proper pattern descriptor definition. However, since we are working with 1) machines with finite precision and 2) on surface discretization, the way we approximate the concept of curvature is of major importance and it needs to be deepened.

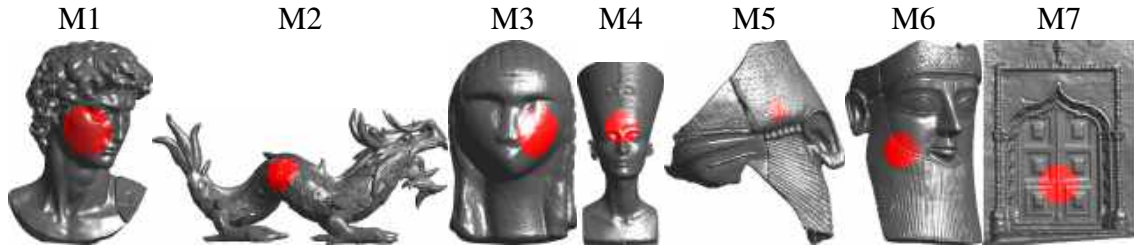


Figure 2.5: The test-beds for the curvature estimation benchmark for small reliefs. *Models 5 & 6* come from the GRAVITATE use case [UUTCIC<sup>+</sup>]. The red areas contain 20000, approximately.

## 2.3 Comparative analysis of curvature evaluation methods

The obstacle in actually using the curvature in a real world scenario is that the way it is computed on triangulations (and point clouds) is not trivial. For example, triangulations are flat faced (with zero curvature), thus the curvature is concentrated along the edges and vertices (on which the surface is not differentiable), while point clouds are not even surfaces. However, in the current literature, there are mainly three approaches to the computation of curvature on meshes [GG06]. The first deals with the problem by moving the problem in the continuum, with local surface fitting. The second looks for equivalent descriptors starting from basic properties of continuous operators but directly applied to the discrete settings, with an eye of regard for the validity of differential properties during the computation. Finally, the third looks for discrete approximations of differential properties of surfaces, from which the curvature can be derived. An analysis of these approximations performances is necessary for the use-cases of our work.

For the work described in this thesis, the existing benchmarks for the comparison of curvature estimators presents a limitation. In particular, the curvatures estimations are evaluated on almost smooth surfaces and by looking at the overall surface without focusing on the details. On the contrary, the following comparison focuses on the practical behaviour of the algorithms on *local geometric variations*, such as chiselling, incisions, small bumps on the surface, to assess the capability of different approaches to identify local features or, as expected, patterns. Differently from [GG06, MSR07] and [VVP<sup>+</sup>16], the models considered in these comparisons do not correspond to smooth surfaces and are all obtained with scans of real objects. Unfortunately, this means that it is impossible to know the exact curvature values to compare the estimations obtained. For this reason, the evaluation is mainly visual and quantitative metrics are provided only for the localization of curvature values on specific features and the frequency of the outlier values. Seven 3D models are selected, trying to cover a wide range of different reliefs that are significant when looking for the characterization of local features and details. Figure 2.5 overviews the selected models. Triangulated meshes are used to represent these surfaces, which is a standard-de-facto representation of models reconstructed from laser scans and for which there is a rich variety of algorithms.

All surfaces are sampled with 1M vertices, with the exception of  $M1$  and  $M7$ , which have  $\sim 550K$  vertices. Figure 2.5 visually represents the vertex density for each model; the red regions contain 20000 vertices, approximately. The local variations in these surfaces come in various shapes and degrees, ranging from sharp variations (facial traits of  $M1$  and  $M4$ ) to smooth ones (hairs of  $M3$  or scales of  $M2$ ), from small ones (like the hair of  $M1$ ,  $M5$  and  $M6$ ) to bigger ones (scales of  $M2$ ), from frequently repeated (circlets of  $M5$  and  $M6$ ) to more localized ones (door decorations of  $M7$ ).

We selected eight representatives of the various curvature estimator strategies. Far from being exhaustive, our selection falls on implementations that are freely available and, in our knowledge, of common use in the geometry processing community. Namely, we are considering:

- the Algebraic Point Set Surface (AP) fitting method [GG07, GGG08] as implemented in [CCC<sup>+</sup>08];
- the curvature discretization based on the *cotan* discretization of the Laplace-Beltrami operator (MA) [MDSB03] following the implementation provided in [CCC<sup>+</sup>08];
- the pseudo-inverse quadratic fitting method (QF) in [CCC<sup>+</sup>08];
- the normal cycles approach (NC) [CSM03] as in [Pey];
- the curvature estimation based on an adaptive re-weighting of the vertices in the neighborhood proposed in [KSNS07] (KA) (authors' implementation);
- the discrete estimation of the second fundamental form proposed in [Rus04] (TR) (author's implementation);
- the normal curvature estimation based on the Euler formula proposed in [DW05] (DO) (authors' implementation);
- the least-squares curvature tensor approximation and iterative diffusion smoothing proposed in [Chu01] (CH) (authors' implementation).

As a broad classification, the proposed methods are divided in: *fitting* methods, based on the fitting of mathematical surface primitives (for instance quadratic surfaces like spheres, or cubic Bézier patches or Hermite RBF fitting) [GI04, CP03, GG07, GGG08, PV18]; *direct discretization* methods of the curvature in a vertex (e.g., in terms of the angle excess or variations [MPS<sup>+</sup>04]) and of the curvature tensor [MDSB03, Tau95a, Rus04, KSNS07, DW05, ZGYL11]; *indirect approximations* of related quantities (e.g. second form) [CSM03, CSM06, LBS07].

For every method that allows the user to tune its parameters (namely NC, KA, AP, MA) the default algorithm settings as proposed by the authors are used, without altering the neighbour size or the smoothing intensity. While this is in itself a fair way for comparing different methods,

	M1	M2	M3	M4	M5	M6	M7
AP	5.5%	17.1%	8.3%	1.6%	54.7%	51.3%	76.8%
CH	> 0.05%	0.9%	1.0%	> 0.05%	10.3%	10.5%	35.1%
DO	0.1%	2.6%	0.7%	0.1%	28.2%	26.6%	52.4%
KA	> 0.05%	1.1%	> 0.05%	> 0.05%	26.4%	24.3%	49.0%
MA	0.8%	5.8%	4.7%	0.3%	31.2%	29.4%	60.7%
QF	0.4%	2.3%	0.5%	> 0.05%	24.7%	23.7%	49.3%
NC	0.7%	2.4%	0.1%	> 0.05%	> 0.05%	0.1%	0.2%
TR	0.1%	2.0%	0.3%	> 0.05%	26.2%	24.6%	51.1%

Table 2.1: Percentage of mesh vertices classified as a outliers.

there could be a parameter optimization that better suits the proposed problem. Anyway, the lack of criteria for tuning the parameters and the lack of ground-truths suggested to keep the default settings. Depending on the technique, different surface variations are highlighted: this implies there is not a best method for all applications. Not only, the peculiarity of a method like the sensitivity to small scales of the geometric variations, could become a detriment in applications that need, for example, a noise estimation.

The comparison is limited to the *mean curvature* values. Figure 2.6 visually overviews the results we obtained. The mean curvature is represented with colors and it ranges from  $-2$  (blue) to  $2$  (red). Such an interval intuitively spans the visible curvature variations in the 3D models (i.e.: a rough approximation of the theoretical extreme values of the curvature). Curvature values that exceed from that interval are considered as *outliers*. An exception is the NC method, which approximates the tangent bundle of the curvature tensor rather than the curvature values. Looking at the curvature variation we select the interval  $[-0.05, 0.05]$  as a reference interval for NC.

As already highlighted in [VVP<sup>+</sup>16] every approximation algorithm suffers from ambiguities, like the non-uniqueness of the fitting surface or the sensitivity of the method to local perturbations. Looking at Figure 2.6, it is possible to see that DO, CH, KA, MA, QF and TR output very similar mean curvature estimations. This group of methods is able to effectively highlight small repeated variations (beard and circlets in *M5* and *M6*), while consistently keeping at 0 the curvature estimation on the flat areas. A downside is that these methods seem to output mainly extreme curvature values (around  $\pm 2$ ) and 0, rarely passing through other values (it can be observed by the lack of orange/light blue vertices). DO (see *M2*) presents a greater continuity of the colors.

In this sense, the best approximation is obtained by NC (especially visible in *M1* and *M2*). Also, NC is one of the few methods that put high contrast in the curvature values on convex/concave areas of the model, together with AP. A downside of AP is its sensitiveness to really small variations, especially visible in the flat areas in *M7*.



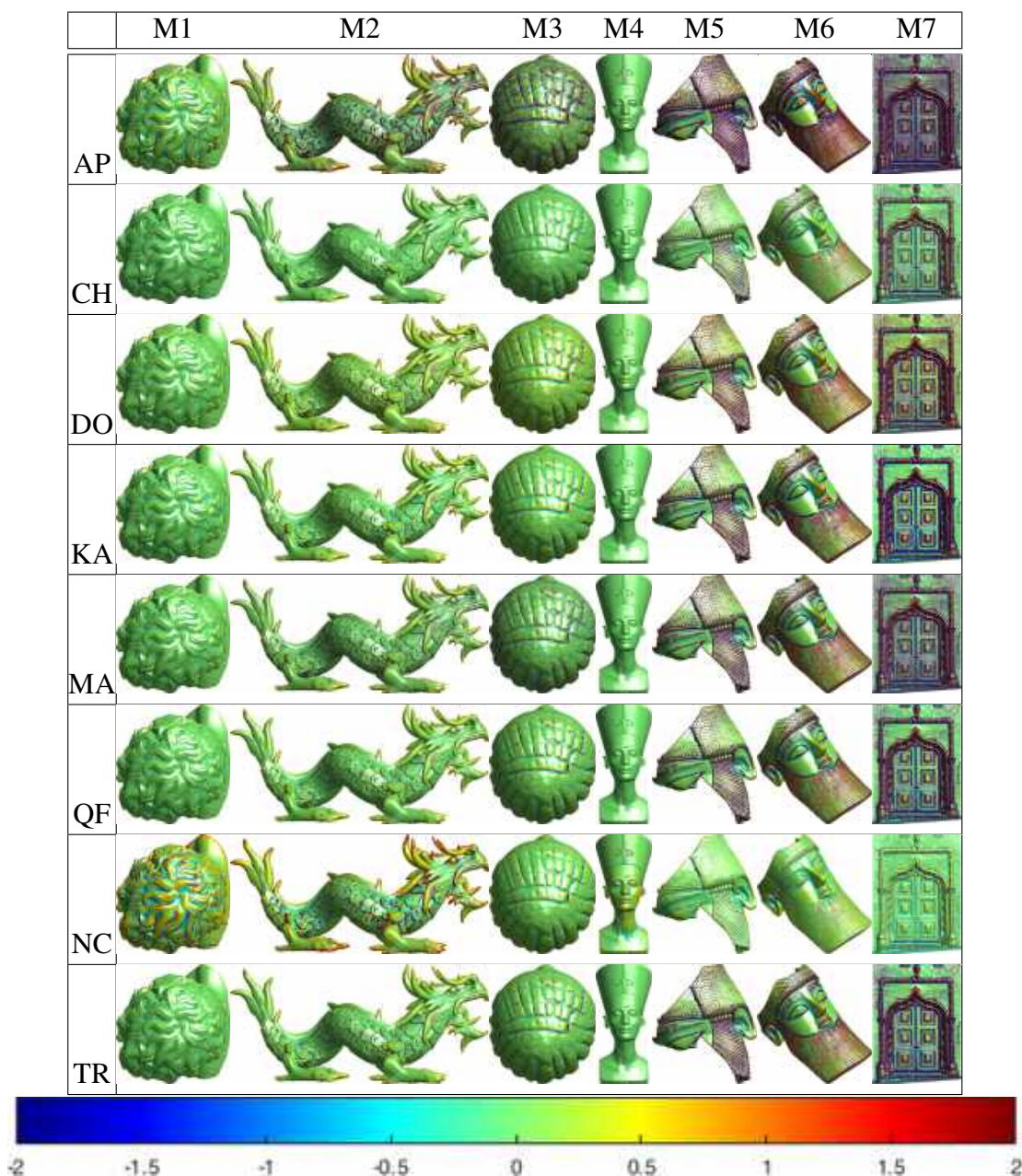


Figure 2.6: Visual representation of the mean curvature values for the eight algorithms. The color-bar is reported at the bottom. The values for NC are re-scaled into  $[-2, 2]$  from  $[-0.05, 0.05]$ .

Tables 2.1 and 2.2 present some statistics on the outlier distribution. For this analysis, the absolute value of the curvature estimations is considered. The values in Table 2.1 represent the percentage of vertices that are out of the expected curvature interval. The values in Table 2.2

give an idea of the range of variation of the mean curvature in correspondence of the outliers (in terms of the 5 – th decil, last permil and maximum value). The results in Table 2.2 suggest that for meshes with dense vertices like those adopted in our experiments, the NC approach is the most stable in terms of the absolute variation of the mean curvature. Considering both Tables 2.1 and 2.2, the 5 – th decil, which is an approximation of the average of the mean curvature in the outliers, is very close to 2 in most cases. Note that the adopted definition of the curvature outlier is based on the empirical observation that for many smooth surfaces the mean curvature values are enclosed in the interval  $[-2, 2]$ ; however, the presence of many outliers does not necessarily imply that a method is unstable. Also, as in the case of NC, the values estimated by a given method could be consistent in terms of variation even if the values are not those that we expect from the theory behind the curvature on surfaces. This fact confirms that all the methods provide a reasonable estimation of the curvature values, with the exception of NC, which captures the curvature variations rather than its values.

The results show that no single estimator is suitable for all possible input data but the methods that smooth the curvature estimation in a vertex neighbour provide visually and quantitative better performances when highlighting features on scans of manufacts. The NC method, for the combination of availability, relief characterisation and quickness in computation, results as the best method between those compared.

For the rest of this manuscript, the curvature will be estimated with the NC method.

## Related publications

- E. Moscoso Thompson and S. Biasotti, *A Preliminary Analysis of Methods for Curvature Estimation on Surfaces With Local Reliefs*. In The Eurographics proceedings, 2019.

	M1	M2	M3	M4	M5	M6	M7
AP	2.56 [9.28] < 10 <sup>5</sup> >	2.98 [15.73] < 10 <sup>6</sup> >	2.71 [11.74] < 22.58 >	2.47 [8.19] < 11.55 >	4.61 [30.53] < 77.27 >	4.77 [31.66] < 87.44 >	8.10 [72.47] < 363.7 >
CH	2.30 [14.04] < 14.04 >	2.50 [14.61] < 39.77 >	2.70 [13.65] < 36.71 >	2.34 [6.30] < 6.30 >	2.66 [12.18] < 112.9 >	2.76 [12.21] < 185.1 >	3.56 [25.53] < 2649 >
DO	2.24 [31.08] < 31.94 >	2.44 [10.45] < 22.73 >	2.47 [37.97] < 187.1 >	2.31 [8.88] < 8.92 >	3.28 [22.31] < 164.1 >	3.33 [20.86] < 139.2 >	4.76 [35.11] < 582.4 >
KA	2.27 [55.71] < 55.71 >	2.36 [7.49] < 17.42 >	2.33 [40.09] < 40.09 >	2.33 [6.95] < 6.95 >	3.17 [12.92] < 78.18 >	3.20 [12.73] < 56.50 >	4.38 [27.48] < 5296 >
MA	2.34 [15.89] < 84.21 >	2.66 [14.65] < 26.51 >	2.99 [38.48] < 1097 >	2.40 [12.20] < 23.34 >	3.38 [93.92] < 10 <sup>8</sup> >	3.47 [130.8] < 10 <sup>7</sup> >	5.06 [119.1] < 10 <sup>5</sup> >
QF	2.34 [23.67] < 32.23 >	2.57 [22.70] < 74.22 >	2.39 [8.90] < 182.9 >	2.31 [7.30] < 7.30 >	3.24 [17.15] < 1074 >	3.27 [18.37] < 6465 >	4.44 [37.53] < 9568 >
NC	0.056 [0.246] < 0.260 >	0.056 [0.459] < 0.821 >	0.056 [0.112] < 0.115 >	0.052 [0.060] < 0.060 >	0.054 [0.092] < 0.092 >	0.057 [0.213] < 0.213 >	0.064 [0.303] < 0.337 >
TR	2.22 [12.93] < 12.93 >	2.44 [8.32] < 14.23 >	2.35 [14.84] < 42.34 >	2.24 [6.61] < 6.61 >	3.18 [14.29] < 29.34 >	3.23 [14.49] < 43.14 >	4.61 [34.26] < 6485 >

Table 2.2: Approximated 5 – th decil, [last permil] and <maximum values> of the curvature in correspondence of the outliers.

# **Part II**

## **Pattern Descriptors**

## Chapter 3

# Edge Local Binary Pattern

The recognition of geometric patterns requires a good characterization of the local shape properties, which has to be: *robust* to different model representations; *sensitive* to the *local* geometric variations that characterize the surface; as much as possible *independent* of the surface bending, while keeping a reasonable computational complexity. A descriptor with these characteristics is not enough to solve the pattern recognition problem by itself, but it is sufficient to face the pattern retrieval one. The retrieval of image texture is largely addressed and many methods exist in the literature. This chapter proposes a novel extension of one of the most famous of these methods, namely the Local Binary Pattern description [OPH96, OPM02]. Such an operator is easy and quick to compute, while being very robust and flexible.

The proposed extension, called *edge Local Binary Pattern* (edgeLBP), is:

- able to deal with surface tessellations whose faces are made of convex polygons;
- robust to non-uniform surface samplings;
- invariant to object Euclidean transformations (roto-translations)
- able to deal with surface tessellations characterized by non-uniform vertex distributions and different types of faces, such as triangles, quadrangles and, in general, convex polygons,
- very competitive with the at the time state of the art.

With respect to a previous extension of the LBP to triangle meshes, the so-called meshLBP [WTBdB15, WBB15], the edgeLBP evaluation is based on the vertices of the tessellation. Then it adopts a sphere-mesh intersection approach to determine the rings around a vertex and define

a re-sampling criterion to obtain the same number of samples on each ring. A MatLab implementation of this method can be found on GitHub at <https://github.com/EliamTH/edgeLBP>.

Experimental results exhibit very good performances on various datasets, showing the good potential of the proposed approach for real world applications. Before starting the presentation of the method, we outline the Local Binary Pattern as implemented on images.

## Local binary Pattern

The LBP is a reference description for texture recognition in still images [OPH96]. A lot of LBP variations exist [BJNL14] and they all share a similar idea: comparing a pixel with its neighbor pixels in a particular order, then treat a statistic on the results of the comparison as a descriptor (usually histograms). In the following, two variations of the LBP on gray scale images are presented. These are the ones used in the first iteration of the meshLBP and are used in the rest of the manuscript.

Let  $I$  be a gray-scale image characterized by a pattern,  $i \in I$  a pixel and  $h$  the function such that  $h(i)$  is the gray-level value of  $i$ . We denote  $RING_1^i = \{i_1, \dots, i_8\}$  the set of 8 pixels adjacent to  $i$ , see Figure 3.1(a). Usually, the ring  $RING_1$  is clockwise ordered moving from the top left pixel. A binary string  $\mathbf{str}$  of 8 bits is associated with  $i$  to code the variations of the gray-scale values of  $i$  and the pixels in  $RING_1^i$ .

For each  $i_j \in RING_1^i$ , the value  $\mathbf{str}(j)$  is defined as follows:

$$\mathbf{str}(j) = \begin{cases} 1 & \text{if } h(i) < h(i_j) \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

The LBP operator labels the pixel  $i$  with a scalar value derived from  $\mathbf{str}$  as follows:

$$LBP(i) = \sum_{j=1}^8 \mathbf{str}(j) \alpha_k(j), \quad (3.2)$$

where  $\alpha_k$  is a weight function that determines the size of the descriptor. The most popular choices for  $\alpha_k$  are:  $\alpha_1(j) = 1 \forall j$  and  $\alpha_2(j) = 2^j \forall j$ .

To achieve a multi-resolution description, it is possible to extend the LBP operator through a multi-ring coding [OPM02]. For each pixel  $i$ , a sets of concentric rings centered in  $i$  ( $RING_2^i, RING_3^i, \dots, RING_{n_r}^i$ ) with increasing radii values is considered, see Figure 3.1(a-b). Note that the rings are non necessarily square rings of pixels. Each ring is sampled with a pre-defined number  $p$  of pixels, so that the string  $\mathbf{str}$  has the length  $p$  on every ring, as shown in Figure 3.1(c) for  $p = 8$ . Finally, the multi-ring LBP descriptor is a matrix whose  $k$ -row corresponds to the LBP descriptor relative to  $RING_k$ . For completeness, note that the way rings are defined is

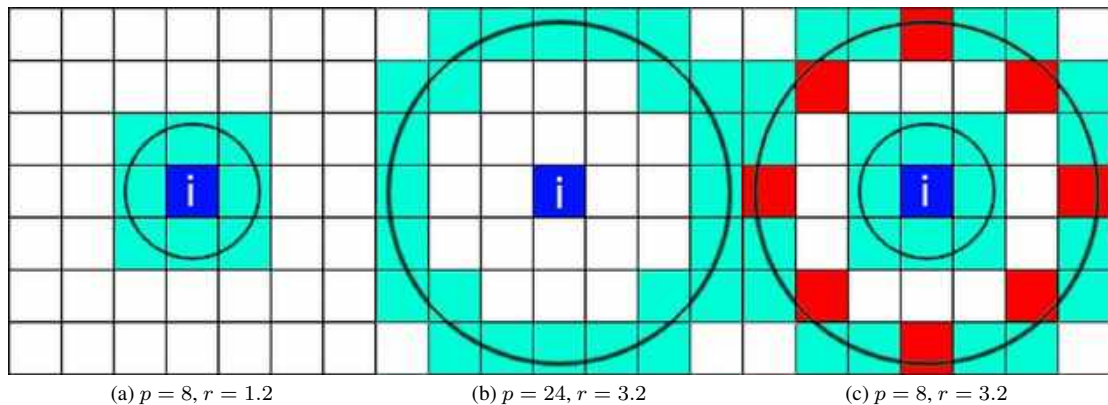


Figure 3.1: (a-b): Rings with different radii  $r$  relative to the pixel  $i$ . (c): Uniform down-sampling (from 24 to 8) for the pixels of the ring in (b). The down-sampled ring is represented by the red pixels.

not unique: we consider concentric rings of pixels, but is also possible to consider the Euclidean distance between the central pixel and its neighborhood and to select the pixels that contribute to a given ring by proximity.

### 3.1 Method description

This Section is dedicated to the description and setup of the edgeLBP. The following Sections explore the method pipeline in details, as well as the parameter settings and the experimental results.

#### 3.1.1 The edgeLBP description

The multi-ring LBP operator is extended to deal with surface tessellations using a sphere-mesh intersection technique. With the term *surface tessellation*, we mean a polygon mesh  $T = (V, E, F)$  which is a collection of vertices  $V$ , edges  $E$  and faces  $F$  that defines the surface of a polyhedral object. In our settings, we also assume that the faces are convex polygons. Popular examples of surface tessellations are triangle meshes, quadrangulations, or Centroidal Voronoi Tessellations (CVT) [DFG99], some examples of possible surface representation are shown in Figure 3.2. While a pixel grid has the same connectivity anywhere, surface tessellations can be widely *irregular*. By irregular we mean that the vertices can be non uniformly distributed over the surface. Furthermore, the faces of the tessellation may have different area, shape and number of edges. We also assume that each pattern property can be coded as a scalar function  $h$  defined on the vertices of the tessellations, formally,  $h : V \rightarrow \mathbb{R}$ . The notion of ring is crucial for

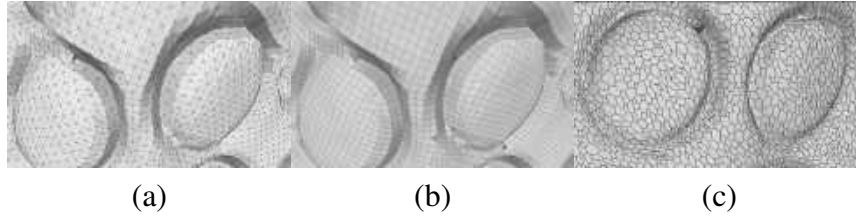


Figure 3.2: Detail of three surface tessellations of the same pattern: (a) a triangle mesh, (b) a quadrangle mesh, (c) a convex polygon mesh.

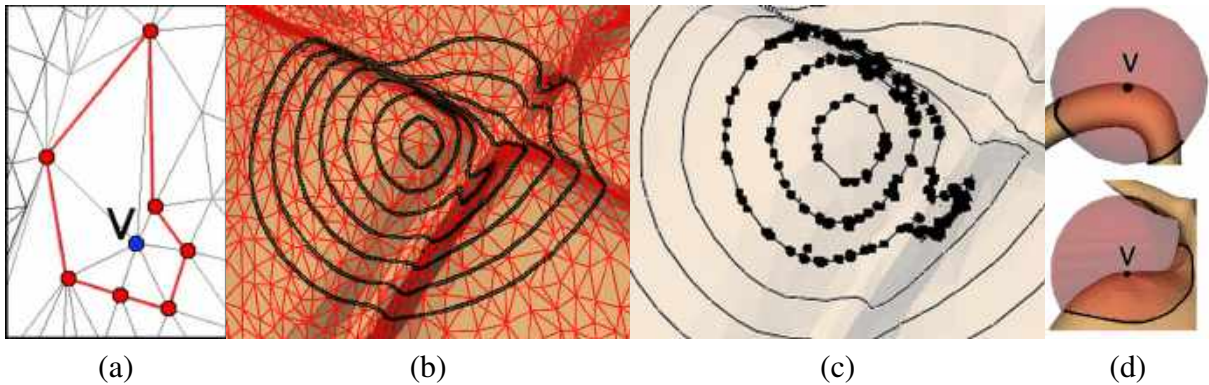


Figure 3.3: (a): The ring of the vertex  $v$  (in blue) formed by vertices over a triangle mesh; (b): multiple rings of a single vertex are shown; the black dots in (c) ( $p_i$  in our notation) represent the intersections between black rings in (b) and the edges of the mesh; (d), from top to bottom: first,  $\mathcal{S}_k^v$  corresponds to the connected component that contains  $v$ , the other component is discarded; second,  $\mathcal{S}_k^v$  is non-simply connected and  $v$  is non-admissible.

the LBP description. In case of triangle meshes, an intuitive transposition of the notion of ring would be a ring defined as a set of vertices. In Figure 3.3(a) we show the ring of the vertex  $v$  with the sequence of red edges and vertices. Indeed, the irregularity of the mesh elements strongly influences rings defined on mesh elements only, because it would not be invariant to different tessellations of the surface, even simple edge swaps. Rings that are associated to different elements of the tessellation could carry information about surface portions with a significantly different shape.

To overcome these limitations, given a surface tessellation  $T$ , we define the ring of a vertex  $v \in V$  as the intersection of the surface tessellation with a sphere centered in  $v$  of a given radius  $r$ . Then, we look at the intersections  $p_i$  between the sphere and the *edges* of the tessellation, creating a set of points  $Q = \{p_1, p_2, \dots, p_k\}$ , that approximates the curve that is the intersection between the sphere and the surface. We linearly interpolate the set  $Q$  to obtain a continuous and closed curve  $\mathcal{C}$  that represents the ring;  $\mathcal{C}$  is oriented counterclockwise with respect to the vector in  $v$  normal to  $T$ . The value  $h(p_i)$  on the points of  $Q$  is determined by the weighted average of



the value the function  $h$  assumes on the vertices that limit the edge  $e \in E$  such that  $p_i \in e$ . In general, the number of elements of  $Q$  varies from one ring to another, because of the increasing radius and the irregularity of the tessellation. To keep the number of elements constant on every ring, we sample  $\mathcal{C}$  with a fixed number  $p$  of points; we call  $p$  the *spatial resolution*.

To achieve a multi-ring representation, for any vertex  $v \in V$  we consider  $n_r$  rings,  $RING_k^v$ . Let  $\mathcal{S}_k^v$  be the surface portion of  $T$  that contains  $v$  and has the  $ring_k^v$  as its boundary,  $k = 1 \dots n_r - 1$ , then the relation  $\mathcal{S}_k^v \subset \mathcal{S}_{k+1}^v$  holds for each  $k$ . We take advantage of this relation to optimize the sphere-tessellation intersection adopting a region growing expansion around the vertex  $v$ . Examples of the intersection of the sphere at increasing radius around a vertex are shown in Figure 3.3(b-c).

Similarly to the standard LBP approach and to avoid a possible ambiguity close to the surface boundaries, we consider as *admissible* only the vertices for which all the  $n_r$  rings are closed curves.

In general, the sphere-surface intersection can produce multiple, closed curves that bound either a multiple connected or a disconnected portion of surface; some examples are shown in Figure 3.3(d), for a detailed vertex classification based on a sphere-mesh intersection approach we refer to [MPS<sup>+</sup>04]. Using a region growing approach,  $\mathcal{S}_k^v$  is the portion of the sphere-surface intersection that contains  $v$ . If the boundary of  $\mathcal{S}_k^v$  is a closed curve,  $v$  is considered an admissible vertex, otherwise it is *non-admissible* for the edge-LBP. Note that with the edge-LBP we are interested to code local geometric variations on the surface (like corrugations, incisions, and so on), therefore the radius  $r$  should be kept small with respect to the overall dimension of the surface. This implies that the choice of the radius  $r$  is crucial for the type (and the size) of the patterns we are going to identify; indeed it must be not too large to avoid mixing global and local surface information and not too small to be significant. In practice, multiply-connected regions appear only in case of topological noise, like small handles and self-intersections of the mesh and in our experiments over thousands of tessellations we never met admissibility problems.

The routine that is adopted to evaluate the edgeLBP over a single ring of a vertex and how to extend it to a multi-ring representation is detailed in the following.

1. *RingExtraction* - The ring  $Q$  given by the intersection between the sphere of radius  $r$  centered in  $v$  and the tessellation edges is computed according to the procedure detailed in Algorithm 1. The function  $VE(v)$  calls the basic function to the data structure that returns the list of all the edges that are incident to the vertex  $v$ . Once the edges that intersect the sphere with center  $v$  of radius  $r$  are identified (lines 2-15 of Algorithm 1), the coordinates of the intersection points  $p_i$  are computed (the function *EdgeSphereIntersection* numerically evaluates the intersection of an edge with a sphere).

We propose a method to sort  $Q$  with respect to a starting point  $\tilde{p}$ .  $\tilde{p}$  is selected according to a shape-based criterion and therefore is rotation and translation invariant. Starting from  $\tilde{p}$  the function *SortingP* counter clock-wisely sorts the points of  $Q$  with respect to the normal

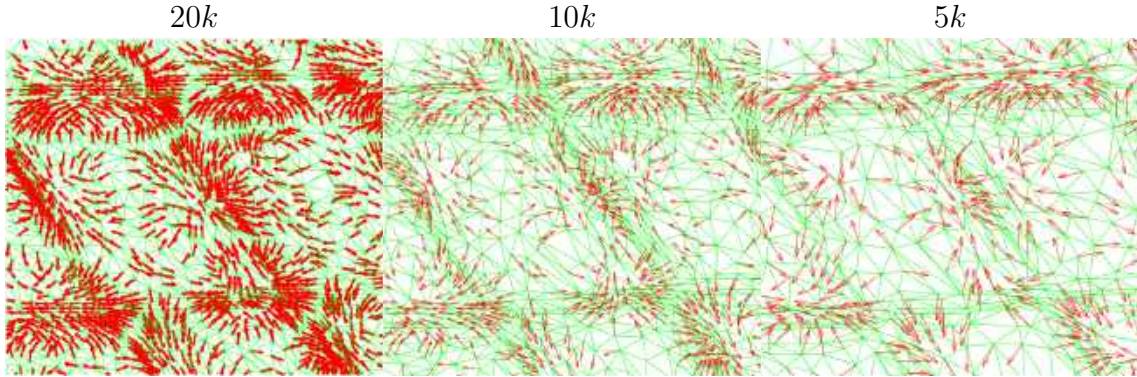


Figure 3.4: Arrows represent the orientation of vector that connect a vertex  $v$  with the starting point of the rings centered in  $v$ ; from left to right, details on a surface mesh of  $20K$  vertices and two re-samplings with  $10K$  and  $5K$  vertices: the choices of  $\tilde{p}$  is robust to mesh decimation and depends on the local geometry of the surface.

to the surface in  $v$ . Even if this sorting would not influence the  $\alpha_1$  representation we adopt in this chapter, it would become crucial if considering the  $\alpha_2$  one. The point  $\tilde{p}$  verifies the relation:

$$\tilde{p} = \arg \max_{p_i \in Q} h(p_i).$$

In Figure 3.4 we detail the choice of  $\tilde{p}$  for three meshes that correspond to three different resolutions of the same model. There, for each vertex we depict the unit vector defined as  $\mathbf{n} = \frac{\tilde{p}-v}{\|\tilde{p}-v\|}$ . As expected, the starting point of the rings is stable in most of the vertices of the mesh. This fact was confirmed in numerous experiments we performed on meshes of different resolutions. In case of symmetries around a vertex, multiple choices of the starting point are possible: we select the candidate point that is the farthest from the other elements of  $Q$ .

2. *RingResampling* - Given a ring  $Q$  representing a simple, closed curve we identify  $p$  equidistant samples  $s_k$ ,  $k = 1, \dots, p$  on  $Q$  as detailed in Algorithm 2. The values  $h(s_k)$  are linearly approximated (function *Sample* in Algorithm 2) as follows: denoting  $p_i$  and  $p_{i+1}$  the two consecutive points of  $Q$  on which the sample  $s_k$  falls, the value  $h(s_k)$  is equal to the weighted mean of  $h(p_i)$  and  $h(p_{i+1})$ . At the end of this procedure, the values  $h(s_k)$ ,  $k = 1, \dots, p$  are returned in the array  $S$ .
3. *edgeLBP Evaluation* - Once the value of the function  $h$  is known on the sample set  $S$ , the evaluation of the edgeLBP on the vertex  $v$  is straightforward. Here the function  $\alpha$  represents the weight function  $\alpha_1$ .

When extending the edgeLBP evaluation to multiple rings, the *RingExtraction* procedure is modified to take advantage of the nested nature of the rings; i.e., rings are computed increasingly with

---

**Algorithm 1: RingExtraction.**

*Notes:*  $d$  is the Euclidean distance,  $U$  is the list of the edges that may have a  $p_i$  on them (initially empty),  $L$  is the list of edges the algorithm has already checked.

---

**Input :** A tessellation  $T = (V, E, F)$ , a vertex  $v \in V$ , a radius  $r > 0$ .

**Output:** A set  $Q$  of samples  $p_i$  of the ring with center in  $v$  of radius  $r$ .

---

```
1 begin
2    $L \leftarrow \emptyset$ 
3    $U \leftarrow VE(v)$ 
4   while  $U \neq \emptyset$  do
5     for  $e = (v1, v2) \in U$  do
6       if  $(d(v1, v) - r) * (d(v2, v) - r) < 0$  then
7          $L \leftarrow L \cup \{e\}$ 
8       end
9       if  $(d(v1, v) - r) < 0$  or  $(d(v2, v) - r) < 0$  then
10         $U \leftarrow U \cup VE(v1) \cup VE(v2)$ 
11      end
12      mark  $e$ 
13    end
14     $RemoveMarked(U)$ 
15  end
16  for  $e \in L$  do
17     $Q \leftarrow EdgeSphereIntersection(e, r, v)$ 
18  end
19  return  $Q \leftarrow SortingP(Q, T)$ 
20 end
```

---

respect to the radius  $r$ . The initialization in Algorithm 1 of the set  $U$  of edges that are suitable for the sphere-surface intersection, starts from the edges that originated the previous ring and does not take into account edges already visited. Moreover, only the biggest ring  $RING_{n_r}$  is sorted: we sort all the other rings centered in the vertex  $v$  consistently with this sorting. In particular, we consider the plane  $\pi$  passing through  $v$  with  $\mathbf{w} = \mathbf{n}(v) \times (v - \tilde{p})$  as its directional vector, where  $\mathbf{n}(v)$  is the normal, unit vector to the surface in  $v$  and  $\tilde{p}$  is the starting point of  $RING_{n_r}$ . Then, we choose as the starting point on each ring the closest point to  $\tilde{p}$  and order all the rings counterclockwise with respect to the normal in  $v$ .

In our settings, we opted for a uniform distribution of the ring radii. For instance, denoting  $r_{max}$  the maximum radius is will be  $\frac{r_{max}}{n_r}, 2\frac{r_{max}}{n_r}, \dots, r_{max}$ .

---

**Algorithm 2: RingResampling**

---

**Input** : A set  $Q$  of intersection points, the function  $h$ , the spatial resolution  $p$ .

**Output**: An array  $S$  of  $p$  scalar values  $s_k$ .

```
1 begin
2    $length \leftarrow \sum d(p_i, p_{i+1})$ 
3    $dl = \frac{length}{m}$ 
4    $idx_{end} \leftarrow 2$ 
5    $index \leftarrow 1$ 
6    $s(1) \leftarrow h(p_1)$ 
7   while  $size(S) \neq m$  do
8     while  $d_r(p_1, p_{idx_{end}}) - dl \cdot index \leq 0$  do
9        $idx_{end}++$ 
10    end
11     $index++$ 
12     $S \leftarrow Sample(h(p_{idx_{end}-1}), h(p_{idx_{end}}))$ 
13     $idx_{end}++$ 
14  end
15 return : S
16 end
```

---

### 3.1.2 Parameter settings

While the choice of the number of rings  $n_r$  follows the classic LBP approaches, the values of  $r_{max}$  and  $p$  are set on the basis of the following reasonings:

- $p$  corresponds to the number of samples over each ring. In the case of a circle on a flat surface, it would correspond to the number of sectors that would divide the angle  $2\pi$ . Based on our tests, this parameter should be included between 12 and 18 (for flat surfaces, this would correspond to a uniform sampling with an angle that ranges from  $\frac{\pi}{10}$  to  $\frac{\pi}{6}$ );
- $r_{max}$  represents the radius of the biggest sphere used to define the rings. It can be chosen by the user on the basis of the size of the variations (patterns) to be coded. Nevertheless, we also suggest two possible automatic ways to define  $r_{max}$ , both based on the assumption that a pattern on a surface should be quite small with respect to the global size of the model. Namely:

- $r_{max} = \frac{1}{10} \sqrt{\frac{a}{\pi}}$ . This is a scale-invariant radius based on a fraction of the area of the whole surface, where  $a$  represents the area of the surface model.
- $r_{max} = el\epsilon$ . This is a calibration of the radius based on the average length  $el$  of the tessellation edges and  $\epsilon$  is a constant,  $\epsilon \in [10, 20]$ .

---

**Algorithm 3:** edgeLBP Evaluation

---

**Input** : The array  $S$ , the pivot value  $h(v)$ .

**Output:** The value  $\text{edgeLBP}(v)$ .

```
1 begin
2   for  $idx = 1 : \text{numel}(S)$  do
3     if  $h(S(idx)) < h(v)$  then
4        $str(idx) \leftarrow 0$ 
5     else
6        $str(idx) \leftarrow 1$ 
7     end
8   end
9   return :  $\text{edgeLBP}(v) \leftarrow \sum str(j)\alpha(j)$ 
end
```

---

Notice that, while the second choice is influenced by the tessellation, the first definition is not, since it only depends on the area of the model.

Given the surface tessellation  $T$ , its edgeLBP descriptor  $D_T$  is defined as a feature vector; in particular, the value  $DT(n, m)$  corresponds to the number of vertices that assume edgeLBP value  $m$  on the  $RING_n$ . The size of  $DT$  is equivalent to  $n_r(p + 1)$ . Since in the experiments we are mostly interested in a probability histogram of the distribution of the edgeLBP values, we adopt  $\frac{D_T}{n_v}$  as the edgeLBP descriptor, where by  $n_v$  we mean the cardinality of the set  $V$  of the vertices of  $T$ . Through this normalization of  $T$  we achieve robustness to the number of vertices of the surface representation.

We define the (dis)similarity between two tessellations  $A$  and  $B$  as the distance between their corresponding edgeLBP descriptors  $D_A$  and  $D_B$ . Since the edgeLBP can be thought of as a matrix, any feature vector distance is suitable to evaluate the similarity between two edgeLBP descriptions. In the experiments shown in this paper, we adopt the Bhattacharyya and the  $\chi^2$  distances [DD09], which are widely used in image processing. For the discrete case, the Bhattacharyya distance between two distributions  $\phi$  and  $\psi$  of a scalar random variable  $X$  has the following formulation:

$$d_{Bha}(\phi, \psi) = \sqrt{1 - BC(\phi, \psi)}, \quad BC(\phi, \psi) = \sum_{x \in X} \sqrt{\phi(x)\psi(x)},$$

where  $BC$  is called the *Bhattacharyya coefficient*. We also tested other distances (like the Euclidean distance and the Earth Mover's Distance [DD09]) but the results obtained with the Bhattacharyya and the  $\chi^2$  distances provided the best performances. In most of the experiments, the Bhattacharyya and the  $\chi^2$  distances behave equivalently, when different we specify in the text the distance adopted.

For a set of surface tessellations, the (dis)similarity values are stored in a *distance matrix*  $Dist$ , where  $Dist(i, j) = d_{Bha}(D_i, D_j)$  is the distance between the descriptor of the tessellation  $i$  and  $j$ . The diagonal values of  $Dist(i, i)$  are zero.

### 3.1.3 Computational cost

Given a surface tessellation  $T$  with  $n_v$  vertices, we briefly discuss the computational complexity of the routines involved in the edgeLBP evaluation.

We assume that the tessellation in input is stored in an appropriate data structure, therefore the cost of computing the relations among the elements of the tessellation (e.g., vertex-edge, face-vertex, vertex-face) is constant or  $O(n_v)$ , depending on the relations. Also, the shape properties are precomputed: the curvature estimation proposed in [CSM03] has computational complexity  $O(n_v \log n_v)$ .

If the tessellation  $T$  has a boundary, the creation of the list of the vertices that are admissible for the edgeLBP operator is based on the distances of the vertices from the boundary. This preprocessing phase is done using the kd-tree, which cost is  $O(n_v \log n_v)$ .

For each vertex that is admissible, the computation of the intersection between a sphere and the edges of the tessellation has complexity  $O(n_v)$  (in the worst case), and, therefore, the cost is  $O(n_v^2)$  for the whole surface. It is worth noticing that this cost holds if the sphere intersection processes all the vertices of the tessellation for every vertex. On average, the radius of the sphere is considerably small and diminishes the average computational complexity to  $O(n_v \log n_v)$ . Ring re-sampling has linear complexity with respect to the number of elements of the rings; in the worst case the number of elements in the rings of a vertex  $v$  can become  $O(n_r \cdot n_v)$ , where  $n_r$  is the number of rings (and it is constant). Thus, the re-sampling of all the rings costs at most  $O(n_v^2)$  operations and, in average, the computational complexity is again  $O(n_v \log n_v)$ .

Finally, the computation of edgeLBP histogram is linear in the number of samplings of the tessellations that are  $O(n_r \cdot p \cdot n_v)$ ; since  $n_r$  and  $p$  are two parameters that are constant, the cost is  $O(n_v)$ .

## 3.2 Experimental results

The edgeLBP is able to deal with both geometric and colorimetric patterns. For example, performances are evaluated on the SHREC'17 dataset on relief retrieval [BMTA<sup>+</sup>17], which contains geometric patterns. It is composed of 720 triangle meshes derived from knitted objects, grouped into 15 classes (see Figure 3.5(b)), each one made of 48 textile patterns. Each class has been created from 15 base surfaces (embedding a single textile pattern into 12 different positions);

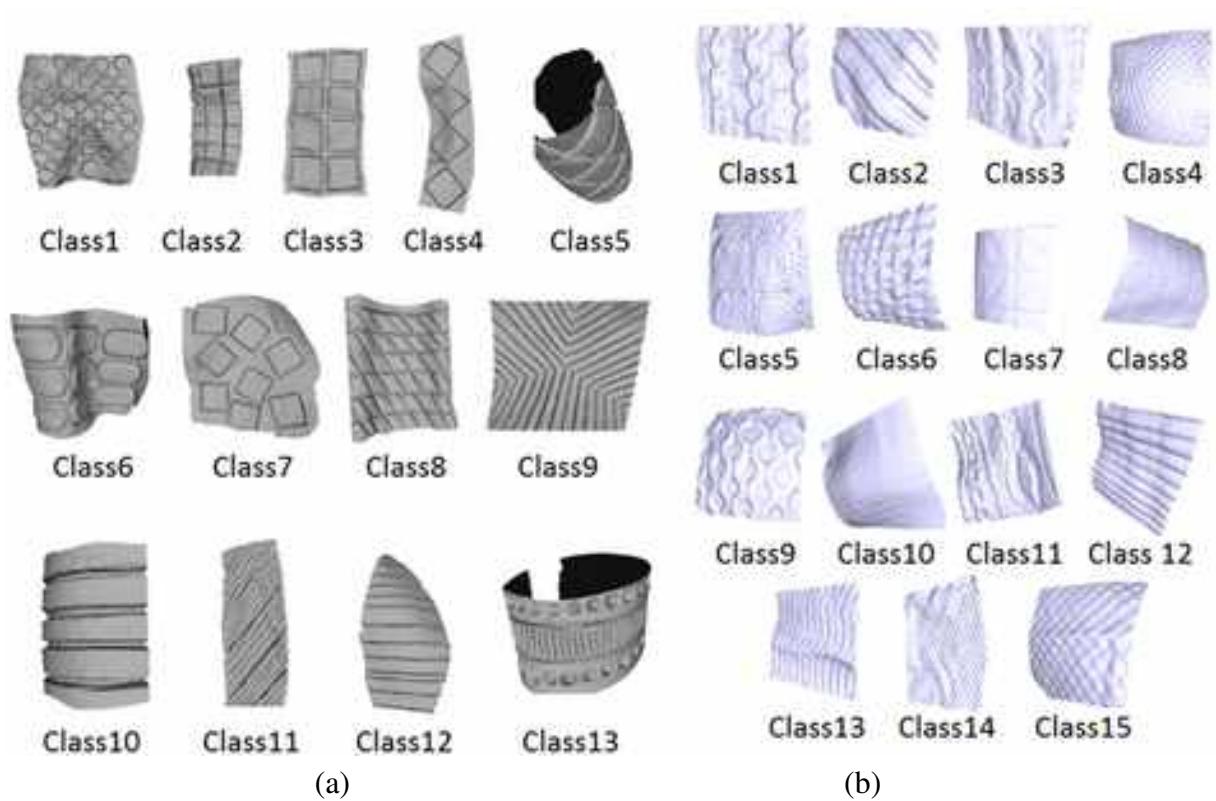


Figure 3.5: (a): the 13 models used to originate the first dataset. (b): the knitted patterns of the SHREC17 contest [BMTA<sup>+</sup>17].

then, each surface was modified with four mesh re-samplings. Again, two datasets can be derived: the first one is related to the complete dataset of 720 models and aims at evaluating the overall robustness and stability of methods with respect to different mesh representations. The second one groups the 180 original meshes according to their textile pattern and it is better suited to analyze the capability of a method of effectively recognizing a pattern independently of the overall surface embeddings.

The evaluation tests have been performed using a number of classical information retrieval measures, namely the Nearest Neighbor, First Tier, Second Tier, Discounted Cumulative Gain, e-measure, Precision-Recall plot, confusion matrices and tier images.

To compare the edgeLBP with the other participants of the SHREC'17 pattern retrieval benchmark, we use the two configuration parameters: *run1* ( $p = 12$  and  $n_r = 7$ ) and *run2* ( $p = 15$  and  $n_r = 5$ ).  $r_{max}$  is set  $10mm$  for both runs and is obtained by measuring the size of the patterns of three randomly selected surfaces. The maximum curvature is used as  $h$  function. Other curvatures (mean and minimum curvature) lead to similar performances. For more details on why we use curvature, please refer to Chapter 2.

Original Dataset							Complete Dataset						
Method	NN	1-Tier	2-Tier	mAP	e	DCG	Method	NN	1-Tier	2-Tier	mAP	e	DCG
LBPI	0.339	0.207	0.353	0.250	0.237	0.250	LBPI	0.828	0.248	0.400	0.283	0.232	0.697
GI HOG	0.089	0.069	0.130	0.118	0.097	0.373	GI HOG	0.686	0.107	0.176	0.131	0.102	0.561
IDAH-2	0.339	0.182	0.271	0.215	0.181	0.503	IDAH-2	0.306	0.141	0.244	0.163	0.127	0.559
CMC-1	0.600	0.342	0.461	0.371	0.274	0.641	CMC-1	0.718	0.258	0.372	0.260	0.247	0.673
CMC-2	0.633	0.363	0.494	0.390	0.293	0.662	CMC-2	0.763	0.272	0.389	0.271	0.261	0.686
CMC-3	0.533	0.281	0.394	0.308	0.242	0.596	CMC-3	0.647	0.219	0.323	0.218	0.208	0.639
SQFD-HKS	0.106	0.066	0.137	0.123	0.102	0.376	SQFD-HKS	0.536	0.117	0.192	0.139	0.110	0.558
KLBO-FV-IWKS	0.522	0.295	0.412	0.307	0.247	0.603	KLBO-FV-IWKS	<b>0.986</b>	0.333	0.449	0.339	0.332	0.759
KLBO-SV-IWKS	0.489	0.249	0.375	0.273	0.235	0.570	KLBO-SV-IWKS	0.978	0.287	0.409	0.296	0.283	0.732
edgeLBP - <i>run1</i>	<b>0.922</b>	0.683	0.825	0.716	0.580	0.863	edgeLBP - <i>run1</i>	0.979	0.619	0.763	0.651	0.413	0.894
edgeLBP - <i>run2</i>	0.911	<b>0.689</b>	<b>0.844</b>	<b>0.725</b>	<b>0.590</b>	<b>0.865</b>	edgeLBP - <i>run2</i>	<b>0.986</b>	<b>0.634</b>	<b>0.780</b>	<b>0.669</b>	<b>0.421</b>	<b>0.902</b>

Table 3.1: Retrieval Performances obtained by the edgeLBP compared with the scores of the best performing methods in [BMTA<sup>+</sup>17] on the SHREC17 datasets. The best runs are in bold.

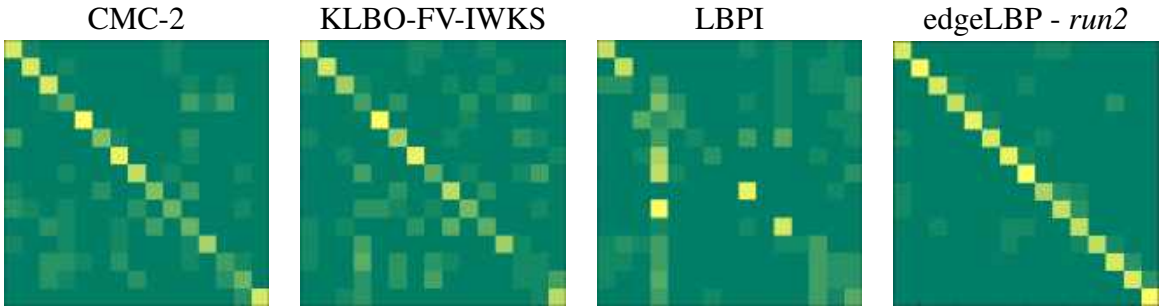


Figure 3.6: Best confusion matrices in the SHREC17 retrieval pattern contest (Original Dataset) in comparison to that of the edgeLBP.

Table 3.1 reports the edgeLBP performances on the Original and Complete datasets and the best runs obtained by the SHREC’17 participants, who are indicated with the same label used in the SHREC’17 report [BMTA<sup>+</sup>17]. In the case of the Original Dataset, our method provides the best results in all the scores, showing a good capability of discriminating the geometric patterns. In the case of the complete dataset, our results significantly overcome the other participants in each measure, only the NN measure is equivalent to the (KLBO-\*) runs. As stated in the SHREC report [BMTA<sup>+</sup>17], the NN value of methods that analyze the global geometry (such as the KLBO-\* run) is biased by the presence of three variations of each mesh in the original dataset. In this case, each mesh sampling keeps the same overall embedding but degrades the mesh connectivity and approximates the original reliefs. For this reason, global methods are made easy to find as the Nearest Neighbor one of the mesh variations; however, they rapidly degrade when the query range increases, like reflected by the FT and ST scores.

Figure 3.6 compares the confusion matrix derived from the best edgeLBP run (*run2*) against



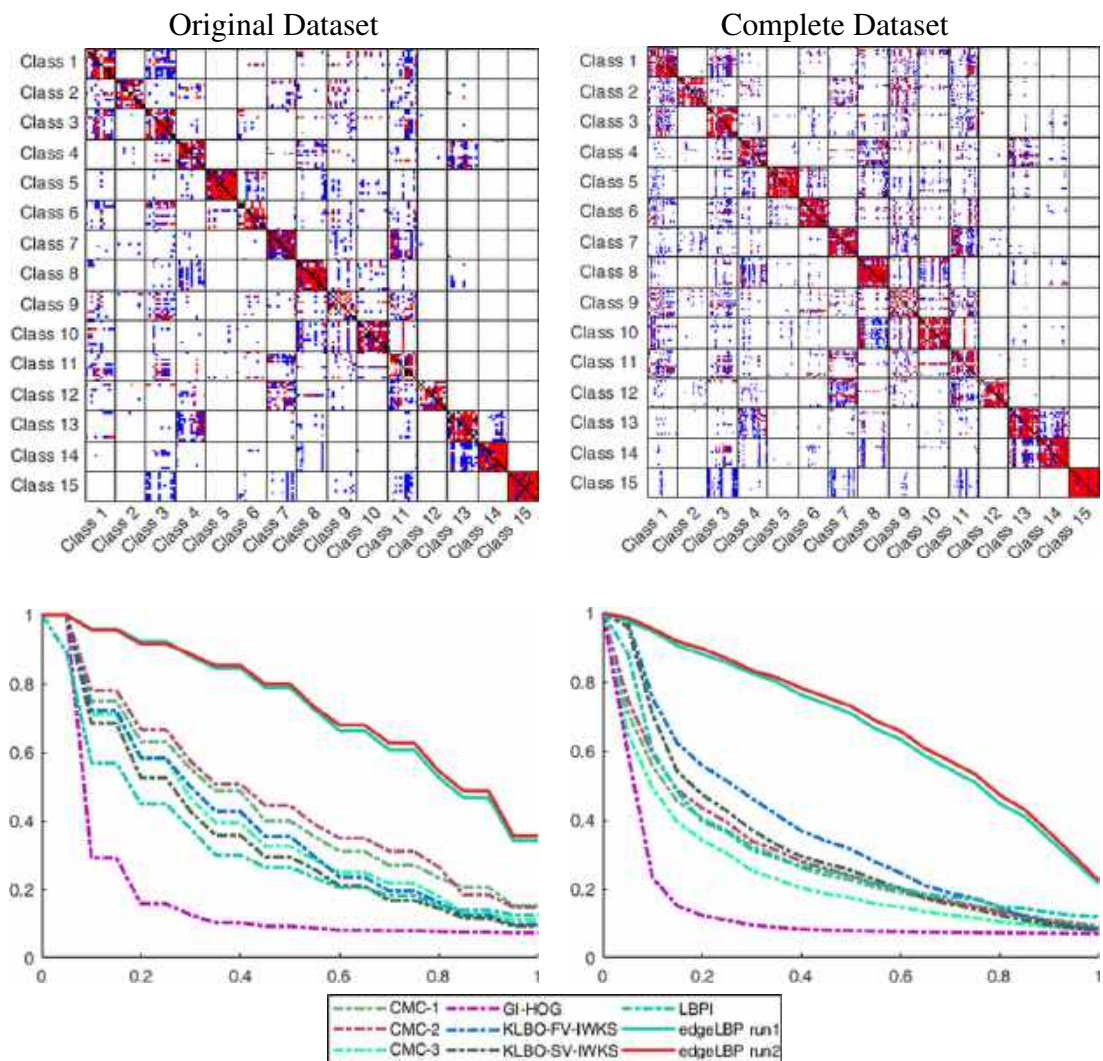


Figure 3.7: Tier image and Precision-Recall plots of the original dataset (left) and complete dataset (right), using the edgeLBP - *run2*. In the tier images, rows represent the queries, the NN is marked in black, the FT in red and the ST in blue.

the three best runs in [BMTA<sup>+</sup>17]. Similarly, Figure 3.7(top) depicts the tier images both on the Original and the Complete datasets of the edgeLBP, *run2*. In the case of the complete dataset, it is worth noticing that adding three variations of the original patches does not alter the overall mask of the tier image, highlighting the coherence of the retrieval performance among the original dataset and its variations.

Finally, Figure 3.7 (bottom) plots the Precision-Recall curves in both the dataset configurations and compares the best edgeLBP performance (*run2*) with the best runs in [BMTA<sup>+</sup>17]. The

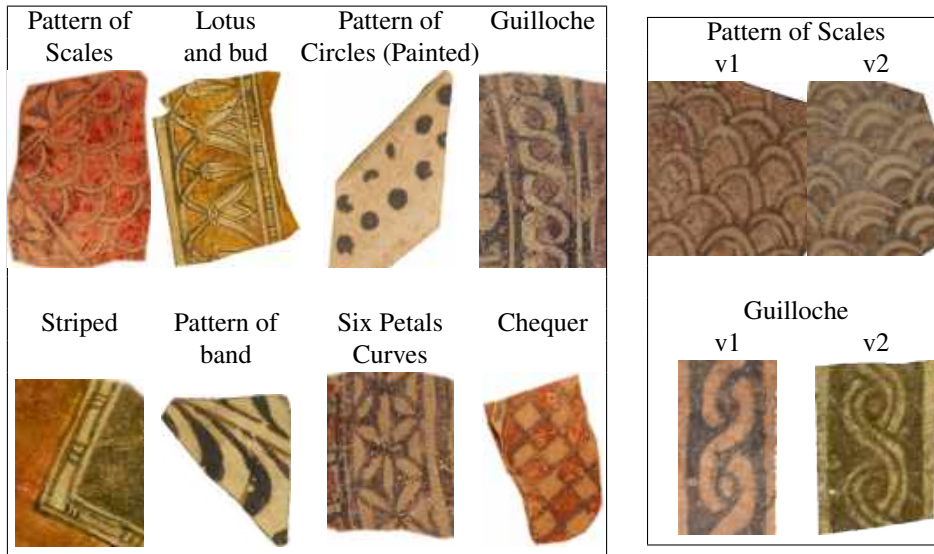


Figure 3.8: Left: The colorimetric patterns in the GRAVITATE dataset used for an evaluation of the edgeLBP description performances. Right: The scale and guilloche classes contain decorations that are semantically the same but differ from the geometric point of view (for instance the number of curved lines, for this reason we subdivided them in two different classes).

overall performance of the edgeLBP successfully deals with the SHREC’17 dataset and generally improves with respect to the other participants of more than 20%; in our opinion this reveals that the method combines the efficacy of a local pattern characterization with a mesh independent and embedding invariant description.

The edgeLBP description of a pattern is also validated on multiple colorimetric pattern datasets. For example, in [MTBS<sup>+</sup>18] we used the edgeLBP to classify pre-segmented fragments from the GRAVITATE [UUTCIC<sup>+</sup>] dataset. We selected the most frequent patterns and created a dataset of colorimetric ones (CP dataset). A list of the classes is represented in Figure 3.8(Left). In particular, two of the classes (labelled *Pattern of Scales* and *Guilloche*) present a large intra-class variation that from the geometric point of view suggests to split them into two sub-classes (see Figure 3.8(Right)). We then consider four classes instead of two (Scale v1, Scale v2, Guilloche v1 and Guilloche v2). From models with colorimetric patterns, we extracted a dataset of 49 patches grouped into 10 classes, where each class contains from 4 to 7 elements.

The classification performance obtained by the edgeLBP is evaluated with respect to the *Nearest Neighbor*, *First Tier* and *Second Tier* per class of pattern. Moreover, we report the *Confusion Matrix* over the pattern classification obtained with the NN classifier.

Multiple settings are used in our tests, with encouraging results. In Figure 3.9 we detail the performances for the best edgeLBP runs. We see that specific patterns like the Guilloche ones are well classified while more complex decorations like the Six petals, the Lotus and Bud and

edgeLBP settings: $n_{rad} = 5, p = 15, r = 0.5\text{cm}$					
Class Label	NN	FT	ST	e	nDCG
Guilloche v1	1.000	1.000	1.000	0.171	1.000
Six Petals	0.500	0.500	0.767	0.270	0.720
Chequer	0.857	0.571	0.738	0.271	0.811
Striped band	0.800	0.200	0.400	0.222	0.544
P.o.C. (Painted)	1.000	1.000	1.000	0.171	1.000
Lotus and Bud	0.500	0.333	0.583	0.171	0.666
Scales v1	0.429	0.310	0.619	0.316	0.579
Scales v2	1.000	1.000	1.000	0.171	1.000
Guilloche v2	1.000	1.000	1.000	0.171	1.000
Pattern of Curves	0.500	0.250	0.250	0.086	0.424
Overall	0.735	0.582	0.723	0.217	0.758

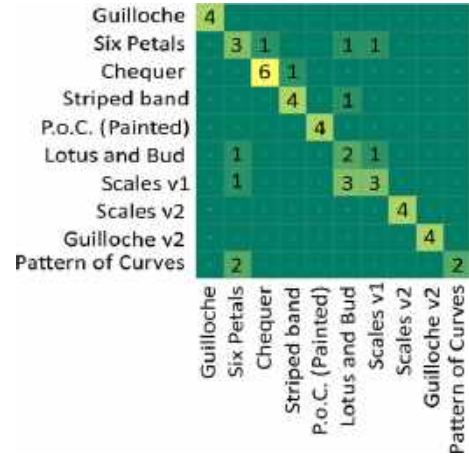


Figure 3.9: Classification performances of the edgeLBP over the colorimetric patterns selected.

the Scales (v1) are often confused. We think that this effect depends on two effects: the non uniform decoration (there are thin and fat lines together) and the fact at the moment we are able to consider only one channel at a time (the L-channel corresponds to the luminosity thus forgetting the other colorimetric information).

More tests on colorimetric patterns with the edgeLBP description are shown in Chapter 5.

## Conclusions

In this chapter we extensively presented the edgeLBP, a novel pattern descriptor that performs very well also on very recent benchmarks. The edgeLBP descriptor is based on local samples on the model mesh and is based on the intersection of the edges of the mesh and spheres of different radii. This induces a solid local sampling scheme that is then described using the Local Binary Pattern. The major downside of the edgeLBP is the computational cost, which is high due to the number of sphere-edge intersections required for each vertex of the mesh.

## Related publications

- E. Moscoso Thompson, S. Biasotti, *Description and retrieval of geometric patterns on surface meshes using an edge-based LBP approach*. Pattern Recognition, 2018.
- E. Moscoso Thompson, S. Biasotti, *Edge-based LBP Description of Surfaces with Colorimetric Patterns*. Eurographics Workshop on 3D Object Retrieval, 2018.
- E. Moscoso Thompson and S. Biasotti, *Retrieving color patterns on surface meshes using edgeLBP descriptors*. Computers & Graphics, 2019.
- E. Moscoso Thompson, S. Biasotti, G. Sorrentino, M. Polig, S. Hermon, *Towards an Automatic 3D Patterns Classification: the GRAVITATE Use Case*. Eurographics Workshop on Graphics and Cultural Heritage, 2018.

## Chapter 4

# Mean Point Local Binary Pattern

A further extension of the LBP description to surfaces is introduced, which is able to deal with any surface described as a set of points. This description is designed to deal with point clouds from, for instance, laser scans of archeological fragments. However, if the surface is given as a tessellation, this set of points can be the set of vertices, supplemented by additional points sampled on the faces if the number of vertices is low. This implies that this method is able to work with both point clouds and tessellations. The points are organized in a kd-tree structure that permits an efficient search of the neighbors [FBF77] to extract concentric rings needed by the LPB descriptor. Rings are adaptively sampled so that an "equal sector" area is preserved along the neighbour rings without changing the width of the rings. The experiments show how the new descriptor, named *mean point Local Binary Pattern (mpLBP)* for short, considerably reduces the computational cost with respect to its direct competitor, the *edgeLBP*, while preserving competitive retrieval and classification performances.

The robustness of the mpLBP descriptor is further challenged, considering different surface bendings, presenting the mpLBP performance on models obtained from scans of archaeological fragments and by analyzing the efficiency of the descriptor and its characteristics when different neighborhood shapes and sampling rules are chosen.

### 4.1 Method description

This section is devoted to the description and setup of the mpLBP. The following sections explore the method pipeline in details, as well as the parameter settings and the experimental results.

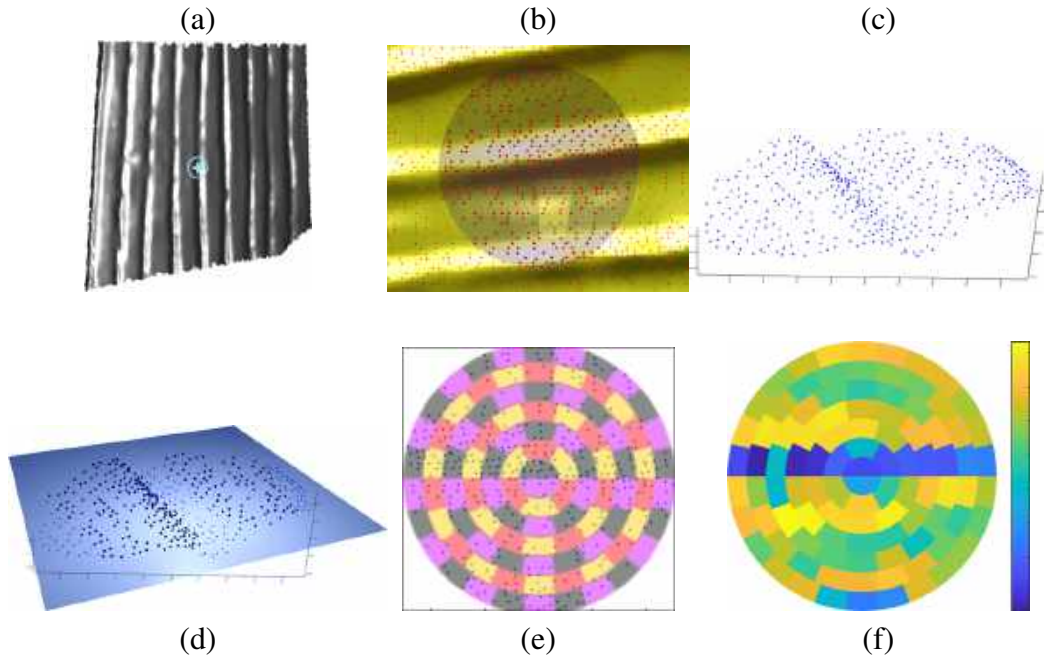


Figure 4.1: A mpLBP descriptor at point  $\tilde{v}$  (marked with a light-blue star in (a)). (b): neighborhood  $S[\tilde{v}]$  of  $\tilde{v}$  is shown with a dark sphere. (c): point density in  $S[\tilde{v}]$ . (d): regression plane  $\pi$ , (e): clustering into sectors. (f): resulting punctual descriptor, represented as a ‘circular’ feature vector.

### 4.1.1 The mpLBP descriptor

This section introduces a statistical descriptor that aggregates values of the local descriptors computed at a set of positions on the surface. The input surface model can be a point cloud or a tessellation (in the last case, the vertices are input points for the descriptor).

Similarly to the edgeLBP, the mpLBP procedure can be described by two main steps: the creation of the punctual descriptor and the LBP evaluations that are further combined to create the mpLBP descriptor.

#### 4.1.1.1 mpLBP punctual descriptor

Let  $S$  be a point set embedded in the 3D Euclidean space and a surface property defined on  $S$ ,  $h : S \rightarrow \mathbb{R}$ , a function defined on  $S$  whose values depends on the pattern we want to describe (e.g.: curvature-based values in case of geometric patterns, a color-based property in case of depicted decorations, etc.). Let us consider the point  $\tilde{v} \in S$  and the set  $S[\tilde{v}]$  of the points  $v_i \in S$  at a distance from  $\tilde{v}$  at most equal to  $r$ , i.e.,  $S[\tilde{v}] = \{v_i \in S | d(\tilde{v}, v_i) \leq r\}$ . We will discuss the

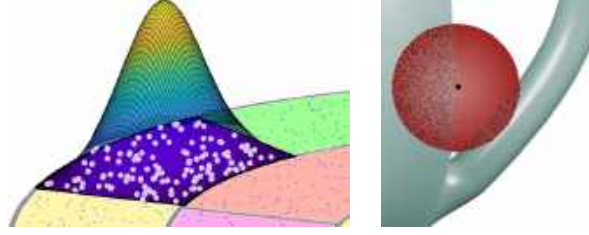


Figure 4.2: Left: illustration of the Gaussian filter adopted to weight the points (in white) in a given sector (in purple). The colors of the Gaussian range from blue (0) to yellow (1). Right: example of a neighborhood that could occur if the radius  $r$  is larger than  $\tilde{r}_{max}$ : points are samples on two disconnected parts.

choice of the radius  $r$  later in this section. Gathering the sets  $S[\tilde{v}]$  means visiting the points of  $S$  several times. Since 3D models with patterns need to be at high resolution (thus described by a high number of points), the way the distance relations between points are computed must be efficient. In our implementation, these relations are computed using a kd-tree. This structure is computed once per model (with a computational cost of  $n \log(n)$ ).

Points in  $S[\tilde{v}]$  are projected on a plane  $\pi$ , obtained using linear regression on  $S[\tilde{v}]$ . When the density is high enough and if the radius is chosen appropriately, that plane may be interpreted as an approximation of the tangent plane. The projected points are sorted in  $n_r$  concentric rings based on their distances from  $\tilde{v}$ .

The number of rings is given by the parameter  $n_r$ , that we call *radial resolution*. Each ring is defined, for  $j = 1 \dots n_r$ , as follows:

$$S[\tilde{v}]_j = \{v_i \in S[\tilde{v}] | d(\tilde{v}, v_i) \in [r_{j-1}, r_j]\}, \quad r_j = j \frac{r}{n_r}$$

Each  $S[\tilde{v}]_j$  is divided in  $p_j$  sectors, delimited by some regularly spaced angle values  $\theta_k$ . Note that  $p_j$  may vary along the rings, in order to obtain sectors with similar areas. We call  $p_j$  the *spatial resolution*. More formally, we define the sector  $k$  of the ring  $j$  (sector  $(j, k)$  for short) of the point  $\tilde{v}$  as:

$$S[\tilde{v}]_j^k = \{v_i \in S[\tilde{v}] | d(\tilde{v}, v_i) \in (r_{j-1}, r_j], \theta_i \in (\theta_{k-1}, \theta_k)\},$$

where  $\theta_k = k \frac{2\pi}{p_j}$ ,  $k = 1, \dots, p_j$ . Finally, we assign to each sector  $(j, k)$  a value  $sec(\tilde{v})_j^k$  as the representative of the function  $h$  in that sector. Figure 4.1 represents the pipeline to build the punctual descriptor. Note that the punctual descriptor can be seen as a feature vector by simply stacking the values of the descriptor on each ring.

As it usually happens in the LBP implementations, we excluded the computation of the punctual descriptor at points that are close to the boundary of the model (if any). If the boundary of the model is known, it is enough to consider only the points that are at least at distance  $r$  from the

boundary. In addition, if a point punctual descriptor has more than  $\frac{1}{4} \sum_j p_j$  empty sectors, we consider it *invalid* and discard that point. When the intersection of the sphere of radius  $r$  with the point cloud generates multiple surface components like those in Figure 4.2 (Right), we consider such a configuration non acceptable and refine the point neighborhood by selecting a smaller value for  $r$ . Indeed, for a given model  $M$  we assume that the projection onto  $\pi$  is injective and that the surface locally captured by the sphere is locally homeomorphic to a topological disk. Moreover, we assume the existence of a radius  $\tilde{r}_{max}$ , which is the maximum value for the parameter  $r$  such that all the points on  $M$  have an acceptable description.

#### 4.1.1.2 Local Binary Pattern evaluation

The mpLBP punctual descriptor is the point neighborhood representation to which we apply the LBP encoding technique. If the radius  $r$  is small enough with respect to the curvature and the thickness of the object, we can suppose that the rings of the punctual descriptor are locally close to concentric rings using geodesic distance to  $\tilde{v}$ . Thus, each sector can be seen as the evaluation of  $h$  at a sample of the surface. For all the points  $\tilde{v}$  in  $S$ , we define  $LBP(\tilde{v})$  the feature vector of  $n_r$  elements as follows:

$$LBP(\tilde{v})_j = \sum_k (str[\tilde{v}]_j)_k,$$

$$(str[\tilde{v}]_j)_k = \begin{cases} 0 & \text{if } sec(\tilde{v})_j^k < h(\tilde{v}) \\ 1 & \text{otherwise} \end{cases}$$

Then, the mpLBP descriptor of  $S$  ( $mpLBP(S)$ ) is the histogram of the LBP values of the points of  $S$ . As a final step, the mpLBP is normalized, i.e., all the entries of  $mpLBP(S)$  are divided by the number of points considered in the histogram, enhancing the stability of the descriptor.

The  $mpLBP(S)$  is a  $\sum_j (p_j + 1)$  sized feature vector. Intuitively, we can visualize it as a horizontal concatenation of the rings of the multiple feature vectors in Figure 4.1(f). In particular, the  $j$ -th ring generates a feature vector of  $p_j + 1$  entries, where  $mpLBP(S)_{(j,m)}$  is equal to the number of points  $\tilde{v}$  in  $S$  such that  $LBP(\tilde{v})_j = m$  (with  $j = 1, \dots, n_r$  and  $m = 0, \dots, p_j$ ).

It is worth mentioning that if the neighborhood of a point is rotated significantly around the normal of the point, its punctual descriptor changes. On the contrary, if the rotation is small, the punctual descriptor is stable; indeed, the Gaussian filter adopted to weight the points is stable under rotations smaller than a fraction of the angular sector. In other words, if the grid of sectors (Figure 4.1(e)) is slightly rotated, the punctual descriptor does not vary significantly. Moreover, we recall that the LBP value per ring (as intended in this chapter) is rotation invariant (because it is a sum of 0 and 1 values on the whole ring), we can conclude that the pattern descriptor of the mpLBP is robust to rotations of the surface. This fact has been verified by applying a random rotation to each point neighborhood: the results indicate an almost perfect stability in this sense.



## 4.1.2 Parameter settings

The three parameters of the mpLBP are: the radius  $r$  (used to set the neighborhood size around each point of  $S$ ), the radial resolution  $n_r$  and the spatial resolution  $p_j$ . This is similar to the parameter set of the edgeLBP as described in Chapter 3, the main difference being that for the edgeLBP the parameter  $P$  is fixed across all the rings. In the following, we present some hints on how these parameters should be tuned. The intuition suggests that the mpLBP ability of detecting a pattern depends on the size of the neighborhood of each point, i.e, the size of the disk must be related to the pattern size. Moreover, the denser the ring sampling, the more complete information is stored, at the cost of a larger storage size.

- $r$ : neighborhood radius shown as a dark bubble in Figure 4.1(b).  $r$  should be set so that neighborhoods contain at least one part of the pattern that we want to describe (e.g.: if the pattern is defined by chiseled circles, the bubble should contain at least one circle entirely).
- $n_r$ : it defines the radial resolution and should be fixed together with  $p_j$  (see below).
- $p_j$ : it represents the spatial resolution and varies over the different rings.

**Choice of the rings and sampling scheme** The mpLBP description adopts as the point description a set of circular rings such that the spatial resolutions  $p_j$  guarantee that all the sectors have the same area. For this reason, we selected  $p_j = multP(2j - 1)$ ,  $multP \in \mathbb{N}_+$ . In this case,  $p_j$  depends on  $n_r$ . This degree of freedom was tuned by the parameter  $multP$  (that replaces the  $p_j$  parameters). For instance, in Figure 4.1(c) the parameters are  $n_r = 7$  and  $multP = 2$ , which means that  $S[\tilde{v}]$  has 7 rings, where  $S[\tilde{v}]_1$  has 2 sectors,  $S[\tilde{v}]_2$  has 6 sectors and  $S[\tilde{v}]_3$  has 10 sectors, etc.

However, similarly to the LBP for images, different types of rings may be used. Both the shape of the ring (square, elliptical...) and the sampling scheme (or rather considering only part of the neighborhood) may be changed in order to better suit a given dataset. In general, not symmetrical sampling schemes are a valid option in this context. Indeed, the punctual descriptor converts a patch of a surface into a sort of image. Therefore, it would make sense to straightforwardly adopt a square neighborhood and its variations, following the image literature on pattern recognition. In general, it is also possible to define various scheme variations of the standard punctual descriptor. We focus on different sampling areas of the point neighborhood. In the following, we depict some possible neighborhood and sampling strategies.

- *Scheme 1*: same concept of the mpLBP punctual descriptor, but both the descriptor and the sectors are shaped as squares. Figure 4.3(a) shows how the square neighborhood of the points are divided in sectors. The parameters of this descriptor are half the diagonal of the square (a sort of radius, thus we still refer to it as  $r$ ) and the square root of the number of sectors (or rather, the number of sectors along the sides of the square) labelled with  $pxres$ .

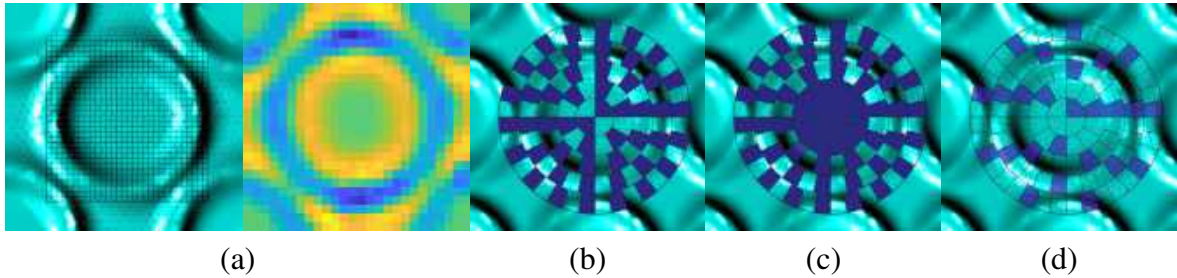


Figure 4.3: Various sampling schemes. In (a) we show the grid used to create the square punctual descriptor (Left) and the final punctual descriptor (Right). Color goes from blue to yellow, representing the value of the sector. In (b,c,d) we highlight (in blue) the sectors that are considered in Schemes 2,3, and 4.

- *Scheme 2*: same as the mpLBP punctual descriptor, but only one every two sectors is kept on each ring (e.g., sectors 1, 3, etc.). In our tests we selected only odd sectors; anyway also even indices could be equally considered. In practice, adopting such a strategy we half the number of samplings.
- *Scheme 3*: similar to Scheme 1, but the first rings are kept in their entirety. The implicit assumption behind this scheme is that smaller rings need a denser sampling.
- *Scheme 4*: same as the mpLBP punctual descriptor, only the sectors with an index  $j$  such that  $j = 1 + 4n$  with  $n \in \mathbb{N}$ , are considered. Similarly, to Scheme 1, this strategy aims at decreasing the number of samples.

A representation of these variations are shown in Figure 4.3.

## 4.2 Experimental results

The mpLBP is validated on the same benchmarks the edgeLBP was validated on, for a better comparison between the two methods. Further stability tests are reported, to confirm the stability of the descriptor despite its simpler definition with respect to the edgeLBP. After looking at the performances of the different sampling schemes on the SHREC'17 benchmark, the computational times of the two methods on the same model are compared.

**Performances on the SHREC'17 benchmark** In addition to the methods of [BMTA<sup>+</sup>17] that obtained the best performances, we compare the mpLBP with the edgeLBP and the SIFT-based method in [Gia18]. Among all the settings tested, the best performing ones are  $r = 14$ ,  $n_r = 7$  and  $multP = 4$ . Since most of the models in the dataset have low resolution, we resampled them

Original Dataset						
Method	NN	FT	ST	mAP	e	nDCG
CMC-2	0.633	0.363	0.494	0.390	0.293	0.662
KLBO-FV-IWKS	0.522	0.295	0.412	0.307	0.247	0.603
edgeLBP - <i>run2</i>	0.911	0.689	0.844	0.725	0.590	0.865
T/mC/SIFT/FV	0.872	0.710	0.849	0.741	0.457	0.883
mpLBP - Cmax	0.933	0.706	0.845	0.744	0.449	0.871
mpLBP - Cmin	0.922	0.732	<b>0.861</b>	0.738	0.442	0.862
mpLBP - Cmean	0.911	<b>0.733</b>	<b>0.861</b>	<b>0.763</b>	0.447	<b>0.888</b>
mpLBP - Cmean*	0.917	0.733	0.863	0.761	0.427	0.875
mpLBP - Cmean*	0.922	0.728	0.862	0.763	0.426	0.877
mpLBP - Cgauss	<b>0.939</b>	0.707	0.845	0.744	<b>0.483</b>	0.872
mpLBP - SI	0.911	0.729	0.835	0.749	0.440	0.876
mpLBP - HF	0.672	0.353	0.451	0.431	0.281	0.658

Original Dataset						
Method	NN	FT	ST	mAP	e	nDCG
CMC-2	0.763	0.272	0.389	0.271	0.261	0.686
KLBO-FV-IWKS	0.986	0.333	0.449	0.339	0.332	0.759
edgeLBP - <i>run2</i>	0.986	0.634	0.780	0.669	0.421	0.902
T/mC/SIFT/FV	0.993	<b>0.712</b>	<b>0.850</b>	0.739	0.647	0.929
mpLBP - Cmax	0.994	0.678	0.820	0.738	0.653	0.931
mpLBP - Cmin	0.997	0.677	0.815	0.733	0.653	0.931
mpLBP - Cmean	0.994	0.702	0.841	<b>0.759</b>	<b>0.665</b>	<b>0.938</b>
mpLBP - Cgauss	0.994	0.678	0.820	0.738	0.653	0.931
mpLBP - SI	0.997	0.688	0.813	0.737	0.650	0.930
mpLBP - HF	<b>0.999</b>	0.383	0.480	0.431	0.419	0.802

Table 4.1: Results on the SHREC’17 benchmark, both the Original (Top) and the Complete (Bottom) Dataset. We mark with \* the runs to which we added to each point neighborhood a small rotation of the tangent parametrization around the point normal (from 0 to  $2\pi$ ).

to 40000 vertices using the Remesh tool [AF06]. Table 4.1 reports the mpLBP scores together and compares them with the other methods, with respect to NN, FT, ST, e-measure, mAP and nDCG. In this table we also include the results obtained when testing the independence from the cut of the grid on the neighborhood of the points. The results show that adding this rotation does not influence critically the results of our method.

The mpLBP scores equivalently or slightly better than the edgeLBP and T/mC/SIFT/FV over the SHREC’17 benchmark on geometric patterns, with small variations depending on the surface property chosen. Overall, over this benchmark, the mpLBP performs well with all the curvature-based properties; in particular, the mean curvature provides slightly better retrieval performances. However, the NN performance over the complete dataset of mpLBP with the height field highlights how this property is able to characterize a model and its re-samplings but it is less robust to different surface bendings.

**Performances on the SHREC’18 benchmark** The performance of mpLBP on this benchmark is compared against those obtained in [MTW<sup>+</sup>18] and [MTB19]. The parameters settings with the best evaluations are  $r = 0.10$ ,  $n_r = 7$ ,  $multP = 1$  (*set1*) and  $r = 0.14$ ,  $n_r = 7$ ,  $multP = 1$  (*set2*). Table 4.2 summarizes the best scores obtained.

Over this benchmark, mpLBP and edgeLBP perform equivalently, even if the time for evaluating the edgeLBP on this dataset is approximately twenty times higher than the mpLBP (details on the computation costs are provided in Section 4.2.2). Indeed, to guarantee the decorations were intelligible, the original surfaces were densely sampled (100K vertices, each) and this corresponds to a demanding task for the edgeLBP. On the contrary, this is a good basis for mpLBP because

Single Pattern Dataset						
Run	NN	FT	ST	mAP	e	nDCG
TWB3	0.755	0.502	0.688	0.577	0.455	0.795
V2	0.82	0.51	0.731	0.593	0.481	0.808
edgeLBP-R4	0.915	0.717	0.879	0.766	0.60	0.898
edgeLBP-R5	0.950	0.740	<b>0.892</b>	<b>0.790</b>	<b>0.606</b>	<b>0.911</b>
mpLBP - <i>set1</i>	<b>0.965</b>	0.739	0.862	0.781	0.600	0.910
mpLBP - <i>set2</i>	0.960	<b>0.744</b>	0.864	0.762	0.590	0.900

Complete Dataset						
Run	NN	FT	ST	maP	e	nDCG
TWB3	0.593	0.417	0.564	0.460	0.376	0.711
V2	0.79	0.433	0.594	0.493	0.39	0.753
edgeLBP-R4	0.903	0.673	0.827	0.722	<b>0.557</b>	<b>0.878</b>
edgeLBP-R5	<b>0.923</b>	0.667	0.805	<b>0.727</b>	0.546	<b>0.878</b>
mpLBP - <i>set1</i>	0.903	<b>0.739</b>	<b>0.862</b>	0.668	0.520	0.850
mpLBP - <i>set2</i>	0.907	0.573	0.735	0.639	0.510	0.840

Table 4.2: Performance scores over the Single pattern dataset and Complete dataset of the SHREC'18 benchmark.

the key issue for its success is that the point cloud is dense enough, i.e., most of the sectors of the descriptors should not be empty.

**Performances on the GRAVITATE dataset** We run the mpLBP on both datasets with 4 different settings. The results are compared with those of the edgeLBP on the GRAVITATE datasets. In particular, on GRAVITATE(geo) we used the Shape Index as the  $h$  function. In Table 4.3, the best mpLBP run we had is compared to the edgeLBP results.

As a general premise, since the dataset classes contain few models (fewer than 10), even a variation of 0.1 in the scores means that only one model is miss-classified. The performances of the edgeLBP and mpLBP are comparable, with differences that depend on the type of pattern and the quality of the model. This can be observed by the overall performances of the runs. Looking at the single class performances, the NN measures are almost identical, with the edgeLBP being slightly more efficient on the geometric patterns. The opposite is true on the colorimetric ones, where the patterns are more degraded. This may be explained by the fact that since the mpLBP scouts an entire area for each sampling (sector) rather than a single point to code the evolution of the ring properties, it slightly averages the local surface properties simulating a slight smoothing effect. This can be positive in case of noisy colorimetric patterns but it could lead to a confusion between geometric noise and actual small surface variations.

edgeLBP settings: $n_r = 5, P = 15, r = 0.5\text{cm}$						mpLBP settings: $n_r = 7, P = 15, r = 0.5\text{cm}$					
Class Label	NN	FT	ST	e	nDCG	Class Label	NN	FT	ST	e	nDCG
Guilloche v1	1.000	1.000	1.000	0.171	1.000	Guilloche v1	1.000	1.000	1.000	0.171	1.000
Six Petals	0.500	0.500	0.767	0.270	0.720	Six Petals	0.500	0.400	0.767	0.270	0.709
Chequer	0.857	0.571	0.738	0.271	0.811	Chequer	1.000	0.714	0.976	0.316	0.924
Striped band	0.800	0.200	0.400	0.222	0.544	Striped band	0.600	0.250	0.500	0.189	0.567
P.o.C. (Painted)	1.000	1.000	1.000	0.171	1.000	P.o.C. (Painted)	1.000	1.000	1.000	0.171	1.000
Lotus and Bud	0.500	0.333	0.583	0.171	0.666	Lotus and Bud	0.500	0.417	0.583	0.143	0.566
Scales v1	0.429	0.310	0.619	0.316	0.579	Scales v1	0.571	0.357	0.643	0.263	0.680
Scales v2	1.000	1.000	1.000	0.171	1.000	Scales v2	0.750	0.333	0.667	0.171	0.673
Guilloche v2	1.000	1.000	1.000	0.171	1.000	Guilloche v2	1.000	1.000	1.000	0.171	1.000
Pattern of Curves	0.500	0.250	0.250	0.086	0.424	Pattern of Curves	0.500	0.167	0.250	0.100	0.427
Overall	0.735	0.582	0.723	0.217	0.758	Overall	0.742	0.693	0.918	0.211	0.755

Table 4.3: Results on the GRAVITATE datasets of colorimetric patterns, and comparison with those of the edgeLBP.

## Robustness of the descriptor

We analyzed the robustness of the pattern descriptor of a fixed pattern when the latter lies on surfaces with different bendings.

**Robustness to noise** The popularity of scanning devices that can digitize objects increased the number of acquired 3D models available. However, acquired data can be corrupted by acquisition noise. Furthermore, objects can be degraded by time or other factors, which can lead to corruption of the patterns that lie on the surface of the object.

Noise in general is an unwanted variation of a signal (the surface, in this case) usually of high frequency. Patterns also are usually small, thus the noise is a problem that is worth addressing. We added noise with various amplitudes to some of our test models. Such variations change the  $h$  function and as a consequence the whole punctual descriptor. Depending on the pattern nature, we considered different noise addition. The geometrical patterns are corrupted with a Gaussian noise on the vertices, based on a parameter  $\lambda_g$ , expressed as a percentage of the diameter of the smallest bounding sphere. The values of  $\lambda_g$  considered are 0.2 and 0.4. See Figure 4.4 (Left) for an example of mesh degradation.

Colorimetric patterns are instead corrupted by adding small variations to the RGB values stored on the model vertices. Such variation is bound to the parameter  $\lambda_c$ , an integer value added to each RGB channel (we assume the three channels to range from 0 to 255). For example,  $\lambda_c = 5$  added three random offsets in the interval  $[-5, +5]$  to each color channel. In our tests, we used  $\lambda_c \in \{5, 7\}$  (see Figure 4.4 (Right)).

The noise tests are run on the SHREC'17 Original dataset for the geometric patterns and on the

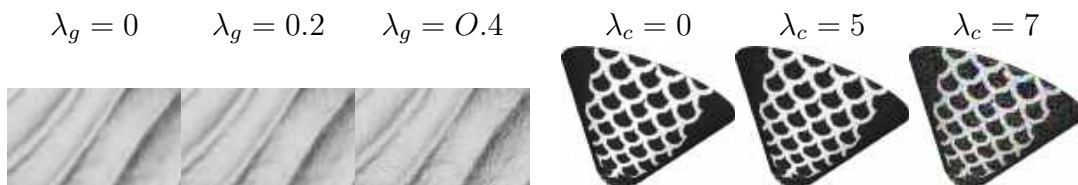


Figure 4.4: Pattern distortion when noise is randomly added. Left: a geometric pattern is corrupted using increasing Gaussian noise. Right: an increasing random noise is added to each RGB color channel.

SHREC'17: Original Dataset, geometric noise						
Method	NN	FT	ST	mAP	e	nDCG
mpLBP - <i>set1</i> Clean	0.917	0.711	0.859	0.743	0.420	0.861
mpLBP - <i>set1</i> , $\lambda_g = 0.2$	0.911	0.693	0.846	0.733	0.380	0.790
mpLBP - <i>set1</i> , $\lambda_g = 0.4$	0.872	0.618	0.769	0.664	0.350	0.753

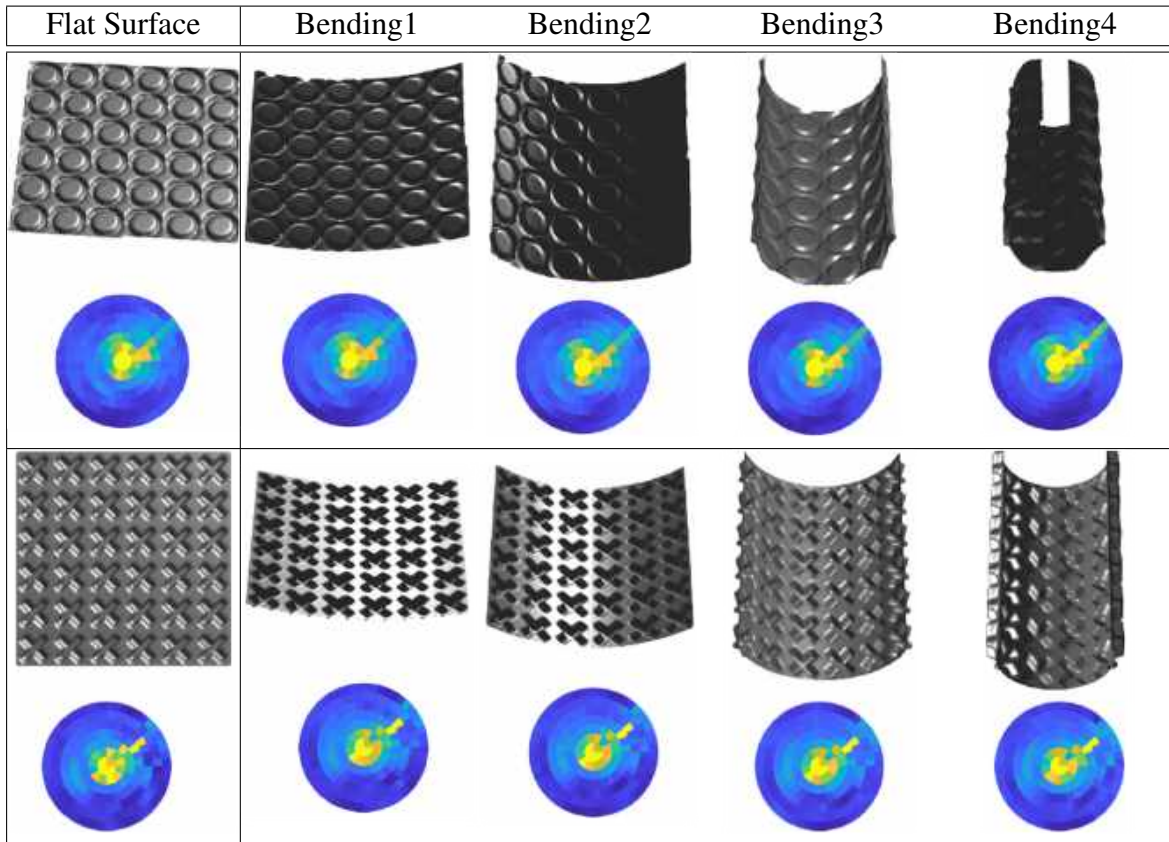
  

SHREC'18: Single Pattern Dataset, colorimetric noise						
Method	NN	FT	ST	mAP	e	nDCG
mpLBP - <i>set1</i> Clean	0.965	0.739	0.862	0.781	0.600	0.910
mpLBP - <i>set1</i> , $\lambda_c = 5$	0.915	0.514	0.653	0.586	0.440	0.822
mpLBP - <i>set1</i> , $\lambda_c = 7$	0.75	0.332	0.445	0.457	0.355	0.741

Table 4.4: mpLBP performance for data corrupted with noise. Top: the Original Dataset of the SHREC'17 benchmark, Bottom: Single Pattern Dataset of the SHREC'18 benchmark.

SHREC'18 Single pattern dataset for the colorimetric one. Results are reported in Table 4.4. We observe that the performances significantly decrease in presence of heavy noise, while the mpLBP is robust in case of lighter one. We think that this behaviour derives from the strategy we adopt to evaluate a property in the sectors. Indeed, the use of the weighted mean for each sector balances the small variations of the  $h$  function, while it starts being less efficient in case of higher variations (in this case, all the sectors become similar).

**Robustness to surface bendings** A very relevant feature of a pattern descriptor on surfaces is its robustness to different bendings of the underlying surface. To test this, we created a pattern of circlets and applied it to surfaces with more and more severe bendings. These models are reported in Figure 4.5 (First row). Since they are generated from a synthetic surface and adopting an isotropic bending we are guaranteed that the models possess the same pattern. In the second row of Figure 4.5, the corresponding descriptors are shown. A similar test is done using a pattern made up by small crosses (see Figure 4.5 - Third and Fourth rows). From the results, one can observe that the pattern descriptor changes depending on the embedded pattern. Moreover, it



$L^2$ norm of each pattern descriptor of the bent surfaces from the same pattern embedded into a flat one				
Flat Surface	Bending 1	Bending2	Beinding3	Bending4
Circles	0.237	0.032	0.058	0.0794
Crosses	0.089	0.084	0.100	0.105

Figure 4.5: Examples of two patterns embedded on surfaces with gradually stronger bendings. Colors of the descriptor go from blue to yellow. The table reports the distances in  $L^2$  norm between the pattern descriptor of the pattern embedded on the flat surface and on the bent surfaces. The mpLBP is stable to different bendings.

remains quite close, although not perfectly identical, for the same pattern across the bending changes. This shows that the descriptor is more sensible to changes in the patterns it describes and far less sensitive to the surface bending.

**Robustness to different model samplings** The models with a low resolution were re-sampled, increasing the vertex density to reach a reasonably dense representation. Therefore, it is worth exploring the behaviour of the pattern descriptor when it is computed for different sampling

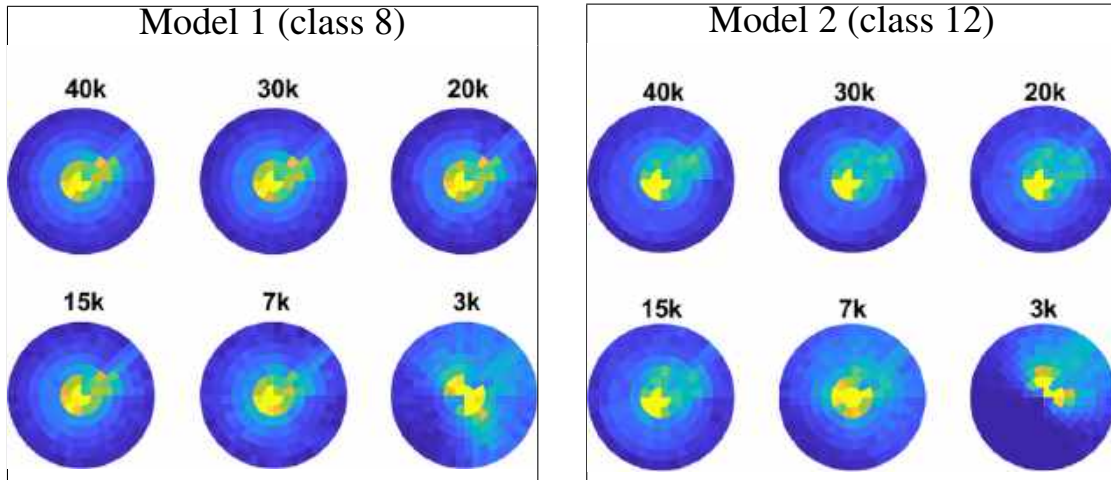


Figure 4.6: Pattern descriptor robustness to different samplings. The number of vertices of the model is on top of the respective pattern descriptor.

density of the same model. We used the SHREC17 Original Dataset for this experiment. We selected two of the models of this dataset (one from class 8 and one from class 12) and down-sampled it from 40000 vertices to 3000 vertices, with various steps, using the default sampling scheme implemented in [AF06]. Figure 4.6 shows the pattern descriptor computed. It is easy to notice that the pattern descriptor holds its shape and changes when the number of vertices considerably decreases (although in the examples 7000 is still sufficiently stable with respect to the higher samplings we noticed that around 10000 vertices is a good quality compromise). This test shows that it is possible to have similar performances when re-sampling with different vertex resolutions, but it is necessary to have a minimum vertex density in order to have more stable pattern descriptors (in this case, approximately 10000 vertices or more).

**Robustness to different choices of the parameters** The choice of the mpLBP parameters ( $r$ ,  $n_r$ , and  $multP$ ) is crucial for the performance of the method because they are closely related at the resolution a pattern is analysed. Anyway, we noticed that the performance of the mpLBP is quite stable for small variations of the parameters. In other words, slightly changing the parameters (all three of them) will not jeopardise the performances of the method. This fact was experimentally confirmed by selecting 27 variations of the best mpLBP run over the SHREC17 original dataset (see Table 4.1). The performance in terms of NN, FT and ST scores of the mpLBP are reported in Figure 4.7 for all these 27 settings. On the horizontal axis we report the parameter setting, while the vertical axis represents the performance score (different colours are used for the NN, FT and ST, respectively). The performances are very similar for all the settings. The maximum discrepancy observed across all the evaluation measures is around 0.05, which is very tiny, especially considering that it occurs only once in 27 runs.



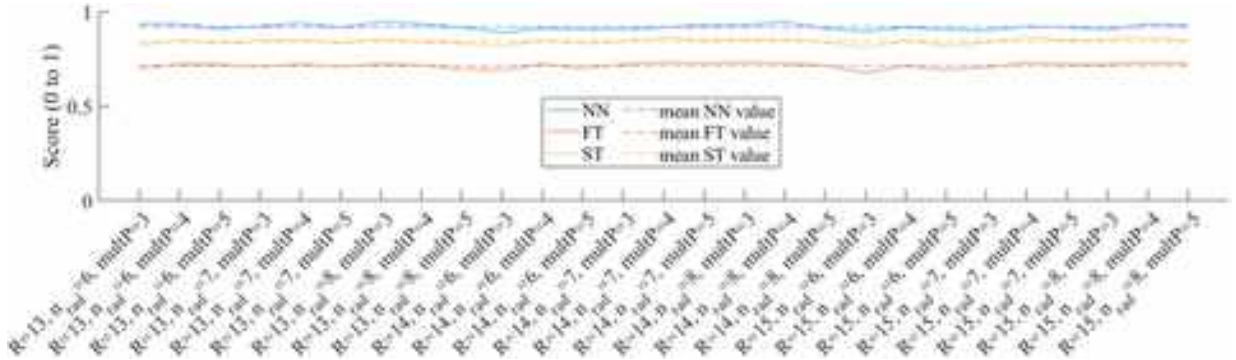


Figure 4.7: The performances of 27 different mpLBP runs on the SHREC17 dataset, with different parameter settings. The dashed lines represent the mean value of the respective evaluation measure.

#### 4.2.1 Different choices of the ring sampling scheme

We evaluated four variants of the point neighborhood sampling schemes on the SHREC'17 Original dataset. As shape properties, we considered the height-field because of its simplicity and rough shape description and the Mean curvature because it generally performs well over geometric patterns. Scheme 1 is computed using  $r$  ranging from 7 to 15 and  $pxres$  ranging from 10 to 16. To evaluate Schemes 2, 3 and 4 we extracted the standard punctual descriptor with parameters  $r = 14$ ,  $n_r = 7$  and  $multP = 4$  and we considered only the sectors highlighted in the schemes. In this section, we report only the most significant runs. Table 4.5 summarizes the results of the tests of mpLBP with these settings. In particular, when using sampling scheme 1 with  $h$  equal to the Mean Curvature, we use  $r = 10$  and  $pxres = 16$ , while when using sampling scheme 1 with  $h = HF$ , we set  $r = 7$  and  $pxres = 12$ . Note that the size of the mpLBP descriptor varies according to the different neighborhood sampling schemes. Looking at the retrieval performances, we notice that mpLBP with scheme 1 (with a square-based point neighborhood) performs poorly compared to the other schemes. This is not really surprising because a square-like point neighborhood inserts an orientation and an anisotropic sampling of the model. Unlike images where a square-like neighborhood is compliant with the intrinsic grid structure, for a surface, a circle or a geodesic neighborhood better reflect the intrinsic surface metric. Schemes 2 and 3 highlight that other sampling strategies over a circular neighborhood are possible and lead to good performances. On the contrary an excessive sparse sampling like the one proposed by Scheme 4 jeopardises the mpLBP performance. These results highlight that the mpLBP technique can be adapted to different schemes.

SHREC'17: Original Dataset, scheme variants						
Parameters	NN	FT	ST	mAP	e	nDCG
Scheme 1 - Mean C	0.739	0.395	0.534	0.481	0.324	0.705
Scheme 1 - HF	0.794	0.504	0.622	0.560	0.360	0.755
Scheme 2 - Mean C	0.928	0.707	0.835	0.746	0.447	0.878
Scheme 3 - Mean C	0.928	0.667	0.804	0.715	0.436	0.861
Scheme 4 - Mean C	0.856	0.596	0.745	0.658	0.418	0.813

Table 4.5: Retrieval results of the various sampling schemes on the SHREC'17 Original Dataset.

### 4.2.2 Computational cost

The mpLBP algorithm is implemented in around 200 MATLAB lines of code. The most expensive part, in terms of computational cost is the creation of the punctual descriptor, which is based on a kd-tree (with a computational cost of  $n \log(n)$ ). This characteristic allows the mpLBP algorithm to run in a much shorter time if compared, for example, with the edgeLBP, while keeping similarly high evaluation scores. By running both edgeLBP and mpLBP on meshes with different number of vertices (from 5000 to 120000 vertices) and different parameter settings, we can see the huge gap between the timings of the two methods (see Table 4.6). Tests are run on a personal computer Intel Core i7 processor (at 4.2 GHz) with 32Gb RAM. The edgeLBP (as currently implemented) has the number of sectors per ring constant across all the rings. In order to have a fair comparison, we also set the number of sectors to be constant for mpLBP (i.e.,  $p_j = p$  with  $p \in \mathbb{N}$  fixed). We observed that  $n_r$  and  $P$  do not affect the computation times that much. Indeed, the radius size and the number of vertices are the biggest bottlenecks. Figure 4.8 provides another computational time comparison between edgeLBP and mpLBP showing the much more severe increase of the edgeLBP computational cost compared to the cost increase of the mpLBP. These timings are those obtained on the 120k vertices mesh, with  $r = 4.5$ ,  $n_r = 4$   $P = 15$ . Trends obtained by changing the parameters of both methods are almost identical to the ones reported (only the time scale (y-axis scale) changes based on the radius). It is worth also mentioning the different number and size of inputs demanded both methods. On one hand, the edgeLBP requires many relations between the triangulation elements (vertex-face, edge-edge, etc.), which may require a significant computational effort when computed on large meshes. On the other, the mpLBP barely requires the normal of the point clouds or on the vertex of the triangulation.

<b>5K</b>	r=2,5	r=3,5	r=4,5
$n_r = 4, P = 12$	22.04/2.88	16.89/1.38	19.06/1.35
$n_r = 7, P = 12$	15.74/1.59	19.91/1.55	24.54/1.60
$n_r = 4, P = 18$	11.40/1.27	15.89/1.48	17.12/1.38
$n_r = 7, P = 18$	16.14/1.95	20.39/1.88	30.18/2.55

<b>10K</b>	r=2,5	r=3,5	r=4,5
$n_r = 4, P = 12$	59.33/4.23	79.09/4.62	92.31/5.09
$n_r = 7, P = 12$	71.69/4.35	95.58/4.93	116.51/5.46
$n_r = 4, P = 18$	52.92/3.95	76.55/4.77	83.54/4.95
$n_r = 7, P = 18$	72.43/5.01	95.86/5.53	140.23/6.25

<b>15K</b>	r=2,5	r=3,5	r=4,5
$n_r = 4, P = 12$	81.13/5.31	118.42/7.48	143.29/8.00
$n_r = 7, P = 12$	107.26/6.63	143.08/7.52	178.01/8.40
$n_r = 4, P = 18$	81.92/5.96	115.85/7.38	128.10/7.49
$n_r = 7, P = 18$	107.83/7.53	143.77/8.19	188.56/9.32

<b>30K</b>	r=2,5	r=3,5	r=4,5
$n_r = 4, P = 12$	341.81/19.90	516.53/28.52	651.99/33.08
$n_r = 7, P = 12$	454.23/23.30	618.36/28.36	805.07/33.72
$n_r = 4, P = 18$	348.93/20.43	507.31/28.21	583.39/30.31
$n_r = 7, P = 18$	456.26/25.10	621.50/29.75	811.99/35.25

<b>90K</b>	r=2,5	r=3,5	r=4,5
$n_r = 4, P = 12$	2378.79/109.32	3661.28/158.43	4344.93/196.08
$n_r = 7, P = 12$	3024.61/122.58	4142.54/157.74	5200.46/194.75
$n_r = 4, P = 18$	2344.02/110.05	3481.22/160.97	3989.87/179.15
$n_r = 7, P = 18$	3034.85/128.34	4145.79/163.19	5704.31/201.03

<b>120K</b>	r=2,5	r=3,5	r=4,5
$n_r = 4, P = 12$	4314.18/165.65	6612.18/260.30	8341.62/335.82
$n_r = 7, P = 12$	5583.24/189.33	7812.26/260.18	9954.04/332.90
$n_r = 4, P = 18$	4236.92/170.22	6586.75/262.25	7626.82/309.25
$n_r = 7, P = 18$	5596.74/198.12	7806.80/266.27	10438.45/348.40

Table 4.6: Computational times for edgeLBP and mpLBP (in seconds). The top-left cell of each table indicates the number of vertices.

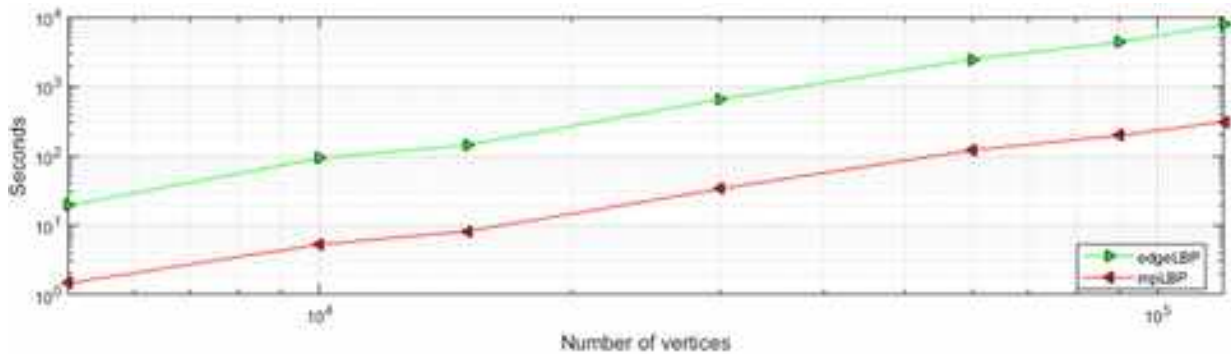


Figure 4.8: Computational time trends for the mpLBP and edgeLBP (in logarithmic scale).

## Conclusions

In this chapter we presented the mpLBP, a pattern descriptor with good performances on recent benchmarks. It is similar in spirit to the edgeLBP (see Chapter 3) with a way lower computational cost. Indeed, the edge-sphere intersection sampling scheme is replaced with a local average of the property that describe the patterns. The neighborhood of each point is computed using the kd-tree. While the performances are at most ‘only’ on par with those of the edgeLBP, the mpLBP remarkable quickness makes it an important contribution when working with very high resolution models or with point clouds.

## Related publications

- E. Moscoso Thompson, S. Biasotti, J. Digne, R. Chaine, *mpLBP: An Extension of the Local Binary Pattern to Surfaces based on an Efficient Coding of the Point Neighbours*. Eurographics Workshop on 3D Object Retrieval, 2019.
- E.Moscoso Thompson, S. Biasotti, J. Digne, R. Chaine, *mpLBP: A point-based representation for surface pattern description*, Computers & Graphics, 2020.

# Chapter 5

## Benchmarking activities

The creation of datasets and their ground truth is the basis on which to build a competitive and fair evaluation system, driving also the identification of new research directions. The intrinsic goal of developing a new benchmark is not unique: it ranges from the need of a summary of the methods able to face a given task to highlighting what is still missing or needs further developments.

With this goals in mind, we organized five SHREC tracks whose challenges revolve around our research topic. This chapter presents them, sorted by year of publication of the respective final reports. One of the reports, in particular, addressed the geometric pattern recognition problem: for this reason, the thoughts and conclusions developed in that contest are presented as a preliminary discussion of that problem and deepened in Chapter 6.

This chapter is organized as follows: we initially introduce the performance evaluation measures we use in most of our benchmarks (Section 5.1). Then, our first benchmark on colorimetric patterns is described in Section 5.2, followed by the one focused on aspects that are preliminary to our research as addressed the problem of feature curve recognition Section 5.3. Finally, we proposed two contest on geometric patterns (Section 5.4 and Section 5.5). These three contributions are described in the following sections, after the performance evaluation measures we used to compare the methods in each benchmark.

### 5.1 Performance measures for pattern retrieval methods

The research in information retrieval has led a number of performance metrics [Rij79, BYRN99] that can be adopted also to evaluate the performance of 3D retrieval and classification methods [TV04, SMK04]. While a single measure is not enough to fully assess the quality of a method, the combination of multiple measures gives a global view of the various methods, highlighting different properties (goodness of the method per model, class or overall with respect to

multiple criteria). In the following we list the performance measure used in our benchmarks.

*Nearest Neighbor (NN), First tier (FT), Second tier (ST).* These measures check the fraction of models in the query class also appearing within the top  $k$  retrievals [SMKF04]. In the case of NN,  $k$  is 1 and corresponds to the classification rate if the nearest neighbor classifier would be performed. Given a class of  $|C|$  elements,  $k$  is  $|C| - 1$  for the FT and  $k$  is  $2 * (|C| - 1)$  for the ST. Higher values of the NN, FT and ST measures indicate better matches. These measures range in the interval  $[0, 1]$ .

*Normalized Discounted Cumulated Gain (nDCG).* This measure [SMKF04] is based on the assumption that relevant items are more useful if appearing earlier in the list of the retrieved items. The nDCG is based on the graded relevance of a result with respect to the query. Then, the value is normalized with respect to the ideal outcome of that query.

*Average precision-recall curves, mAP and e-Measure (e).* Precision ( $P$ ) is the fraction of retrieved items that are relevant to the query. Recall ( $R$ ) is the fraction of the items relevant to the query that are successfully retrieved. More formally,

$$P = \frac{|A \cap B|}{|B|}, \quad R = \frac{|A \cap B|}{|A|}, \quad (5.1)$$

where  $A$  is the set of relevant elements (true positive and false positive items) and  $B$  is the set of positive items. By plotting the precision value with respect to the recall value we obtain the so-called recall vs. precision curve: the larger the area below such a curve, the better. In particular, the precision-recall curve of an ideal retrieval system would result in a constant curve equal to 1. For each query, we have a precision-recall (PR) curve. In our context, results are evaluated on the mean of all the PR curves. The *mean Average Precision (mAP)* corresponds to the area between the horizontal axis and the average precision-recall curve and ranges from 0 to 1. The higher, the better. The e-Measure ( $e$ ) derives from the precision and recall for a fixed number of retrieved results (32 in our settings), [Rij79]. For every query, the e-Measure considers the first 32 retrieved items and is defined as  $e = \frac{1}{P^{-1} + R^{-1}}$ , where  $P$  and  $R$  represent the precision and recall values over those results, respectively.

*Confusion matrix.* To each run we associate also a confusion matrix  $CM$ , that is, a square matrix whose order is equal to the number of classes in the dataset. For a row  $i$  in  $CM$ , the element  $CM(i, i)$  gives the number of items which have been correctly classified as elements of the class  $i$ . The elements  $CM(i, j)$ , with  $j \neq i$ , count the items of the class  $i$  which have been misclassified and  $j$  corresponds to the class in which they were wrongly classified. An ideal classification system should be a diagonal matrix. The sum  $\sum_j CM(i, j)$  equals the number of items in the class  $i$ . Generally, the confusion matrix is non-symmetric.

*Tier images.* Similar to the confusion matrix, the tier image visualizes the matches of the NN, FT and ST. The models of a class are grouped along each axis so it is easier to interpret, following the definition in [SMKF04]. With this configuration, the optimal tier image clusters the pixels related to NN and FT on the diagonal.

*Receiver Operating Characteristic (ROC) curve and AUC value.* ROC curves are largely used to evaluate the classification performance of a method and are suitable to assess retrieval issues, too. The ROC curve shows the ratio between False Positive Rate and True Positive Rate for each model at different classification thresholds. In our scenario, the classification thresholds are the number of models in each class (20) multiplied by a scalar value that goes from 1 to the number of classes in the dataset (11). The higher the curve, the better. A coarse comparison between the methods based on the ROC curves can be derived also by the AUC value (namely the *area under curve* value), which is the measure of the area under the ROC curve. The higher this value is, the better. Anyway, note that an AUC value of 0.5 means that the corresponding method is not able to classify the models at all. In this work, we consider the mean of all ROC curves.

## 5.2 Retrieval of gray patterns depicted on 3D models

The aim of this benchmark [MTW<sup>+</sup>18] is to evaluate the performance of retrieval algorithms for 3D surfaces decorated with one or more gray patterns. The web page is available at [http://shrec.ge.imati.cnr.it/shrec18\\_color/](http://shrec.ge.imati.cnr.it/shrec18_color/).

### 5.2.1 Dataset

The dataset consists of 300 surfaces characterized by different patterns. We created a set of 20 base models, selecting triangles meshes from the *cups and vases* classes of the SHREC'07 Watertight model contest [GBP07] and the *goblets* class of the COSEG [WAvK<sup>+</sup>12] datasets, see Figure 5.1. All models are oriented, connected and locally regular, triangle meshes with at most, two boundary components. The rationale behind the choice of these base models is the fact that these shapes do not possess a geometrically privileged point of view, many of them (e.g., cups) have an interior and exterior face and trivially projecting them in a plane is not possible without distorting the surface.

We created 15 black and white textures (obtained via manual painting): 10 textures (called *Single textures*) are characterized by a single pattern while 5 more textures (called *Double textures*) are obtained joining the 10 textures two by two, see Figure 5.2. The 15 textures have been applied to all the 20 base meshes using a semi-automatic procedure, which required a manual fix of the geometry of some models. Each pattern is applied so that it keeps the same scale over all the models. Then, the colorimetric information is stored in the vertices of the triangle mesh  $M$ . At the end of this process, every model is covered by a black and white pattern on at least the 30% of its surface, while the rest of the surface is only black or only white (see Figure 5.3d as an example). Assuming that the maximum value of the luminosity is 100, we uniformly alter the luminosity of each model with a fixed amount that is a randomly selected from the set  $\{0, 8, 16, 24, 32, 40, 48, 56, 64, 72\}$ .

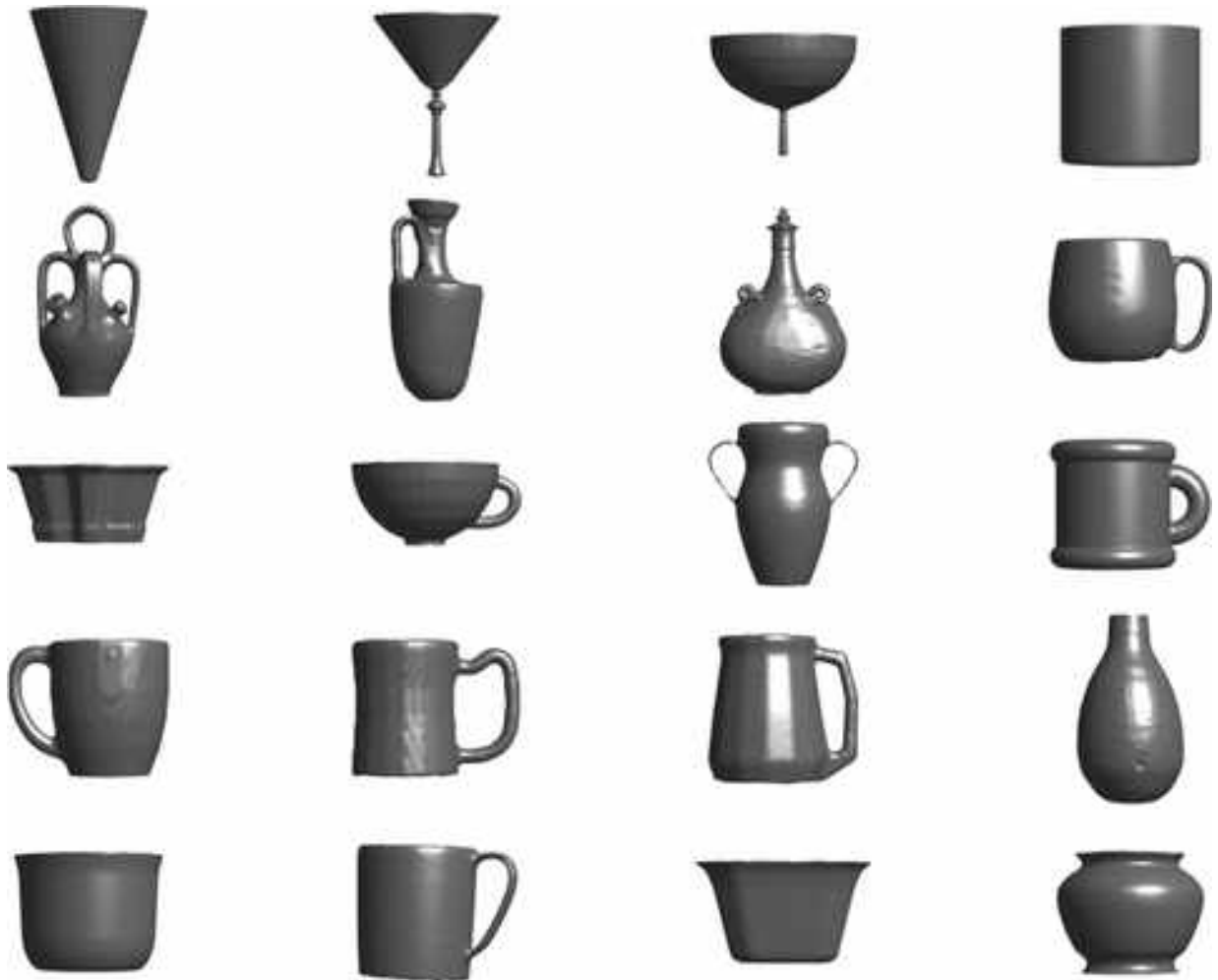


Figure 5.1: The models used as base surfaces.

This results in a more or less faded version of the original model, see examples in Figure 5.3(b-c).

The number of vertices of the 300 models in this dataset ranges from  $95K$  to  $107K$ . Similarly to the outcome of many laser scan systems, the colorimetric information related to each model is stored as a RGB value associated with each vertex (no texture images are provided).

## 5.2.2 Results

The challenge of this contest is to group the objects of the dataset according to the pattern impressed on them. The overall geometric shape of the object is not relevant; in practice the methods are asked to classify an object using the local, colorimetric properties. 9 runs are evaluated



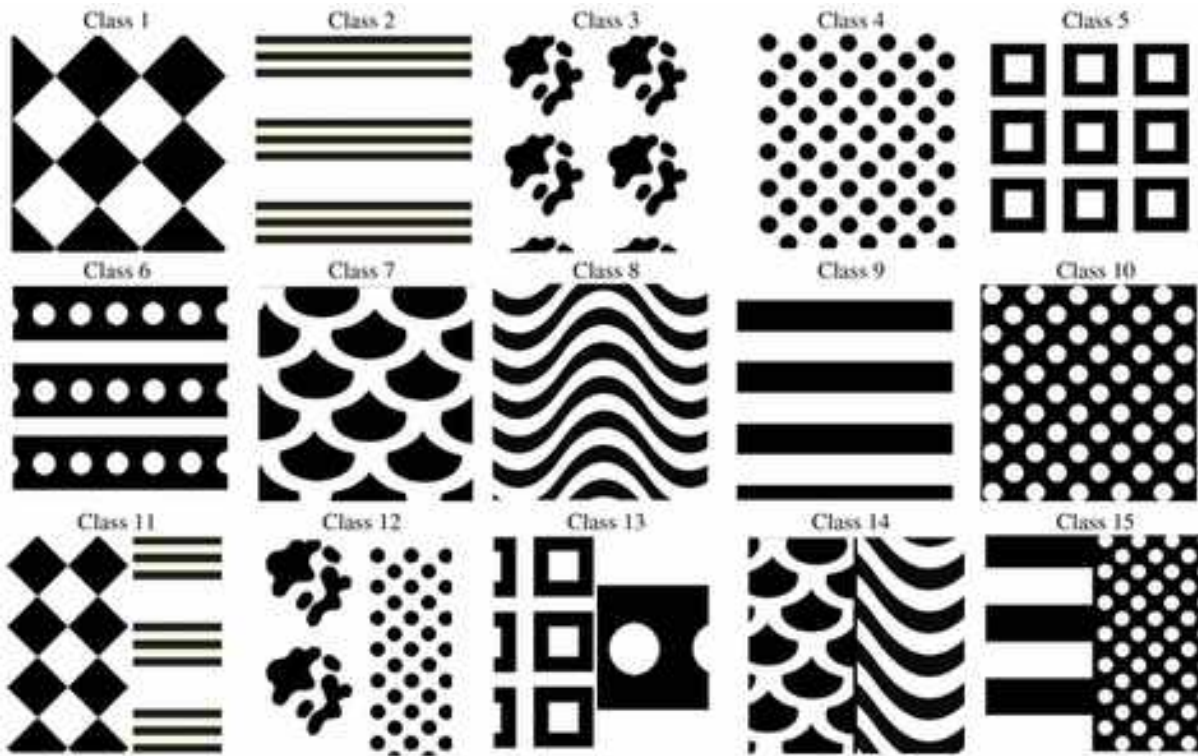


Figure 5.2: Textures (patterns) depicted on the models. Class 11 to 15 are Double Textures.



Figure 5.3: Models in the dataset of this contest, with different fades. The first two models are fully covered by a single pattern, the third presents a faded double pattern while the surface of the last model is partially covered with the pattern decoration and the rest is uniformly black.

according to the following classifications:

- The *Single Pattern Dataset* contains the 200 models of the dataset that are characterized by a single pattern (*Single textures*). Models in this dataset are grouped in 10 classes of 20 elements.

- The *Complete Dataset* contains also the 100 models characterized by *Double textures* that mix the 10 single textures. Models in this dataset are grouped in 15 classes, each class contains 20 elements.

A summary of the evaluated methods is reported in the following. For additional details we refer to [MTW<sup>+</sup>18].

1. Color distribution on a model in CieLAB space color (*HBl* and *HnBl*). A baseline method based on color histograms (*HBl*) and normalized histograms (*HnBl*).
2. *From Mesh to Image using Ordered Ring of Facets (ORF) for gray Texture Retrieval* by Claudio Tortorici, Naoufel Werghi and Stefano Berretti (*TWB\**). The algorithm extracts images holding the color texture information of a region of the surface. The images are extracted using the Ordered Ring Facet (ORF) [WTBdB15, TWB15]. The comparison between two models is done using Bhattacharyya distance. The runs differ for the number of sampled images.
3. *Histogram of Double Distance Transform* by Santiago Velasco-Forero (*V\**). This method is described in Chapter 1. The runs differ for the number of bins of the concatenated histograms.
4. *Edge Local Binary Pattern (edgeLBP)* detailed in Chapter 3.

For each setting, given a query model, its relevant models are the models with the same pattern(s) of the query. We highlight that, when considering the complete dataset, the models characterized by a double texture are not in the same class of the models that have its single texture components.

The retrieval performance of each run has been evaluated in both the Single Pattern and Complete datasets. With respect to the results in [MTB18], the results obtained on these dataset are those reported in the respective extended issue [MTB19]. The settings used in the tests are the following:

- R1:  $n_{rad} = 5, P = 15, R = 0.04$ ;
- R2:  $n_{rad} = 7, P = 12, R = 0.04$ ;
- R3:  $n_{rad} = 5, P = 15, R = 0.03$ ;
- R4:  $n_{rad} = 5, P = 15, R = 0.05$ ;
- R5:  $n_{rad} = 5, P = 15, R = 0.08$ .

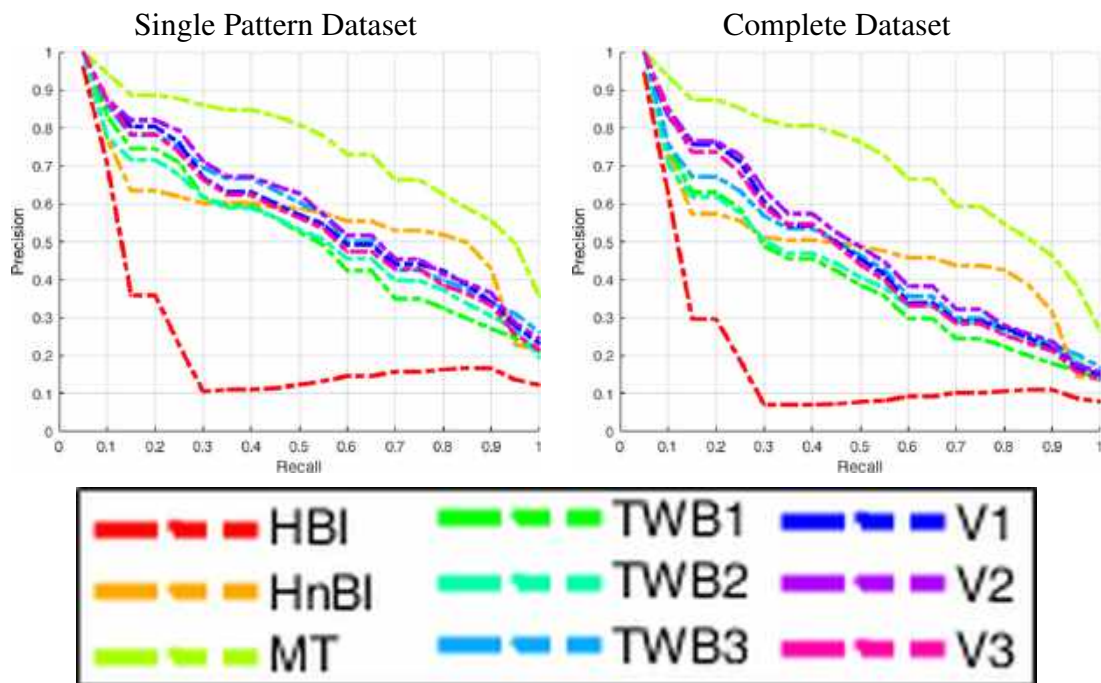


Figure 5.4: The Precision-Recall plots for all the runs reported in this contest.

Single Pattern Dataset							Complete Dataset						
Run	NN	FT	ST	e	nDCG	mAP	Run	NN	FT	ST	e	nDCG	mAP
HBI	0.52	0.141	0.23	0.097	0.501	0.233	HBI	0.467	0.115	0.16	0.139	0.548	0.184
HnBI	0.595	0.52	0.773	0.425	0.698	0.564	HnBI	0.557	0.437	0.646	0.509	0.75	0.481
TWB1	0.71	0.454	0.632	0.338	0.679	0.516	TWB1	0.593	0.367	0.514	0.419	0.758	0.408
TWB2	0.615	0.46	0.675	0.362	0.681	0.523	TWB2	0.45	0.379	0.553	0.439	0.75	0.424
TWB3	0.755	0.502	0.688	0.376	0.711	0.577	TWB3	0.593	0.417	0.564	0.455	0.795	0.46
V1	0.82	0.45	0.725	0.386	0.743	0.571	V1	0.77	0.412	0.584	0.474	0.798	0.472
V2	0.82	0.51	0.731	0.39	0.753	0.593	V2	0.79	0.433	0.594	0.481	0.808	0.493
V3	0.815	0.474	0.71	0.376	0.733	0.557	V3	0.78	0.394	0.569	0.463	0.786	0.462
<i>edgeLBP-R1</i>	0.905	0.699	0.863	0.594	0.888	0.749	<i>edgeLBP-R1</i>	0.87	0.646	0.805	0.551	0.865	0.697
<i>edgeLBP-R2</i>	0.905	0.706	0.870	0.590	0.889	0.749	<i>edgeLBP-R2</i>	0.893	0.647	<b>0.807</b>	0.542	0.864	0.696
<i>edgeLBP-R3</i>	0.900	0.631	0.827	0.555	0.858	0.682	<i>edgeLBP-R3</i>	0.857	0.563	0.741	0.489	0.820	0.610
<i>edgeLBP-R4</i>	0.915	0.717	0.879	0.60	0.898	0.766	<i>edgeLBP-R4</i>	0.903	<b>0.673</b>	0.827	<b>0.557</b>	<b>0.878</b>	0.722
<i>edgeLBP-R5</i>	<b>0.950</b>	<b>0.740</b>	<b>0.892</b>	<b>0.606</b>	<b>0.911</b>	<b>0.790</b>	<i>edgeLBP-R5</i>	<b>0.923</b>	0.667	0.805	0.546	<b>0.878</b>	<b>0.727</b>

Table 5.1: Summary evaluation over the single pattern dataset (left) and over the complete dataset (right).

Table 5.1 reports the outcome of the algorithms on this benchmark. Average PR curves are displayed in Figure 5.4. Confusion matrices are reported in Figure 5.5 and Figure 5.6. Finally,

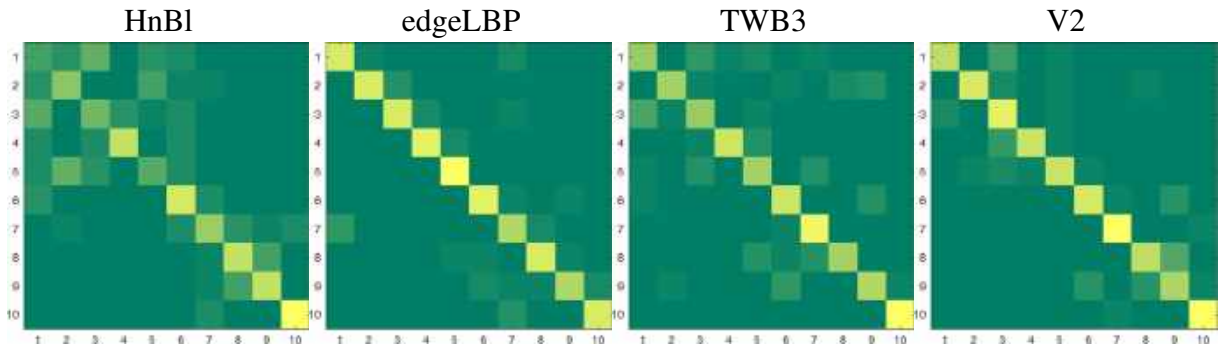


Figure 5.5: The best confusion matrices over the single pattern dataset.

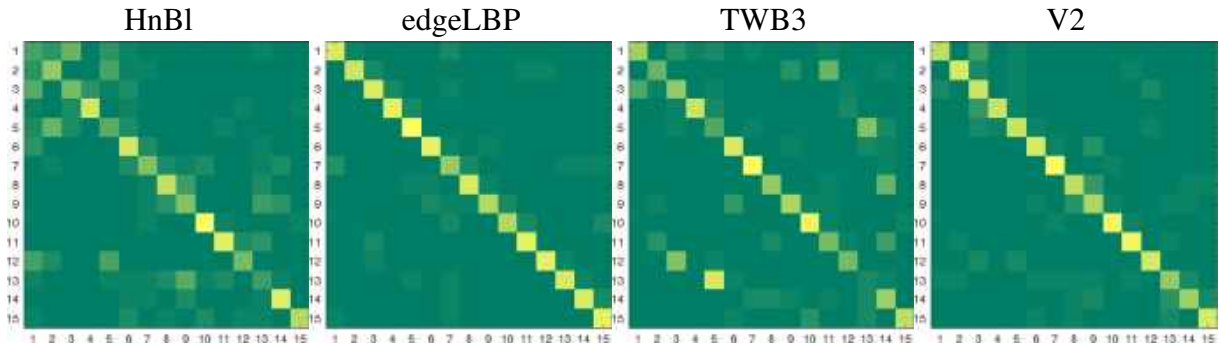


Figure 5.6: The best confusion matrices over the complete dataset.

the best tier images per participant is reported in Figure 5.7 and Figure 5.8.

The recall-precision curves in Figure 5.4 and the statistics in Table 5.1 indicate that the pattern retrieval task is challenging on this dataset. Even if the proposed dataset is not a real world dataset (with a huge variety of shapes and color patterns) it presents two important challenges that the pattern retrieval on 3D surfaces problem must face: (i) the impossibility of encoding the colorimetric information of a mesh in a single image and (ii) the presence of the same pattern over a number of different shape embeddings.

Interestingly, the performance of the methods is quite stable over the single pattern and the complete datasets, with a quite limited degradation on the complete one (the largest decrease for the NN value is approximately 15% for the *TWB\** runs). Besides the NN evaluation, in which there is an average variance of values, the FT, ST, e and nDCG measures seem to be similar for most methods, polarizing in a range of values for each measure ([0.45,0.50] for FT, [0.65,0.75] for ST and so on). The outliers are HBl, whose performances are really low, as expected for an histogram-based method without any local analysis of the gray distribution, and edgeLBP, whose performances overcome the other runs of approximately 10%.

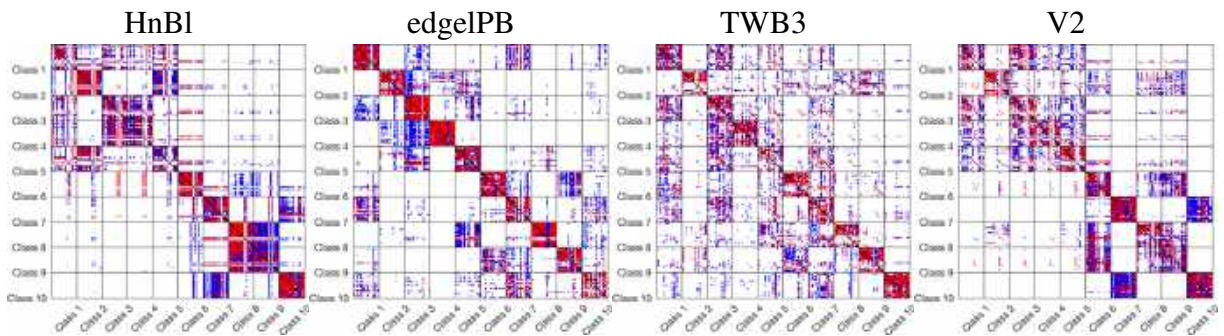


Figure 5.7: The best tier image per participant on the single pattern dataset. NN are marked in black, FT in red and ST in blue.

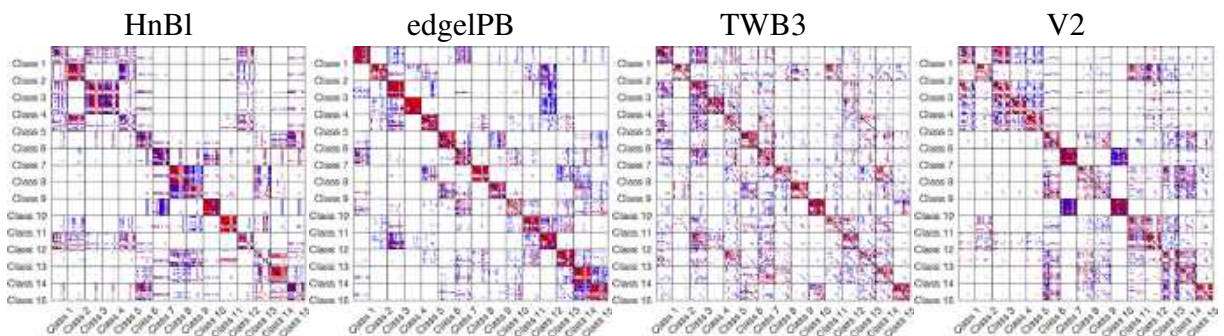


Figure 5.8: The best tier image per participant on the complete dataset. NN are marked in black, FT in red and ST in blue.

More details of the performance of the methods over the different classes are shown in the confusion matrices and tier images, see Figures 5.5, 5.6, 5.7 and 5.8. Confusion matrices help us understand which are the hardest patterns to correctly deal with. For the baseline method, the confusion matrices show that there are multiple pattern classes that are tricky for them. This is especially true for the HBl run on the Complete dataset. For the methods who submitted more runs, the confusion matrices show a quite stable behavior on the classes of the benchmark, despite the runs different settings. On all runs, the tier images confirm the trends revealed by the confusion matrices.

All runs submitted to this track are based on feature vectors. In the case of the baseline methods (simple histograms of the L-channel values) the description does not encode any information of the gray distribution in the vertex neighbor. All other methods propose strategies for coding the colorimetric information evolution in a region around the vertices. From the retrieval results it seems that methods that works directly on the mesh are independent of the tessellation (MT and V\*) achieve better performances.

## 5.3 Feature Curve Extraction on Triangle Meshes

In the context of this manuscript, a *surface feature* is a local surface variation that is not repeated more than a couple of times. In other words, a surface feature is intended as a bending of the surface that shapes a part of interest (e.g.: the eye of a statue is a feature, while the hand of a statue is not). Related to this definition, a *surface feature curve* (or feature curve) is a line that delineates a surface feature. Feature curves derive from the human perception and interpretation of a surface, both in terms of localization and width. On a more practical level, feature curves are mostly defined by ridges or valleys, thus characterized by principal curvature extremes. Psychologists and computer vision scientists who studied how humans perceive a shape have identified curvature variations, in terms of changes from convex to concave regions, as a key element of the human perception [PT96].

Since local variations are a component of both patterns and surface features, the research field of finding feature curves could help the pattern analysis by removing parts of the surface that are not of interest for the pattern analysis. While the identification of feature curves can be related to the identification of a relative feature, this is not mandatory. Most methods tackle the problem by looking directly for the feature curves. For example, in [APM15] groups the salient points into a curve skeleton that is fitted with quadric spline approximation. It is also possible to consider entire families of curves, like in [HT10, HT11], proposing them as a natural way to describe line drawings and silhouettes. However, most of the proposed methods still have difficulties in dealing with partial information or fails in correspondence of noise.

This Section presents the benchmark we designed to evaluate methods for the feature curve detection [MTAM<sup>+</sup>19]. We meant this analysis as a complement of pattern recognition because the analysis of archaeological findings is born from their combination. Since this task differs from the rest in terms of result structure, we briefly introduce the evaluation measures we considered in this benchmark.

### 5.3.1 Evaluation of feature curve characterization methods

We consider as a feature curve a set of vertices of a model that jointly define a contour, a valley or a ridge on the mesh. The mesh boundary (if it exists) is considered nor a feature curve, neither the noise artifacts (e.g., from scan inaccuracy and/or smoothing/resampling operations). An efficient method for feature curves extraction, as intended in this benchmark, needs to group sets of mesh vertices so that each set identifies a single feature curve.

As a further evaluation, we compare the similarity estimation between the feature curves each method found, if it allows such a comparison.

The dataset consists of 15 surfaces characterized by at least one feature curve. Some of the













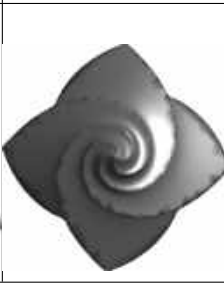


M1 (50541)	M2 (83556)	M3 (92902)	M4 (25397)	M5 (25411)
				
M6 (33910)	M7 (18319)	M8 (18659)	M9 (25280)	M10 (214868)
				
M11 (11144)	M12 (11433)	M13 (7919)	M14 (30174)	M15 (5192)
				

Figure 5.9: Overview of the dataset of the SHREC'19 on feature curve detection. In the brackets of each model is the corresponding number of vertices.

models are obtained through scans, while others are made in *silico*. Some models are derived from the Visionair shape workbench (<http://visionair.ge.imati.cnr.it/>) and the Turbosquid repository of 3D models (<https://www.turbosquid.com/Search/3D-Models>). The original models of the ornaments from which derive the models from 4 to 10 are available thanks to the courtesy of the prof. Karina Rodriguez Echavarria. Both vertex distribution and density vary from mesh to mesh. Figure 5.9 shows an overview of the models, together with their number of vertices.

The definition of a ground truth for this task is a challenging job, since no formal definition of a feature curve on surfaces exists. The ground truth was defined by people from the IMATI-CNR (Italy) staff, requested to highlight the vertices of each model if, in their opinion, they were part

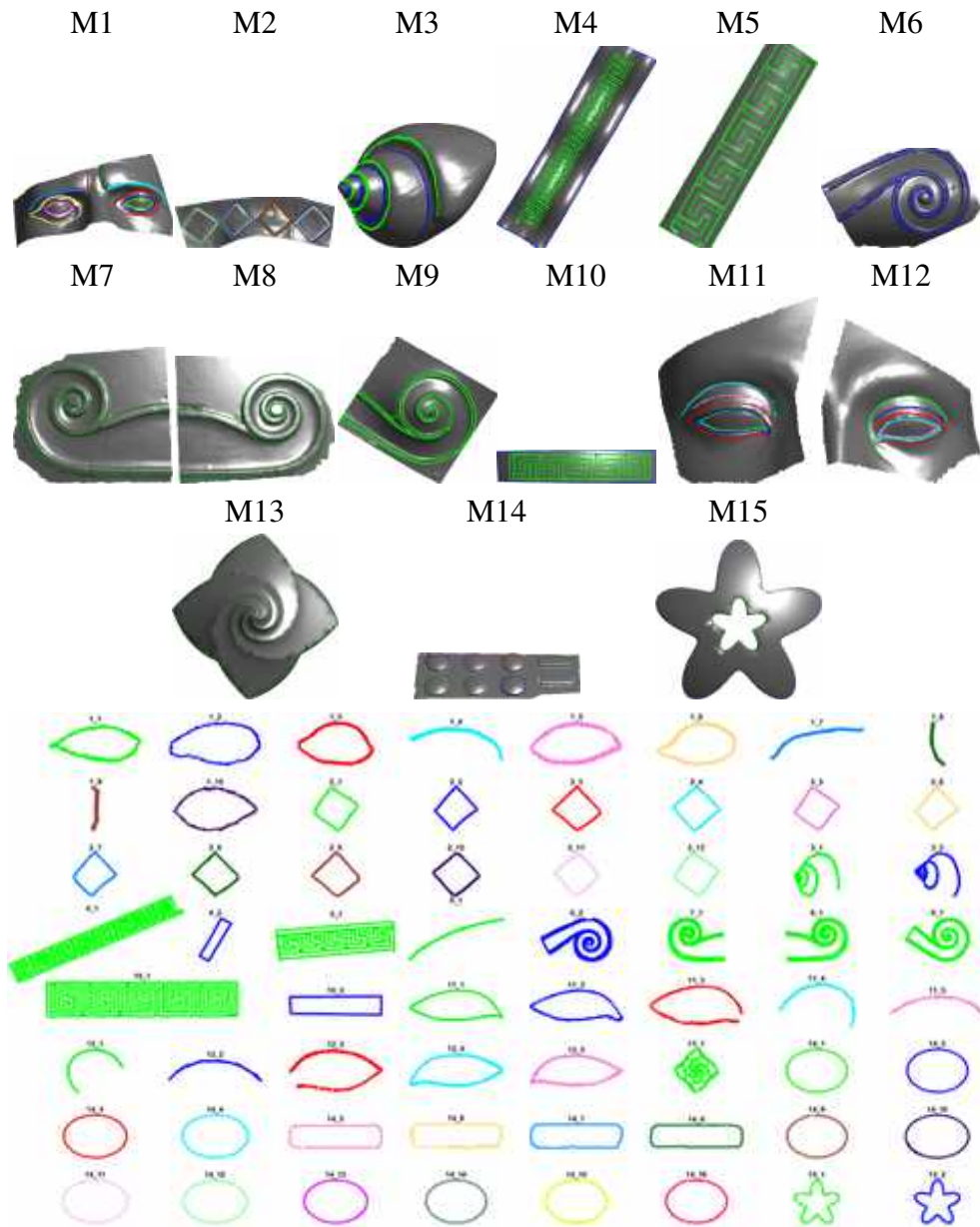


Figure 5.10: The final ground truth of the contest. Top: the feature curves on the models; Bottom: the feature curves represented one by one.

of feature curves. Then, a ground truth based on these individual annotations has been created. An overview of the final ground truth is shown in Figure 5.10.

For a given model  $M$ , the outcome of each method is expected to be a set of  $n_M$  separate lists  $f_{p_i}$  of vertices, resembling the set of  $n_M$  feature curves highlighted in the ground truth. More for-



mally, the expect result format is a set of lists  $P(M) = \{fp_1, fp_2, \dots\}$  for each  $M$  in the dataset. The evaluations are done by comparing this set with the set  $GT(M) = \{fc_1, fc_2, \dots, fc_{n_M}\}$  of feature curves defined in the ground truth. We consider two classifications:

- *[Overall Comparison](O-comp)*: all the feature curves found on each model are jointly evaluated with the described evaluation measures, matching them with the ground truth data. More formally, the sets  $\cup_i fp_i$  and  $\cup_i fc_i$  are compared.
- *[Curve-by-curve Comparison](CbC-comp)*: let us consider a feature curve  $fc_j$  of the model  $M$  and the set  $P(M)$  of feature curves proposed by a participant. In the lists in  $P(M)$ , the closest to  $fc_i$  is selected and compared. The closest curve is selected by the same people that defined the ground truth by voting the curve in  $P(M)$  which overlaps  $fc_i$  the most.

## Methods

Four methods were considered. Depending on the design choices, the results vary in precision, sensitiveness and overall quality. The methods description is now reported, following [MTAM<sup>+</sup>19].

### Spectral based saliency estimation for the identification of features (SBSE)

This method works in two steps. The first step estimates the saliency of each vertex using spectral analysis. The magnitude of the estimated saliency identifies if a vertex is a feature or not. Based on the geometry, it is possible to say that the feature vertices represent the edge of a feature curve (both crests and valleys) or corners. At the second step, the mean curvature of the extracted features is estimated and it is used to classify the different feature curves (if they exist). Additionally, the information related to the mean curvature and the saliency of each feature curve are used to find similarities with feature curves of other models. The execution time of the algorithm depends on: (i) the size of the mesh and (ii) the size of the patches, but generally, it is very fast.

**Definition and computation of vertex saliency** For each of the  $n$  vertex  $v_i$ , a patch  $\mathcal{P}_i = \{\mathbf{v}_i, \mathbf{v}_{i_1}, \dots, \mathbf{v}_{i_k}\}$  vertices is created, which consists in the  $k$  geometrical nearest vertices to the vertex  $\mathbf{v}_i$  based on their coordinates (typically  $k = 15$ ). These points are used to define a matrix  $\mathbf{N}_i \in \mathbb{R}^{(k+1) \times 3}$  for each vertex:

$$\mathbf{N}_i = [\mathbf{n}_i, \mathbf{n}_{i_1}, \dots, \mathbf{n}_{i_k}]^T, \quad \forall i = 1, \dots, n$$

where the normal  $\mathbf{n}_i$  of the vertex  $\mathbf{v}_i$  is defined as:

$$\mathbf{n}_i = \frac{\sum_{j \in \mathcal{N}_i} \mathbf{n}_{c_j}}{|\mathcal{N}_i|}, \quad \forall i = 1, \dots, n,$$

where  $\mathbf{n}_{c_j}$  is the normal of the  $j$ -th face of the mesh and  $\mathcal{N}_i$  is the first-ring area of the vertex  $i$ . For each vertex, the associated covariance matrix  $\mathbf{R}_i = \mathbf{N}_i^T \mathbf{N}_i$  is decomposed:

$$eig(\mathbf{R}_i) = \mathbf{U}_i \Lambda_i \quad \forall i = 1, \dots, n$$

where  $\mathbf{U}_i \in (R)^{3 \times 3}$  denotes the eigenvectors matrix and

$$\Lambda_i = diag(\lambda_{i1}, \lambda_{i2}, \lambda_{i3})$$

The value  $s_i$  is the *saliency* of  $\mathbf{v}_i$  and it is defined as the value given by the inverse *norm* – 2 of the corresponding eigenvalues:

$$s_i = \frac{1}{\sqrt{\lambda_{i1}^2 + \lambda_{i2}^2 + \lambda_{i3}^2}}, \quad i = 1, \dots, n.$$

$s_i$  is normalized to be in the  $[0, 1]$  range as follows:

$$\bar{s} = \frac{s_i - \min(s_i)}{\max(s_i) - \min(s_i)}, \quad i = 1, \dots, n$$

This method assumes that a small value of saliency means that the vertex lies in a flat area, while a big value indicates that the vertex belongs to an edge or corner. This characterization depends by the number of dominant (or main) eigenvalues. For example, considering a cube, a vertex that "has" three, two or one dominant eigenvectors is, respectively, on a corner, on an edge or on a flat area.

A k-mean algorithm is used for separating the normalized values of the saliency into five different classes. The first two are considered as non-feature vertices, while the other are actual feature vertices.

**Clustering of salient vertices** The feature curves are identified by grouping the feature vertices based on mean curvature  $m_c$  values. Since the initial number of feature curves is unknown for each model, the optimal number of clusters is supposed to range from 1 to 5 and it is estimated using the Calinsky-Harabasz clustering evaluation criterion, followed by the k-means algorithm that performs the actual clustering.

Moreover, the feature curve similarity among models is assessed through the histograms of saliency and mean curvature. More specifically, for a given model the histograms of the  $\bar{s}$  value

and the normalized mean curvature are computed (respectively  $\dot{s} \in \mathbb{R}^{10 \times 1}$  and  $\dot{m} \in \mathbb{R}^{10 \times 1}$ ). Then they are horizontally stacked in the vector  $q = [\dot{s}, \dot{m}]$ . The correlation coefficient  $r$  of two models A and B defined by the vectors  $q_A$  and  $q_B$  is:

$$r = \frac{\sum_{i=1}^{20} (q_{A_i} - \bar{q}_A)(q_{B_i} - \bar{q}_B)}{\sqrt{\left(\sum_{i=1}^{20} (q_{A_i} - \bar{q}_A)\right)\left(\sum_{i=1}^{20} (q_{B_i} - \bar{q}_B)\right)}}$$

where  $\bar{q}_*$  represents the mean value. The lower  $r$  is, the higher is the similarity between A and B.

### Point aggregation based on angle and curvature saliency (PCs)

Two methods were proposed, labelled PCs:A and PCs:C, respectively. Both have the same approach: defining a set of candidate vertices with a significant difference in a given property. Then candidate vertices that might be on a flat region and/or small fragments are removed, to reduce noise in the output. With this approach, all the feature curves obtained on a single model are grouped, thus we consider these methods only in the overall comparison. The core difference between the two methods is how the candidate vertices are determined.

**Angle-based vertex saliency (PCs:A)** The first method works in three steps. First, the angle  $\theta_i$  between each pair of connected triangles (by one edge) is computed. Then, if  $\theta_i > \alpha \text{mean}_i(\theta_i)$  (with *mean* equal to the average value), the two extremes of the relative edge are considered as candidate vertices.  $\alpha$  is set equal to 1.3 in most cases, aside from 1.6 for *Model2* and 2.6 for *Model3*. Second, if two candidate vertices share an edge larger than the double average length of the edges the two candidate vertices are removed. Finally, a graph with each pair of candidate vertices as nodes is created. All the connected components of this graph are computed and, if the number of vertices in a component is less than 1% of the number of vertices of the mesh, the vertices are removed.

**Normal curvature-based vertex saliency (PCs:C)** The second method runs in three steps. First, the oriented normals per-triangle are computed and for each vertex the normal of a vertex is the average of the weighted sum of its incident faces, with weights being proportional to a face's area. For each edge in the mesh, if its extremes are  $p_1, p_2$  with normals  $n_1, n_2$ , an estimation of its curvature is given by:

$$\text{curv} = \frac{(n_2 - n_1)(p_2 - p_1)}{|p_2 - p_1|^2}$$

Second, the average curvature of each vertex  $v_i$  is estimated as the geometric mean of the absolute values of all the edge curvatures of incident edges at the selected vertex. This evaluation is

smoothed by averaging the value with those of its immediate neighbors. This is repeated multiple times. Vertices with large touching triangles indicate that the surrounding area is relatively flat and thus filtered away, checking if their adjacent vertices have length larger than some value proportional to the average of the edge length. Of the remaining vertices, those that have a curvature value larger than  $a + k(\text{mean}_i(\text{curv}(v_i)))$  are flagged as possible elements of some feature curves. This formulation derives from the following observation: if the curvature value is larger than the average, at some point then it is highly possible that it is part of some curve, but in a sample with mostly noisy texture, this limit needs to be relaxed. In this method,  $a = 0.025$  and  $k = 0.7$ . Finally, to reduce noise, the components with fewer vertices are removed. Also, large components that have no nearby other flagged vertices are removed.

### Point-based multi-scale curve voting (PMCV)

This method extracts feature lines from meshes using a voting system based on a set of 3D curves generated in the direction of minimal curvature in anisotropic regions.

**Point cloud sampling and curve generation** Each mesh  $\mathcal{M}$  is converted to a dense point cloud with a uniform point cloud  $\mathcal{P}$  through a uniform sampling weighted by the face area. Feature curves as intended in this benchmark are characterized by a small curvature along the feature and a large curvature in orthogonal direction. The curvature is evaluated using the APSSlocal surface estimation [GG07]. Curves are generated at five levels of scale, based on the size of the neighbor used to approximate the surface with APPS, namely  $t_i = \frac{\bar{e}}{2}(2 + i\bar{e})$ ,  $i = 0, \dots, 4$  where  $\bar{e}$  is the median edge lengths in  $\mathcal{M}$ .  $\mathcal{P}$  is sub-sampled in 5 sparse point clouds  $\mathcal{P}_i$  using a Poisson disk sampling, with radius  $r_i = \frac{t_i}{10}$  plus an additional cloud  $\mathcal{P}_l$ , with  $r_l = \frac{t_0}{2}$ . For each  $t_i$ , curves are iteratively generated from each point in  $\mathcal{P}_l$  as follows:

$$\mathbf{p}_{j+1} = \text{proj}(\mathbf{p}_j + \Delta \mathbf{v}(\mathbf{p}_j))$$

with  $\Delta = \frac{t_0}{2}$ .  $\mathbf{v}(\mathbf{p}_j)$  is the direction of minimal principal curvature computed on  $\mathcal{P}_i$ . *proj* projects a point on the APPS surface approximation of  $\mathcal{P}_i$ , ensuring that the curve remains close to the surface. The iterations stop after reaching a maximum, set at  $10^5$ , or if the curve leaves the curved area, i.e. if  $\frac{|\kappa_1 - K_i|}{K_i} > \alpha$ , where  $\kappa_1$  is the maximal curvature,  $K_i$  is set to the 90<sup>th</sup> centile of maximal curvature absolute values calculated in  $\mathcal{P}_l$  at scale  $t_i$  and  $\alpha$  is set to 0.5. In order to filter noise or insignificant features, if the number of iterations is lower than 50, the curve is discarded.

**Voting-based feature line extraction** The vertices of  $\mathcal{M}$  accumulate votes from the extracted neighbor curves. Each vertex of each curve accumulates a vote in its neighboring mesh vertices. The size of the spherical neighborhood is  $\bar{e}$ . A vote is a negative scalar coefficient for valley lines

and positive for crest lines, with absolute values ranging from 0 to 1 according to the distance between the curve vertex and the mesh vertex. Sign is used to balance the sum of vertices close to both valley and crests. Finally, a region growing process delineates an individual set of vertices based on these votes. A region grows from a vertex to its neighbor if the sum of the votes has the same sign and if its absolute value is greater than  $\frac{1}{20}V_{max}$ , where  $V_{max}$  is the maximal absolute value of votes on the vertices of the mesh.

### **Feature curve characterization via mean curvature and algebraic curve recognition via Hough transforms (MHT)**

This feature curve recognition method derives from the technique described in [TBF18] and works in three steps.

**Feature point characterization** The vertices at which the mean curvature is significant (e.g. with high maximal and low mean curvature values) are selected as feature points. The curvature is estimated using [Pey]. Feature points are obtained by filtering the distribution of the mean curvature by means of two thresholds  $m$  and  $M$ . Note that  $m$  and  $M$  are two input parameters. Their value varies according to the precision threshold set for the property used to extract the feature points (e.g., in our case, two typical values of  $m$  and  $M$  are 15% and 85%, respectively).

**Feature curve aggregation** Feature points are aggregated to determine the elements that potentially correspond to a curve. Once detected, the set of feature points is subdivided into smaller clusters (that is, groups of points sharing some similar properties) by using a clustering algorithm. The *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) method [EK96], is adopted which groups together points that lie close by marking as outliers isolated points in low-density regions. The DBSCAN algorithm requires two parameters: a threshold used as the radius of the density region, and a positive integer that represents the minimum number of points required to form a dense region. As feature curves, the output of the DBSCAN algorithm is submitted. To estimate the density of the feature points and, therefore, the minimum number of points in a region, the *K-Nearest Neighbor* (KNN) [FBF77] is used. In general, the K value of the KNN search is set to 15 for MHT1 and to 4 for MHT2.

**Curve approximation using Hough transforms** Finally, the feature curves are fitted with template curves to recognize their type and quantify the parameters characterizing such a feature. This step is obtained following the procedure based on the Hough transform described in [TBF18]. In this context, the following dictionary of curves is considered: circles, Lamet curve, citrus curve, geometric petal and Archimedean spiral, see [TBF18] for details. In general, combinations or additional families of curves are possible [Shi95]. The peculiarity of the Hough

transform is to estimate in a family of curves, the parameters of the curve  $\mathbf{a} = (a_1, \dots, a_n)$  that better fits a given set of points. The curves considered have at most one or two parameters. Depending on the curve, these parameters estimate its bounding box, diagonal, radius, etc.

The distance between the two curves  $\mathcal{C}_1$  and  $\mathcal{C}_2$  is defined as the norm  $L^1$  of the parameters corresponding to these curves, i.e.,  $d(\mathcal{C}_1, \mathcal{C}_2) = |\mathbf{a}_{\mathcal{C}_1}, \mathbf{a}_{\mathcal{C}_2}|_1$ , where  $\mathbf{a}_{\mathcal{C}_1}$  and  $\mathbf{a}_{\mathcal{C}_2}$  are the parameters of the curves  $\mathcal{C}_1$  and  $\mathcal{C}_2$ , respectively. Note that such a notion of distance assumes the curve parameters are homogeneous in terms of the properties measured; this implies that the distance between two feature curves is computed only if they belong to the same family.

## Performance evaluation

Apart from the well-known Hausdorff distance between two sets of points, there is not a standard measure for the evaluation of this kind of task. Also, notice that despite being curves, the ground truth for this contest is defined by sets of points that are not sorted, thus distances for ordered polylines like the Fréchet distance are not suitable [EGHP<sup>+</sup>02]. We used the Direct Hausdorff distance [DD09], the Dice coefficient [TH15] and the Jaccard index [TH15]. More precisely:

- *The Direct Hausdorff distance* from the points  $a \in A \subset \mathbb{R}^3$  to the points  $b \in B \subset \mathbb{R}^3$  is defined as follows:

$$d_{dHaus}(A, B) = \max_{(a \in A)} \min_{(b \in B)} d(a, b),$$

with  $d$  the Euclidean distance. As a reference, the well known *Hausdorff distance* between A and B is

$$\max\{d_{dHaus}(A, B), d_{dHaus}(B, A)\}$$

In order to have coherent evaluation through different models, measurement are done on resized models so that their loads are as close as possible.

- *The Dice coefficient* between two sets (let them be  $S$  and  $R$ ) is defined as

$$dice(S, R) = \frac{2|S \cap R|}{|S| + |R|},$$

where  $|\cdot|$  denotes the number of elements. It ranges from 0 (no match), to 1 (perfect match).

- *The Jaccard index* is defined as:

$$jaccard(S, R) = \frac{dice(S, R)}{2 - dice(S, R)},$$

that varies from 0 to 1, the higher the better.

The Direct Hausdorff distance score is used to evaluate how well the overall shape of the curves extracted by the different methods fit the ground truth and vice-versa. The other two measures are used to evaluate the precision of the methods. While a score of 1 is not mandatory for a method to be considered good, the higher the value is, the better the extracted curve fits its ground truth counterpart. Six runs are evaluated. For the SBSE and MHT\* we had runs for both the feature curve characterization and the feature curve similarity task. The PCs results report a unique feature curve per model so they are considered in the *overall comparison* only. The results for the curve-by-curve classification are reported in Table 5.2. Those for the overall evaluation are splitted into two tables for format reasons and are reported in Table 5.3 and Table 5.4.

If interested in the strongest features (in terms of bending) of a model, SBSE provides a quick overall preview of the related feature curves, quite robust to noise.

SBSE is able to extract the jointed feature curves even in presence of acquisition noise, with good precision, as shown in Figure 5.11(Top)(a). PMCV has impressive precision in its extraction process. Such a precision could be ideal to identify different features that share a jointed feature curve. Figure 5.11(Top)(b) shows how this method is able to separate the L-shaped bumps of the mesh with different feature curves. It usually detects more feature curves than those selected in the ground truth. The main reason is that this method extracts the set of valley and crest lines in the mathematical sense, while the ground truth focuses on a user-specified subset. It may also happen that the curve generation stops at non anisotropic areas such as corners. In that case, a feature line is separated in several curves. The feature lines provided by the PMCV are generally thicker than those in the ground truth. If required, a thinner set of lines can be obtained by reducing the distance used for the curve voting, although representative features could be discarded in this way. About the PCs runs, while they do not separate the feature vertices in different feature curves, they almost always provide a super-set of the vertices of the ground truth jointed feature curves. Also, as shown in Figure 5.11(Top)(c,d), the methods are very precise in case of very sharp features, as those in Models 5 and 9. A good balance between precision and vertex clustering is obtained by MHT, which recognizes most of the expected feature curves, balancing the number of vertices recognized and the curve fragmentation (with respect to our ground truth). An example of this is shown in Figure 5.11(Top)(e,f).

For the feature curve similarity task, SBSE provides a global distance between two models based on histogram-based feature vectors. For example, M4 and M10 are considered similar based on this evaluation, as well as M11 and M12, M7 and M8, M6 and M9. MHT provides instead a similarity measure among the single feature curves, even those in the same model. In other words, it performs a local similarity evaluation of the models. The similarity evaluation is doable with curves that are obtained using the same family of curves. For instance, the eyes in Model 11 and 12 are mutually considered similar, as well as each pair of rings on Model 14. An example of curves sorted by similarity in a single family is shown in Figure 5.11(Bottom).

As a final remark, the feature curve characterization contributes to give an overall interpretation of a model and combines with the pattern recognition problem allowing for the removal of un-

<i>O-comp - <math>d_{dHaus}</math> from GT to Parts</i>						
Model	SBSEPCs	APCs	CPMCVMHT1	MHT2		
M1	0.068	<b>0.054</b>	0.105	0.675	1.570	1.570
M2	0.054	0.060	<b>0.032</b>	0.079	0.071	0.060
M3	0.074	0.006	0.005	<b>0.001</b>	0.048	0.048
M4	3.047	3.694	2.771	<b>0.162</b>	3.555	3.555
M5	<b>0.887</b>	1.019	2.100	0.921	1.019	1.019
M6	1.229	<b>0.033</b>	1.049	2.246	0.650	0.089
M7	0.018	0.010	0.028	0.016	<b>0.012</b>	<b>0.012</b>
M8	0.062	<b>0.006</b>	0.027	0.037	0.011	0.011
M9	1.622	0.165	1.699	1.716	<b>0.081</b>	<b>0.081</b>
M10	2.427	<b>0.003</b>	0.582	2.348	4.585	4.585
M11	0.091	<b>0.009</b>	0.035	0.013	0.045	0.045
M12	0.062	0.036	0.064	0.011	0.040	<b>0.010</b>
M13	0.035	0.057	0.023	0.070	<b>0.013</b>	<b>0.013</b>
M14	0.024	<b>0.004</b>	0.027	<b>0.004</b>	0.010	0.010
M15	<b>0.004</b>	0.009	0.077	0.145	0.030	0.030

<i>O-comp - <math>d_{dHaus}</math> from Parts to GT</i>						
Model	SBSEPCs	APCs	CPMCVMHT1	MHT2		
M1	<b>0.225</b>	5.924	<b>0.225</b>	0.311	0.280	0.280
M2	1.407	0.128	1.643	<b>0.012</b>	0.041	0.020
M3	0.388	<b>0.001</b>	0.184	0.278	0.061	0.061
M4	1.055	0.969	1.055	0.258	<b>0.209</b>	<b>0.209</b>
M5	0.166	0.037	<b>0.029</b>	<b>0.029</b>	0.250	0.250
M6	1.399	1.101	1.101	<b>0.680</b>	1.101	1.276
M7	<b>0.017</b>	0.043	0.031	0.026	0.078	0.078
M8	0.019	0.039	0.028	<b>0.016</b>	0.044	0.044
M9	4.288	0.260	0.496	<b>0.215</b>	1.442	1.442
M10	0.229	0.723	0.422	0.411	<b>0.022</b>	<b>0.022</b>
M11	<b>0.013</b>	0.043	0.015	0.039	0.015	0.015
M12	<b>0.009</b>	0.030	0.012	0.220	0.036	0.038
M13	0.068	0.060	0.091	<b>0.054</b>	0.067	0.067
M14	<b>0.007</b>	0.019	0.013	0.008	0.008	0.008
M15	0.090	0.090	0.051	0.022	<b>0.009</b>	<b>0.009</b>

<i>O-comp - Dice coefficient</i>						
Model	SBSEPCs	APCs	CPMCVMHT1	MHT2		
M1	0.345	0.352	0.354	<b>0.479</b>	0.452	0.452
M2	0.421	<b>0.494</b>	0.475	0.482	0.210	0.213
M3	0.411	0.492	<b>0.508</b>	0.383	0.292	0.292
M4	0.342	0.496	<b>0.513</b>	0.392	0.449	0.449
M5	0.427	<b>0.586</b>	0.582	0.563	0.555	0.555
M6	0.279	0.446	0.467	<b>0.525</b>	0.445	0.451
M7	0.306	0.426	0.508	<b>0.550</b>	0.501	0.501
M8	0.316	0.412	0.498	<b>0.543</b>	0.518	0.518
M9	0.221	<b>0.533</b>	0.502	0.447	0.474	0.474
M10	0.425	0.466	0.498	<b>0.516</b>	0.402	0.402
M11	0.389	<b>0.579</b>	0.554	0.562	0.562	0.562
M12	0.548	0.711	<b>0.727</b>	0.666	0.667	0.637
M13	0.298	0.553	0.537	0.405	<b>0.976</b>	<b>0.976</b>
M14	0.304	0.565	0.517	0.512	<b>0.882</b>	<b>0.882</b>
M15	0.584	0.659	0.631	0.536	<b>0.917</b>	<b>0.917</b>

<i>O-comp - Jaccard index</i>						
Model	SBSEPCs	APCs	CPMCVMHT1	MHT2		
M1	0.209	0.213	0.215	<b>0.315</b>	0.292	0.292
M2	0.266	<b>0.328</b>	0.312	0.318	0.118	0.119
M3	0.259	0.326	<b>0.340</b>	0.237	0.171	0.171
M4	0.207	0.330	<b>0.345</b>	0.244	0.289	0.289
M5	0.271	<b>0.414</b>	0.411	0.392	0.384	0.384
M6	0.162	0.287	0.305	<b>0.356</b>	0.286	0.291
M7	0.181	0.270	<b>0.341</b>	0.379	0.334	0.334
M8	0.188	0.260	0.332	<b>0.372</b>	0.350	0.350
M9	0.124	<b>0.363</b>	0.335	0.288	0.311	0.311
M10	0.270	0.304	0.331	<b>0.347</b>	0.252	0.252
M11	0.241	<b>0.407</b>	0.383	0.391	0.391	0.391
M12	0.378	0.552	<b>0.571</b>	0.500	0.501	0.467
M13	0.175	0.382	0.367	0.254	<b>0.953</b>	<b>0.953</b>
M14	0.179	0.394	0.348	0.344	<b>0.788</b>	<b>0.788</b>
M15	0.412	0.491	0.461	0.366	<b>0.847</b>	<b>0.847</b>

Table 5.2: The evaluation measures of the O-comp classification. The  $d_{dHaus}$  distance measure is computed from groundtruth (GT) to the feature curve proposed by the participants (Parts) and vice-versa. The lower its score is (0 at best), the better. For the Dice coefficient and the Jaccard index, the higher the score is (1 at best), the better. Refer to Figure 5.10 to see which model is evaluated in each cell. Bests results are highlighted with bold font.

wanted parts of the models that causes variations in the geometry of the model comparable to those that define patterns as well.



CbC-comp classification - Part 1																			
$d_{dHaus}$ from GT to Parts				$d_{dHaus}$ from Parts to GT				Dice coefficient				Jaccard index							
FC id.	SBSEPMCV	MHT1	MHT2	FC id.	SBSEPMCV	MHT1	MHT2	Mod	SBSEPMCV	MHT1	MHT2	Mod	SBSEPMCV	MHT1	MHT2				
1.1	0.604	0.014	<b>0.013</b>	<b>0.013</b>	1.1	0.035	<b>0.006</b>	<b>0.006</b>	<b>0.006</b>	1.1	0.121	<b>0.650</b>	0.636	0.636	1.1	0.064	<b>0.482</b>	0.466	0.466
1.2	0.601	0.036	<b>0.034</b>	<b>0.034</b>	1.2	0.016	<b>0.006</b>	<b>0.006</b>	<b>0.006</b>	1.2	0.145	<b>0.450</b>	0.413	0.413	1.2	0.078	<b>0.290</b>	0.260	0.260
1.3	0.575	0.084	<b>0.039</b>	<b>0.039</b>	1.3	0.052	<b>0.021</b>	0.110	0.110	1.3	0.244	0.435	<b>0.503</b>	<b>0.503</b>	1.3	0.139	0.278	<b>0.336</b>	<b>0.336</b>
1.4	n.c.	<b>0.041</b>	0.064	0.064	1.4	n.c.	0.021	<b>0.020</b>	<b>0.020</b>	1.4	n.c.	<b>0.499</b>	0.445	0.445	1.4	n.c.	<b>0.332</b>	0.286	0.286
1.5	0.555	0.056	<b>0.035</b>	<b>0.035</b>	1.5	0.031	<b>0.006</b>	<b>0.006</b>	<b>0.006</b>	1.5	0.180	0.434	<b>0.578</b>	<b>0.578</b>	1.5	0.099	0.277	<b>0.406</b>	<b>0.406</b>
1.6	0.527	0.128	<b>0.050</b>	<b>0.050</b>	1.6	<b>0.057</b>	0.113	0.118	0.118	1.6	0.214	0.458	<b>0.481</b>	<b>0.481</b>	1.6	0.120	0.297	<b>0.316</b>	<b>0.316</b>
1.7	0.541	<b>0.027</b>	0.028	0.028	1.7	0.083	<b>0.020</b>	0.062	0.062	1.7	0.041	<b>0.382</b>	0.338	0.338	1.7	0.021	<b>0.236</b>	0.204	0.204
1.8	n.c.	$\sim 0$	n.c.	n.c.	1.8	n.c.	<b>0.049</b>	n.c.	n.c.	1.8	n.c.	<b>0.447</b>	n.c.	n.c.	1.8	n.c.	<b>0.288</b>	n.c.	n.c.
1.10	0.547	0.035	<b>0.024</b>	<b>0.024</b>	1.10	<b>0.028</b>	0.082	0.081	0.081	1.10	0.126	<b>0.354</b>	0.334	0.334	1.10	0.067	<b>0.215</b>	0.200	0.200
2.1	0.673	<b>0.005</b>	0.674	0.016	2.1	<b>0.011</b>	0.171	0.012	0.012	2.1	0.094	<b>0.296</b>	0.025	0.048	2.1	0.049	<b>0.174</b>	0.012	0.025
2.2	0.467	<b>0.005</b>	0.468	0.013	2.2	0.006	0.081	0.004	<b>0.003</b>	2.2	0.194	<b>0.542</b>	0.147	0.449	2.2	0.108	<b>0.372</b>	0.079	0.289
2.3	0.461	<b>0.013</b>	0.461	0.015	2.3	0.006	<b>0.004</b>	0.007	0.014	2.3	0.164	<b>0.633</b>	0.119	0.332	2.3	0.089	<b>0.463</b>	0.063	0.199
2.4	0.681	<b>0.003</b>	0.681	0.019	2.4	0.005	0.118	<b>0.004</b>	<b>0.004</b>	2.4	0.265	0.457	0.176	<b>0.461</b>	2.4	0.153	0.296	0.096	<b>0.299</b>
2.5	0.686	<b>0.002</b>	0.688	0.019	2.5	0.018	0.155	0.018	<b>0.017</b>	2.5	0.060	<b>0.102</b>	0.033	0.082	2.5	0.031	<b>0.054</b>	0.017	0.043
2.6	0.473	<b>0.002</b>	0.475	0.018	2.6	<b>0.010</b>	0.127	0.017	0.016	2.6	0.077	<b>0.537</b>	0.009	0.029	2.6	0.040	<b>0.367</b>	0.004	0.015
2.7	0.461	<b>0.004</b>	0.462	0.017	2.7	<b>0.015</b>	0.166	<b>0.015</b>	0.017	2.7	0.041	0.076	0.016	0.035	2.7	0.021	<b>0.040</b>	0.008	0.018
2.8	0.475	<b>0.005</b>	0.476	0.022	2.8	<b>0.013</b>	0.146	0.017	0.018	2.8	0.062	<b>0.337</b>	0.018	0.044	2.8	<b>0.032</b>	0.202	0.009	0.023
2.9	0.453	<b>0.002</b>	0.454	0.017	2.9	<b>0.016</b>	0.169	0.017	<b>0.016</b>	2.9	0.029	<b>0.339</b>	0.015	0.019	2.9	0.015	<b>0.204</b>	0.007	0.010
2.10	0.674	<b>0.002</b>	0.674	0.018	2.10	<b>0.015</b>	0.185	0.020	0.018	2.10	<b>0.104</b>	0.032	0.022	0.038	2.10	<b>0.055</b>	0.016	0.011	0.019
2.11	0.686	$\sim 0$	0.686	0.022	2.11	<b>0.011</b>	0.146	0.017	0.017	2.11	0.078	<b>0.248</b>	0.037	0.044	2.11	<b>0.040</b>	0.142	0.019	0.022
2.12	0.683	<b>0.003</b>	0.685	0.016	2.12	<b>0.004</b>	0.131	0.005	0.005	2.12	0.174	<b>0.474</b>	0.107	0.282	2.12	0.095	0.310	0.056	<b>0.164</b>
3.1	n.c.	<b>0.092</b>	n.c.	n.c.	3.1	n.c.	<b>0.172</b>	n.c.	n.c.	3.1	n.c.	<b>0.443</b>	n.c.	n.c.	3.1	n.c.	<b>0.284</b>	n.c.	n.c.
3.2	0.414	<b>0.007</b>	0.062	0.062	3.2	0.035	0.097	<b>0.016</b>	<b>0.016</b>	3.2	<b>0.586</b>	0.487	0.497	0.497	3.2	<b>0.415</b>	0.322	0.331	0.331
4.1	0.048	<b>0.005</b>	0.036	0.036	4.1	<b>0.023</b>	0.313	0.035	0.035	4.1	0.300	0.126	<b>0.426</b>	<b>0.426</b>	4.1	0.176	0.067	<b>0.270</b>	<b>0.270</b>
4.2	0.124	<b>0.027</b>	n.c.	n.c.	4.2	0.092	<b>0.005</b>	n.c.	n.c.	4.2	0.159	<b>0.590</b>	n.c.	n.c.	4.2	0.086	<b>0.419</b>	n.c.	n.c.
5.1	0.021	$\sim 0$	0.102	0.102	5.1	0.887	<b>0.470</b>	1.086	1.086	5.1	0.381	0.304	<b>0.414</b>	<b>0.414</b>	5.1	0.235	0.179	<b>0.261</b>	<b>0.261</b>
6.1	n.c.	<b>0.005</b>	n.c.	0.561	6.1	n.c.	0.079	n.c.	<b>0.008</b>	6.1	n.c.	<b>0.663</b>	n.c.	0.046	6.1	n.c.	<b>0.496</b>	n.c.	0.023
6.2	0.053	<b>0.006</b>	1.021	1.021	6.2	<b>0.060</b>	0.279	1.213	1.213	6.2	0.318	0.357	<b>0.516</b>	<b>0.516</b>	6.2	0.189	0.217	<b>0.348</b>	<b>0.348</b>
7.1	0.012	<b>0.007</b>	0.010	0.010	7.1	0.070	0.220	<b>0.048</b>	<b>0.048</b>	7.1	0.333	0.479	<b>0.528</b>	<b>0.528</b>	7.1	0.199	0.314	<b>0.359</b>	<b>0.359</b>
8.1	0.012	<b>0.007</b>	0.017	0.017	8.1	0.064	0.314	<b>0.046</b>	<b>0.046</b>	8.1	0.313	0.454	<b>0.533</b>	<b>0.533</b>	8.1	0.185	0.294	<b>0.363</b>	<b>0.363</b>
9.1	2.459	<b>0.005</b>	0.016	0.016	9.1	1.638	0.228	<b>0.071</b>	<b>0.071</b>	9.1	0.218	0.270	<b>0.390</b>	<b>0.390</b>	9.1	0.123	0.156	<b>0.242</b>	<b>0.242</b>

Table 5.3: The first part of the evaluation measures of the CbC-comp classification. The  $d_{dHaus}$  distance measure is computed from groundtruth (GT) to the feature curve proposed by the participants (Parts) and vice-versa. The lower its score is (0 at best), the better. For the Dice coefficient and the Jaccard index, the higher the score is (1 at best), the better. Refer to Figure 5.10 to see which model is evaluated in each cell. Model 1.9 and is not reported since no one was able to detect it. Bests results are highlighted with bold font. The value  $\sim 0$  indicates a value lower than 0.001.

## Related publications

- E. Moscoso Thompson, G. Arvanitis, K. Moustakas, N. Hoang-Xuan, E. R. Nguyen, M. Tran, T. Lejemble, L. Barthe, N. Mellado, C. Romanengo, S. Biasotti, and B. Falcidieno, *SHREC'19 track: Feature Curve Extraction on TriangleMeshes*. In Eurographics Workshop on 3D Object Retrieval, 2019.

CbC-comp classification - Part 2																			
$d_{dHaus}$ from GT to Parts				$d_{dHaus}$ from Parts to GT				Dice coefficient				Jaccard index							
FC id.	SBSE	PMCVMHT1	MHT2	FC id.	SBSE	PMCVMHT1	MHT2	Mod.	SBSE	PMCVMHT1	MHT2	Mod.	SBSE	PMCVMHT1	MHT2				
10_1	5.031	<b>0.001</b>	0.019	0.019	10_1	0.108	0.150	<b>0.049</b>	<b>0.049</b>	10_1	0.331	<b>0.394</b>	0.311	0.311	10_1	0.199	<b>0.245</b>	0.184	0.184
10_2	3.192	<b>0.285</b>	n.c.	n.c.	10_2	<b>2.427</b>	13.239	n.c.	n.c.	10_2	0.131	<b>0.433</b>	n.c.	n.c.	10_2	0.070	<b>0.276</b>	n.c.	n.c.
11_1	0.102	0.035	<b>0.033</b>	<b>0.033</b>	11_1	0.007	~ 0	0.005	0.005	11_1	0.497	0.627	<b>0.680</b>	<b>0.680</b>	11_1	0.331	0.457	<b>0.515</b>	<b>0.515</b>
11_2	<b>0.028</b>	0.062	0.048	0.048	11_2	0.018	<b>0.008</b>	0.011	0.011	11_2	0.367	0.393	<b>0.404</b>	<b>0.404</b>	11_2	0.225	0.245	<b>0.253</b>	<b>0.253</b>
11_3	n.c.	<b>0.011</b>	<b>0.011</b>	<b>0.011</b>	11_3	n.c.	<b>0.013</b>	0.015	0.136	11_3	n.c.	0.546	<b>0.553</b>	<b>0.553</b>	11_3	n.c.	0.375	<b>0.382</b>	<b>0.382</b>
11_4	0.171	0.039	<b>0.014</b>	<b>0.014</b>	11_4	0.091	<b>0.009</b>	0.062	0.062	11_4	0.043	<b>0.511</b>	0.489	0.489	11_4	0.022	<b>0.343</b>	0.323	0.323
11_5	n.c.	<b>0.005</b>	<b>0.005</b>	<b>0.005</b>	11_5	n.c.	0.028	<b>0.006</b>	<b>0.006</b>	11_5	n.c.	<b>0.778</b>	0.705	0.705	11_5	n.c.	<b>0.637</b>	0.544	0.544
12_1	0.137	0.083	<b>0.065</b>	0.143	12_1	0.064	<b>0.005</b>	0.040	0.006	12_1	0.190	<b>0.588</b>	0.535	0.230	12_1	0.105	<b>0.416</b>	0.366	0.130
12_2	n.c.	0.006	<b>0.005</b>	0.155	12_2	n.c.	<b>0.008</b>	0.069	<b>0.008</b>	12_2	n.c.	<b>0.817</b>	0.713	0.131	12_2	n.c.	<b>0.690</b>	0.554	0.070
12_3	n.c.	0.023	<b>0.004</b>	0.078	12_3	n.c.	<b>0.011</b>	0.019	0.016	12_3	n.c.	0.137	<b>0.286</b>	0.237	12_3	n.c.	0.074	<b>0.167</b>	0.134
12_4	<b>0.027</b>	0.051	0.029	0.029	12_4	0.006	<b>0.005</b>	0.007	0.007	12_4	0.553	<b>0.572</b>	0.514	0.514	12_4	0.382	<b>0.400</b>	0.346	0.346
12_5	0.082	<b>0.026</b>	0.031	0.097	12_5	0.005	~ 0	0.004	0.006	12_5	0.557	0.766	<b>0.774</b>	0.450	12_5	0.386	0.621	<b>0.631</b>	0.290
13_1	~ 0	0.022	0.067	0.067	13_1	0.035	0.076	<b>0.013</b>	<b>0.013</b>	13_1	0.574	0.567	<b>0.976</b>	<b>0.976</b>	13_1	0.403	0.396	<b>0.953</b>	<b>0.953</b>
14_1	0.716	0.006	~ 0	~ 0	14_1	0.002	~ 0	0.003	0.003	14_1	0.227	0.559	<b>0.979</b>	<b>0.979</b>	14_1	0.128	0.388	<b>0.958</b>	<b>0.958</b>
14_2	0.464	<b>0.006</b>	0.011	0.011	14_2	~ 0	~ 0	0.004	0.004	14_2	0.224	0.529	<b>0.987</b>	<b>0.987</b>	14_2	0.126	0.359	<b>0.974</b>	<b>0.974</b>
14_3	0.464	0.007	~ 0	~ 0	14_3	0.003	~ 0	0.003	0.003	14_3	0.219	0.509	<b>0.978</b>	<b>0.978</b>	14_3	0.123	0.341	<b>0.957</b>	<b>0.957</b>
14_4	0.470	0.007	<b>0.006</b>	<b>0.006</b>	14_4	0.004	~ 0	0.004	0.004	14_4	0.002	0.450	<b>0.942</b>	<b>0.942</b>	14_4	0.001	0.291	<b>0.891</b>	<b>0.891</b>
14_5	0.672	0.008	<b>0.006</b>	<b>0.006</b>	14_5	0.023	<b>0.003</b>	0.004	0.004	14_5	0.001	0.489	<b>0.795</b>	<b>0.795</b>	14_5	< 0.001	0.323	<b>0.660</b>	<b>0.660</b>
14_6	0.654	<b>0.007</b>	0.015	0.015	14_6	0.007	<b>0.004</b>	<b>0.004</b>	<b>0.004</b>	14_6	0.027	<b>0.657</b>	0.646	0.646	14_6	0.013	<b>0.490</b>	0.477	0.477
14_7	0.655	<b>0.007</b>	0.008	0.008	14_7	0.011	~ 0	0.010	0.010	14_7	0.178	0.655	<b>0.739</b>	<b>0.739</b>	14_7	0.098	0.487	<b>0.586</b>	<b>0.586</b>
14_8	0.667	<b>0.006</b>	0.008	0.008	14_8	0.024	~ 0	0.069	0.069	14_8	0.153	0.576	<b>0.667</b>	<b>0.667</b>	14_8	0.083	0.404	<b>0.500</b>	<b>0.500</b>
14_9	0.721	<b>0.007</b>	0.012	0.012	14_9	0.004	~ 0	0.002	0.002	14_9	0.002	0.478	<b>0.897</b>	<b>0.897</b>	14_9	0.001	0.314	<b>0.814</b>	<b>0.814</b>
14_10	0.703	0.007	~ 0	~ 0	14_10	0.004	~ 0	0.006	0.006	14_10	0.227	0.516	<b>0.919</b>	<b>0.919</b>	14_10	0.128	0.348	<b>0.851</b>	<b>0.851</b>
14_11	0.708	0.007	<b>0.003</b>	<b>0.003</b>	14_11	0.004	~ 0	0.004	0.004	14_11	0.004	0.465	<b>0.952</b>	<b>0.952</b>	14_11	0.002	0.303	<b>0.908</b>	<b>0.908</b>
14_12	0.500	<b>0.007</b>	0.012	0.012	14_12	0.004	0.003	<b>0.002</b>	<b>0.002</b>	14_12	< 0.001	0.446	<b>0.956</b>	<b>0.956</b>	14_12	< 0.001	0.287	<b>0.916</b>	<b>0.916</b>
14_13	0.494	<b>0.007</b>	0.011	0.011	14_13	0.004	~ 0	0.004	0.004	14_13	0.226	0.510	<b>0.954</b>	<b>0.954</b>	14_13	0.127	0.343	<b>0.911</b>	<b>0.911</b>
14_14	0.513	0.007	~ 0	~ 0	14_14	0.003	~ 0	0.004	0.004	14_14	0.225	0.465	<b>0.983</b>	<b>0.983</b>	14_14	0.127	0.303	<b>0.966</b>	<b>0.966</b>
14_15	0.519	0.007	<b>0.004</b>	<b>0.004</b>	14_15	0.004	0.003	<b>0.002</b>	<b>0.002</b>	14_15	< 0.001	0.464	<b>0.980</b>	<b>0.980</b>	14_15	< 0.001	0.302	<b>0.961</b>	<b>0.961</b>
14_16	0.470	<b>0.006</b>	0.009	0.009	14_16	<b>0.004</b>	<b>0.004</b>	<b>0.004</b>	<b>0.004</b>	14_16	0.002	0.466	<b>0.907</b>	<b>0.907</b>	14_16	0.001	0.304	<b>0.830</b>	<b>0.830</b>
15_1	0.260	n.c.	<b>0.009</b>	<b>0.009</b>	15_1	<b>0.010</b>	n.c.	0.030	0.030	15_1	0.512	n.c.	<b>0.913</b>	<b>0.913</b>	15_1	0.344	n.c.	<b>0.840</b>	<b>0.840</b>
15_2	0.151	0.022	<b>0.007</b>	<b>0.007</b>	15_2	0.010	~ 0	0.013	0.013	15_2	0.732	0.662	<b>0.921</b>	<b>0.921</b>	15_2	0.577	0.494	<b>0.853</b>	<b>0.853</b>

Table 5.4: The second part of the evaluation measures of the CbC-comp classification. The  $d_{dHaus}$  distance measure is computed from groundtruth (GT) to the feature curve proposed by the participants (Parts) and vice-versa. The lower its score is (0 at best), the better. For the Dice coefficient and the Jaccard index, the higher the score is (1 at best), the better. Refer to Figure 5.10 to see which model is evaluated in each cell. Model 1.9 and is not reported since no one was able to detect it. Bests results are highlighted with bold font. The value  $\sim 0$  indicates a value lower than 0.001.

## 5.4 Retrieval of surface patches with similar geometric reliefs

We built a benchmark for geometric pattern retrieval [MBG<sup>+</sup>20]. Our collection of 3D models is characterized by different classes of reliefs on the models surface. These reliefs represent different kinds of materials, like bark wood or rocks, and structures, like bricks. The peculiarity of the models proposed in this contest is that a realistic geometric pattern (derived from real texture images) is applied to a number of base models, some of which have a non-trivial topology (with handles, tunnels, boundaries, etc.).

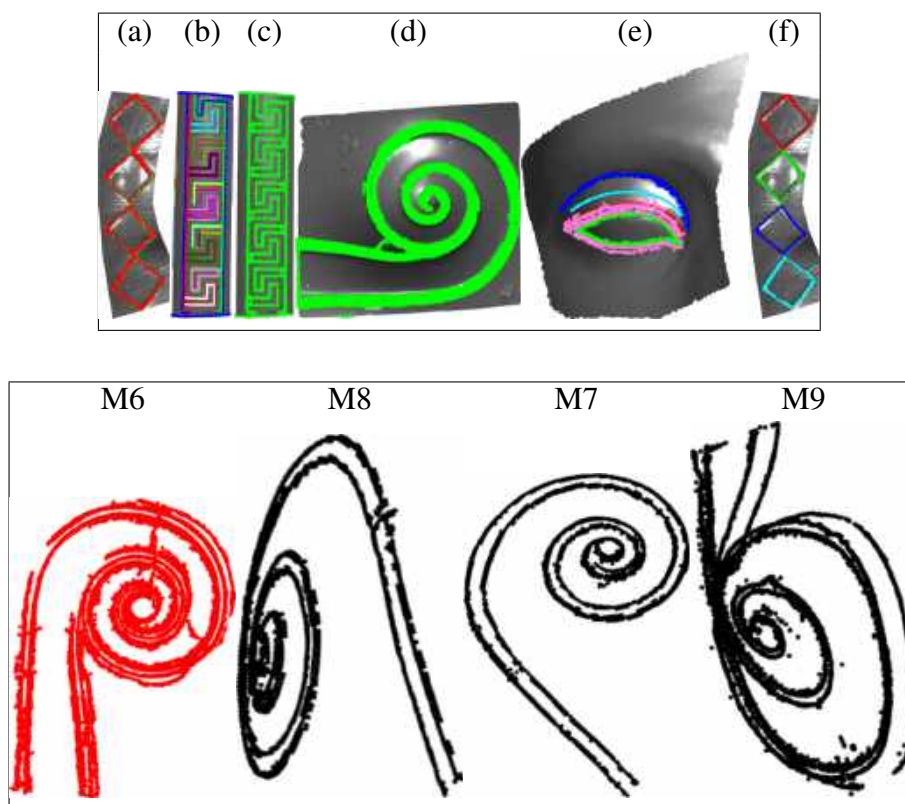


Figure 5.11: Top: An example of the results from SBSE (a), PMCV (b), PCs:A (c), PCs:C (d), MHT1 (e) and MHT2 (f). Bottom: the most similar feature curves to the feature curve extracted on Model 6 (in red) according to the MHT similarity evaluation. From the second images, from left to right, the extracted feature curves on other models from closest to farthest, are shown.

Twenty runs were evaluated in this contest. The performances of the methods reveal that good results are achieved with a number of techniques, both with and without learning approaches.

### 5.4.1 Dataset

The dataset consists of 220 triangulated surfaces. Each one of them is characterized by one of 11 different geometric reliefs.

To create the models, we selected the 20 base models already used in Section 5.2. Then, a set of 11 textures is selected from the free dataset of textures available online from the site Texture Haven [Rob20] that contains a set of natural, high quality texture images made from scanned maps. Most of these textures represent real bricks, floors, roofs surfaces and rock or wood materials. We transform each texture in height values suitable to create a geometric relief by converting



Figure 5.12: An example of the transformation process from texture to height map. For each couple of images: on the left, the original textures are shown. On the right, the final height-map obtained. This process can end with a binary image (just black and white, as in the example at the Top) or a gray-scale one (like that at the Bottom).

each texture into a gray-scale image. The brightness and the contrast values of each image were tuned for each image, based on the values that better enhance the details of the respective color texture. The obtained height field map is applied to the models: initially, the texture is projected onto the target model. Depending on the surface bending, this procedure deforms the texture. To limit this effect, each model is fixed by hand, in particular, in correspondence of significant distortions and parts of the surface with complex geometry (like tight handles). Finally, we raise the vertices of the triangle mesh based on the gray-scale value of the previously processed image along the normal vectors of the models. The same process is repeated for all the textures. A couple of examples of the conversion of a texture into a height map are depicted in Figure 5.12. Finally, the models are slightly smoothed to minimize the perturbations in the color derived from the gray-scale conversion of the textures and the models are sampled with 50000 vertices. Base models, height fields and examples of the final 3D models are shown in Figure 5.13.

## 5.4.2 Results

The challenge proposed in this contest is to group the models only according to the geometric reliefs impressed on them, rather than their shape. In other words, a perfect score is obtained if a method is able to define 11 groups of 20 models each, each group with the models characterized by one of the 11 different geometric reliefs.

A summary of the evaluated method is reported in the following. For additional details, we refer to [MBG<sup>+</sup>20].

1. *Augmented Point Pair Feature Descriptor (APFFD – FK)*. This is a 3D object descriptor made of local features that capture the geometric characteristics or properties of a set of surface patches, each centred at a point, which incorporates the geometrical relation between the latter and its nearest neighbors. Different runs differ in the number and kind

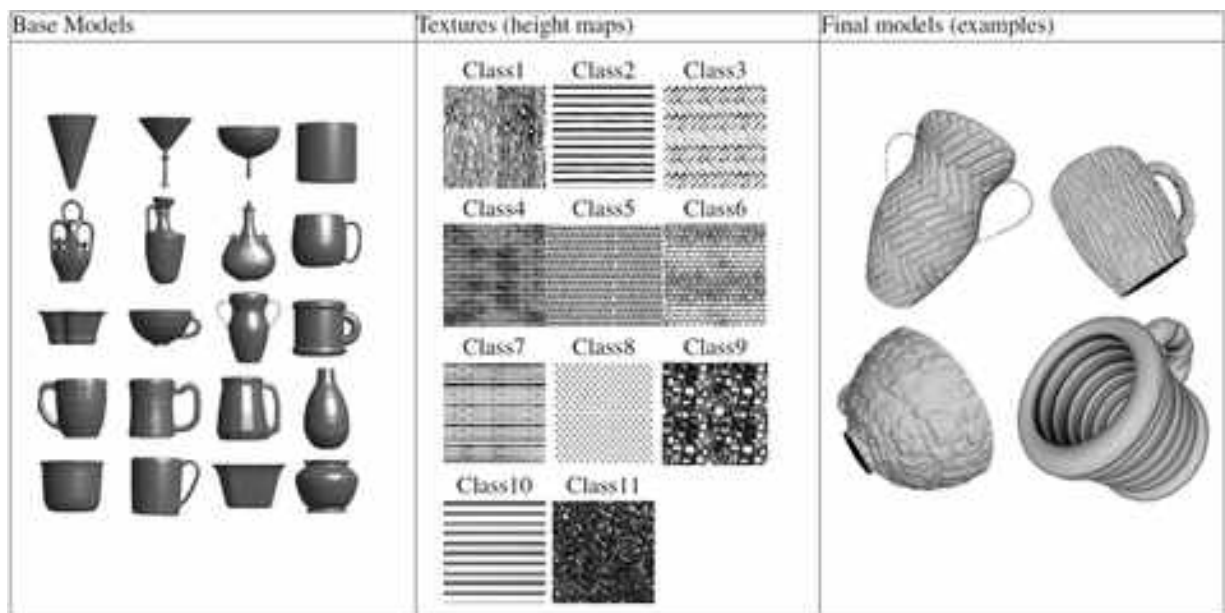


Figure 5.13: (Left): the 20 base models on which the reliefs are applied. (Center): the 11 transformed textures used as height-fields on the base models (the brighter the color, the higher is the value of the field in that point). (Right): a sample of the final models of the dataset of the contest.

of histograms for their feature description.

2. *Orientation Histogram (OH)*. Images are extracted from the models and their gradient is used to compute histograms of modules and angles of the latter. The runs differ from the number of bins of the histogram.
3. *Deep Feature Ensemble (DFE)*. Images are extracted from the models and the output of multiple CNNs are used together to classify the patterns. Runs differ from the kind of DenseNet used. This method is described in Chapter 1.
4. *Deep Patch Metric Learning (DPML)*. It trains a Siamese neural network using images extracted from the models. The runs differ in the way the patches generated are pre-processed and in the use of data augmentation. This method is described in Chapter 1.
5. *Signature Quadratic Form Distance and PointNet (PointNet+SQFD)*. Sipiran et al. proposal consists of computing the distance between two shapes using the Signature Quadratic Form Distance [BUS09] (SQFD) over descriptions of local patches. Runs differ for the number and size of the patches.
6. *Smooth-Rugged Normal Angle (SRNA)*. This method outlines the geometric texture by using a per-vertex quantity and extracts a representative feature vector which is used to test

against every other model in the database.

7. *Variations of the meshLBP (meshLBP)*. It builds a LBP description from the triangles of a mesh on the basis of different local filters and uses an histogram to characterize the patterns. Runs differ for the kind of filter applied. This method is described in Chapter 1.
8. *Correspondence matching based on kd-tree Fast Library for Approximate Nearest Neighbors (kd - treeFLANN)* This method is based on using the kd-tree Fast Library for Approximate Nearest Neighbors (FLANN [FBF77]) correspondence matching to match the query of the other objects in the database. Local features are then matched using the KD Tree FLANN correspondence matching method.

Table 5.5 summarizes the performances of all the twenty runs submitted for evaluation. The best performances for each measure are highlighted in bold. With respect to the results reported in [MBG<sup>+</sup>20], we add the performances of the edgeLBP descriptor (with parameter setting:  $R = 0.11$ ,  $n_{rad} = 5$ ,  $P = 15$ ). Many methods achieve good or very good performances. For example, 6 methods have an NN value above the level of 0.9, i.e. they have a classification rate above 90%. Similarly, the same 6 methods have the mAP value greater than 0.7 and the nDCG greater than 0.8. Also, note that 2 methods have the ST score above 0.99 which, having all the classes 20 models each, means that the models with the same 3D texture as a query are generally found within the first 39 retrieved models, with very few exceptions.

For a better visual comparison of the methods, only the Confusion Matrix and Tier Image of the best run of each method are reported in Figure 5.15 and Figure 5.16 respectively. ROC curves and Precision-Recall plots of the best run for each method are shown in Figure 5.14. As also reflected by the area under the ROC curve, methods with AUC greater than 0.97 provide a better classification than other methods. For completeness, the PR plots and the ROC curves of all the runs submitted are listed in the Appendix. This more complete overview of the runs highlights that the performances of a method show the same trend for the different runs, with small qualitative variations between the different parameter choices.

Overall, the best performances are obtained by the DFE method, which uses a pre-trained neural network. We observe that the NN, FT and ST scores for the methods based on transfer learning do not change significantly. This fact suggests that, if they have success, these methods have a larger capability of ranking the models that contains a texture similar to the query at the beginning of the list of the items retrieved, while the other methods drop around 0.3 from NN to FT. However, also methods that do not use learning techniques perform well (like the meshLBP, OH, SRNA and edgeLBP). We notice that these methods are all based on feature vectors. Some methods share some background, for instance, the meshLBP-so run and the OH methods use of the Sobel Filter. However, among the three meshLBP-based runs submitted to this track, the best performances are reached by the meshLBP run that is based on convolution-like operations extended to a triangle mesh. Among the methods that extend to 3D models the LBP description, the edgeLBP

	NN	FT	ST	mAP	nDCG	e	AUC
APPFD-FK(run1)	0.186	0.204	0.332	0.235	0.523	0.211	0.672
APPFD-FK(run2)	0.132	0.186	0.299	0.212	0.497	0.192	0.632
APPFD-FK(run3)	0.186	0.192	0.318	0.228	0.507	0.203	0.682
OH(run1)	0.791	0.406	0.567	0.470	0.737	0.377	0.817
OH(run2)	0.750	0.374	0.517	0.418	0.709	0.341	0.779
OH(run3)	0.714	0.405	0.575	0.469	0.732	0.382	0.818
DFE(run1)	<b>0.982</b>	<b>0.920</b>	<b>1.000</b>	<b>0.930</b>	<b>0.974</b>	<b>0.715</b>	<b>0.987</b>
DFE(run2)	<b>0.982</b>	0.913	<b>1.000</b>	0.926	0.973	0.714	0.986
DFE(run3)	<b>0.982</b>	0.865	<b>1.000</b>	0.896	0.963	0.693	0.980
DPML(run1)	0.900	0.836	0.990	0.868	0.941	0.686	0.974
DPML(run2)	<b>0.982</b>	0.887	0.992	0.912	0.968	0.690	0.978
PointNet+SQFD(run1)	0.095	0.095	0.184	0.168	0.440	0.113	0.569
PointNet+SQFD(run2)	0.077	0.099	0.203	0.171	0.442	0.122	0.582
PointNet+SQFD(run3)	0.173	0.119	0.225	0.190	0.470	0.137	0.605
SRNA(run1)	0.905	0.493	0.670	0.548	0.802	0.447	0.869
SRNA(run2)	0.923	0.494	0.683	0.563	0.811	0.453	0.882
meshLBP-so	0.909	0.631	0.764	0.687	0.872	0.516	0.870
meshLBP-sh	0.895	0.601	0.759	0.656	0.853	0.522	0.875
meshLBP	0.905	0.671	0.832	0.726	0.884	0.570	0.909
kd-tree FLANN	0.686	0.312	0.424	0.359	0.656	0.283	0.690
edgeLBP	<b>0.982</b>	0.809	0.945	0.855	0.949	0.651	0.959

Table 5.5: Nearest Neighborhood, First Tier, Second Tier, mAP, nDGC, e-measure and AUC value of all the submitter runs.

confirms its superior performances; moreover, the edgeLBP obtains the best scores among the non learning-based methods and, in many measures, is comparable to learning-based methods.

A common characteristic of most methods is the sampling of one or more representative patches as a pre-processing step. It consists of a single patch (like in the case of the DFE, OH, DPML, kd-tree FLANN runs) or multiple ones (like in the APPFD-FK, PointNet+SQDF, SRNA, meshLBP runs). In general, the selection of a single patch seems to lead to good results with the exception of the SRNA and meshLBP methods, which compute a more statistical approach on the representative patches.

Methods that convert the model into point clouds (APPFD-FK) or that are based on CNNs trained on point clouds (PointNet) seem to be sub-optimal for this task. Probably these methods lose information on local details (for instance, the sampling process in the APPFD-FK focuses on the representation of the global geometry) and do not capture the subtle geometry and structure variations of local patterns and reliefs. On a similar note, the authors of the kd-tree FLANN

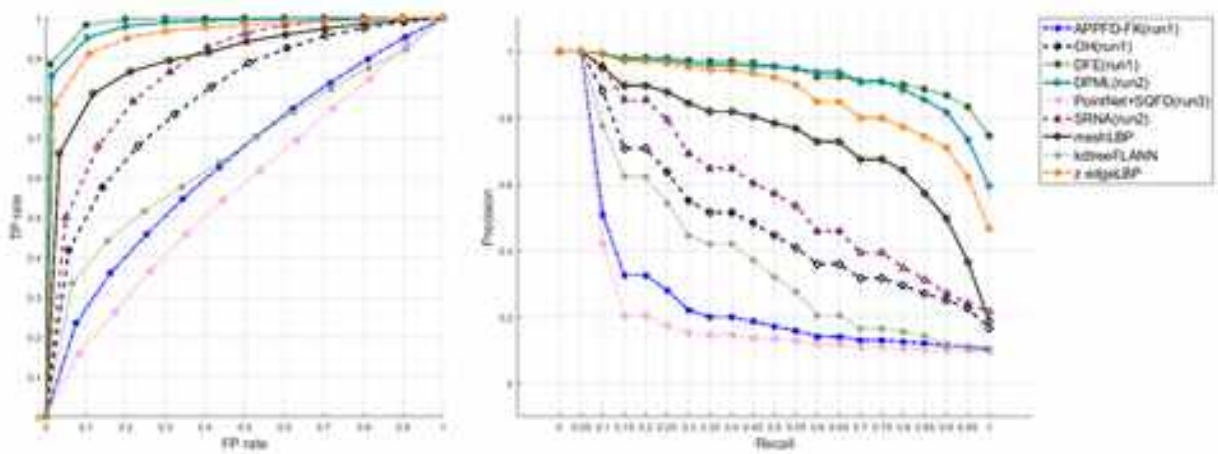


Figure 5.14: Left: Overview of Precision-Recall plots of the best run for each method. Right: overview of ROC curves of the best run for each method.

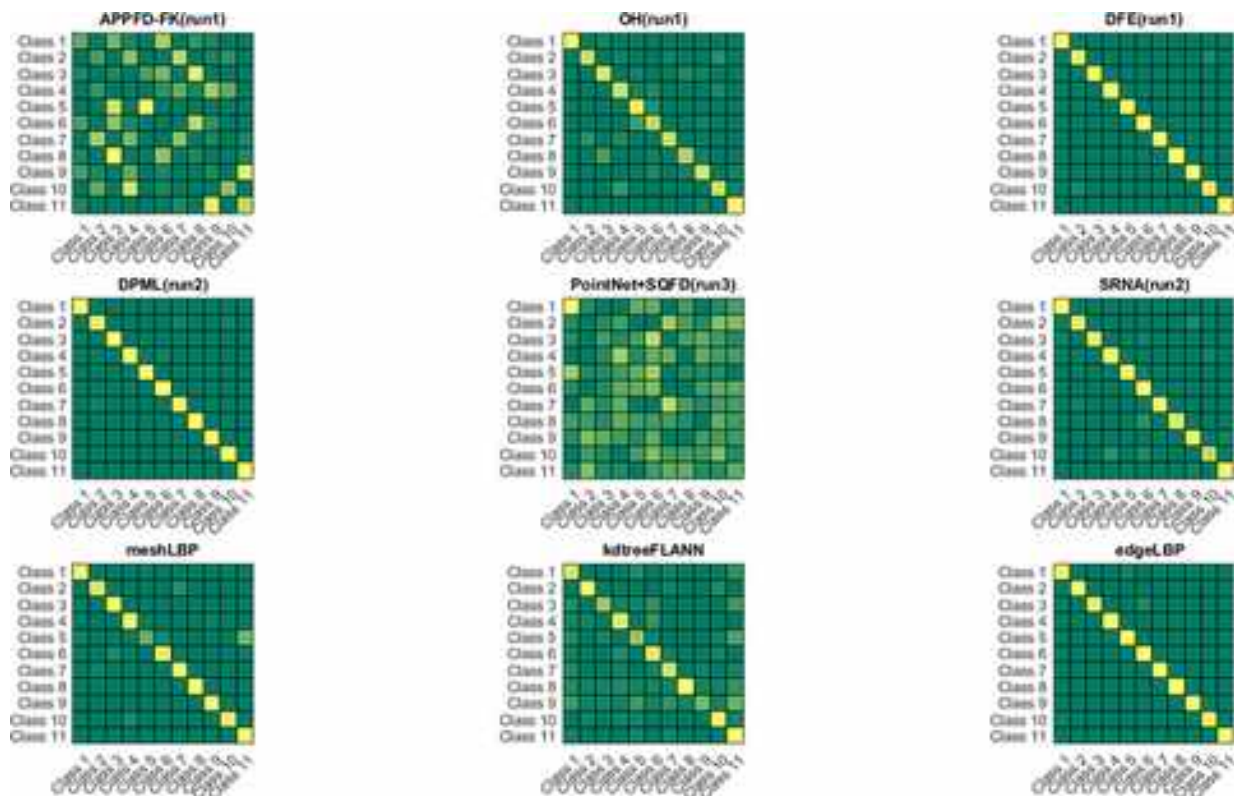


Figure 5.15: Overview of the confusion matrices of the best run for all the methods.

method suggest that the performance of their methods will probably be improved by considering



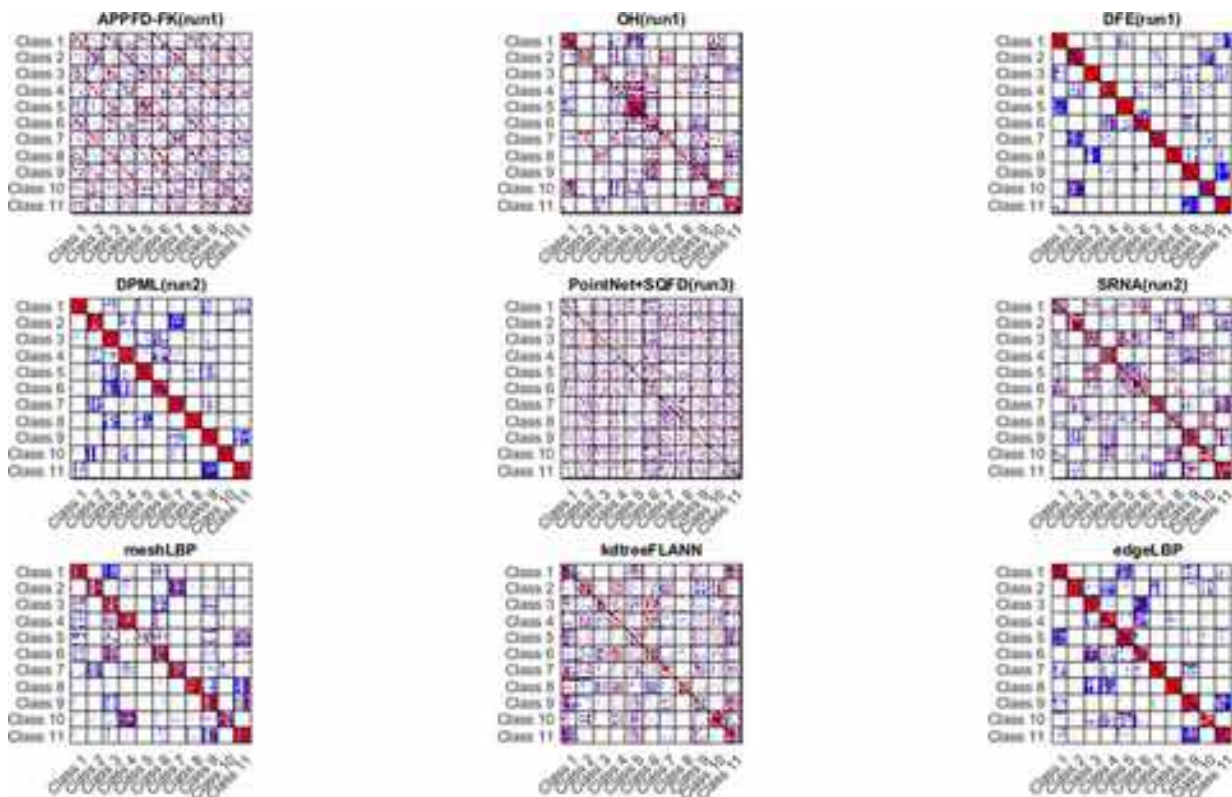


Figure 5.16: Overview of the tier images of the best run for all the methods.

a smaller representative patch. With the current size of the patch, the global geometry of the model is still kept in consideration and it biases the results. This fact highlights the importance of analysing a surface with reliefs by local approaches (but that are robust to noise).

From the Confusion Matrices we observe that no class was impervious to all approaches. On the other hand, Tier Images highlight that some methods (DFE and meshLBP in particular) tend to confuse class 10 (straight horizontal lines with some double lines) and class 2 (just straight lines) or class 4 (bricks). Indeed, all these classes have a set of horizontal and parallel lines which lead to some uncertainty in the classification (especially classes 10 and 2).

## 5.5 River gravel characterization

The quantitative analysis of the distribution of the different types of sands, gravels and cobbles shaping river beds is the task performed by hydrologists to derive useful information on fluvial dynamics and related processes (e.g., hydraulic resistance, sediment transport and erosion, habitat suitability) [CCB16]. It is worth mentioning that traditional tools and methods, based



Figure 5.17: Example details of photographs used for the photogrammetric reconstruction. Top left Class1. Top right: class 3. Bottom left: Class 5. Bottom right: Class 8.

on handling of samples and sieves ([Wol54]), are expensive, time-consuming and bed invasive. Thus, the development of fast and accurate methods able to provide a reasonable estimate of the gravel distribution based on images or 3D scanning data would be extremely useful to support hydrologists in their work. In the following, we describe the benchmark proposed in [GBMT<sup>+</sup>20] to evaluate the suitability of state-of-the-art geometry processing tools to estimate the gravel distribution from digital surface data.

### 5.5.1 Dataset

Surface bed-material sampling produces particle frequency distributions of their sizes that are not directly applicable to those obtained through volumetric sampling, which is the reference method to sediment-transport models. In particular, a finer fraction is more probable to entirely appear on the surface, whereas coarser ones are likely to protrude or to be mostly covered by other grains. The conversion model is still an open issue [MF97]. In this work, we tried to address the first step of a methodology that from surface images finally attains the volumetric grain-size distribution.

The starting point for the realization of our benchmark has been the creation, in the laboratories

of DICAM Department, at the University of Trento, of realistic mock-ups of river beds, built within rectangular wooden boxes  $60 \times 80\text{cm}^2$  and composed of gravels and cobbles with known size ranges and fractions. The aim is the identification of the size of sediment mixture of the grains dwelling on the surface of the bed samples. Attention was paid, in the preparation of the sample, to assure a frequent appearance of any grain-size fraction. At this initial stage, simple bi-modal mixtures were selected. The surfaces, nonetheless, were not levelled by templates, but coarsely arranged by hands, to make the mock-up surface as natural as possible. 8 different mixtures of gravels with size selected in known ranges have been employed:

- Class 1: 3-5 cm
- Class 2: 3-5 cm and 4-6 cm mixed
- Class 3: 3-5 cm and 6-10 cm mixed
- Class 4: 4-6 cm
- Class 5: 4-6 cm and 10-20 cm mixed
- Class 6: 6-10 cm
- Class 7: 6-10 and 10-20 cm mixed
- Class 8: 10-20 cm

These mock-ups have been captured with a digital camera from different viewpoints also capturing the box, which is of known size. Figure 5.17 shows example details of the captured images. 3D models of the mock-ups have been subsequently created using photogrammetric software (Agisoft PhotoScan). Thanks to the box reference, the correct scale factor of the surface has been established. The resulting untextured 3D models have been divided in patches, each one associated to the corresponding grain size composition. The resulting dataset is a set of 256 patches, representing 8 different classes of grain size distributions. Patches are triangulated meshes representing surfaces of about  $25 \times 25$  cm with a single connected component and no holes and about 10K vertices like those shown in Figure 5.18.

Given the 256 surface patches obtained with the previously described procedure, we created a small set with two examples of each class that was available to the participants with the associated ground truth labelling. The remaining 240 patches were distributed as test data in random order with no associated labels.

## 5.5.2 Performances and results

This benchmark consists in 5 methods, together with 2 two baseline approaches.

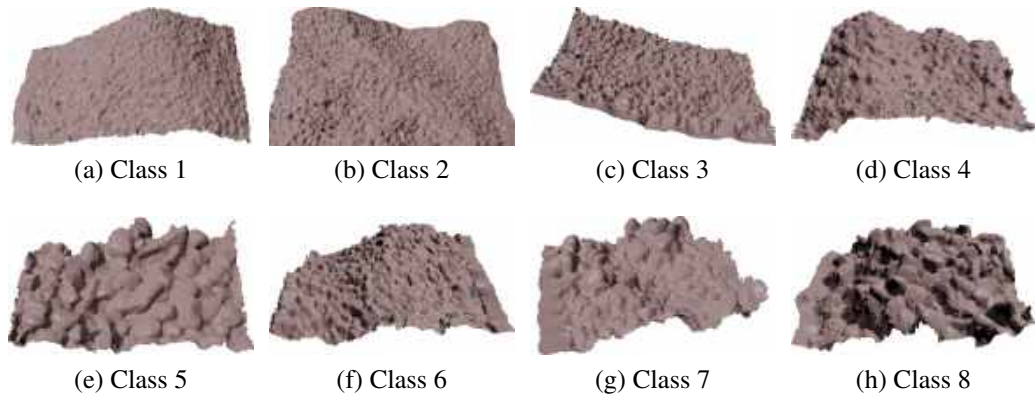


Figure 5.18: Example renderings of surface patches belonging to the 8 classes.

1. *Baseline 1: (Joint) Histograms of Min/Max Curvature (CH, CJH)*. A basic method to detect surface features related with gravel size is to measure curvature values. This method encodes the patterns as histograms of minimum and maximum curvatures at two different scales.
2. *Noise characterization (NC)*. The method tries to categorize each model assuming that the given dataset consists of surfaces which have been affected by different levels of noise, instead of estimating geometric features.
3. *Baseline 2: Mean Point LBP (mpLBP)*. Different runs have different  $n_{rad}$ ,  $P$  and  $R$  values.
4. *Mesh-Local Binary Patterns (meshLBP)*. Runs differ the descriptor shape.
5. *Maximum Inscribed Circle on Depth Image for Gravel Size Estimation (MIC – DI)*. The key idea of this method is to find circles with maximum sizes that can be fit in non-edge areas of the depth image of a gravel patch.

We evaluated the retrieval scores (see Table 5.6) from the dissimilarity matrices submitted by the authors. The method providing the best results is the joint curvature-based benchmark, getting all the highest scores. This may appear surprising, as it is relatively simple, but it should be considered the lack of clear shape priors for the gravels. Methods based on variants of the local binary pattern designed for surface meshes also provided good results. For the MeshLBP method, it is interesting to note that the simple global histogram of the standard version performed significantly better than the Improved Center-Symmetric Binary Pattern. Figure 5.19 shows the precision vs recall plot for all the methods on the whole database. Precision is not too high at intermediate recall values: the effects of the difficult "mixed" classes as well as the irregular depth of the base surface make the task not too trivial.

The difficulty of correctly classifying gravel mixtures is visible in Figure 5.20, displaying the mean average precision estimated on the examples of the different classes by all the methods.

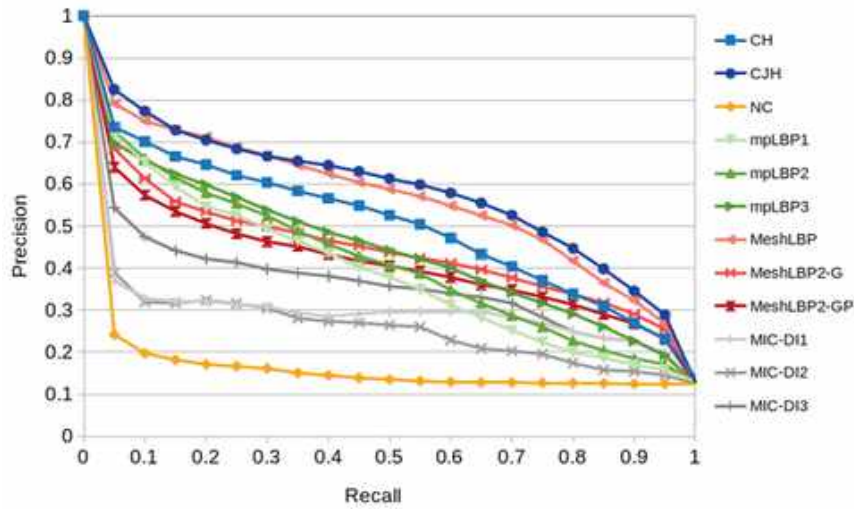


Figure 5.19: Precision-recall plot showing the performances of all the methods proposed on the whole database.

Method	NN	FT	ST	e	DCG	mAP
CH	0.63	0.51	0.74	0.52	0.79	0.53
CJH	<b>0.73</b>	<b>0.58</b>	<b>0.83</b>	<b>0.59</b>	<b>0.84</b>	<b>0.61</b>
NC	0.16	0.15	0.29	0.16	0.53	0.19
mpLBP1	0.59	0.41	0.60	0.41	0.74	0.42
mpLBP2	0.61	0.42	0.63	0.43	0.75	0.44
mpLBP3	0.57	0.45	0.68	0.46	0.76	0.47
MeshLBP	0.68	0.55	0.81	0.56	0.82	0.58
MeshLBP2-G	0.56	0.44	0.72	0.45	0.75	0.46
MeshLBP2-GP	0.53	0.41	0.68	0.42	0.73	0.44
MIC-DI1	0.12	0.29	0.57	0.30	0.62	0.37
MIC-DI2	0.20	0.29	0.50	0.29	0.61	0.31
MIC-DI3	0.45	0.36	0.64	0.38	0.69	0.40

Table 5.6: Retrieval scores obtained with the different baselines and submitted dissimilarity matrices. Joint histograms of curvatures provided the best scores (highlighted in bold)

Higher values are often found for single type classes (1,4,6,8). It is also possible to see that while for the pure classes 1 and 8 the ranking of the method is the same and results are good, for the intermediate mixed and pure classes, the different descriptors present variable rankings.

If we create a pure class retrieval task by considering only the dissimilarity entries of the 4 pure classes, we can evaluate the performances of the different methods in such task using the same

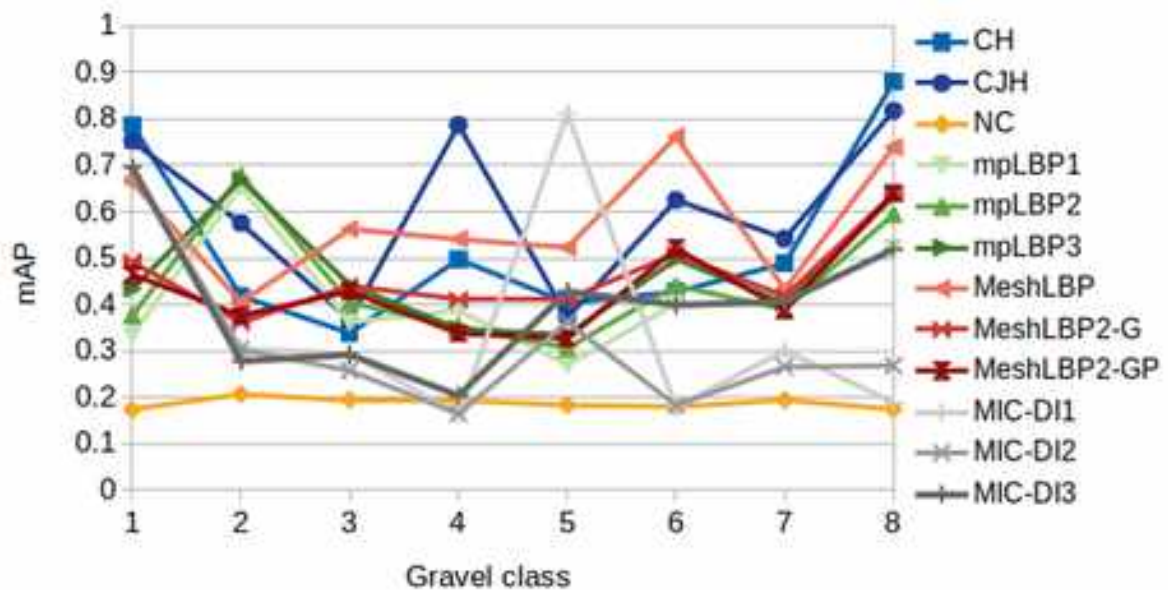


Figure 5.20: Plot showing the mean average precision obtained on each different gravel size class (1-8) with the different methods.

scores (see Table 5.7(Left)) and the precision vs recall plot (Figure 5.21(Right)). As expected the results, in this case, are better, and the ranking of the methods only slightly changed. It is possible to see that the simplest 1D concatenated histograms perform here better than the LBP-based methods and the image-based method is ranked higher than in the global task.

If we create a mixed class only retrieval task by considering only the dissimilarity entries of the 4 classes mixing gravels of different size ranges, we get relatively poor scores (see Table 5.7(Right)) and the precision vs recall plot shows a different ranking of the methods (Figure 5.21(Right)). Simpler surface descriptors and image based techniques seem more sensitive to the increased complexity related to the more complex gravel distributions. More sophisticated methods are probably needed to better capture generic distributions of gravels.

Method	NN	FT	ST	e	DCG	mAP
CH	<b>0.95</b>	0.82	<b>0.98</b>	0.81	0.95	0.84
CJH	0.93	<b>0.83</b>	<b>0.98</b>	<b>0.82</b>	<b>0.96</b>	<b>0.85</b>
NC	0.32	0.26	0.50	0.27	0.63	0.31
mpLBP1	0.83	0.50	0.72	0.50	0.82	0.55
mpLBP2	0.73	0.53	0.76	0.52	0.83	0.59
mpLBP3	0.76	0.57	0.80	0.57	0.84	0.63
MeshLBP	0.93	0.79	0.97	0.78	0.94	0.82
MeshLBP2-G	0.93	0.75	0.95	0.74	0.92	0.78
MeshLBP2-GP	0.90	0.73	0.94	0.72	0.91	0.76
MIC-DI1	0.24	0.61	0.84	0.64	0.76	0.59
MIC-DI2	0.24	0.58	0.79	0.58	0.80	0.61
MIC-DI3	0.73	0.64	0.84	0.65	0.85	0.68

Method	NN	FT	ST	e	DCG	mAP
CH	0.57	0.50	0.84	0.51	0.78	0.52
CJH	<b>0.62</b>	0.53	0.93	0.55	0.80	0.56
NC	0.30	0.27	0.51	0.28	0.64	0.31
mpLBP1	0.59	0.41	0.60	0.41	0.74	0.42
mpLBP2	0.61	0.42	0.63	0.43	0.75	0.44
mpLBP3	0.57	0.45	0.68	0.46	0.76	0.47
MeshLBP	0.60	<b>0.58</b>	<b>0.96</b>	<b>0.59</b>	<b>0.84</b>	<b>0.62</b>
MeshLBP2-G	0.49	0.55	0.95	0.58	0.80	0.59
MeshLBP2-GP	0.51	0.54	0.94	0.56	0.80	0.57
MIC-DI1	0.12	0.29	0.57	0.30	0.62	0.37
MIC-DI2	0.20	0.29	0.50	0.29	0.61	0.31
MIC-DI3	0.45	0.36	0.64	0.38	0.69	0.40

Table 5.7: Left: retrieval scores obtained with the different baselines and submitted dissimilarity matrices by considering only rows and columns corresponding mixed type gravel classes (2,3,5,7). Best scores are highlighted in bold. Right: retrieval scores obtained with the different baselines and submitted dissimilarity matrices by considering only rows and columns corresponding to single range gravels (classes 1,4,6,8).

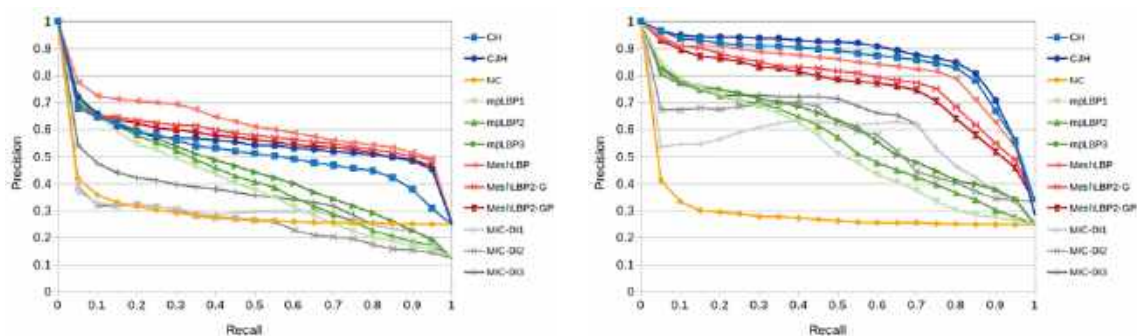


Figure 5.21: Left: precision-recall plots estimated on the dissimilarity sub-matrices corresponding only to single range gravels (classes 1,4,6,8). Right: precision-recall plots estimated on the dissimilarity sub-matrices corresponding only to mixed size gravels (classes 2,3,5,7).

## Conclusions

In this chapter we showcased most of our benchmarking activity, which mainly focuses on pattern retrieval of geometric and colorimetric patterns. These works highlighted the high interest of this problem in the Computer Graphics community. This task is faced with different strategies and, especially in the last years, we observed the rise of machine-learning based approaches.

## Related publications

- E. Moscoso Thompson, C. Tortorici, N. Werghi, S. Berretti, S. Velasco-Forero, S. Biasotti, *SHREC'18: Retrieval of Gray Patterns Depicted on 3D Models*. Eurographics Workshop on 3D Object Retrieval, 2018.
- E. Moscoso Thompson, S. Biasotti, A. Giachetti, et al., *Shrec 2020: Retrieval of digital surfaces with similar geometric reliefs*. Computers & Graphics, 2020.
- A. Giachetti, S. Biasotti, E. Moscoso Thompson, et al., *SHREC 2020 Track: River Gravel Characterization*. Eurographics Workshop on 3D Object Retrieval, 2020.



**Part III**

**Pattern Recognition**

# Chapter 6

## The surface pattern recognition problem

As anticipated in the Introduction of this thesis, the continuation of patterns (decorations either painted or incised, etc.) along the external face of an archaeological fragment is a crucial aspect of the restoration process. When these fragments are stored in different locations or under separate administrations, a physical restoration is almost impossible. Therefore, it is necessary to develop automatic methods that support the digital restoration of fragmented artefacts, making the process faster and more accurate. To reliably accomplish this task, methods must be based on quantitative analysis rather than on visual observation.

In algorithmic terms, this translates in the need of methods and algorithms for feature and pattern recognition on the surface of digital models. Algorithmically, this means to be able to identify if and where a given geometric pattern (represented as a query patch fully characterized by a single pattern) is located over the surface of a 3D model. This is further highlighted by the track of the SHape REtrieval Contest of 2018 [BMTB<sup>+</sup>18] on pattern recognition over 3D models that proposed this exact scenario. The initial response to the contest shows that this is a very lively problem, as 11 groups positively reacted to the contest call. Despite this, at the results delivering deadline no one had a complete output to show (only a membership matrix was submitted, with quite poor performances). However, this contribution exemplifies what the geometric pattern recognition task is, together with its challenges including the problems related to the kind of data used (meshes). In this introductory chapter to the pattern recognition problem, all the aspects as described in [BMTB<sup>+</sup>18] are listed.

### 6.1 The dataset

The 3D models considered in this benchmark come from fragments of archaeological artifacts adopted as use cases in the EU H2020 project GRAVITATE [UUTCIC<sup>+</sup>]. The choice of a real

archaeological case study makes the contest particularly challenging; indeed, the 3D models are laser scan acquisitions of fragments of statues that are degraded and partially abraded,

- *Noise and data acquisition*: being the patterns considered in this contest defined by small geometric variations on the surface, the presence of noise can significantly alter the nature of the pattern. In general, it is really hard to remove noise from a mesh without also affecting the geometric variations that define a pattern. A possible solution could be the definition of noise invariant descriptors, a task which is almost impossible without knowing the nature of the noise and the pattern. Similarly, data acquisition conditions (the resolution of instruments, various sources of imprecision) or incompleteness (presence of occlusions, misalignments, surface specularities) challenge the efficiency of the descriptions and increase the difficulty of the pattern recognition task. Therefore, the understanding and modeling of all the sources of uncertainty and incompleteness is the starting point to improve the existing description methods, the results they produce and their quality.
- *Data size*: some of the models in the dataset are modeled at high definition (more than 6 million vertices), thus dealing with them is computationally demanding. The reason for providing the participants with the highest resolution available as they are stored in the STARC dataset [Ins] was twofold: first, to limit the approximation error due to the algorithm used for the data simplification; second: to provide the best quality of the data currently available. However, the computational complexity is a bottleneck aspect of most methods and calls for the definition of approaches able to take advantage from parallel architectures.
- *Need of a training set*: the track participants agree that an initial training set on which to set the parameters could significantly help. However, in the Cultural Heritage domain, artifacts are often unique and the creation of training sets can be very difficult or limited to specific models. While now this dataset can be used as ground truth for feature testing, it would be almost mandatory to have bigger datasets with a sufficiently high variety of pattern embeddings.

The models are organized into two sets: the Query set (QS) and the Model set (MS).

- *QS*: it contains 8 triangle meshes representing a single pattern. All the patches are surfaces with one boundary, almost flat from a global point of view. The triangulations representing the patches have no fixed number of vertices. These patches were tailored from fragments of the dataset set. Overall, there are 6 possible patterns. The pattern classes are: *eyebrows*, *oblique fringe*, *fringe*, *long incisions*, *spirals*, *stamped circlets*. In particular, there are two different patches for the *stamped circlets* and *oblique fringe* classes. Figure 6.1 illustrates the definition of the patches in QS.









$Q_1$ ( <i>v.r.</i> = 88950)	$Q_2$ ( <i>v.r.</i> = 20250)	$Q_3$ ( <i>v.r.</i> = 147810)	$Q_4$ ( <i>v.r.</i> = 204740)
			
CR	EB	OF	FR
$Q_5$ ( <i>v.r.</i> = 118000)	$Q_6$ ( <i>v.r.</i> = 31640)	$Q_7$ ( <i>v.r.</i> = 124650)	$Q_7$ ( <i>v.r.</i> = 180800)
			
OF	SP	CR	LI

Figure 6.1: Summary table for QD. *v.r.* stands for *vertex resolution*.

- *MS*: it contains 30 triangle meshes, representing 30 different archaeological fragments. Of these, 25 models are characterized by at least one geometric pattern, while the others have no patterns on them. From a geometrical point of view, all the models in *MS* represent a single, closed surface. All triangle meshes are watertight and do not contain self intersections or degenerate triangles. All triangle meshes are provided at the highest resolution available and there is no fixed number of vertices, also called vertex resolution (reported in Figure 6.2). The number of vertices of the meshes in *MS* ranges from 150356 to 6800671. Besides full resolution models, simplified versions with 50K and 100K vertices of the models in *MS* were available; these simplified meshes were generated with the tool [MPS17]. All the models in *MS* are reported in Figure 6.2, with detailed information on which pattern is chiseled on each one of them (if any).

## 6.2 Geometric pattern recognition

Recognizing geometric patterns over surfaces is more complex than simply matching two surfaces. The straightforward extension to 3D models of the techniques adopted for image object detection is not possible because 3D models have peculiar characteristics that require ad-hoc techniques. For instance, the use of meshes instead of grids prevent the adoption of methods that































Pattern inventory: 4 <i>eyebrows</i> (EB), 5 <i>oblique fringe</i> (OF), 3 <i>fringe</i> (FR), 5 <i>long incisions</i> (LI), 6 <i>spirals</i> (SP), 7 <i>stamped circlets</i> (CR)				
1 ( <i>v.r.</i> = 410k)  LI	2 ( <i>v.r.</i> = 1.75M)  -	3 ( <i>v.r.</i> = 1.4M)  -	4 ( <i>v.r.</i> = 4.3M)  OF	5 ( <i>v.r.</i> = 446k)  OF
6 ( <i>v.r.</i> = 414k)  CR	7 ( <i>v.r.</i> = 385k)  FR	8 ( <i>v.r.</i> = 303k)  CR	9 ( <i>v.r.</i> = 492k)  SC EB	10 ( <i>v.r.</i> = 946k)  OF
11 ( <i>v.r.</i> = 152k)  CR	12 ( <i>v.r.</i> = 2.3M)  CR	13 ( <i>v.r.</i> = 536k)  SP	14 ( <i>v.r.</i> = 1M)  LI	15 ( <i>v.r.</i> = 2M)  -
16 ( <i>v.r.</i> = 5M)  LI CR EB	17 ( <i>v.r.</i> = 491K)  LI	18 ( <i>v.r.</i> = 1.4M)  -	19 ( <i>v.r.</i> = 897k)  CR	20 ( <i>v.r.</i> = 172k)  SP
21 ( <i>v.r.</i> = 1.3M)  -	22 ( <i>v.r.</i> = 1.6M)  SP	23 ( <i>v.r.</i> = 150k)  SP	24 ( <i>v.r.</i> = 6.8M)  EB LI CR	25 ( <i>v.r.</i> = 500k)  EB
26 ( <i>v.r.</i> = 622k)  SP	27 ( <i>v.r.</i> = 771k)  LI	28 ( <i>v.r.</i> = 1.5M)  OF	29 ( <i>v.r.</i> = 1M)  FR	30 ( <i>v.r.</i> = 2.4M)  FR

Figure 6.2: Summary table for MD. *v.r.* stands for *vertex resolution* (approximated).

take advantage of the regular, grid structure of the images. The following is a list of issues that deal with geometric pattern recognition.

- *Type of representation*: while for images the grid structure is unique, predictable and regular, this is no longer true for boundary representations of 3D objects, like mesh tessellations, point clouds, and so on. Moreover, two representations of the same object are not unique and can be really different from each other (number of vertices, vertex distribution, etc.). Also, model acquisitions can be affected by noise and/or errors. Not having an ideal, exact reference template both for patterns and models increases the difficulty of the task.
- *Pattern definition and size*: the concept of pattern is quite vague and is necessary to distinguish patterns from local features. Patterns contain a repeated configuration of some surface property over the surface while a feature is a local change on the surfaces without a repetition rule. With reference to the model of a statue head like that in Figure 6.2(12), the stamped circles on the helmet delimit a region with a geometric pattern, while the eyes, the mouth, the ears, etc., represent features chiseled on the model. This definition of pattern, despite being intuitive, is not formal. In particular, the size of a pattern is a crucial point to be identified because it cannot be easily generalized to every model and type of pattern and it is necessary to distinguish what is shape and what is decoration. Methods in the literature for local feature characterization that tackle the problem of considering local regions instead of a point wise characterization (e.g. [SPS16]) strongly depend on the size of the area considered. Moreover, the same model could present patterns of different sizes. This means that the pattern size cannot be uniquely defined and needs to be tuned according to the type of query given in input, calling for adaptive approaches.
- *Local-global nature of the characterization*: Similarly, while it is common to think that the characterization of the object belongs to the local or global description, this task seems to fall in both fields, without belonging to either of those entirely. Most methods that attempt to perform such a task extend or specialize techniques initially born for local or global description. This new point of view, which is really hard to grasp, is still unexplored and could be the key to define effective pattern recognition techniques
- *Surface embedding*: the same pattern can be chiseled on surfaces with different manifold embeddings, for instance a helmet and a cuirass. Differently from images that can be considered flat everywhere, surfaces may present a very general bending in the space. This fact also implies that planar projections may introduce significant distortions of the geometric pattern over the surface. Unless the problem clearly states that only a limited number of configurations is admissible, it is not possible to specialize the pattern recognition problem as a surface fitting or registration problem. In general, an effective description of the pattern must be able to distinguish between the variation of the surface decorations from the overall variation of the surface they lay in.

## 6.3 Open challenges

Although only one, partial solution was submitted, several solutions to the surface pattern recognition problem were considered, ranging from statistical, multi-scale characterization [MGB<sup>+</sup>12] to divide-and-conquer techniques aimed at isolating the sub-regions with a uniform pattern and then applying retrieval techniques to the single components.

Noise, data incompleteness, variability of the patterns size/scale, variation of the surface embedding, necessity of training, computational complexity are all crucial aspects to be considered when dealing with geometric pattern recognition.

In this contest various open issues came to light and are listed below. For each issue, we briefly mention the solutions we have adopted in the next chapters.

1. *Noise*. Typical ways to address noised data include: i) local, patch fitting methods with parametric surfaces and ii) mesh denoising strategies (e.g., [Tau95b, HP04]) prior to descriptor extraction, even if these methods are unable to differentiate between geometric features and noise. One possible solution may be an iterative denoising scheme (e.g., [AsLMF18]) that should automatically estimate noise level and geometric feature subspace size (statistical characteristics) or data-driven approaches for mesh denoising (e.g., [WLT16]) that, however, are limited to cases in which all data exhibit similar type of noise. In the method proposed in Chapter 7 we face this issue by considering the local shape descriptor of the mpLBP, whose robustness to noise has been discussed in Chapter 4. In Chapter 8 we deal with this issue by pre-processing and augmenting the data used as the training set.
2. *Data incompleteness*. Building local geometric pattern descriptors that are robust to missing data is challenging. Approaches following the principles of texture inpainting, for instance extracting the shape description on the basis of the surface normal distributions, could be the basis for potential solutions for this problem. In our approaches, we propose methods that do not require to consider the models in their entirety: they focus instead on patches derived from the model that can be considered independently.
3. *Multi-scale characterization*. Extending single point characterization approaches to a regional level is not straightforward. Volumetric or multi-ring characterizations of the model are worth to be considered [MPS<sup>+</sup>04, GMGP05]. To extend point-wise descriptors at a more global level, a top-down analysis procedure could be an appropriate strategy. High-level primitives could be computed from local descriptors so that the retrieval problem is more simple and intuitive. For instance, this approach has been used for symmetry detection using lines [ASC11] and one can imagine comparing models and queries based on such a line feature. In the methods proposed in Chapter 7 and Chapter 8, we locally

analyze the surface with a neighbor of a radius comparable to the size of the patterns of interest. Moreover, we focus on approaches able to ignore the underlying surface curvature, even subtracting the underlying ground.

4. *Reducing the problem to image pattern recognition.* A possible solution is to adopt a local parametrization/projection of the pattern on a flat surface and evaluate the texture descriptors over that projection. This strategy should provide a method not depending on the quality of point cloud/meshes and the possibility of exploiting the wide amount of texture analysis methods in the literature. Apart from the computational complexity and the possible pattern distortion introduced with this procedure, that might be high depending on the parametrization choice and the density of the patch sampling, the biggest issue of the method is related to the metric used to compare the patch descriptors. In Chapter 8, we propose a local description of the surface that converts a vertex neighborhood into an image.
5. *Definition of proper training sets.* A texture descriptor is typically a vector with a huge dimensionality and in order to derive a way to localize patterns in a model it is necessary to be able to characterize the pattern versus the other potential "background" patterns that can be found in the models of interest. This cannot be done using just *positive* examples, but requires also the *negative* ones. However, 3D data have a lot of information that can be extracted, for example, by taking different local views of the models. In Chapter 8 we build a large dataset by taking multiple images from a single sample of the model, tackling this issue with a multi-class approach.
6. *Learning.* In image processing, recognition tasks, e.g. finding and classifying objects inside images, are usually accomplished working on large databases of annotated images [EEVG<sup>+</sup>15] with localization and labeling of objects from which algorithms can learn characterizations of both objects and background. The solution in this case can rely on convolutional neural networks, with methods not directly exploitable in mesh processing, even if there are recent works applying CNN-like approaches to the mesh domain [MBBV15, BBL<sup>+</sup>17]. For instance, deep learning techniques have recently given remarkable results in shape segmentation and classification [QYSG17]. Although this contest does not contain a large dataset, it could be interesting to see how deep neural networks can be trained on these only 8 queries, still made of hundreds of thousands of vertices. In Chapter 7 and Chapter 8, we tested the ability of our proposed method to characterize the local information of the models with respect to these 8 queries. This is deepened in the respective Chapter, however the results suggest that, for such a challenging task, the queries size may be too small (i.e.: the complexity of the pattern requires larger and more varied query patches).



## Conclusions

In this Chapter we detailed the main challenges of dealing with the problem of pattern recognition. This list contains the obstacles found by 11 groups of research that faced this problem during a SHREC track.

## Related publications

- S. Biasotti, E. Moscoso Thompson et al., *SHREC'18 track: Recognition of geometric patterns over 3D models*. In Eurographics Workshop on 3D Object Retrieval, 2018.

## Chapter 7

# Patch Characterization via Energy Optimization and Local Similarity

This chapter proposes a novel approach to the pattern recognition problem, based on energy optimization and samples of the surface in peculiar points selected according to their local geometry. Then, a punctual descriptor is computed on such points. The set of these descriptors is used as the basis for a energy minimization method called *graph-cut*, that balances the need of having similar descriptors in the same class and the requirement of having continuous segments on the surfaces. The graph-cut results naturally induce a clustering of the query points (based on the label computed by the graph-cut), which induces a segmentation of the surface vertices (labeling each vertices with that of the closest query point). Indeed, the goal is to segment the surface in patches characterized by none or one pattern. Each segment then is described using a pattern retrieval method, which helps identifying and locating a pattern on a surface. Moreover, to adapt our analysis on the patterns of interest, we assume the existence of a sphere of radius  $r$  that is large enough to include a significant portion of the patterns themselves. By ‘include’, we mean that the part of the surface with that detail can be enclosed in a sphere of radius  $r$  centered in a the query point placed on the pattern. For example, if the pattern is made by small circlets, the radius  $r$  must be large enough to include the whole circle, while, if the pattern is made by a more complex detail, the sphere must enclose most of it.

We call this method *PATCH driven similarity via graph-CUT* (PATCH-CUT for short) and it works in 4 steps (shown in Figure 7.1).

Sections 7.2 to 7.5 detail these steps, while Section 7.6 showcases examples of the application of the method on the *model set* described in Chapter 6. Before entering the complete explanation of the PATCH-CUT, in Section 7.1 we briefly overview the graph-cut method.

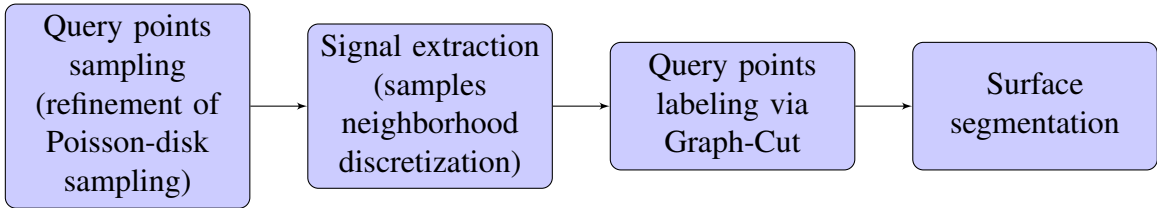


Figure 7.1: Pipeline of the PATCH-CUT method.

## 7.1 Preliminaries on the Graph-cut definition and its application to 3D models

Intuitively, the graph-cut is an efficient graph-based technique aimed at segmenting a graph in 2 or more parts, keeping a ‘smooth segmentation’. More formally, let us have a graph with a set of nodes and arcs, some observed data attached to each node and a finite set of labels  $L$  based on the totality of the nodes data. The goal is to find a labeling  $f$  that assigns a label  $f_p \in L$  to each node so that  $f$  results both piecewise smooth and consistent with the observed data.

Initially the graph-cut was developed for images and eventually it has been used to tackle problems like image segmentation, object co-segmentation, and other problems that can be formulated in terms of energy minimization [YM12]. In this scenario, the graph is the one in which the nodes are the pixels of the image and two nodes are connected if they are in their respective 8-neighborhoods. The graph-cut is formalized in [BVZ01, KZ04, BK04]; in the following we recap the main aspects of this tool. We slightly change the notation with respect to the cited works for an easier understanding of the use we do of this method in our work.

The minimization problem at the core of the graph-cut is  $\min_{f \in F} E(f)$ , where  $E(f)$  is defined as follows:

$$E(f) = \sum_{p \in P} dc(p, f_p) + \sum_{p, q \in N} sc(f_p, f_q), \quad (7.1)$$

with the following notation.

- $P$  is the set nodes to be labeled.
- $F$  is the set of all the possible labeling setups of the nodes  $f$  for the nodes in  $P$ .
- $dc$  is called the *data cost* and it is lower the better  $p$  fits the label  $f_p$ .
- $sc$  is called the *smoothness cost* and it is the term in charge of keeping  $f$  smooth. In other words, it penalizes two neighbors labels  $f_p$  and  $f_q$  if they are too different.
- $N$  is the set of interacting pairs of nodes. Typically, in images, this set consists of adjacent pixels, but it can be arbitrary.

$sc$  has three constraints (where  $\alpha, \beta$  and  $\gamma$  are labels in  $L$ ):

1.  $sc(\alpha, \beta) = 0 \iff \alpha = \beta$ , or  $sc(\alpha, \beta) \neq 0 \iff \alpha \neq \beta$ ,
2.  $sc(\alpha, \beta) = sc(\beta, \alpha) \leq 0$ ,
3.  $sc(\alpha, \beta) \leq sc(\alpha, \gamma) + sc(\gamma, \beta)$ .

There are mainly two algorithms that are used to solve this problem. If these conditions are verified,  $sc$  is a metric and it is possible to use the so-called *alpha-expansion* algorithm. The main idea behind the alpha-expansion algorithm is to successively segment the nodes dividing them in those labeled with a given label and those that are not. At each iteration, the label changes. Its main advantage is that it is possible to guarantee that the local minimum is within a known factor.

However, the triangular inequality is not always verified. In this scenario, the *alpha-beta swap* algorithm is used. The main idea is to iteratively segment the nodes labeled a given label  $\alpha$  with respect to those of another label  $\beta$ . The two given labels change after each iteration, scouting all the possible combinations. This algorithm is detailed in [BVZ01].

It is worth mentioning that it is possible to add a weight to the arcs that connects two nodes. This weight influences  $sc$ , which lead the second term to be re-written as follows:

$$\sum_{p,q \in N} sc(f_p, f_q) ss(p, q), \quad (7.2)$$

where  $ss(p, q)$  correspond to the weight shared by the nodes  $p$  and  $q$ . The set of these weights, called *sparse smoothness* serves a second purpose, other than influencing  $sc$ . Indeed, the structure of the graph can be encoded in this last component, since by setting  $ss(p, q) = 0$  (for two fixed  $p$  and  $q$ ) the smoothness cost estimation does not count the label differences between that of  $p$  and  $q$ . In a way, we removed the link  $p - q$  from the  $N$ .

This is relevant when extending the graph-cut from images to other data in which the proximity of the nodes is not trivial. Indeed, by setting  $N$  equal to the set of nodes, one can dictate the connections between nodes just by setting  $ss$  equal to 0 for unconnected nodes (and vice-versa). In our use-case, since we are using the graph-cut on points in the 3D Euclidean space, we use this trick to efficiently define the points neighborhood.

Finally, it is worth mentioning that there are already contributions towards the extension of graph-cut to 3D data, mainly aimed at segmentation, reconstruction and generation. For example, in [LSZU15] the graph-cut is extended to triangulations and used to segment the models. Data-cost is based on the ratio between geodesic distance and euclidean space of the nodes, while smooth cost is defined by the neighborhood information of faces.

## 7.2 Query point sampling

We sample the surface with points that we characterize based on their neighborhood. It is possible to identify the set samples as the set the whole set of vertices. However, given the resolution of the mesh, this option leads to an unnecessarily high computational cost. Thus, we prefer to start from a simplified set of samples, using the Poisson-disk sampling [CCS12]. Briefly, this sampling technique produces a random and evenly distributed set of samples over a 3D mesh.

The goal of this step is to sample a number of points on the surface that will represent it locally. This is done in two phases. First, a Poisson-disk sampling method is applied to the vertices of the model. Such a technique creates a set of random samples uniformly distributed on the surface. However, we look for samples that fell in pits of the local geometry (i.e., the patterns). Thus, for each sampled point, we consider its vertex neighbors and we slightly shift the position of the sampled point toward the vertex neighbor with the lowest height.

In more details, let us consider the set  $V$  of vertices of a mesh or a point cloud. The Poisson-disk sampling algorithm is used to sample a set  $Q^1 = [q_i^1, \dots, q_{n_q}^1]$  of  $n_q$  points. For each  $q_i^1$ , we consider its neighborhood  $\mathbf{N}(q_i^1)$  in  $V$ , made by points whose Euclidean distance from  $q_i^1$  is at most  $r_q$ . We then look for the best plane of approximation  $\lambda$  of the points in  $\mathbf{N}(q_i^1)$  and we rotate the neighborhood so that the normal vector of  $\lambda$  corresponds to  $[0 \ 0 \ 1]$  and its direction agrees with that of the normals of the  $q_i^1$ . We then replace  $q_i^1$  with the point  $q_i^2 \in \mathbf{N}(q_i^1)$  whose value of the third component is minimum. If the minimum is not unique, we randomly choose one of the points corresponding to the minimum. Once we have iterated this process on all points, we have the new set  $Q^2$ . Note that if a point alternates 2 (or more) fixed positions in these iterations, all these recurrent positions are considered as samples and the point is no longer shifted during the iteration process. This process repeats until all the points are fixed (i.e., when  $Q^t = Q^{t+1}$ ). We track the point evolution and stop to shift the points when all the possible positions (i.e.: all the points of the model) have been visited, In the practice, to limit the computational cost, we prefer to adopt a greedy stop strategy and we end the shifting algorithm if a fixed maximum number of iterations  $t$  is reached. We call the points in the final set  $Q^t$  the *query points*. For brevity, we use the following notation:  $Q^t = Q = [q_1, \dots, q_{n_t}]$ . Duplicate points are removed. Notice that this means that, usually, the final number of query points in  $Q$  is lower than the number of elements of  $Q^1$ .

The parameters that characterize the selection of the query points are listed in the following:

- $n_q$ : the number of samples is based on the area of the surface we are working on. For our tests on archaeological fragments, we use 500 points per  $cm^2$ . Other values are possible and small variations of  $n_q$  do not significantly affect the results.
- $r_q$ : the radius of the neighborhood of a point  $q$  is set as the half the radius  $r$ , so that the maximum shift is correlated to the level detail we are considering.



Figure 7.2: The neighborhood of a query point before and after the flattening based on the quadric approximation of the surface. Both representations point of view is perpendicular to the normal of  $\lambda$ . Left: the neighborhood of points (in blue) and the estimated quadric. Right: the resulting flattened neighborhood.

- $t$ : the maximum number of shift iterations is set to 10 in all our tests. We observed that usually these are enough to set the samples in their final position.

### 7.3 Signals extraction

A punctual descriptor similar to that introduced in chapter 4 is computed for each sample in  $Q$  to encode the local geometric variation around a sample point. The parameters  $n_r$  and  $multP$  are fixed on the basis of the type of the patterns of interest. We use the height-field value as the surface property. In particular, the height map is computed as follows:

1. the neighbor of the a query point  $q_i$  of radius  $r$  is computed and aligned to the plane  $\lambda = xy$  plane aligning to it the best fitting plan of the neighbor of  $q_i$ . Let us call  $N_\lambda(q_i)$  this set of points.
2. Then, we estimate the quadric that best fits  $N_\lambda(q_i)$ .
3. we remove the bending of the underlying surface, in order give more importance to the geometric variation of the pattern. This is obtained by subtracting to the z-coordinates of the points in  $N_\lambda(q_i)$  the z-coordinates of their projection on the fitting quadric. We call this new set of points  $N'_\lambda(q_i)$ . The idea behind this step is visualized in Figure 7.2.

The punctual descriptor is computed on  $N'_\lambda(q_i)$ , using the z-coordinate values of the points as the surface property. We call the computed punctual descriptors *signals*  $\mathbf{s}_i \in S_M$ , where  $S_M$  in the space of the signals (equal to  $\mathbb{R}^{\sum_i P_i}$ ) of the model  $M$ . Finally values of each sectors  $\mathbf{s}_i(j)$  of a signal  $\mathbf{s}_i$  are standardized using the following formula (with a little abuse of notation):

$$\mathbf{s}_i(j) = \frac{\mathbf{s}_i(j) - \text{median}(\mathbf{s}_i)}{\sigma(\mathbf{s}_i)} \quad (7.3)$$

where  $\text{median}(\mathbf{s}_i)$  is the median and  $\sigma(\mathbf{s}_i)$  is the standard deviation of the values of the signal. This standardization is commonly adopted to match features with different units and scales.

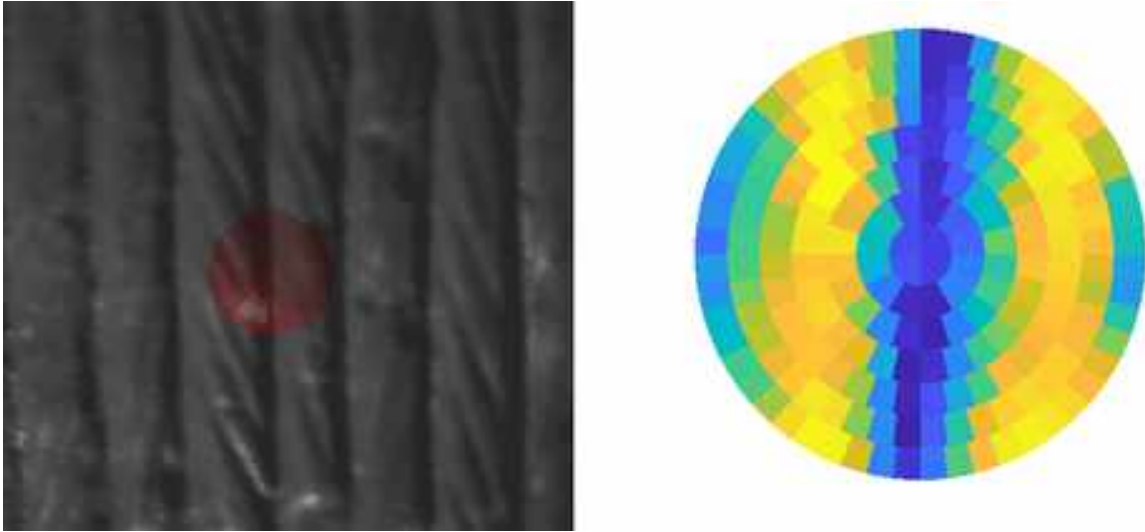


Figure 7.3: Left: an example of a point neighborhood. Right: the signals obtained from the patch to the left.

Since patterns of interest may be of different scales, this standardization helps in having a more complete description of the mesh.

An example of a signal is shown in Figure 7.3.

During the next steps of the PATCH-CUT method, we need to assess similarity between two signals. Despite being sampled by the same model, signals extracted from similar patches are not necessarily oriented in the same way (e.g.: a flat square surface with a pattern of straight lines, horizontally oriented in the left part of the model and vertically oriented in the right part). Thus, a direct comparison using distances such as the  $L^2$  are not optimal for assessing the similarity between signals. The best matching for two signals can be found by computing all the possible rotations of a patch until the patches are properly aligned. This is very demanding in terms of computational cost and we describe how we compare two signals without checking all the possible rotations of the respective patches.

**Signal comparison** Let us assume to have two signals  $s$  and  $q$ . First, among the rings of both  $s$  and  $q$ , we consider a ring close to the middle of the radii of  $s_i$ . Let us call these vectors  $\mathbf{m}_s$  and  $\mathbf{m}_q$  respectively. If we keep  $\mathbf{m}_q$  fixed, we can find the best rotation of  $\mathbf{m}_s$  (by rotating it each time by the angle implied by the arc of the sector) that fits  $\mathbf{m}_q$ . Matches between the rotated vectors are done using the  $L^2$  norm. For example, if our punctual descriptor has 7 rings and  $multP = 4$ , we consider the 4th ring for both  $q$  and  $s$ . Since in this specific case  $\mathbf{m}_q$  and  $\mathbf{m}_s$  have both 20 sectors, we compute the norm of the difference between  $\mathbf{m}_q$  and 20 different rotations of  $\theta = 2\pi/20$  of the signal  $\mathbf{m}_s$ . The rotation of  $\mathbf{m}_s$  that best fits  $\mathbf{m}_q$  defines a rotation

for  $N_\lambda(\mathbf{q})$  that better fits the signal  $\mathbf{s}$  to  $\hat{C}$ . To refine this rotation we do the same comparison with the bigger rings, but we allow a rotation change restricted to one sector of the medium ring. In our example, we consider the 5th ring and we check if, by rotating  $N_\lambda(\mathbf{q})$  by at most  $\theta$  (both clockwise and counterclockwise), there is a better matching *in the 5th ring*. This is done for all the remaining rings (i.e., 5, 6 and 7). Smaller rings are ignored in this comparison mainly because usually they have a very low number of sectors thus they could negatively bias the orientation process or, more simply, add noise to the matching process. In the remainder of this chapter, we call this kind of comparison *Best Fitting Comparison* (BFC).

## 7.4 Graph-cut setup

Till now, we have a set of signals sampled in peculiar positions of the surface (query points). Our goal is to give a label to the query points based on the type of the geometric variations on the neighbor on which they lie. For this step, we do not mind that two labels point to the same pattern, since the similarity analysis in the next steps will join them back into an unique label. Still, we want to avoid one label characterizing more than one pattern at the same time.

To balance similarities in the space of the signals and the proximity of the query points in the space of the model (the Euclidean 3D space in which the vertices/points of the model are), we propose the use of the *graph-cut*, as implemented in [BVZ01, KZ04, BK04]. The MatLab implementation we used for all our tests is available at <https://github.com/shaibagon/GCMex>.

### Label initialization

A fixed number of labels  $L_i, i = 1, \dots, n_l$  is defined. It is worth highlighting that the labels are not related to semantic concepts, as they just characterize the eventual segments of the surface. We scout the distribution of the data both in the space of the signals  $\mathbf{S}$  and in the space of the query points  $\mathbb{R}^3$ . In particular, we first select  $n_l$  signals on the basis of the far-test distance criterion (computing the distances in the space  $\mathbf{S}$ ). The set  $SC$  of  $n_l$  signals selected by this process may be actually different signals or similar signals oriented in different ways. To avoid duplicate cores as much as possible, we proceed to align all the signals with respect to those in  $SC$ , via BFC. The best fitting for each signal  $\mathbf{s}$  (i.e.: the best matching with the lowest  $L^2$  norm value) defines a rotation (based on the BFC result) that is applied to the patch relative to  $\mathbf{s}$ . Finally, the signal is recomputed. This signal re-alignment is repeated a fixed number of times and experimentally shows a good selection of signals for the set  $SC$  in just a limited amount of iterations. This process is speeded up by a preventive similarity estimation of the signals in  $SC$ : if the comparison via BFC of two (or more) of these signals is under a certain threshold  $thr_{SC}$ , then only one among the signals marked as similar (chosen randomly) is kept for the



re-orientation of the signals.

Our goal is to use the signals in the final version of  $SC$  to define a set of *core signals* (*cores* for short) that implicitly define the identity of each label. For example, if the first signal of  $SC$  represents a set of lines, that is the identity of  $L_1$ . However, it is naive to use a single signal as representative for all possible embeddings a pattern may be subjected to.

We refined the cores by computing a k-mean ( $k = n_l$ ) in  $\mathbf{S}$  initialized with the signal in  $SC$ . The new cores are defined as the centroids computed with the k-mean combined with a subset of the signals in their neighborhood. Remember that each centroid has a signal (and thus a query point) of reference. Then, we compute the first decile  $R_S$  of the values of the distances between each signals and the first centile  $R_{\mathbb{R}^3}$  of the distance between all the query points (in  $\mathbb{R}^3$ ). For each centroid  $Centrod_i$ , we consider its  $R_S$ -neighborhood. The signals in the neighborhood correspond to a set of query points in  $\mathbb{R}^3$ . Such a point can be the representative of points that are clustered in a single piece or spread on the model. Through the dbscan algorithm [EK SX96], we identify the different clusters. In particular, clusters must have at least two points and their points can be at most  $R_{\mathbb{R}^3}$  far apart. Finally, for the definition of the cores, we consider only those signals  $\hat{s}_1^i, \hat{s}_2^i, \dots$  which reference query points are in the cluster closer to the query point of reference of the  $Centrod_i$ . The cores  $C_i$  are defined as the mean of the signals  $\hat{s}_i^j$ .

The cores  $C_i$  are the base on which the data cost is computed. The value of  $n_l$  is one of the parameters of the method: it was empirically set to 20 for all our tests. This setting is not restrictive: results similar to those obtained in our tests can be achieved by slightly changing  $n_l$ .

## Data Cost

The data cost is obtained by comparing cores and signals via BFC. Since we want the final label of each query point not to be *0-label*, we add a fake core related to the *0-label*, which cost is set much higher than the maximum cost computed for the other actual cores. Practically, we use 10 times the maximum cost computed for the actual cores. With this, after the graph-cut computation, all the points are labeled according to one of the  $n_l$  cores.

## Smoothness cost

We use the smoothness cost formulation as in equation 7.2. We initialize  $sc(\cdot, \cdot) = 1$  for all the labels, as usual in the graph-cut implementations. The sparse smoothness cost is set to take into count the distance between the nodes (i.e.: the query points) and the similarity of the signals of reference. Moreover, we use  $ss$  to define the strength of the connection between the nodes, setting it to 0 if the respective points  $q_i$  and  $q_j$  are far at most  $2R$  from each other. The weight

shared by these two points is equal to:

$$ss(q_i, q_j) = e^{\left(-\frac{\|s_i - s_j\|}{\sigma_{sig}}\right)} e^{\left(-\frac{\|q_i - q_j\|}{\sigma_{eucl}}\right)}, \quad (7.4)$$

where  $\sigma_{sig}$  is standard deviation of all the possible values of  $\|s_i - s_j\|$  for each existing arc  $ij$  in the graph of the query points.  $\sigma_{eucl}$  is defined accordingly. By our experience, both components in equation 7.4 are influenced by the type of geometric variations that affect a neighbor.

## Other parameters

We use the *alpha-beta swap* algorithm, as it is the one that better fits our purposes. The balance between smoothness cost and data cost is experimentally fixed to  $\lambda=0.015$ . The parameters for the punctual signals depends on the models we are working on. For our tests, we set  $r = 0.7cm$ ,  $n_r = 7$  and  $multP = 4$ . The threshold  $thr_{SC}$  is determined experimentally and we fix it to 10.

## 7.5 Surface segmentation

By applying the graph-cut with the proposed setup, we obtain a numeric label for each query point, with each label associated to a core. The set of query points can then be grouped in  $n_l$  sets of points. Ideally, each set contains all the query points that locally share the same type of geometric variations. In practice, there may be multiple sets of points that correspond to the same geometric variation pattern. For instance, if a model is characterized by a given pattern in two separate areas, the respective query points may be divided in two sets (after the application of the graph-cut). For a more optimized labeling of the query points,

We use the mpLBP descriptor to compare the neighborhood of the query points of each set of points. These descriptors are used to join any couple of labels that define sets of query points that lie on patches that locally share the same geometric appearance. In particular, for each set of points, we consider the points of the base model that are far at most  $r$  from at least one query point of the selected set. We use the Euclidean distance for the neighborhood estimation. It is possible to replace it with the geodesic distance but with little to no significant changes, since  $r$  is usually not big enough with respect to the overall size of the model to make a difference. The settings of the mpLBP are the same as those used to compute the punctual descriptor in Section 7.3.

The final labeling of the points is defined as follows. First, the smallest set of points are removed, since the mpLBP descriptor computed on a small area is less trustworthy. This is done by setting a minimum number of points for each potential segment. For our tests, each set has to count at least 1% of the total amount of query points (with a minimum of 5 points). Then, using the

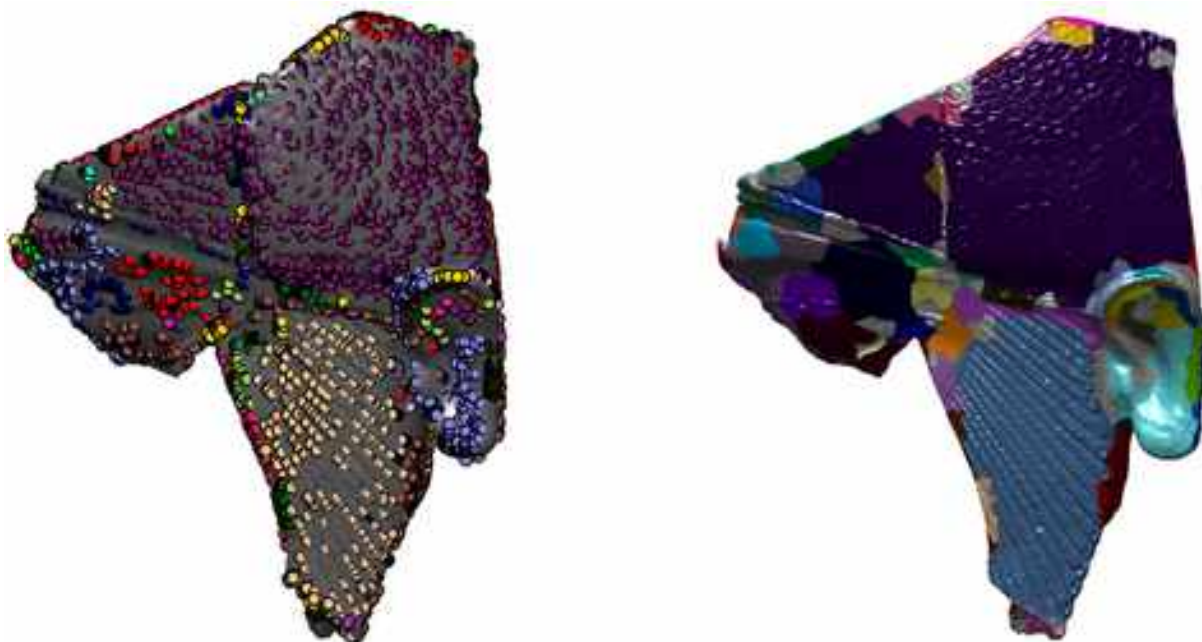


Figure 7.4: The results of each step of the label transferring process. Different labels are represented with different colors. Left: the PATCH-CUT query point labeling. Right: the labeling of the vertices of the mesh. Note that there is no relation between the colors of the query points and those of the patches.

Bhattacharya distance, we evaluate the distance between the mpLBP pattern descriptors. Two segments are considered similar if the distance between two descriptors is lower than a threshold. Such a threshold is experimentally set to the 7th percentile of all the distance values computed for each couple of descriptors. If two segments are similar, then the relative query point sets are joined.

At the end of this process, segments (and query points) with the same label represent patches that have approximately the same local geometric appearance and therefore, a geometric pattern is expected to have the same label everywhere on the surface. By extending the labeling to the neighborhood of each query point, we can obtain a rough segmentation of the models pattern-based.

## 7.6 Examples

We evaluated the PATCH-CUT method on the *Model set* described in Chapter 6. First we analyze the local characterization of each query point with respect to the pattern(s) represented on the

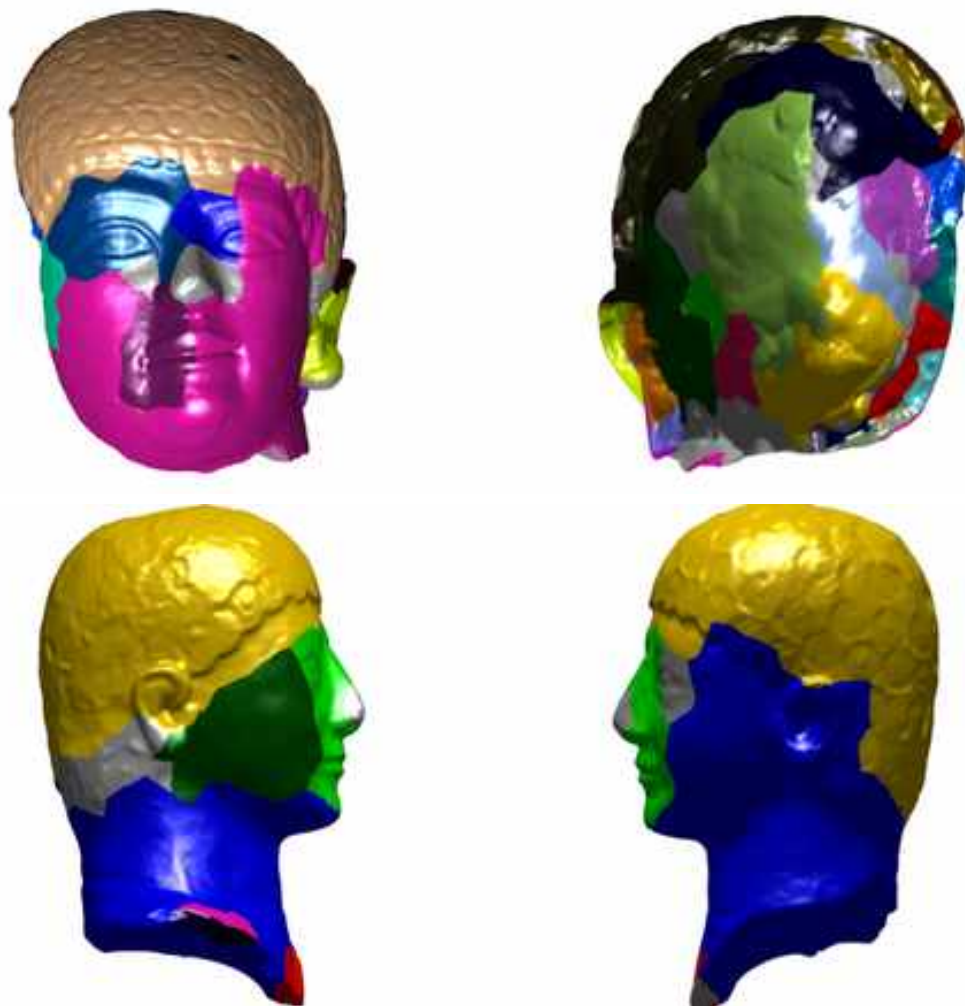


Figure 7.5: Output of the PATCH-CUT method on two models of the *Model set* (front view on the left and back view on the right). Each query point has a different color based on its labeling. There is no relation between the color of the query points and the color that highlights a pattern.

surface. To better visualize the results of the PATCH-CUT, we transfer the labeling from the query points to the vertices of the models. Isolated query points, i.e., the points whose label differ from all its neighbors, are discarded. Then, we label each vertex of the mesh with the label as of the closest query point. In Figure 7.4 we show the mesh labeling obtained with this label transferring process.

A subset of the results is shown in Figures 7.5, 7.6, 7.7, 7.8 and 7.9. The query points have different colors to distinguish the query points labeled with the same label. For simplicity, we refer to a label with the color that we use to represent it. For example, the ‘pink’ query points

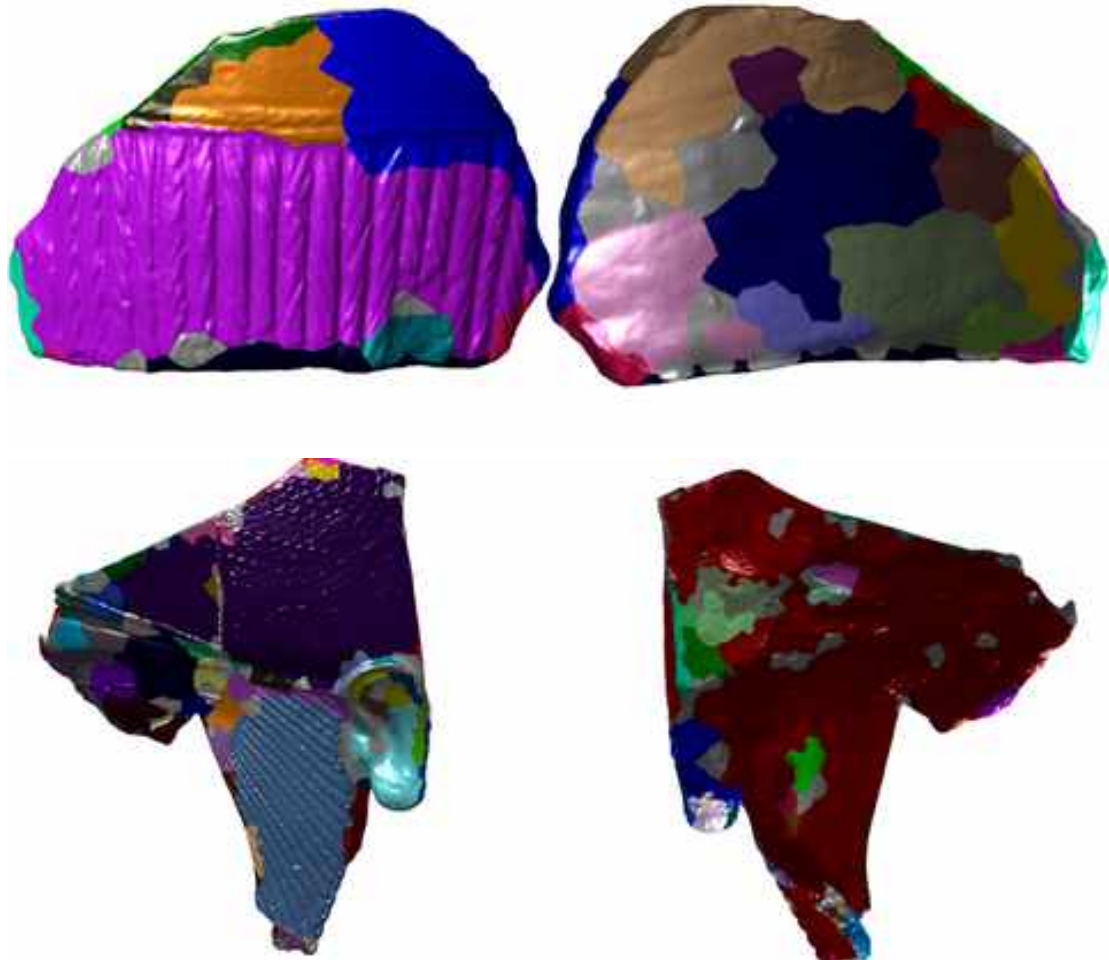


Figure 7.6: Outcome of the PATCH-CUT method on two models of the *Model set* (front view on the left and back view on the right). Each query point has a different color based on its labeling. There is no relation between the color of the query points and the color that highlight a pattern.

are those whose label is represented in pink. Moreover, the patterns of interest on each model are highlighted with colors other than gray. There is *no relation* between the color of the query points and the color that highlights a pattern.

With reference to the example in Figure 7.5 (top row), we are interested in the circlet pattern that is on the head of the model. Notice how the colorization of that area is a single color (peach-pink) and that the same color is not present in other areas of the model (nor in front nor in the back). This means that the characterization defined by the PATCH-CUT groups together this part of the model because it shares the same surface relief (e.g.: the circlets) and also that such a

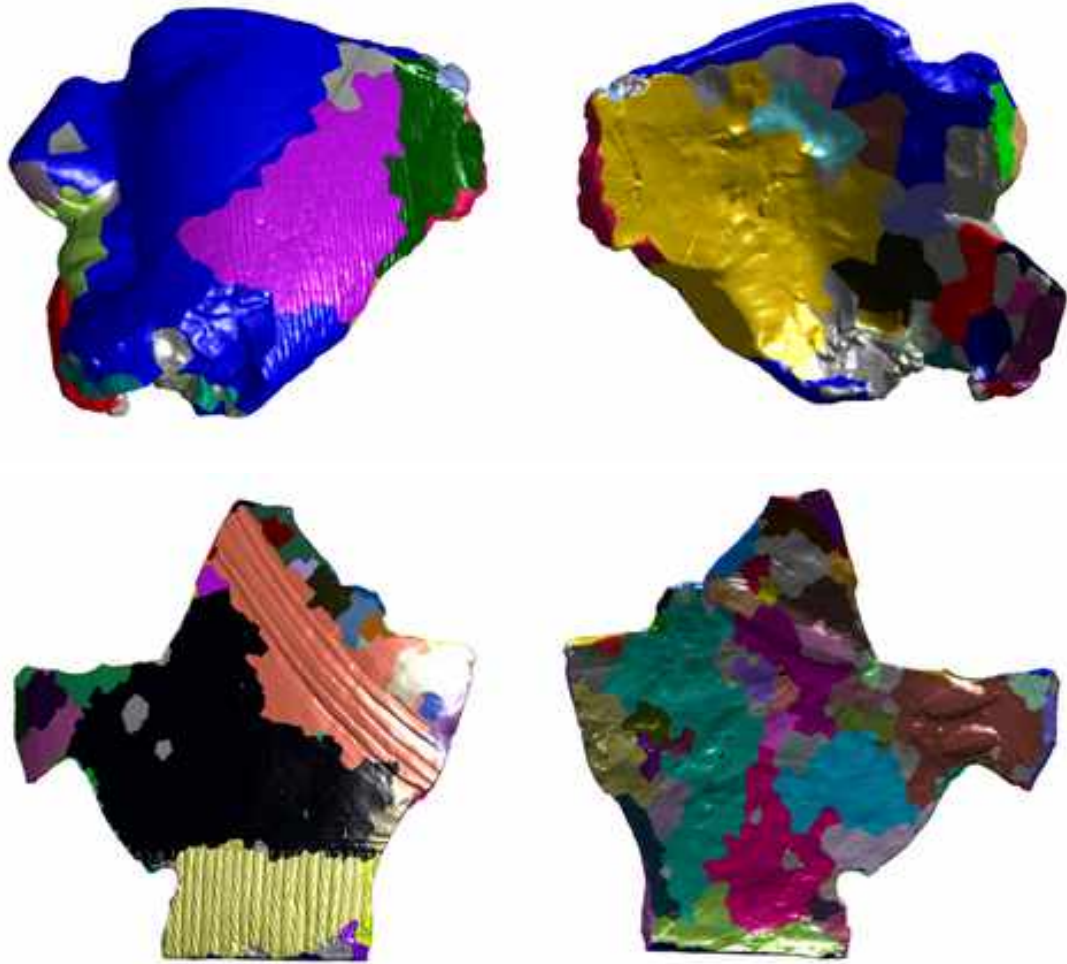


Figure 7.7: Outcome of the PATCH-CUT method on two models of the *Model set* (front view on the left and back view on the right). Each query point has a different color based on its labeling. There is no relation between the color of the query points and the color that highlight a pattern.

relief is not repeated anywhere else. A similar observation can be done for the second example of Figure 7.5(bottom row), in which the pattern of interest is in correspondence of the hairs of the head (represented with a spiral-like pattern). Again, the color yellow spans on the area covered by that pattern.

When a pattern is repeated on a narrow patch, like the beard in Figure 7.7(top row), the labeling can be less precise: see how the beard, represented by straight line, is mostly characterized by the purple label and, in correspondence of the thinnest parts, by the blue label, which mostly characterize the flat parts. Moreover, depending on the example, we observed one dominant

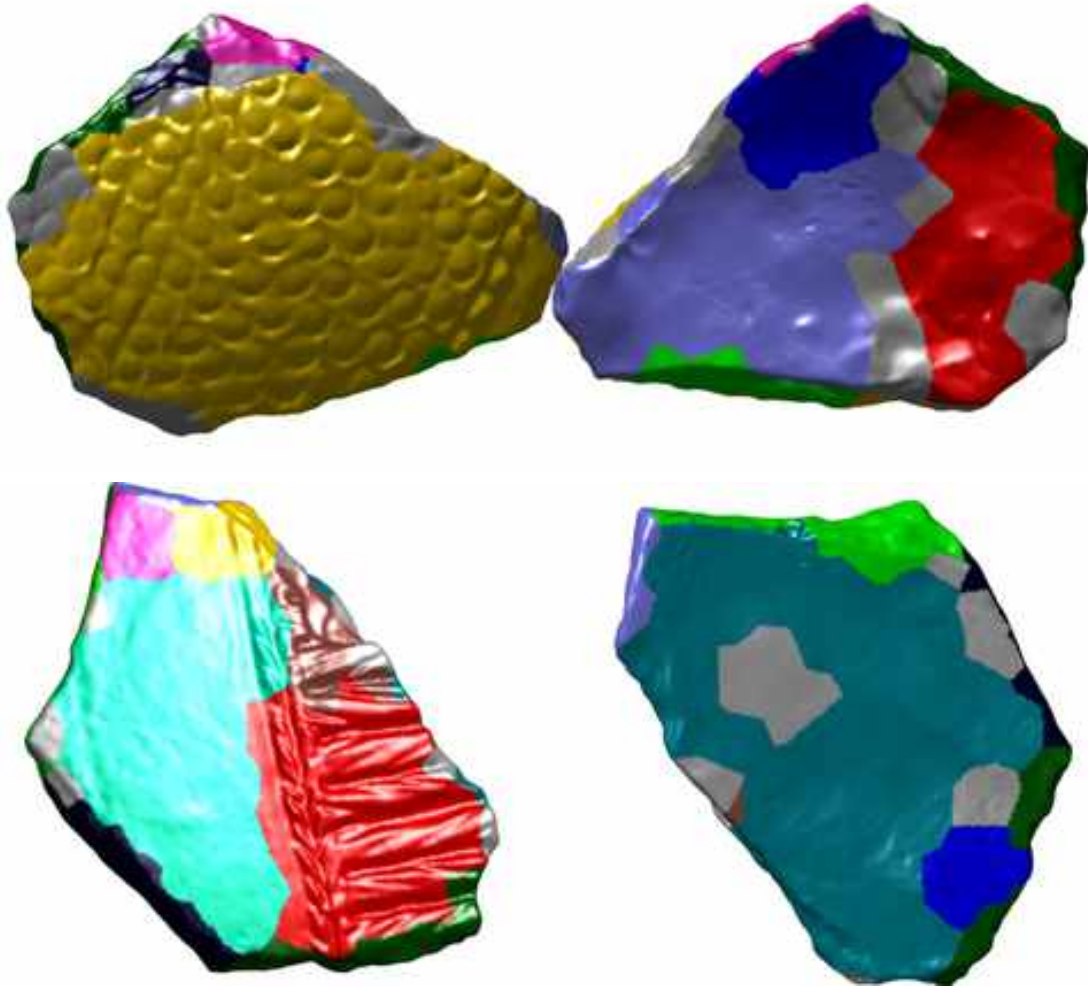


Figure 7.8: Outcome of the PATCH-CUT method on two models of the *Model set* (front view on the left and back view on the right). Each query point has a different color based on its labeling. There is no relation between the color of the query points and the color that highlight a pattern.

label, for instance the back on the model in Figure 7.6(bottom row), or multiple labels (like in Figure 7.6(top row)). The same goes for the flat areas of the model. However, we rarely observed completely wrong labeling of the query points (and thus of the vertices of the models), aside from very specific cases (see later).

Our second test deals with pattern similarity estimation across multiple models. We evaluate the dissimilarity among all the segments extracted in the *Model set*. In our experiments, we use the mpLBP descriptor to measure the dissimilarity between two labels (i.e., segments). The similarity threshold in this test is the same used in Section 7.5 for the similarity estimation among

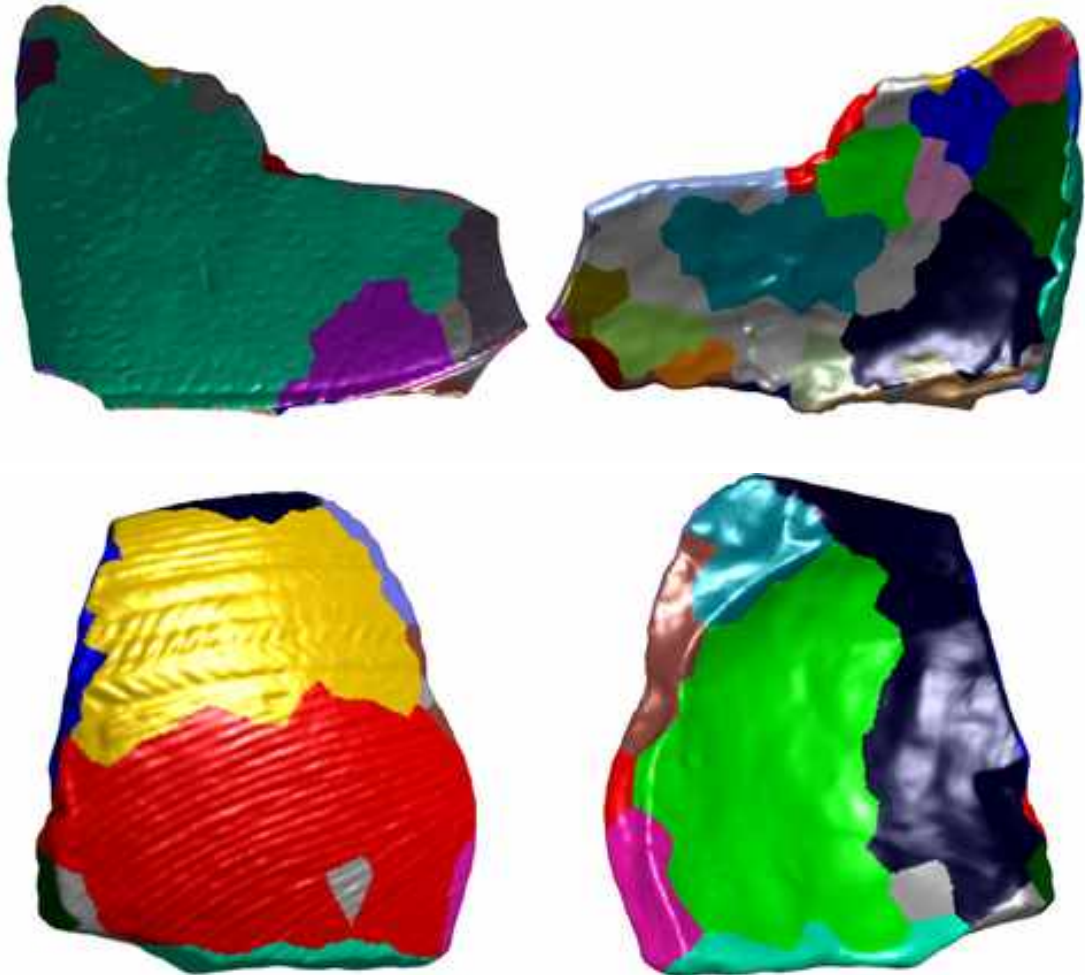


Figure 7.9: Outcome of the PATCH-CUT method on two models of the *Model set* (front view on the left and back view on the right). Each query point has a different color based on its labeling. There is no relation between the color of the query points and the color that highlight a pattern.

patches on the same model via mpLBP descriptor. Since these are early tests, we estimate our results using the query points visualization. Figure 7.10 shows an example for some of the segments extracted with the PATCH-CUT methods. We show an example of cross-correspondence among models for the circles patterns. The experiment shows the results of the search when a segment that lies on the head of the model in the left is used as a query (segments highlighted in red in the models in the middle and right).

One limitation of this method is that its performances are strongly influenced by the core definition. Figure 7.11(left) shows an example of the circle patterns (highlighted in green) not being





Figure 7.10: An example of cross-model pattern recognition, starting from a set of query points (red points) that lie on a pattern, the head of the model in left. The retrieved patches on the other models are shown to the middle and right.

properly characterized since no core is found on the surface characterized by that pattern. This is not surprising because this model is very large and complex, with many outliers. The query points representation of the results allow us to observe that not many query points falls on the pattern area, which makes the respective core even harder to find. When selecting the first 20 core representatives with the farthest sampling in the space of the signal, the signals corresponding to the circlets are not ranked in the top 20. This fact is easily verified considering, for instance, 30 cores. The outcome of the PATCH-CUT method with 30 cores is shown in Figure 7.11(right), in this case a large part of the circlets is recognized as belonging to a much more distinct segment ('orange' query points). Another interesting fact is that the beard is split into two labels. We indeed observed that the beard pattern is not uniform, as some of its lines are wider in some areas and tighter in others, partially justifying the presence of more than one label on the beard. However, the exact 'line of cut' between the 'red' and 'lilac' query points does not represent faithfully these areas. An ad-hoc smoothness cost that takes more into count the signal similarity might help to better separate the segments.

If a pattern is just lightly chiseled on the surface, the graph-cut may not be able to properly balance signal similarity and label 'smoothness'. Figure 7.12 shows an example of this issue.

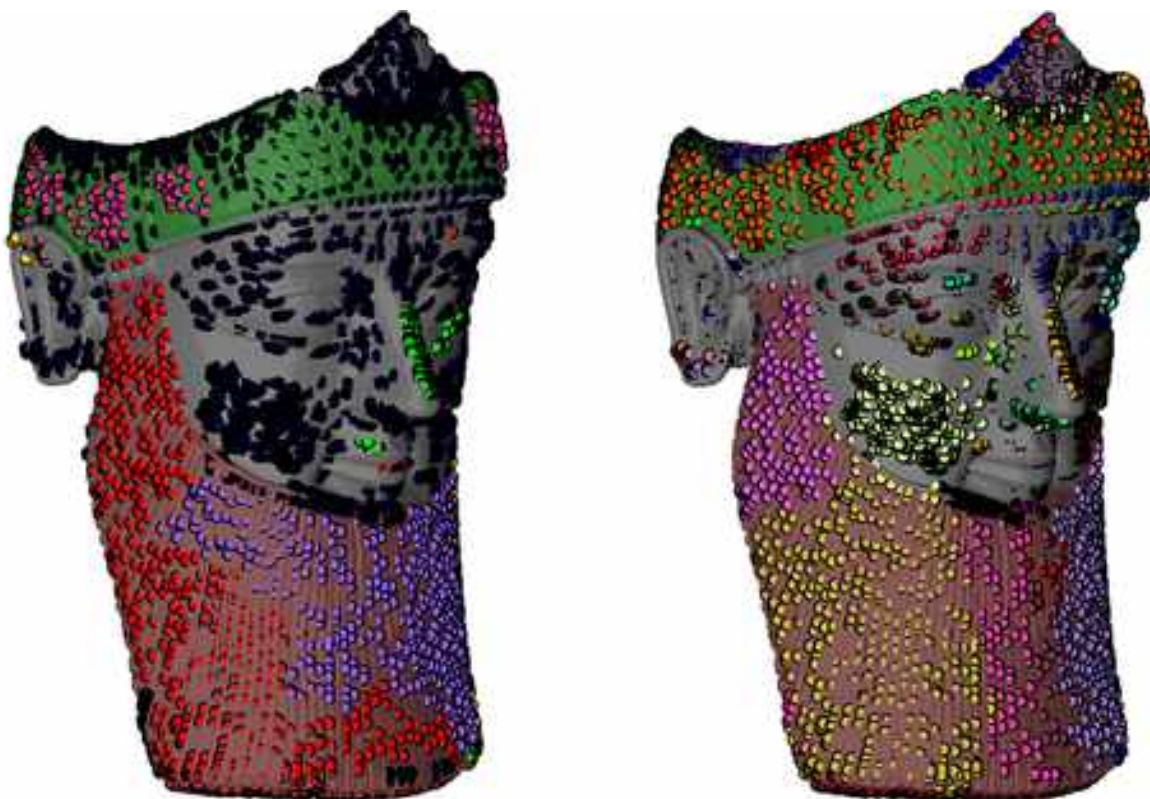


Figure 7.11: An example of non proper labeling of the surface. Left: the final result of the PATCH-CUT method using 20 cores. Right: the final result of the PATCH-CUT method using 30 cores.

## Conclusions

In this Chapter we presented the PATCH-CUT method which, starting from a mesh with none, one or more than one patterns, aims at segmenting the models in patches characterized by at most one pattern. The segmentation is done via local analysis of the neighborhoods of a set of samples on the surface. The graph-cut algorithm is used to balance between similarity and smoothness of the boundaries of the segments. While there is still room for improvement, the current state of the method is enough to correctly identify patterns on the surface of the model and to partially address cross-model pattern recognition, thanks to the combination of PATCH-CUT with pattern retrieval methods.

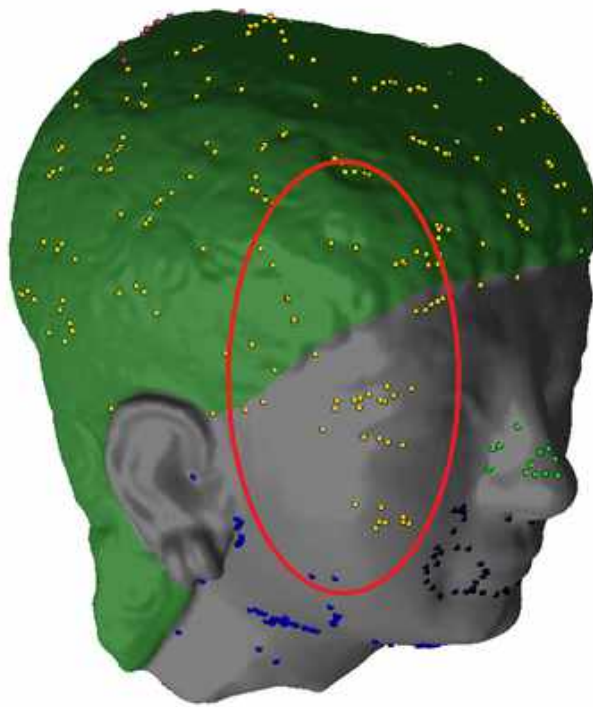


Figure 7.12: An example of a model with a lightly chiseled pattern on the surface. The graph-cut does not segment the pattern (highlighted in green) properly with respect to the flat surface (in gray).

# Chapter 8

## Learning-based approaches

Artificial intelligence has a long development history, that alternates success and setback mostly related to the advancement in the calculators ability in dealing with larger and larger data and their ability to make more operations faster (which usually came together with technology advancements). In the very beginning, Warren Sturgis McCulloch and Walter Pitts [MP43] introduced the concept of Artificial Neural Networks in 1943. The first significant evolution in the field was in 1958 thanks to the introduction of the concept of the perceptron [Ros58]: a binary net with classification purposes built with an input and an output layer. This model of neural network was not largely used, as the perceptron is not able to model simple operations like the XOR [MP17]. Moreover, until the adoption of the back-propagation algorithm and multilayer net designs, there was neither the algorithm nor the necessary amount of data for a proper training. On a slightly different note, in 1967 Cover and Hart lay the foundation for a basic pattern recognition (intended in the more broad meaning of the word) with their Nearest Neighbor Algorithm [CH67]. Around the beginning of the '80s, there was a significant separation between the final goals of Machine Learning (ML) and the rest of the Artificial Intelligence (AI). Rather than working on the realization of general AI, machine learning researchers focused on solving practical problems, focusing on neural networks (abandoned, at those times, by AI researchers) using tools available in probability theory and statistic literature. With the growth in research and technology (starting from the '90s) ML flourished more and more and, in particular, the ImageNet success [DDS<sup>+</sup>09] marks a pivotal point in the machine learning history, boosting immensely the interest in the branch of machine learning called *Deep Learning* and, in particular, on the neural networks.

Currently, ML methods ease the classification process of large amounts of data with very good performances. This rich research field is divided mainly in three branches: *supervised learning*, *reinforcement learning* and *unsupervised learning*. Briefly, supervised learning includes the use of a set of examples provided with a ground truth that can be used to evaluate the goodness of the classification based on the differences between the method output and the ground truth

itself. Reinforcement learning is designed for problems that require decision making: actions are enforced or penalized based on the outcome of the latter. Finally, unsupervised learning is used when a ground truth is not available and the research of patterns in the given data is up to the method itself.

There are multiple challenges in applying learning approaches (in all the three machine learning branches mentioned) to the use case of this manuscript. First and foremost, 3D models contain a huge amount of raw data: each problem may interest different aspects of that, which lead to different pre-processing or completely different setups for the learning method itself. Moreover, especially in the archaeological environment, datasets of 3D models usually are not big enough. Moreover, the elements of interest (i.e.: patterns) are not repeated sufficiently to decently train a learning method. Additionally, the number of data sets containing textures is limited by the effort required to produce high resolution models with detailed textures, as well as their availability, as some of these pieces are unique.

Our research for a learning method able to face the pattern recognition problem culminated in two different attempts. The first is the study of the potential of a sparse representation (unsupervised) learning method that aims at finding a sparse decomposition of a set of signals, called *dictionary learning*. In other words, to find a set of signals (called *atoms*), that can be linearly combined to obtain the original signal set. The fact that dictionary learning does not require a huge amount of labeled data makes it appealing for our problem.

Secondly, given the advent of convolutional neural networks (CNNs), we present a possible approach that characterizes the region around a vertex in a robust way with respect to a pattern and classifies it appropriately. The neighborhood of each vertex is characterized by a surface property (e.g.: curvature) captured in an image. The classification with CNNs is made through transfer learning starting from the ResNet-50 architecture, whose model is fine-tuned for the problem. The classification can be transferred back to the model, obtaining a rough segmentation of the models based on the pattern(s) that characterize them.

In this thesis we report the pros and cons of our applications, presenting the difficulties in applying learning methods to our use-case.

## 8.1 Signal aggregation based on dictionary learning

Let us consider a set of signals  $X = [\mathbf{x}_1, \dots, \mathbf{x}_n]$ ,  $\mathbf{x}_i \in \mathbb{R}^m$ . The *dictionary learning* can be formulated as the following minimization problem:

$$\min_{D \in \mathcal{C}, A \in \mathbb{R}^{p \times n}} \sum_{i=1}^n \frac{1}{2} \|\mathbf{x}_i - D\boldsymbol{\alpha}_i\|_2^2 + \lambda\psi(\boldsymbol{\alpha}_i), \quad (8.1)$$

where  $A = [\alpha_1, \dots, \alpha_n]$  carries the decomposition coefficients of the signals  $\mathbf{x}_1, \dots, \mathbf{x}_n$ ,  $\psi$  is the sparsity-inducing regularization function and  $\mathcal{C}$  is typically chose as the following set:

$$\mathcal{C} \triangleq \{D \in \mathbb{R}^{m \times p} : \forall j \quad \|d_j\|_2 \leq 1\}. \quad (8.2)$$

$D$  is usually called a *dictionary*. In our use case,  $\psi$  (i.e.: sparsity) is driven by a parameter  $k$ . In other words, give a set of signals  $X$  and given a set of parameters  $p$  (size of  $D$ ) and sparsity  $k$ , by solving the minimization problem in Equation 8.1 we obtain a set of  $p$  signals ( $d_j \in \mathbb{R}^m$ , called *atoms*) that can approximate the signals  $\mathbf{x}_i$  with  $\approx D\alpha_i$ . Notice that, due to  $\psi$  and the parameter  $k$ , the vector  $\alpha_i$  has only  $k$  non zero entries, thus  $\mathbf{x}_i$  is approximated using only  $k$  atoms. Of course, to make the most out of the method,  $p$  should be much lower than  $n$ . Finally, to our knowledge, keeping the ratio  $k \approx \frac{p}{4}$  leads to better results.

The goal is to apply sparse decomposition of the dictionary leaning to the signals extracted from a 3D model and to study the components obtained from the learned dictionary ( $D$  and  $A$  in particular), to scout the surface more quickly searching patterns of the surface. For example, since patterns are repeated details on the surface and being the atom shape strongly influenced by repeated signals, we conjectured that the atoms alone could lead the research in this direction. The pipeline of this method is shown in Figure 8.1.

Finally, we distinguish two approaches: one in which  $k = 1$  and another in which  $k > 1$ . Of course, it is almost redundant to talk about dictionary learning if  $k = 1$ . Still, it is interesting to discuss the results obtained with this particular setup of the dictionary computation.

To compute a dictionary, we used an implementation of the K-SVD algorithm [RZE08]. The MatLab implementation we used is freely available at <http://www.cs.technion.ac.il/~ronrubin/software.html>.

### 8.1.1 Surface sampling and signal extraction

As in previous chapters, we want to avoid computing a signal for all the points of the surface, due to the high resolution of the meshes we usually work with. Similarly, we also have to be sure that the signals on the areas of interest are repeated multiple times. This is especially important when working with the dictionary learning, seeing its behaviour with repeated signals. Indeed, intuitively, the more a signal is repeated, the more it is probable that an atom is going to assume its shape.

We sample query points and punctual signals as in the PATCH-CUT method, ( $q_i \in Q$  and  $\mathbf{s}_i \in S$  respectively). Then, inspired by the work done in [DVC18], we sort the signals based on the results on a dictionary computed on them. More precisely, this is an iterative process based on how well the signals fitted the atoms computed in the dictionary. At each iteration, a dictionary is computed on the signals. Then, each signal is compared to each atom. The comparison is done by rotating the signals with respect to the atom until it best fits the atom (based on the  $L^2$

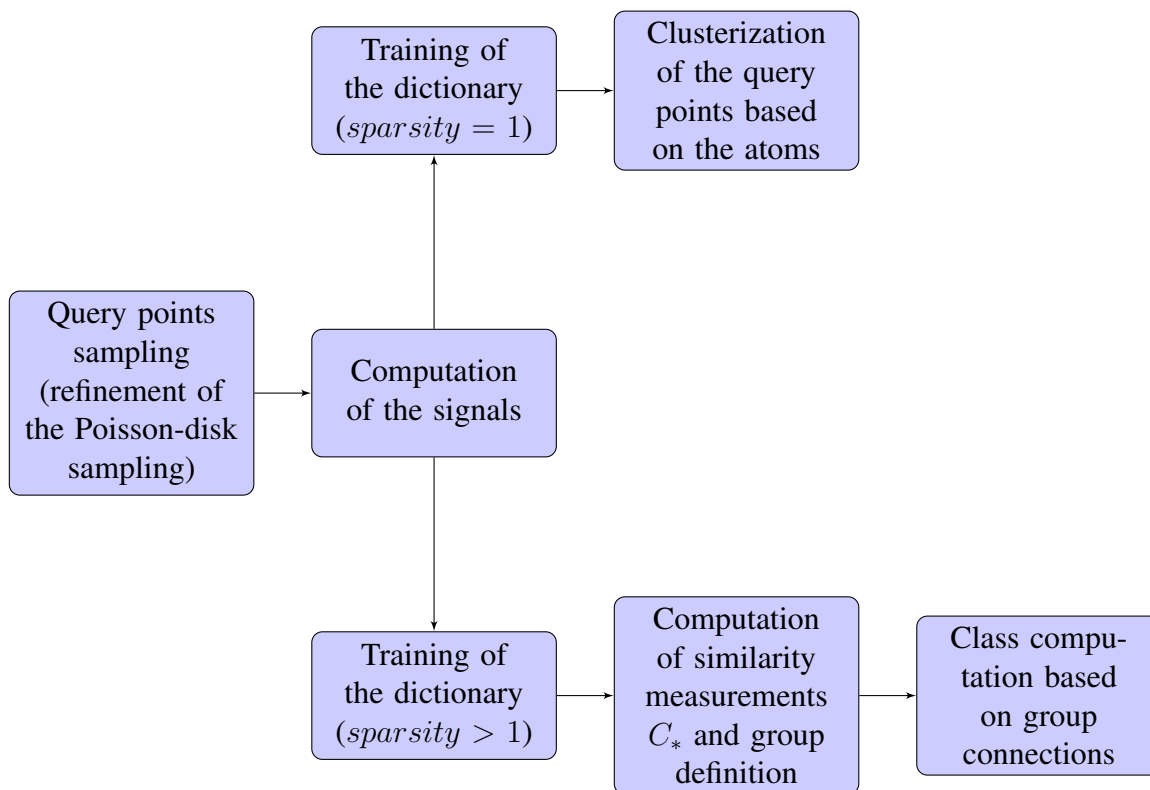


Figure 8.1: Pipeline of the Dictionary Learning based method proposed for pattern characterization and recognition. This method has two possible versions, sparsity 1 and sparsity greater than 1, which both start in the same way.

norm). Then, the rotation that fits best across all the atoms is kept. Once this is done for all the atoms, the next iteration starts, a new dictionary is computed and the cycle repeats. In our tests, we observed that after a given number of iterations the signals stabilize their orientation toward a low number of ‘default’ positions. In our tests, the number of iterations is set to 10.

### 8.1.2 The case of sparsity equal to 1

Our study started from the idea that we can distinguish patterns on surfaces just by evaluating the differences between the atoms of a dictionary trained on the signals of the query points on a given model. Indeed, atoms should be highly influenced by the signals of query points on the patterns most present on the surface (in an ideal case, one atom per pattern). By backtracking which atom approximates which signal, it is possible (in theory) to segment the surface in the areas characterized by a single atom, obtaining a surface segmentation. Such a segmentation could be used as a basis for a retrieval method or, in case of very good atoms, directly as a

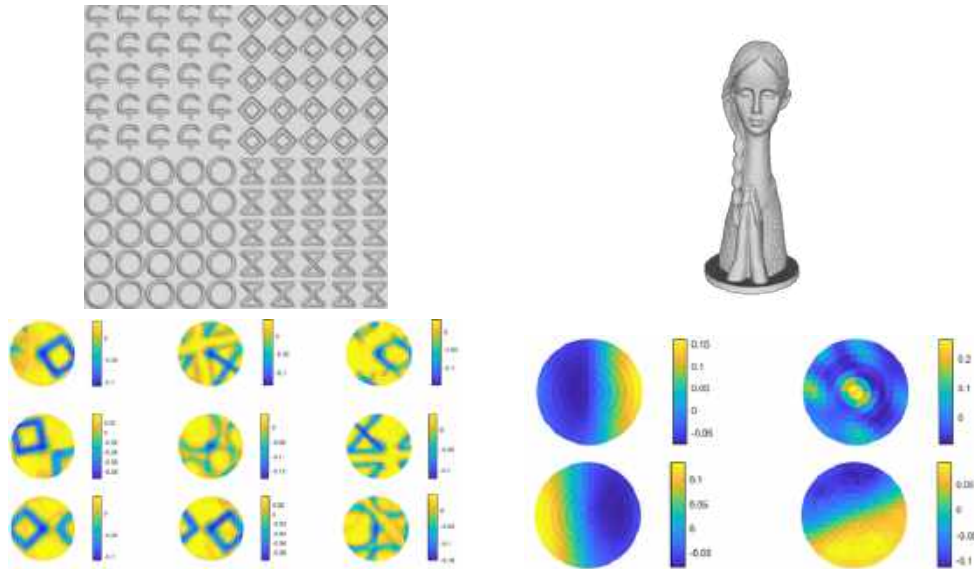


Figure 8.2: Top: the labeling of the query points of two models. Bottom: the atoms of the models above. Notice how the atoms resembles the patterns for both the first model (with 4 different patterns) and the second one (with one pattern).

pattern recognition solution (since different parts of the model with similar pattern should have similar atoms).

In more details, we set  $k = 1$  and  $p$  to a fixed value. A dictionary is computed on the signals extracted from a given model, obtaining  $p$  atoms. Since  $k = 1$ , each atom should represent one of the signals the most repeats in the overall signal set. Indeed, the atoms are selected by their reconstruction error, thus they alone (multiplied by a constant value) should represent at best a specific class of signals. We identify these classes with the patterns on the surface. We then labeled the query points based on the atoms they are describing in the dictionary. Finally, we ‘accept’ a labeling only after checking that the signal is actually well reconstructed by the atoms, based on the reconstruction error between the original signal and its approximation. This method was tested on a synthetic model (with four patterns) and the scan of a statue (with one pattern), respectively shown in Figure 8.2(Top). The surface colors are based on the atoms that characterize that Figure 8.2(Bottom) shows that atoms actually take the shape of the signals we are interested in and that the labeling of the query points is driven by the pattern in most of the surface of the model. The size of the two computed dictionaries, 9 for the first models and 4 for the second one, are different since the number of patterns on the surfaces is different from model to model.

However, despite the promises, this approach became less and less efficient when more complex models came into play. Indeed, models like those in the GRAVITATE models are far less homogeneous in terms of signals (and query point distribution) which demand a more precise



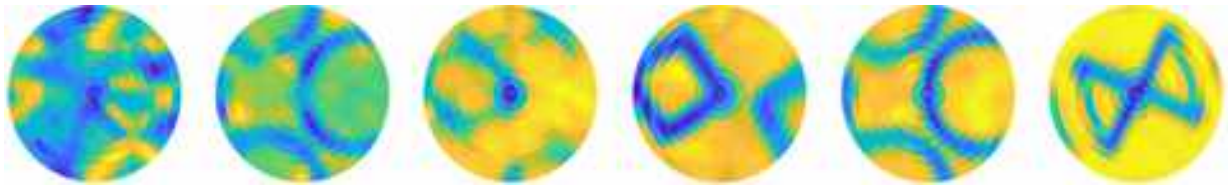


Figure 8.3: A subset of the atoms of a dictionary of size 12 and sparsity 3 computed on the model in Figure 8.2(Bottom). Notice how the first atom can be obtained with a linear combination of other atoms.

reconstruction of the signals that cannot be obtained by setting  $k = 1$ .

### 8.1.3 The case of sparsity greater than 1

The idea behind this path is similar to the previous one, but in this scenario the reconstruction of the signals ( $D\alpha_i$ ) are more precise and should reduce the approximation errors one can obtain for  $k = 1$ . However, the characterization based on multiple atoms is much more complicated. Mainly, already with  $k = 2$ , atoms alone may be misleading in many ways. Let us assume that an atom represents a circle, while another one represents a line and, finally, let us have a signal approximated by a combination of both. A practical example of this issue is shown in Figure 8.3 in which a subset of the atoms extracted from a dictionary of size 12 and sparsity 3 computed on the model in Figure 8.2(Bottom) is represented. Notice how the first atom (from left to right) can be seen as the combination of other atoms. The atoms alone bring contradictory information on the table, which makes the once trivial segmentation much harder. Also, keep in mind that the set of atoms can be seen as a set of generators of  $S$ . Since they do not form (in the mathematical meaning of the word), we cannot assume that each atom is unique. Thus, multiple atoms can represent the same feature in slightly different ways, leading to a redundant segmentation.

In this case, other components of the computed dictionary must be kept more in consideration, such as the coefficients  $\alpha_i$ . For this research path, we looked for a way of including the information of multiple results of the dictionary learning method (reconstructions, reconstruction errors, coefficients and signals themselves) all at the same time. To better understand the approach, we formalize more the concept of reconstruction in a subset of atoms of a given dictionary. Let us assume that  $D$  is a dictionary computed on a set of signals  $\hat{s}$ , with sparsity  $k$ . This means that each  $\hat{s}$  has a set of atoms  $\{\hat{d}_1, \dots, \hat{d}_k\}$  that linearly approximates  $\hat{s}$ . Of course, nothing forbids us from using that set of atoms to approximate a completely different signal, let it be  $\bar{s}$ . While in general this approach does not lead to a good approximation of the  $\bar{s}$ , it is reasonable assuming that if the signals are similar, then also the approximation of  $\bar{s}$  should be at least decent enough to be recognizable. Since, in the following, we are going to use this concept frequently, we add the following notation. We use  $[a]_b$  to indicate the reconstruction of the signal  $a$  in the space of the atoms of the signal  $b$ . Since we always work with one dictionary at the time, we do not

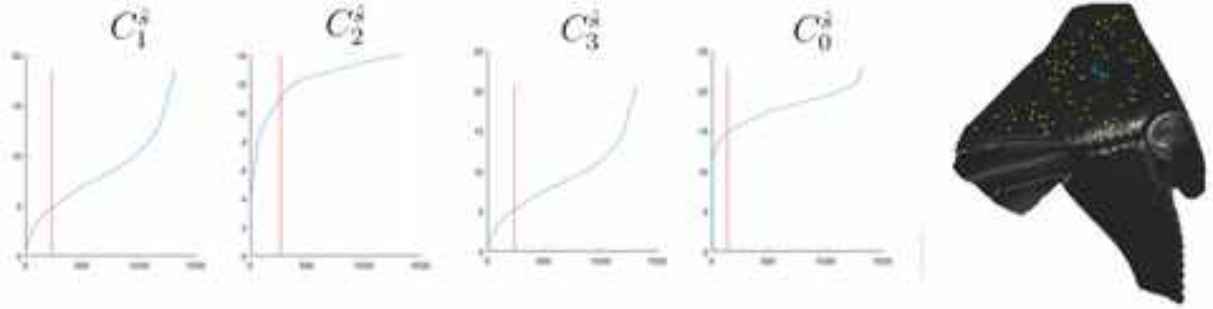


Figure 8.4: An example of the four distances used to compute a group. Left: each curve represents the sorted values of each distance. The thresholds  $thr_0^s$  are represented by the red lines. Right: the point  $\hat{q}$  (in light-blue) and its group (in yellow) based on the distances computed.

indicate it in the notation, to make it more readable.

In our approach, first we compute a dictionary  $D$  on the signals with a fixed set of parameters  $p$  and  $k$ . Then, for each signal  $\hat{s}$  with respect to all the other signals  $s_i$ , we computed the following distances:

- $C_1^s(s_i) = \|\hat{s} - [s_i]_{\hat{s}}\|_2$
- $C_2^s(s_i) = \|s_i - [s_i]_{\hat{s}}\|_2$
- $C_3^s(s_i) = \|\text{coeff}([\hat{s}]_{\hat{s}}) - \text{coeff}([s_i]_{\hat{s}})\|_2$
- $C_0^s(s_i) = \|\hat{s} - s_i\|_2$

where  $\text{coeff}([A]_B)$  are the coefficients given to the atoms shaping  $A$  in the space of the atoms of  $B$ . These distances induce four different sortings for the signals, from the one with the lowest distance value to the higher. Intuitively, we can say that if a signal  $s_i$  stays in the first position in all four sorts, then it is probably similar to  $\hat{s}$ . Since there is no trivial definition for a threshold value that assess similarity based on the proposed distance, we define a set of thresholds  $thr_0^s$  in the following automatic way. For each distance  $C_i^s$  we can define a discredited curve  $\mathcal{L}$  of the sorted values obtained computing the distance evaluations  $C_i^s(s_i)$ . The goal is to approximate  $\mathcal{L}$  with a polynomial function and setting the threshold based on the first flex of the curve. Experimental tests showed that the family of the polynomial of degree 5 are good for approximating curves like  $\mathcal{L}$  (defined by the distances  $C_i^s$ ). Moreover, we went for a more restrictive similarity criteria than the position of the first flex: indeed, we used  $\frac{1}{3}$  of the distance between zero and the x-coordinate of the first flex. Figure 8.4 shows an example of these measures and threshold for a point on top of the head of this model and the respective group.

We call this set of similar points a *group*. We then compute the group of each point (so every time the set of curves changes) and we see them as the nodes of a graph. If a signal is shared by

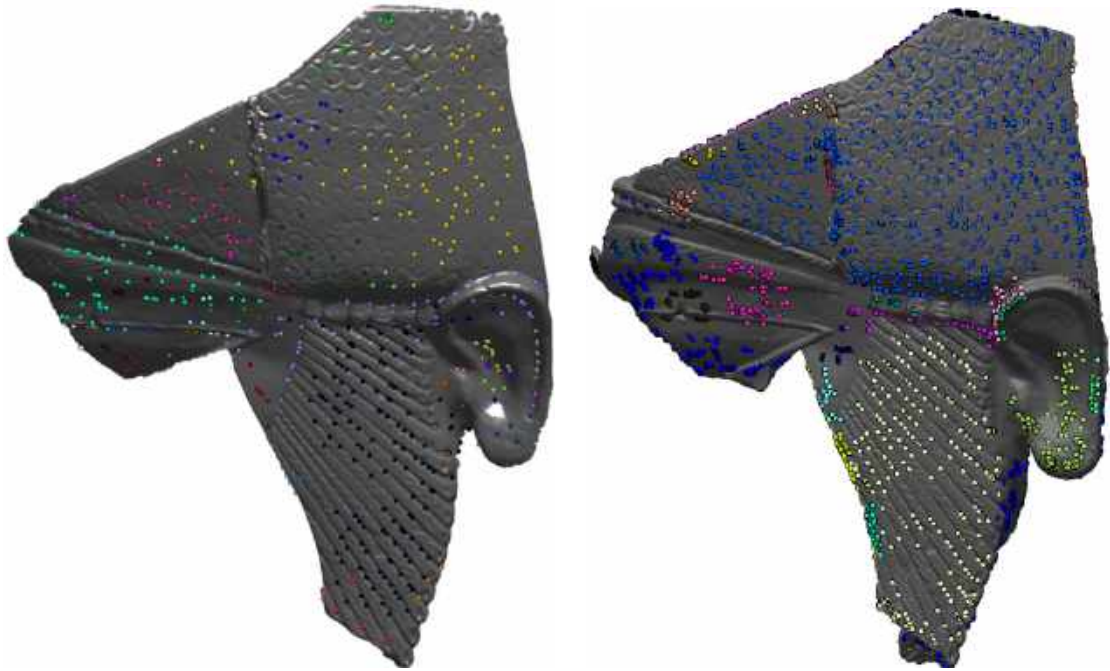


Figure 8.5: The results on a GRAVITATE model with the dictionary learning with sparsity greater than 1 (Left), compared to those obtained with the PATCH-CUT method (Right).

2 groups, we connect them with an arc. Finally, we assume that the connected components of the graph are composed of similar points and, given the nature of the signals, should correspond to areas characterized by the same patterns. Figure 8.5 shows an example of the result obtained by using this pipeline, compared with the results obtained with those obtained in Chapter 7. Notice that, with this method, not all the points are labeled.

## 8.2 Multi-view RESNET50: a description based on CNN

Briefly, an artificial neural network is typically made by an input layer, an output layer and a set of hidden layers. Each layer is made by *neurons* (or interconnected nodes). Each connection has a weight that changes the behaviour of the network. The whole ‘learning paradigm’ is how these weights are set (or how they change) based on the data we feed to the neural network. In this section we focus on *convolutional neural networks* (or CNN for short), which are a branch of the supervised machine learning methods. In it a dataset of examples  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots\}$  (*training data*) is given. Examples of possible data are images, sounds, point clouds, etc.. For any example, there is also a desired output,  $y_i$ . The set of all the outputs is called *ground truth*. It can be a set of labels discrete, which leads to a *classification problem*, or a continue label,

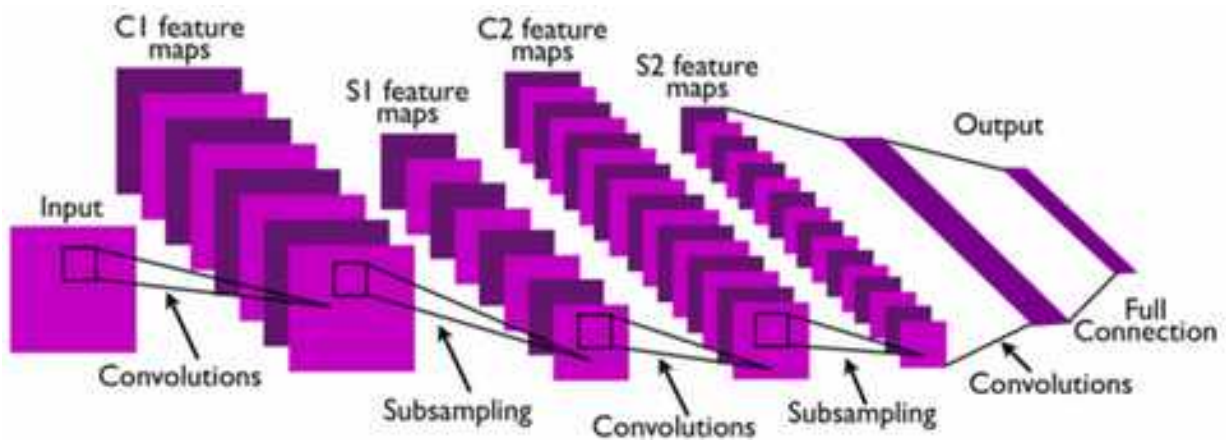


Figure 8.6: A typical CNN architecture with two feature stages. Image from [LKF10].

which leads to a *regression problem*. During the training process, the net generates an output  $\hat{y}_i$  for each  $x_i$ , which is matched with  $y_i$ . Weights are updated using the Stochastic gradient descent so that the difference between  $\hat{y}_i$  and  $y_i$  is lower and lower. The *loss function* quantifies the error between these two entities.

In particular, a CNN does an hierarchic extraction of the features of the inputs. This is reflected in the structure of the net which, with respect to the classic nets, presents layers with a three dimensional structure. There are two main strengths in using this approach. First, a CNN takes advantage of the 2D structure of the images (close pixels are highly correlated), so that neurons in the same layer are connected by just a small region of the previous layer, avoiding unnecessary connection as in the classic nets. Moreover, in a CNN each *feature map* is generated by the convolution of a *kernel*, scaled with the input size. This leads to a number of parameters drastically lower than that of the classic nets. Second, CNN uses a so-called *pooling* layer. This allows for a translation invariant network, which in turn allows the net to focus initially on the smaller details and then on the bigger one. The totality of the objects in the training set can usually be found in the deepest layers of the CNN. A CNN is composed by a number of the following layers: *convolutional layer*, *rectification layer*, *normalization layer* and *pooling layer*. The structure of a CNN is represented in Figure 8.6.

In this section we present our work on pattern recognition in which classification is driven by a particular CNN, called *Residual Network 50* (or RESNET50). The key difference with respect to the other CNNs is that this net, instead of learning features, learns *residuals*. To put it simply, on each layer it learns from subtracting a learned feature from an input of a given layer. There are multiple versions of RESNET, mainly distinguished by the number of layers they have. As an example, a visual representation of the RESNET34 is shown in Figure 8.7. We tested our approach with the RESNET50, which has 50 layers.

In the following, we detail our approach to the pattern recognition problem based on the classifi-

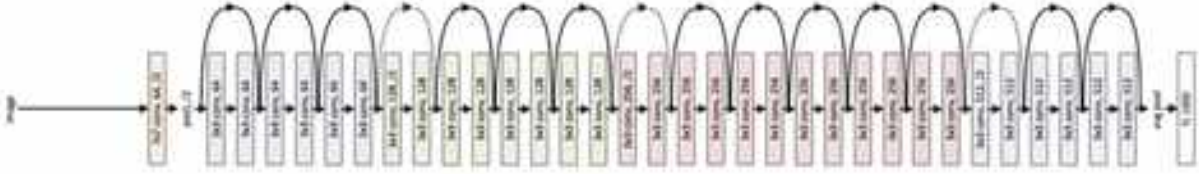


Figure 8.7: A visual representation of the RESNET34. The dotted shortcuts increase dimensions. Image from [HZRS15].

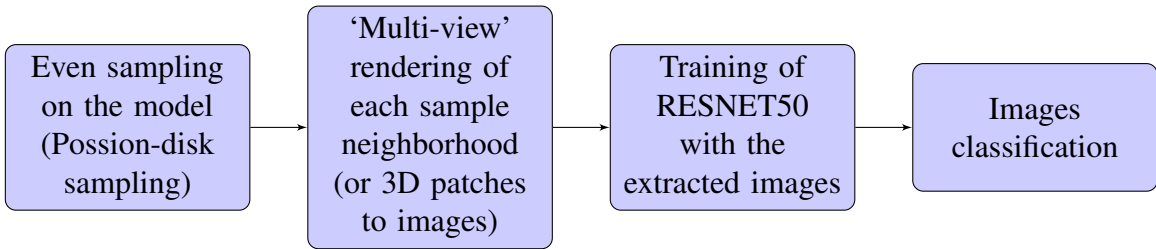


Figure 8.8: Pipeline of the multi-view CNN based method proposed for pattern characterization and recognition.

ation given by the RESNET50 that we call *multi-view RESNET50*. The pipeline of this method is represented in Figure 8.8. The goal of this method is to extract multiple views (from which the term ‘multi-view’) of the point neighborhoods of a given set of vertices  $\tilde{v}_1, \dots, \tilde{v}_n$  of a given triangulation  $M$ . By sampling more view from the same patch, we aim at the generalization of its pattern by considering it from different angles and different views. These views are not just a projection of the whole model with different rotations: only the neighborhood of each  $\tilde{v}_i$  is considered when the model is rendered and projected in the final image. These images are then used to feed a RESNET50 and the classification obtained by the net is used to locally identify the patterns.

### 8.2.1 Local feature characterization

In this section we introduce the concept of a local feature of the 3D model, not to be confused with the concept of feature extracted from a Convolutional Neural Network. The first is a carefully selected feature of the 3D model, the second is part of a feature map and is a feature of the 2D image that is extracted by a convolutional operation against one kernel at a given hidden layer of the CNN. We can identify two situations in which it is required to extract local features: when preparing the training set and when we want to sample and classify a mesh vertex neighbor. The local feature extraction process is very similar in both cases and it starts as follows.

Let us have a model  $M$  (represented with a triangulation  $T = (V, F)$ ) and a vertex  $\tilde{v} \in V$  (see

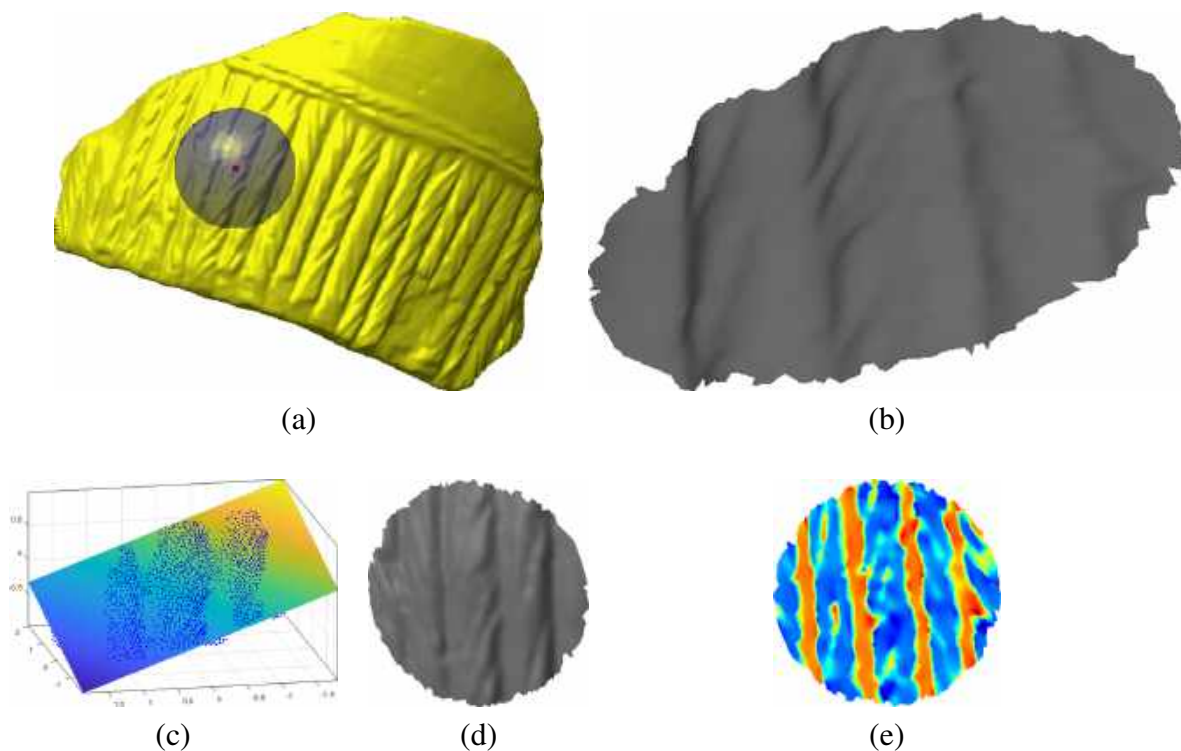


Figure 8.9: (a): a model  $M$ , a vertex  $\tilde{v} \in V$  (in red) and a visual representation of  $N(\tilde{v})$  (the bubble). (b): the triangulation  $T_{\tilde{v}}^r$  selected from  $T$ . (c): the best fitting plane  $\lambda$  with respect to the vertices in  $T_{\tilde{v}}^r$ . For a better visualization, only the vertices of the patch are shown. (d): the change of coordinates effects on both  $T_{\tilde{v}}^r$  and  $\lambda$ . (e): the final rendering of  $T_{\tilde{v}}^r$  (in this example, the colors are based on the shape index estimation).

Figure 8.9(a)). The goal is to extract the feature on which this vertex lies (or it is close to). Moreover, we define a radius  $r$ , which is the radius of the neighbor we are interested in. More details on how this radius is selected are at the bottom of this Section.

First, the kd-tree algorithm is used to compute the neighbor of the point  $\tilde{v}$ , defined as  $N(\tilde{v}) = \{v \in V \mid d(\tilde{v}, v) \leq r\}$ , where  $d$  is the Euclidean distance. A second triangulation  $T_{\tilde{v}}^r$  is selected from  $T$  and its faces are those that have all their vertices also in  $N(\tilde{v})$  (see Figure 8.9(b)).

Moreover, we compute the best fitting plane  $\lambda$  with respect to the vertices in  $T_{\tilde{v}}^r$  (see Figure 8.9(c)). Then,  $T_{\tilde{v}}^r$  translated and rotated so that the center of the coordinates is  $\tilde{v}$  and the normal of  $\lambda$  is  $(0, 0, 1)$  (see Figure 8.9(d)).

The triangulation now rendered: it is colored based on the surface curvature, which is approximated using the Toolbox Graph MatLab toolbox [Pey], using the jet colormap (see Figure 8.9(e)). Lighting is turned off to avoid noises and/or uncertainty caused by shadows or highlights. Our choice regarding the mean curvature and its estimation method for the characterization for pat-

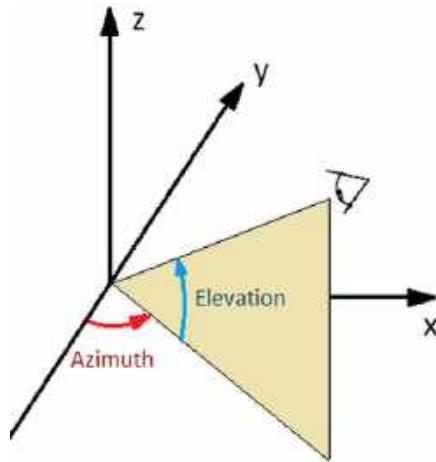


Figure 8.10: The diagram of the coordinate system with the azimuth (red) and elevation (blue) angles.

terns is discussed at the end of this section.

In the scenario in which we are sampling a new vertex neighbor, the final step of this process consist in taking a picture of  $T_v^r$  render, placing a camera along the positive  $z$  - *axis*, pointed toward the origin of the coordinate system, to that all the patch is contained in the field of view. In different terms, we project the model onto the  $z = 0$  plane, obtaining a 2D surface (i.e.: the picture mentioned before).

If instead we are creating a training set, we want to sample  $T_v^r$  from different angles. To do so, we rotate the model changing the azimuth and elevation angles of the viewpoint: see Figure 8.10 to understand the angle placements. For each azimuth-elevation angle combination, the same projection and rendering process is repeated.

Shape index and mean curvature are the two surface descriptors we selected for this approach. The first is much more sensitive to small variations, while the other is better suited to find bigger details of the surface, but still small enough to be patterns.

## 8.2.2 Training of the models

The training took place using a Jupyter Notebook and the popular deep learning library, *Fast.ai* [HG20], based on PyTorch. The hardware used was a GPU node of the new high-performance EOS cluster located within the University of Pavia. This node has a dual Intel Xeon Gold 6130 processor (16 cores, 32 threads each) with 128 GB RAM and 2 Nvidia V100 GPUs with 32 GB of video RAM.

The training was performed starting from 3006x1534 high resolution images rendered by our local feature extractor, properly ordered in labeled subdirectories. We chose to split the dataset randomly into training and validation sets using a 70/30 split.

As previously mentioned, the model chosen for training is a pre-trained version of a ResNet-50 architecture. The APIs of the *Fast.ai* allows users to download the pre-trained architecture and weights in a very simple and automatic way. *Fast.ai* also automatically modifies the architecture so that the number of neurons in the output layer corresponds to the number of classes of the current problem, initializing the new layer with random weights.

The training was performed using the progressive resizing technique, i.e. performing several rounds of training using the images of the dataset at increasing resolutions to speed up the early training phases, have immediate feedback on the potential of the approach, and to make the model resistant to images at different resolutions (i.e. the model generalizes better over the problem). The specific section in [FMN<sup>+</sup>19] explains very well the concept of progressive resizing. For our particular problem, we have chosen the resolutions of  $100 \times 51$ ,  $150 \times 76$ ,  $300 \times 153$ ,  $601 \times 306$ ,  $900 \times 460$  and  $1503 \times 767$  pixels (i.e. 1/30th, 1/20th, 1/10th, 1/5th, 3/10th and 1/2 of the original  $3006 \times 1534$  pixels resolution). We found empirically that raising the resolution above 1/2 of the original resolution provides no further improvement in the classification accuracy and has high training costs.

When performing transfer learning from a pre-trained network, each training round at a given image resolution is usually divided into two phases,  $a$  = convolutional layers are frozen and  $b$  = all layers unfrozen, each made by one or more epochs. For each epoch, all the training set is fed into the network, one batch at a time, with each image in the batch being altered by a random number of data augmentation transformations of different intensity or magnitude. Given that the last layer of the network is new and initialized with random weights, in phase  $a$ , the weights of all the layers of the neural network except those of the new output layer are frozen and therefore are not trained (they are used only in the forward pass of features extraction). During phase  $a$  a larger learning rate (LR) can also be used and for these reasons this phase is much quicker to converge than phase  $b$ , because backpropagation is applied only to the last, untrained layer and the gradient descent algorithm is allowed to converge faster. In phase  $b$ , performed with a lower learning rate, typically of one or two orders of magnitude less, all layers, even the convolutional ones, are trained to improve the network globally. However, for our problem, we found that there was no significant improvement in *unfreezing* the convolutional layers of the ResNet architecture, so we performed only the training of the last layer with fewer epochs and with larger learning rates.

The data augmentation transformations that we chose to apply to the images were: crop, flip left/right, symmetric warp, rotate, zoom, brightness, contrast and jitter (noise obtained by pixel shuffle among adjacent pixels). Examples of some of these augmentations are shown in Figure 8.11. The learning rate used was 0.01 for all the rounds except for the last one at  $1503 \times 767$  pixels resolution. The high amount of VRAM of the GPU we used allowed us to keep a high



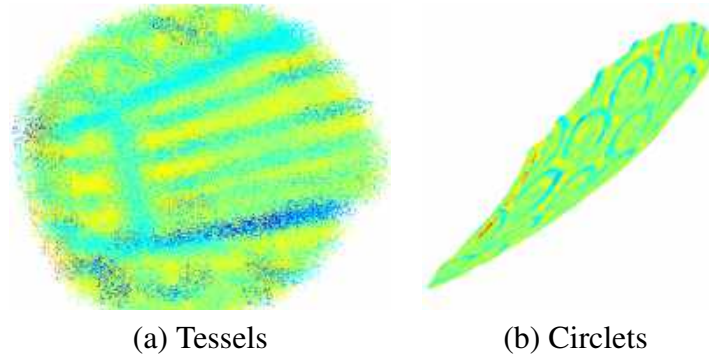


Figure 8.11: (a): Tessels, augmented with jitter noise (b): Circlets, warped.

batch size for all the phases of the training. We started with a batch size of 256 images and we ended with  $bs=12$  at the last round with images at  $1503 \times 767$  pixels.

As the neural network model optimizer, we chose Ranger as it combines two of the best state-of-the-art optimizers, RAdam [LJH<sup>+</sup>19] (Rectified Adam) and Lookahead [ZLHB19], in a single optimizer. Ranger corrects some inefficiencies of Adam [KB14], such as the need for an initial warm-up phase, and adds new features regarding the exploration of the loss landscape, keeping two sets of weights, one updated faster and one updated more slowly, and interpolating between them to improve the convergence speed of the gradient descent algorithm. We also used the *Fast.ai* callback hooks to automatically save the model if the accuracy on the validation set improved with respect to the previous epoch and to stop the training if the accuracy was above a predefined threshold, in this case 0.999 (or 99.9%) accuracy.

Once all the training rounds were completed, the model with the best accuracy was selected for the testing phase.

### 8.2.3 Tests on the Model set

For our tests we used the models in the pattern recognition benchmark dataset [BMTB<sup>+</sup>18]. With respect to the initial task of the benchmark, we only focus on finding areas of the surface that are characterized by different patterns (or no pattern). First, we took a set of samples on each model, using the Poisson sampling algorithm. The number of samples  $n_{samples}$  is obtained by making a proportion between the size of  $r$  and the surface area of the model. In particular, we want to obtain a set of patches able to cover the whole model. Thus, since a patch is roughly the size of a disk ( $\pi r^2$ ), we set  $n_{samples} = \lceil \frac{A(T)}{\pi r^2} \rceil$  (rounded to the closest integer value). Since the geometry of the models may be complex, we usually slightly over sample the surface (110% of the value  $n_{samples}$ ), so we are sure that we are able to cover the whole surface with the selected patches. For this dataset, we set  $r = 2$ . We are able to determine for each patch which

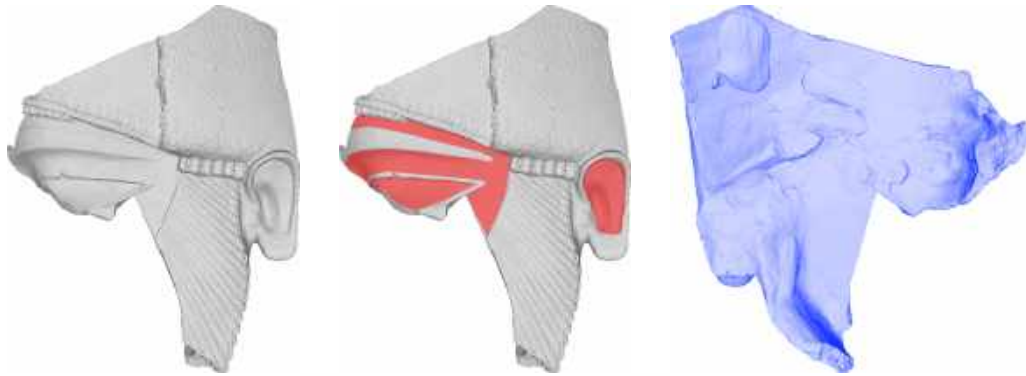


Figure 8.12: Example of the areas that we classify as *NoPattern*. Left: one model of the pattern recognition benchmark dataset [BMTB<sup>+</sup>18]. Center: smooth areas on the models (in red). Right: back facet of the models (in blue). Both red and blue areas on the models are labeled as *NoPattern* in the benchmark ground truth.

pattern it lies on, thanks to the benchmark ground truth. Moreover, with respect to the classes of patterns of the benchmark, during the designing of this test we foresee the need of an additional class resembling the patch with no pattern. In particular, we added the "*NoPattern*" label to our training set, which includes both areas without patterns (sort of "locally smooth" patches) and those on the back facet of the model. With the latter we intend those parts of the models that are not meant to be seen (like the inside of a vase) but that still appears in a complete digitalization of an object. Figure 8.12 shows examples for both kinds of patches that we consider *NoPattern*.

Our first test is on a net trained on the models *model set* (see Chapter 6). From each sampled point, we extract a patch that is rendered in 252 different positions, varying the azimuth angle from 0 to  $2\pi$  and the elevation angle from  $-\frac{\pi}{3}$  to  $-\frac{\pi}{3}$  with steps of  $\frac{\pi}{18}$ . With this test we want to understand the capability of the training process to learn from the data. The accuracy obtained is already above 0.99 on the first image size ( $100 \times 51$ ). The best accuracy (0.997393) is obtained in the second round ( $150 \times 76$ ). In Figure 8.13(Left) we report the normalized confusion matrix related to the accuracy of the training.

In our second experiment we tested the ability of the trained net to classify unseen data. To do this, we removed some models from the training set. The rest of the models are used for both training and validation tests. Again, the accuracy during the training is very solid (already 0.99846 on the second round ( $150 \times 76$ )). The normalized confusion matrix related to the accuracy is shown in Figure 8.13(Middle).

The results on the unseen models are, unfortunately, less steady. The overall accuracy on them is 0.75, thus only  $\frac{3}{4}$  of the points are classified correctly. The confusion matrix for this classification is shown in Figure 8.13(Right). Moreover, for a quick evaluation of the quality of the results, we report the classification results obtained on some of the models not considered in the training

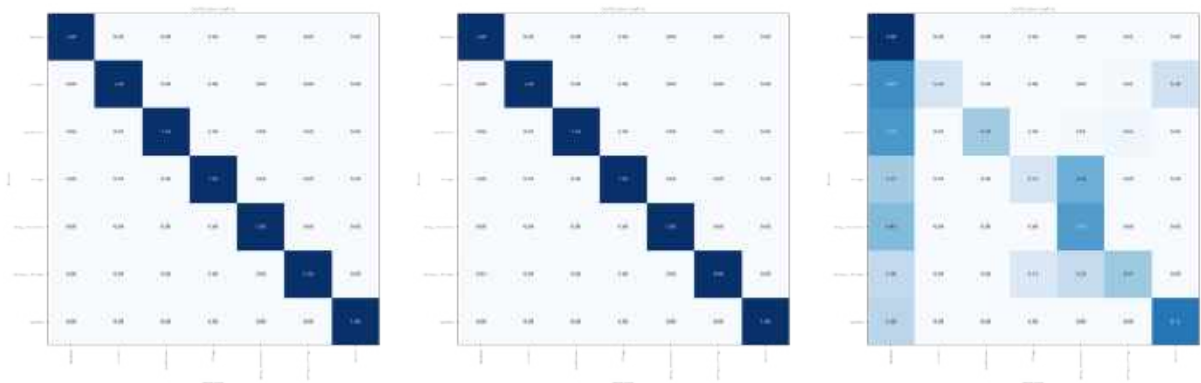


Figure 8.13: The normalized confusion matrices related to the accuracies of different stages in our tests. Left: the normalized confusion matrix of the accuracy of the training after the second round of training of our first test. Center: the normalized confusion matrix of the accuracy of the training after the second round of training of our second test. Right: the normalized confusion matrices on the unseen data after the third round of training of our second test.

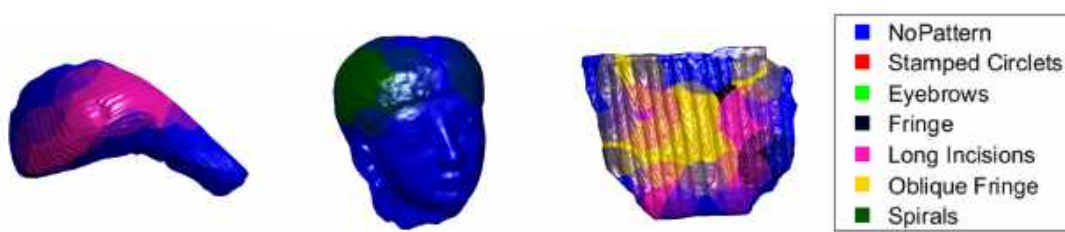


Figure 8.14: Visual representation of the classification obtained with net in our second test on the unseen models.

set. In particular, we painted the models based on the classification results: each patch (i.e.: each vertex in a patch) is painted with a color representing one of the 7 possible labels (*fringe-1*, *fringe-oblique-lines-2*, *long-incisions*, *small-bands*, *stamped-circles*, *spirals*, *NoPattern*). Vertices shared between two (or more) patches may be classified in two different ways, thus in that scenario we blend the results by mixing the two (or more) respective colors. Since there are more samples per patch, the classification of each one of them is not trivially defined. For example, the best view for each patch is the one perpendicular to the best fitting plane. However, it is naive to completely ignore the results obtained from multiple classification estimations of the same patch. However, we attempted both strategies, with almost identical results. Thus, we show the results for the classification obtained with the voting approach in Figure 8.14.

We notice that the different views of the same patch have almost always the same label. This fact may be interpreted in two ways. On one hand, it could mean that training the net with this multi-view approach helped in the identification of the pattern despite the different bendings. On

the other hand, it may mean that the information given by the multi-view approach is redundant and just a single view is enough to properly train the network. Of course, it is very possible that the truth may stay in between these two scenarios: indeed, the multi-view may actually help the net to better learn the patterns, but 252 different views may be too redundant.

## Conclusions

In this Chapter we present two of the learning approaches to the pattern recognition addressed during my PhD research activity. One is based on dictionary learning, which is a well known unsupervised learning method. The signals given to the dictionary are a local discretization of the neighborhood of a set of points sampled on the surface, similar to the mpLBP description of the model (see Chapter 4). Early results are promising because the method shows how it can be applied to pattern synthesis, however, for pattern recognition the method introduces a machinery that is computationally more complex than PATCH-CUT while having results that are comparable to those we can obtain with the PATCH-CUT method. The second proposed approach is based on local views of the neighborhood of samples on a vertex. Each neighborhood is observed from different point of views and an image is extracted from each view. The images are used to build a training set for a RES-NET 50. The method is therefore supervised and, as such, the experiments show a strong sensitivity to the extent and quality of the images used for training.

## **Part IV**

### **Future works and conclusions**

## Chapter 9

# A search engine for 3D shape collections

Visual search engines are the natural tools for supporting comparative studies of object collections and they are generally based on the combination of *shape descriptors* as signatures synthesizing the geometric content of 3D models [BKS<sup>+</sup>05, TV08], and of *similarity measures* for matching descriptors [SJ99, BCBB16]. In the last fifteen years, search engines have been addressed with a large number of content-based techniques aimed at detecting global shape similarity [SMKF04, CRC<sup>+</sup>02] or part-based search and retrieval in 3D object collections [KLM<sup>+</sup>12, LBZ<sup>+</sup>13] or specific contexts, like product design [CGK03, IJL<sup>+</sup>05], parametric shape collections [SSB<sup>+</sup>17] and cultural heritage [BCFS15]. As similarity is a cognitive process, the user's intent during the search should be included in the loop [SWS<sup>+</sup>00]: methods for relevance feedback were introduced for 3D object search [LMT05, GFSF10], while the first 3D search engine able to support user interaction appeared quite recently [Heal1] under the paradigm of the faceted meta-data.

The use of heterogeneous shape collections, like the Princeton Shape Benchmark<sup>1</sup> and 3D Warehouse<sup>2</sup>, is acknowledged by [HSS<sup>+</sup>13, KFLCO13, AKZM14, GCL<sup>+</sup>15] and, at the moment, the collection used in [GCL<sup>+</sup>15] is the largest one adopted for content-based exploration, with its 103738 3D models. These methods do not explicitly declare any particular assumptions on the type of the 3D shapes but do not permit a multi-facet exploration by a user-driven combination of different descriptors.

When reasoning about similarity in a collection of objects with heterogeneous properties, it is useful to both use the combination of some multi-modal information (e.g., geometry and texture) and also to process part of the collection in an interactive manner, by grouping items, or their parts, into meaningful clusters. This scenario is still quite far from the scenario depicted by the current state-of-the-art: most methodologies for comparing, retrieving, or classifying objects

---

<sup>1</sup><http://shape.cs.princeton.edu/benchmark/>

<sup>2</sup><https://3dwarehouse.sketchup.com/>

in repositories are based on a single analysis of the geometric 3D shape, possibly building on specific invariants, such as the presence of axis of symmetry [KPL<sup>+</sup>10] or appendages [KC11]. Even if several recent approaches for similarity assessment aim at identifying (dense or sparse) correspondences among the model elements (e.g., [OBCS<sup>+</sup>12, KLM<sup>+</sup>12, SKVS13]) or combining texture and geometry information (e.g., [BCFS15, BCA<sup>+</sup>16, GG16]), they actually pursue a shape matching at a global level rather than evaluating similarities based on the comparison of specific features. With respect to these strategies, the approach presented in this chapter addresses similarity on the basis of a set of aspects that describe the models, following an approach similar in spirit to the faceted meta-data proposed for images in [YSLH03]. This approach has been pushed by a challenging scenario for visual search, archaeology, where traditional approaches fail and where trendy approaches based on learning are not really effective due to the lack of large training sets and to the inherent complexity of capturing the search intent of users.

This chapter presents the design and experimental results of a search engine for 3D shape collections able to address queries and automatic search relaxation, mainly aimed at scouting archaeological datasets. This is a search framework able to adapt the navigation to the properties of the objects, which may differ for different collections, and adapt also to the goals of the search session, which may differ from session to session even if the user is the same. The search framework defined is flexible with respect to the choice of descriptors, and also to the inclusion of different similarity distances, depending on the properties one might be interested to capture.

## 9.1 Conceptual model

The proposed comparison framework acts as a query-by-example search engine that combines a set of properties, following the ideas of faceted metadata for images [Hea11, YSLH03]. Starting from a model  $A$  (the query model) of a given collection of 3D models  $R$ , the list  $L$  of models in  $R$  that are considered similar to  $A$  is retrieved. The search is based on the activation of a number of properties ( $P_1, P_2, \dots$ ) that the user selects before running the search. The design of the search engine is compatible with prior classifications of the types of the objects collection. For instance, in case of archaeological datasets like the GRAVITATE use case the type of the query object (sherd/non-sherd) determines what properties are considered valid search criteria based on the nature of  $A$  (e.g.: if  $A$  is sherd-like, the filter related to properties that works on non-sherd-like object are not activated).

Each property  $P_i$  is interpreted as a *filter* ( $F_i$ ), which removes from  $L$  the models that are dissimilar to  $A$  with respect to  $P_i$ . If the user judges the filter results too restrictive, it is possible to relax the filter severity incrementally adding more models to  $L$  and gradually enlarge the queue of models shown in the query window. Note that the list  $L$  could be empty.

From a more technical point of view, the single filters are distance matrices among models, one matrix for each property.



When the user selects one or more criteria to compare a query model against the dataset, each criterion acts as a filter. The combination of more filters is done on the basis of the logic *and* operator. In practice, the model  $A$  is similar to the model  $B$  with respect to the two properties  $P_1$  and  $P_2$  if the two models  $A$  and  $B$  are similar with respect to both  $P_1$  and  $P_2$ . If  $L_1$  is the list of the query results for the property  $P_1$  and  $L_2$  is the set of retrieved models for  $P_2$ , the set  $L$  of the models that satisfy both  $P_1$  and  $P_2$  corresponds to the intersection of the two sets, i.e.  $L = L_1 \cap L_2$ . With these settings, the combined distance is a metric if all the distances for the single properties are metrics.

The query results that fulfill all the criteria are ranked according to a combined measure defined as the product of all the distances that are smaller than a threshold  $t$ : this threshold acts as a tuning parameter for the granularity of the search results, using the criteria described below.

A different threshold is automatically set for each shape signature, as follows. Given a distance matrix  $D$ , the  $k$ -th row  $Q_k$  stores the distance of the  $k$ -th model with respect to all the other elements in the dataset. Given a threshold  $t$ , the set  $V_k$  of models such that  $Q_k(j) \leq t$  are considered valid query results for the model  $k$  with respect to the property stored in  $D$ .

The similarity between two models is defined as a score, represented in terms of a non-negative, real value that translates the distance between two signatures in a number. Depending on the shape signature adopted and the variety of the elements in a dataset, the image of the similarity distance spans different intervals of values, unless the similarity distance is normalized into interval  $[0, 1]$ . In this framework, it is preferable to avoid normalizing the similarity distances and then combining the similarity scores. If a unique, global maximum is not known a priori, the normalization would depend on the dataset, and the similarity distance could be biased by the context. For this reason, the following is a description of the procedure to determine the initial thresholds for dataset exploration, adaptively tuning them to the dataset variations. For each matrix  $D$ , the threshold  $t$  is automatically predetermined as:  $t = average_k\{t_k\}$ , where  $t_k = argmin\{\#\{V_k\} \approx 5\% \text{ of the elements in the model collection}\}$ . the symbol  $\#$  means the set cardinality. Since  $t_k$  is an average value, the size of  $L$  varies from model to model and the list  $L$  could be empty.

By tuning the thresholds values, it is also possible to enlarge the search result set, offering more flexibility and interaction to the user. A value  $dt$  is determined as the average of the  $dt_k$  values that increase the number of elements of  $V_k$  of approximately 3%. The value  $dt$  is selected as many times the user decides to relax the threshold  $t$ . The thresholds values  $t$  and  $dt$  are automatically determined for every property and dataset.

A schema of the components and behaviour of the search engine is shown in Figure 9.1. The schema also sketches the layout of the graphical user interface, which has been fully developed in the GRAVITATE project (Figure 9.1(Bottom)).

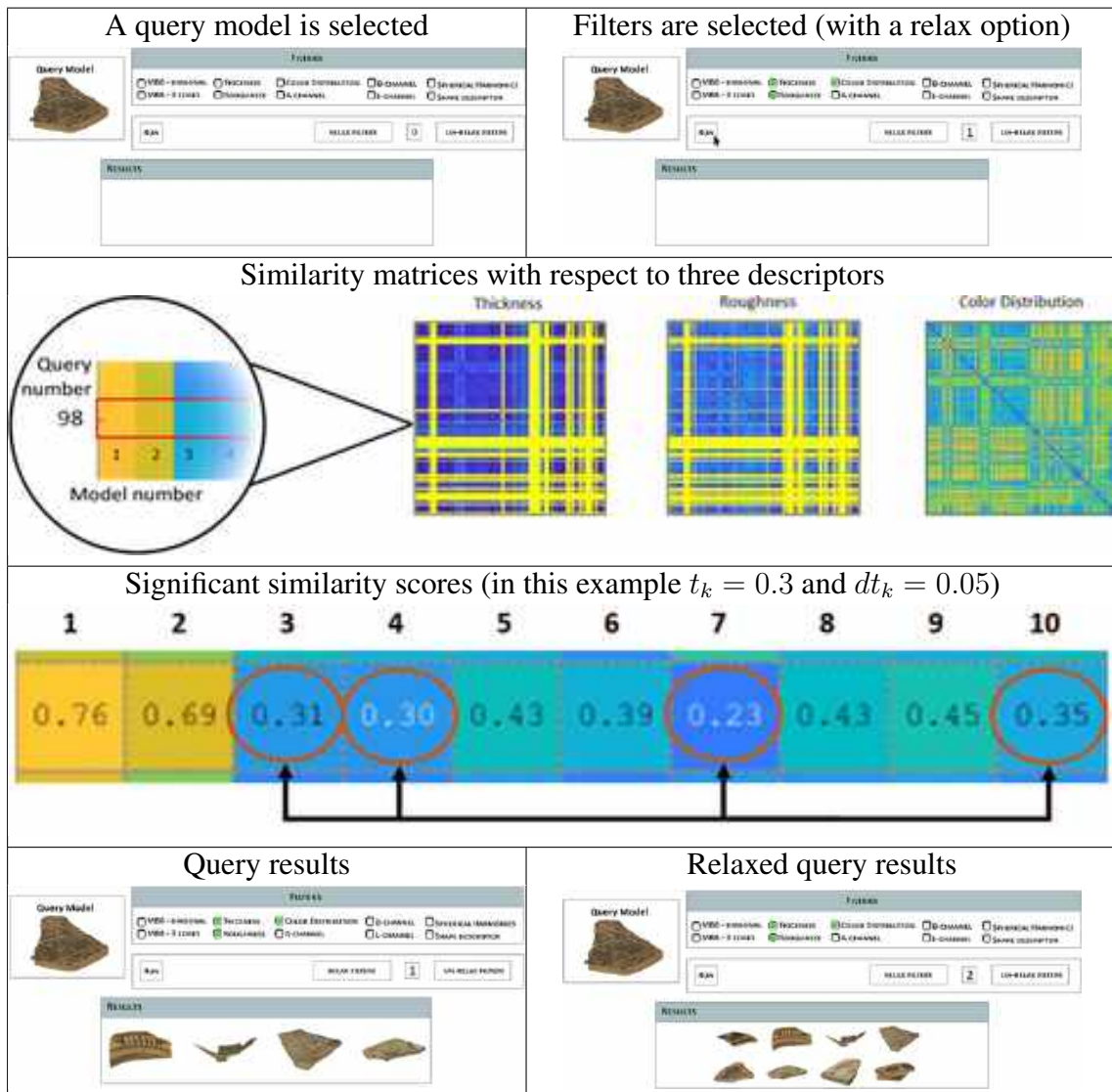


Figure 9.1: An overview of the search engine behavior using three descriptors on a model of the Naukratis collection and the effect of a further relax.

## 9.2 Similarity assets

We need to define a set of similarity criteria that can be identified with or translated into computational tools. This can be done either resorting to state of the art descriptors methods, or by implementing new ones if none can provide the behaviour sought. Since each filter is based on a distance, the properties can be seen also as axes of a Euclidean space of dimension equal to the number of filters. As driving guideline for associating a shape descriptor to a similarity axis, the

following rules are kept in great consideration:

- *simplicity* of the descriptor. If two or more descriptors are suitable for a task, the simplest one is selected, in terms of output storage and matching complexity (e.g., real numbers or feature vectors);
- *scalability*, depending on the target detail (e.g. a chiselled decoration), it is necessary to compare high-resolution 3D models with millions of vertices. Therefore, it is worth favouring the shape signatures that are able to deal with the larger meshes (in term of vertices and faces);
- *coherence*, meaning that the shape signatures should rank the query results with respect to a model as close as possible to the similarity types identified by the archaeologists.

Moreover, it can happen that a single descriptor is not enough to fully capture the complexity of a similarity criterion or is able to capture it only partially; in these cases, either a combination of descriptors or present multiple choices to the user is proposed.

In the description below, the term *compatibility* is used instead of *similarity* to discuss the mapping from criteria to descriptors, to emphasize that each similarity measure contributes to a reasoning rather than retrieving a *crisp* result. In other words, each criterion acts as a filter with respect to a specific property and it is not meant to return the whole similarity assessment. To give an example of this effect, when selecting a colorimetric property when the query model is mainly red, the results are expected to include models with a predominance of the red texture, independently of their shape. In the following, the implemented descriptors are defined.

1. *Compatibility in terms of the overall size*. The object size of the whole object is identified with the volume occupied by the object, therefore the oriented minimal bounding box is used as an approximation of such a volume. Two descriptors are used to map this criterion: the diagonal of the minimal bounding box (MBB) and the vector of the length of the three sorted edges of the MBB (MMB(edge)). The hull packing value (i.e., the ratio between the minimum and maximum edge of the minimal bounding box) is considered as well, as proposed in [CRC<sup>+</sup>02] but, in the experiments shown in this manuscript, the MMB(edge) value seems to provide a higher discriminating capability. Figure 9.2 depicts the meaning of the descriptors based on the minimal bounding box on a GRAVITATE fragment. The distance between the descriptors is the  $L^1$  distance, which works nicely as the descriptors are scalar values or three-dimensional vectors.
2. *Compatibility in terms of the thickness*. The thickness of a model is computed as the shape diameter function (SDF) [SSCO08] that has been shown to provide a stable approximation of the diameter of a 3D object with respect to a view cone centered on the object surface and with the axis aligned to the surface normal. After computing the SDF function on



Figure 9.2: Left: the diagonal and Right: the three edges of the MMB.

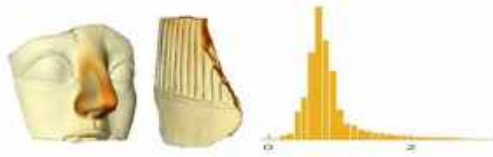


Figure 9.3: Left: Two fragments are colored according to the local SDF value, ranging from white-yellow (low values) to red (high values). Right: the distribution of the SDF function; the abscissa of the maximum value of the histogram corresponds to the thickness descriptor.

each point of the fragment surface, the thickness descriptor is defined as the average of the most frequent value. Other possible choices for the thickness descriptor are possible. For instance, the thickness around fragment profiles or fragment fractures [SA16, ASP17] could be considered. However, these methods generally require a heavy pre-processing step, often not completely automatic, either in terms of the detection of fragmentation orientation or identification of the fracture creeks. With reference to Figure 9.3, the thickness descriptor corresponds to the bin that scores the maximum value of the histogram. The  $L^1$  distance is used to compute the distance between two thickness descriptors.

3. *Compatibility in terms of the roughness.* Roughness is the grain and finish of the fragment surface. Often, this property is quantified by the deviations, in the surface normal direction, of a real surface from its ideal form. It is often associated with local, shape texture-like properties [GKM<sup>+</sup>02]. Among the possible descriptions, the mean curvature and the shape index are two possible geometric properties for representing these local surface variations [PT96, KvD92]. The value of the minimum  $k_1$  and maximum  $k_2$  curvatures at each vertex are computed adopting the implementation presented in [CSM03]. The mean curvature  $K$  and the shape index  $SI$  are derived for each vertex, as  $K = \frac{k_1+k_2}{2}$  and  $SI = \frac{2}{\pi} \arctan\left(\frac{k_1+k_2}{k_1-k_2}\right)$ ,  $k_2 \geq k_1$ , respectively. In the following experiments, it is possible to see that the mean curvature better highlights the roughness of a model, thus it is used as the descriptor for this property. The corresponding descriptor is defined as the histogram of the distribution of the mean curvature, in an interval of values that holds for the whole collection and is determined on the basis of the overall curvature variation over the collection. The Earth Mover's distance [RTG00] is adopted as the distance between



Figure 9.4: Lest: The hairs of the statue are represented on the fragment by a regular relief pattern. Right: the corresponding mean curvature distribution histogram.

two histograms.

4. *Compatibility in terms of the color distribution.* In the considered meshes, colorimetric information is associated with the vertices in terms of a RGB value. However, it is well known that the Euclidean distance in the RGB space does not correspond to the perceived distance between two colors. The CIELAB color space was designed to be perceptually uniform with respect to human color vision [AKK00], meaning that the same amount of numerical change in its values corresponds to about the same amount of visually perceived change. Therefore, the Euclidean distance in this space is a good approximation of the perceived color distance. Among the descriptors adopted in the literature to code the colorimetric information for objects, the concatenated histogram of the three color channels (L, a and b) in the CIELab [Suz01, BCFS15] (see Figure 9.5) space are considered, for its simplicity.

A 100-bins histogram is computed for each color channel, for all the models. Histograms are clamped between the overall minimum bin-value and the overall maximum bin-value that are not empty. The color descriptor is composed by the concatenation of the three clamped histograms. Using the GRAVITATE use case as example the first 60 bins correspond to the interval [20,80] for the variation of the L-channel, the next 35 bins correspond to the interval [-15,20] of the a-channel and the last 65 bins correspond to the interval [-15,50] of the b-channel. The color descriptor for each model of this collection is a 160-bins histogram. The single CIELAB channels contain a specific colorimetric information: the L channel represents the lightness, while the a and b channels respectively represents the green–red component, with green in the negative direction and red in the positive direction and the blue–yellow component, with blue in the negative direction and yellow in the positive direction. For this reason, the histogram of the single color channels are also considered as three separated descriptors, for a total of four color descriptors. Being the CIELab a non uniform space, the Earth Mover’s distance is used to compute similarities between two color histograms [RTG00] to compensate the problem of non-uniformity.

5. *Compatibility in terms of the overall shape similarity:* Due to the large variability of shapes within the collections, the selection of a geometric signature yielding good results for all

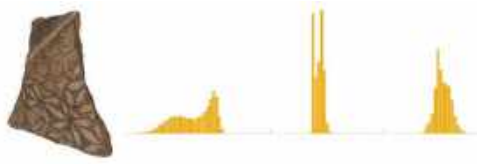


Figure 9.5: A fragment and the corresponding concatenated color histogram with respect to the L, a and b channels.

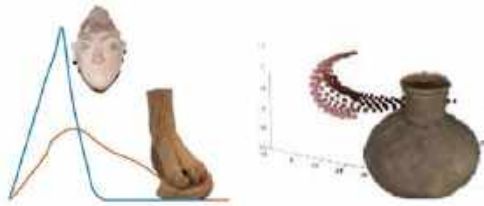


Figure 9.6: Left: The non-normalized  $D_2$  signatures of two fragments; Right: the persistence space with respect to the distance from the center of mass and the average, geodesic distance for a vase model.

shapes is challenging. For instance, in these use cases, the collection is made of heads, feet, legs, busts, vases, broken arms and hands, and these artifacts can have different scales (e.g., small and big statues) or can have cracks and missing parts. The quite limited number of elements per class also prevents the use of learning techniques. A combined descriptor is used, which mixes rough filters with scale invariant and non-rigid descriptions (called simply *Shape* descriptor). Namely, it is a combination of multiple global signatures, like compactness and hull packing [CRC<sup>+</sup>02], the spherical harmonics indexes [KFR03], a non-normalized variation of the shape distribution signature with respect to random chords (this signature codes the probability distribution of the distance between two random points on a surface and is called  $D_2$  in [OFCD02]) and the persistence spaces computed according to the average geodesic distance and the distance from the main axis of an object [BCFG11].

Figure 9.6(left) represents the non-normalized shape distribution  $D_2$  [OFCD02] for a head and a foot model and Figure 9.6(right) shows the persistence space obtained with respect to the distance from the center of mass and the average geodesic distance, evaluated according to [BCFG11]. All the distances adopted in this setting are metrics in the descriptors space.

As a second global shape descriptor, the Spherical Harmonics indexes [KFR03] alone is considered, which is generally efficient for rigid matching and comparison of objects with spherical, or at least cylindrical, symmetries [SSB<sup>+</sup>17].

It is worth noticing that the selections made for descriptors are not, and cannot be, guided by knowledge about the performance of specific descriptors on specific classes. Rather, the selection is made on the prospects opened by their combination in a search engine: there might be

descriptors in the current literature that are better suited for a specific similarity task. However, the goal is to investigate how a user can navigate a dataset based on a given set of computed descriptors that capture different properties, like orthogonal axes in the similarity space.

Finally, we highlight how our research on pattern recognition fit into the experts' requests: their need of searching object based on local characterization, like patterns, is generally not satisfied by the linear combination of the proposed assets and calls for more interactive and flexible tools.

### **9.3 User-driven dataset navigation**

The considered collections are organized with respect to criteria that are different with respect to the similarity criteria that the archaeologists have identified as suitable for their findings. Indeed, these collections are organized through factors like the functionality of the models or the provenance and it is really complex (or impossible) to define a single descriptor that is able to assess such tricky classifications. Since the search results lack a rigid ground-truth, their correctness is judged based on their visual appeal, minding also the similarity criteria considered in each query search.

The outcome of some queries on different datasets is discussed in the following. Although the quality of the query results is obviously dependent on the descriptor(s) selected, those examples are enough to highlight how the combination of multiple properties can tailor the engine to the users needs.

As an exception, the experts provided a limited ground-truth for 46 of the 72 fragments in the Naukratis collection, identifying 8 groups of shards that are likely to belong to the same object. This allows for a quantitative analysis of the combination of two descriptors against the single ones, highlighting the advantages in using the search engine as a navigation tool instead of considering the single descriptors.

#### **The GRAVITATE case studies**

The primary GRAVITATE [UUTCIC<sup>+</sup>] collection is composed by fragments of terracotta figurative statues discovered in Salamis, on the island of Cyprus, dating back to the seventh - early sixth century BC [KKF91]. Most of these statues are fractured, while most shards are faded and eroded. The project use-case has 241 digital models of Salamis statues fragments, acquired by laser scans.

Another set of models come from the Naukratis collection at the British Museum [Mus17]. It is a collection of pottery vessels fragments from Naukratis, a Greek trading port on the Nile Delta, in Egypt, dated from the VII century BC to the VII century AD. The collection available for these

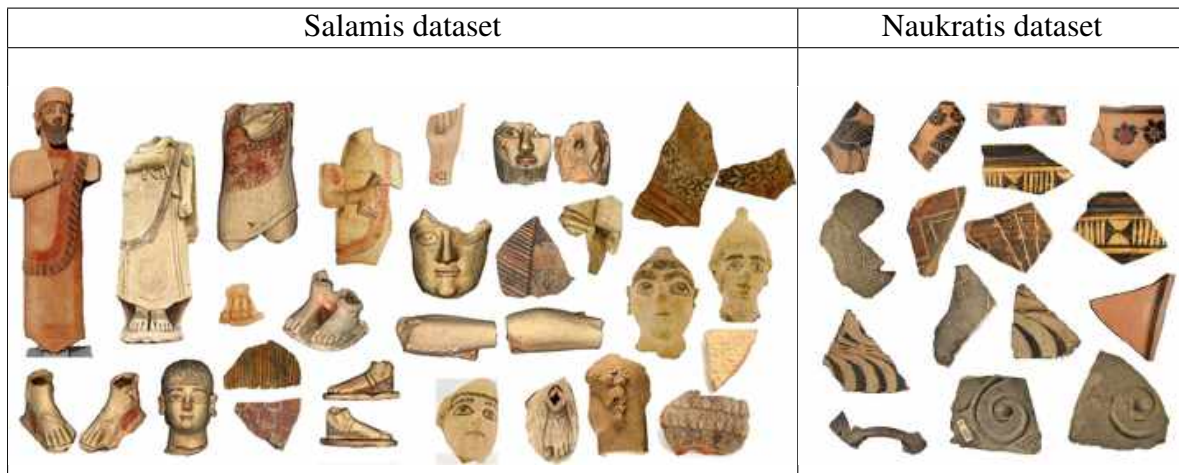


Figure 9.7: Examples of fragments of Salamis terracotta figurines (right) and of Naukratis potteries (left).

studies is made of 72 digital fragments, acquired by photogrammetric scans. Figure 9.7 shows some examples of the fragments in the GRAVITATE use cases.

A preliminary distinction of the fragments in the Salamis collection is done. Two categories are defined, one for fragments shaped like a classical sherd and one for those representing still a volumetric component of the original object. This distinction was validated by archaeological experts. For instance, with reference to Figure 9.7 (right), the heads, the figurines and the busts are examples of volumetric fragments, while the decorated fragments (e.g. top-right and bottom-right) or the fragments in Figure 9.7(Right) are examples of sherd-like fragments.

Figure 9.8(A-F) shows a set of query results for the GRAVITATE use cases. Rows 9.8(A-B) present the same query model with two different query options: in (A) only the Color Distribution filter is adopted, thus admitting models with different geometric shapes. The combination of Color Distribution with Roughness is shown in 9.8(B) and provides a large set of fragments that are intuitively more similar (in terms of red-like color distribution and surface smoothness) to the query model. Row 9.8(C) presents the combination of Color Distribution, Roughness and Thickness at once. Again, the results are quite intuitive. The archaeologists classified the first six fragments (the query models and the first five results) as potential elements of the same vase. Nevertheless, also the other elements of the row 9.8(C) present quite homogeneous properties. Similarly, the example in row 9.8(D) highlights how using only Color Distribution and relaxation on the threshold it is possible to select all the fragments in the dataset that potentially belong to the same group because they are made of the same material. One of the peculiar characteristics of this group is that all its elements are made of fired clay, while the Shape, the Roughness and the Thickness of these fragments is largely variable. Finally, rows 9.8(E-F) look at the different capability of the global shape descriptors. The Shape property corresponds to a combination of



global descriptors that are able to deal with both volumetric and topological shape distribution, thus mixing the head model with other heads but also hands. On the other hand, the Spherical Harmonics is a scale, invariant, global representation where the overall shape of the object is represented with the coefficients on the spherical harmonics computed with respect to the center of mass: besides the heads in the dataset the query results are tricked by cylindrical objects, like the last model in the result set.

## The Hampson collection

The second collection of archaeological artifacts is provided by the Virtual Hampson Museum<sup>3</sup> in the form of textured, triangle meshes. The dataset comprises 442 models, 395 of them are available for download, representing remains of Native Americans groups that were living in the northwestern portion of the central Mississippi valley from about 1450 to 1650 AD and that are referred to by archaeologists as the Nodena phase. An overview of this dataset is shown in Figure 9.10.

Figures 9.8(G-J) report some query results for the Hampson collection. The example 9.8(H) is based on the size of the model (described using the MBB(edges) descriptor) and shows us the possibility to navigate the dataset considering only fragments of a specific overall size. In this case, all the five models in the class are retrieved, without any false positive. The rows 9.8(G,I) combine other properties, such as Shape or Color Distribution, with the model size. The outcome of these queries highlights that the combination of properties targets the query results to the query models. Finally, the row 9.8(J) shows an example of query results when combining Shape with Thickness when the query model is a bowl-like model.

Most of the examples on the Hampson collection highlight that a threshold relaxation is often applied. Indeed, when acting with multiple filters it is possible that the queue of the query results (i.e., the set of models satisfying all the properties selected) is empty. This is not surprising because the collection is not *complete* in the sense that there are no examples for all the possible property combinations. Moreover the thresholds are automatically set roughly evaluating the property variance on the whole collection and the query queue of some models can be very short. To overcome this effect, the *relax* option is crucial, allowing the user to complete the query search desired without being forced to ignore one of the similarity criteria.

## YCB Benchmark

This benchmark contains a set of 71 3D models generated from visual data that are commonly involved for human-robot interaction. These models are available at <http://ycb-benchmarks.s3->

---

<sup>3</sup>VHM, <http://hampson.cast.uark.edu>

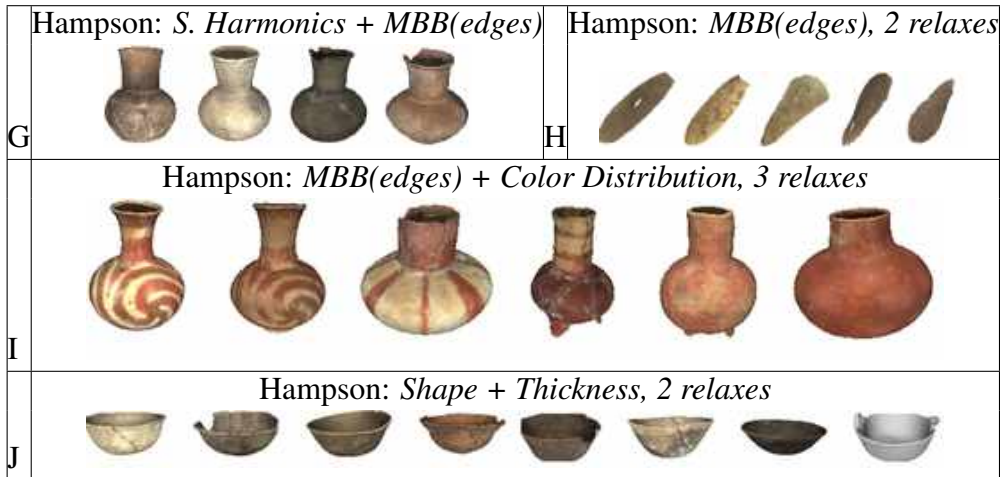


Figure 9.8: Examples of query search on models of the GRAVITATE use case and Hampson collection. The first model in each cell is the query model.

[website-us-east-1.amazonaws.com/](http://website-us-east-1.amazonaws.com/). The dataset contains everyday objects whose common trait is the fact they can be grasped with a robot, see Figure 9.11. Data was acquired with the scanning

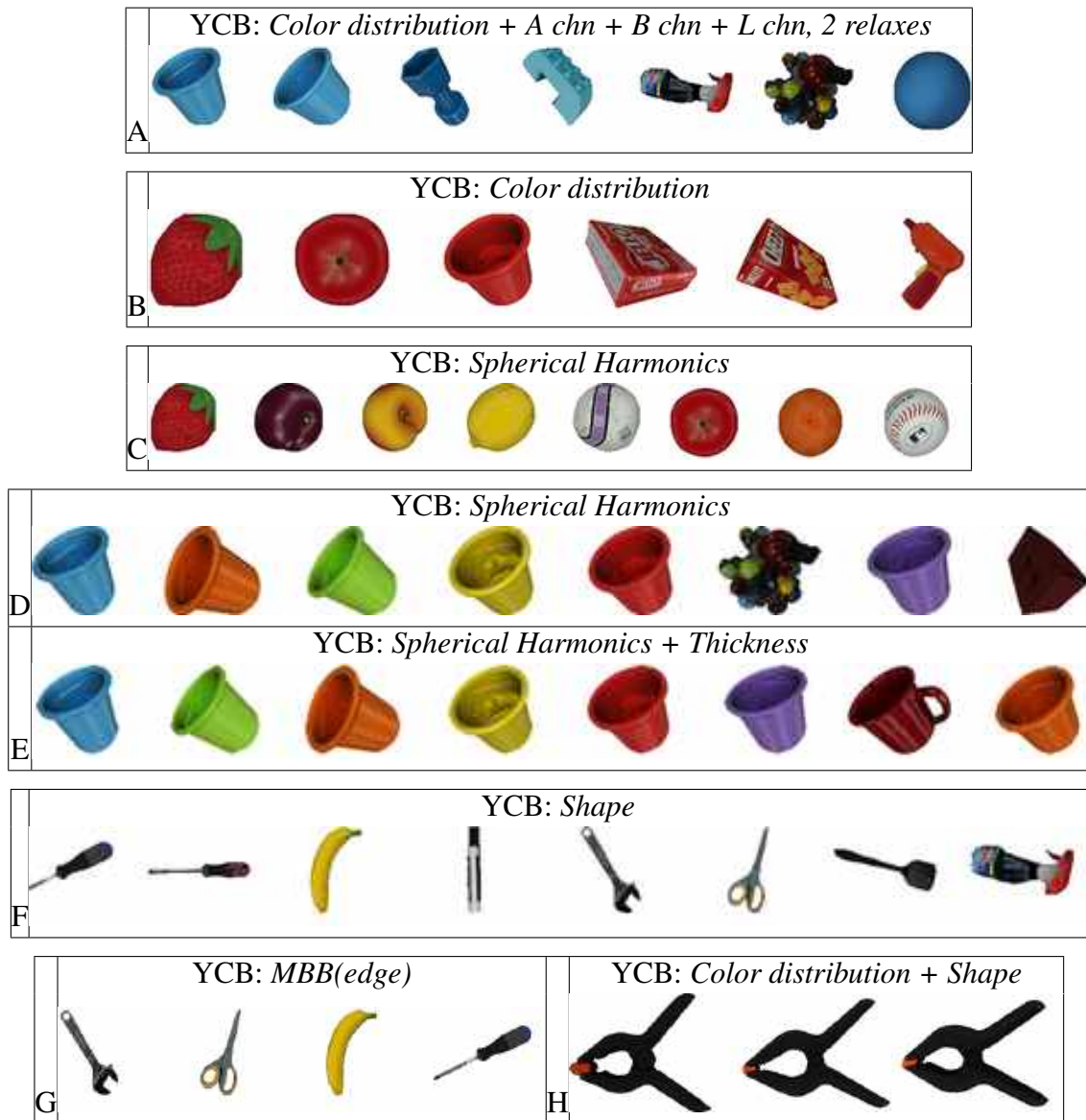


Figure 9.9: Examples of query search on models of the YCB dataset. The first model in each cell is the query model.

rig used to collect the BigBIRD dataset. Details on the dataset can be found in [CWS<sup>+</sup>15, CSB<sup>+</sup>17]. Aiming at a generalization of the criteria identified for archaeological context, the model properties are revisited in order to identify the search axes in the conceptual dimensions of the similarity space that could be useful in generic contexts. This is not meant to argue, obviously, that these are the only and the most relevant ones. The axes are *overall size*, *roughness*, *thickness*, *color* and *overall shape*.



Figure 9.10: Examples of fragments from the VHM dataset.

These criteria are tested over the YCB collection that presents a large variety of colors, shapes (spherical-like, elongated, kitchen utensils, boxes with sharp edges, cans, etc.). The examples in Figure 9.9(A-B) shows that the Color Distribution property is far more effective in a dataset with such a larger variety of colors than the GRAVITATE use case and the Hampson dataset, which are populated of artifacts that are mostly brown (terracotta) with colors ranging from yellow to red, at most. Figure 9.9(C-E) instead shows searches done using the Spherical Harmonics as global shape signature. The results are satisfactory, as the closest models have similar global shape to that of the query model. The addition of the Thickness to the query search 9.9(E) highlights how this property keeps models with a shape distribution similar to the query cup model. Similarly, Figure 9.9(F) provides an example of query results when the combined shape signature is considered. Still, some false positives pop up in the results, but this is due to the nature of the descriptors and the fact that the false positives are actually very similar to the query models in terms of elongated shape distribution. Finally, the examples in 9.9(G-H) show examples of queries under overall size and the combination of color and overall shape for some utensil-like models.

## Discussions and quantitative evaluation

The aim of this search engine is to provide a tool for the navigation of 3D model collections where multiple similarity axes can be identified. Depending on the user needs and tasks, the similarity notion can be different and vary in time; for instance, a professional user could be interested in details like manufacturing aspects and specific decorations, while a more generic user could search for all statues with black stripes. The targeted data collections generally are not fully



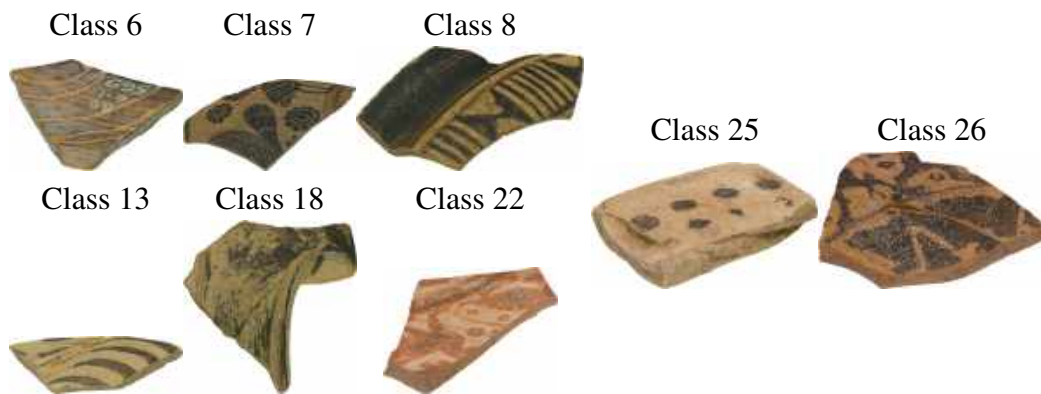


Figure 9.12: The list of classes used for a quantitative evaluation on the Naukratis dataset.

search engine when that model is adopted as the query. Models that belong to the same class have the same color. In the diagrams on the left, for each model, the percentage of the elements of the same class of the query model correctly retrieved (true positives) is reported. The value 1 means that all the elements of the class are shown in the list of the query results, 0.5 means that on a half of the elements are correctly retrieved when that fragment is adopted as the query. In the diagrams on the right, the number of false positives that are present in each query is reported. Theoretically, this number could equal the dataset size less the elements of that query group. It is possible to observe that some of the classes (the first, third, fifth, sixth and seventh classes) for *Search 1* and 2 have comparable percentages of true positives and numbers of false positives. For *Search 3*, while the percentage of true positives is slightly lower overall, the number of false positives is far away lower than in the other searches. Overall, these results show how the combination proposed in this search engine works and highlight that it can manage queries based on multiple descriptors diminishing the number of false positives in the query results.

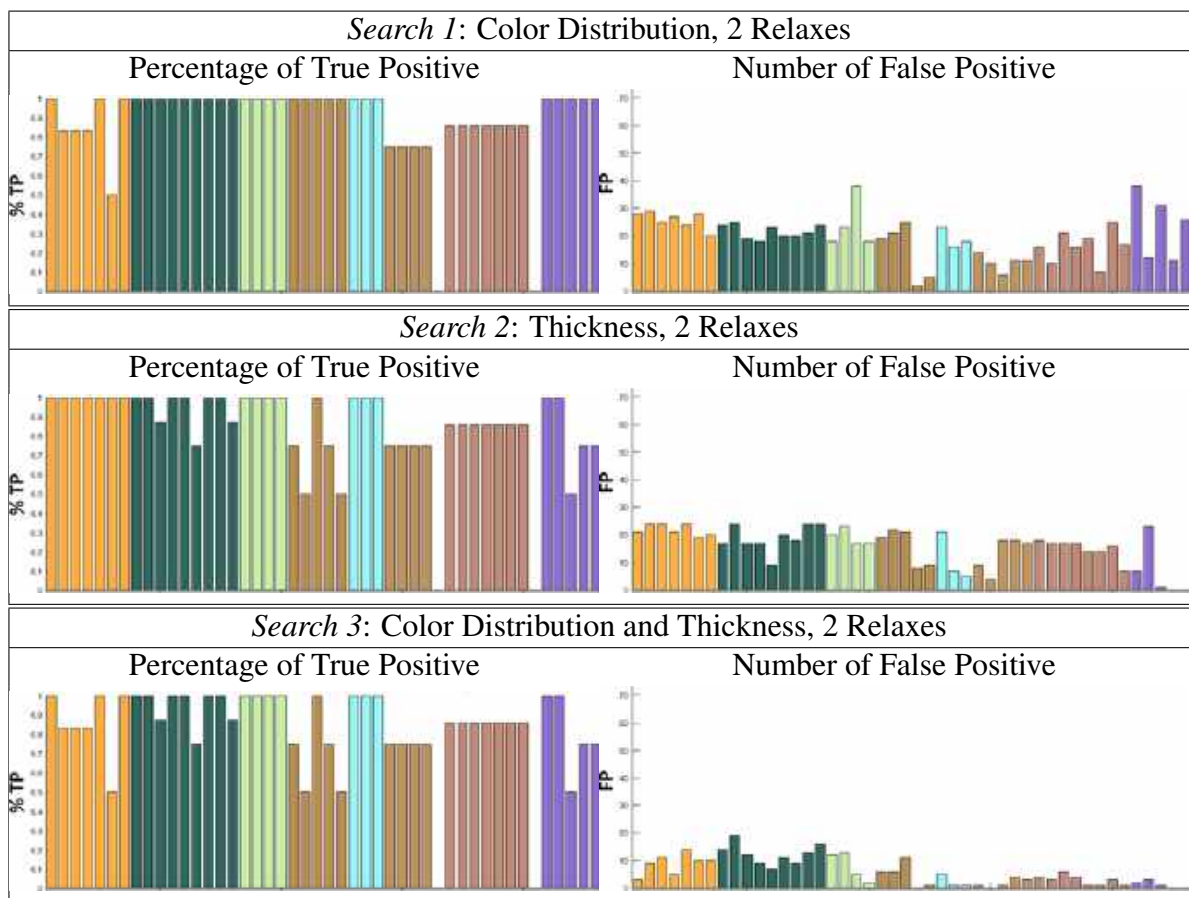


Figure 9.13: Qualitative test on the Naukratis collection. Each column represent the a query search based on a Naukratis model that has been labelled by the experts.

## Conclusions

In this Chapter we presented the architecture of our query-based search engine. For each dataset, a set of similarity criteria are selected and converted into quantifiable similarity measures. These assets are then combined based on the user request. We showed that such search engine can work in multiple domains. At the time of the publication of the search engine [BMTS18], pattern similarity was not included in the search criteria. With the techniques developed in this thesis, and in particular the contribution in the Chapter 7, we foresee the use of techniques such as PATCH-CUT to add pattern similarity to the search engine.

## **Related publications**

- S. Biasotti, E. Moscoso Thompson, M. Spagnuolo, *Experimental Similarity Assessment for a Collection of Fragmented Artifacts*. Eurographics Workshop on 3D Object Retrieval, 2018.
- S. Biasotti, E. Moscoso Thompson, M. Spagnuolo, *Context-adaptive navigation of 3D model collections*. Computers & Graphics, 2019.



# Chapter 10

## Concluding remarks

In this manuscript we faced the characterization and recognition of patterns on the surface of digital 3D models. Our studies on pattern description consolidated in two methods (edgeLBP and mpLBP) able to successfully deal with the pattern retrieval problem. Both the edgeLBP and the mpLBP achieve state of the art performances on several benchmarks and are able to deal with other datasets just by tuning three key parameters, without changing any step of the core pipeline. Moreover, we showed that there is not an unique parameter setup that leads to good results: indeed slight variations of the parameters do not significantly jeopardize the results. Both edgeLBP and mpLBP descriptors are robust to noise and different mesh/point cloud resolutions.

These pattern characterization results act as a basis towards a more general pattern characterization on models with more complex pattern(s) distribution(s). The PATCH-CUT method proposes a general approach to the local characterization of surfaces based on chiseled geometric patterns. To our knowledge, this is the first solution to the pattern recognition problem able to deal with a generic surface. The results obtained so far are promising and allows for an efficient pattern recognition across multiple models.

The whole work is also supported by a significant benchmarking activity that we carried on during these years. Each benchmark starts as a contest and, for each one of them, we created a dataset (great efforts went in both the dataset creation and in its ground-truth definition) and coordinated different teams of researchers. Three benchmarks on pattern retrieval, in particular, serves as the "litmus test" for the respective trends regarding this problem. Each time, we proposed a different kind of pattern or contest to be addressed (colorimetric patterns, geometric patterns, real river-bed scans). Rather than being a failure due to the lack of methods able to solve the pattern recognition contest, it became a starting point for our work and lays the foundations for further research on the topic of pattern recognition. Two other benchmarks address some of the challenges we encountered during the setup of our research and, to our knowledge, consolidated into contributions that at the current state of the art are unique. For instance, curvature

estimation benchmarks usually consider, as test cases, models with known curvature, limiting the judgment of the goodness of the methods performance when they are applied to object scans.

As future research perspectives, we foresee several improvements. As a technical solution, we think that all methods for geometric pattern analysis could benefit from a light pre-process of the models the most damaged or barely noticeable reliefs from a detail enhancement algorithm [BDC18]. However, its use must be carefully pondered because on one hand it emphasizes the model decorations but on the other hand it also emphasizes the model imperfections.

Regarding the pattern recognition methods presented in this thesis, we observe that the characterization of patterns using the dictionary learning method leads to an inefficient pipeline (in terms of the complexity of the pipeline and time with respect to the PATCH-CUT method) but it is still able to achieve satisfying recognition results and contains in its description a wealth of information that would be useful for further researches on pattern reconstruction or synthesis. Indeed, further developments could exploit the synthetic description provided by the dictionary learning to virtually create models decorated on the basis of a given template pattern.

Similarly, the multi-view RESNET50 method can greatly benefit from a more in-depth analysis of the training process possibly considering the creation of synthetic simulations of a model abrasion and, more generally, a model degradation. Besides technical improvements of the PATCH-CUT method like a more precise core definition, which is the main cause of the partial failures on the trickier models, one of the major advantages of the method PATCH-CUT is its applicability to domains where only a few pieces (full artifacts or fragments) are available, such as in the case of Cultural Heritage, and therefore does not require data for training. This opens up interesting perspectives, and although so far validated mainly on the use-cases of the GRAVITATE project, it promises to be easily extended to other archaeological collections, e.g. of Egyptian or Roman origin.

The application of our findings will add a new facet to those available for the exploration of collections of 3D models. We proposed a search engine that, from a query model and a set of constraints (on global and local characteristics of the objects), gives back a list of models in the dataset that are similar to the query model on the basis of the constraints selected. In this context, an application of the PATCH-CUT method on the query model will allow the user to select one of the final labels obtained (i.e.: a pattern on the model) and to compare its descriptor to those computed on the other models. A simulation of the difference between a run with or without the "pattern-based facet" is shown in Figure 10.1. Moreover, our results may fit into already established annotation systems for Cultural Heritage artifacts [CVHS20], adding crucial information that supports the cross characterization of models based on the decorations that can be classified as patterns (as intended in this work). Eventually, a validation of both an enhanced search engine and an enhanced CHAP notation, can greatly benefit from a validation with true field experts, such as the archaeologists involved in the GRAVITATE project.

Other applications of our methods are possible also outside the CH field. The benchmark on



Figure 10.1: An example of a query search in the GRAVITATE dataset with two different sets of constraints. Top: the query model (left) and the results of the search based on the general color distribution on the whole model (right). Bottom: the same search adding the constraint the models retrieved must contain the same colorimetric pattern of the query model.

river beds and previous contributions [OVSP13] show that a number of naturalistic problems may benefit from a general pattern recognition method. Even the biological research field can take advantage of the pattern characterization. For instance, the way bio-molecules interact with each other (docking) are influenced by factors of chemical and geometric nature. While the precise geometric configurations that permit the docking between two bio-molecules are hard to

classify, recent works pose the docking problem as a pattern recognition one and address it with learning-based methods based on curvature and scalar chemical physical properties [GSM<sup>+</sup>20].

# Bibliography

- [AD17] Hiteshwar Kumar Azad and Akshay Deepak. Query expansion techniques for information retrieval: a survey. *Inf. Process. Manag.*, 56:1698–1735, 2017.
- [ADBA09] Vedrana Andersen, Mathieu Desbrun, J. Andreas Bærentzen, and Henrik Aanæs. Height and tilt geometric texture. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Yoshinori Kuno, Junxian Wang, Jun-Xuan Wang, Junxian Wang, Renato Pajarola, Peter Lindstrom, André Hinkenjann, Miguel L. Encarnação, Cláudio T. Silva, and Daniel Coming, editors, *Advances in Visual Computing*, pages 656–667, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- [AF06] Marco Attene and Bianca Falcidieno. Remesh: An interactive environment to edit and repair triangle meshes. In *Proc. SMI'06*, page 41. IEEE Computer Society, 2006.
- [AKK00] Elif Albuz, Erturk D. Kocalar, and Ashfaq A. Khokhar. Quantized cielab\* space and encoded spatial structure for scalable indexing of large color image archives. In *Acoustics, Speech, and Signal Processing, ICASSP '00*, volume 6, pages 1995–1998, 2000.
- [AKZM14] Melinos Averkiou, Vladimir G. Kim, Youyi Zheng, and N. Mitra. Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. *Computer Graphics Forum*, 33, 2014.
- [APM15] Anthousis Andreadis, Georgios Papaioannou, and Pavlos Mavridis. Generalized digital reassembly using geometric registration. In *2015 Digital Heritage*, volume 2, pages 549–556, 2015.
- [ASC11] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1626–1633, Nov 2011.

- [AsLMF18] Gerasimos Arvanitis, Aris s. Lalos, Konstantinos Moustakas, and Nikos Fakotakis. Feature preserving mesh denoising based on graph spectral processing. *IEEE Transactions on Visualization and Computer Graphics*, PP(99):1–1, 2018.
- [ASP17] Luca Di Angelo, Paolo Di Stefano, and Caterina Pane. Automatic dimensional characterisation of pottery. *Journal of Cultural Heritage*, 26:118 – 128, 2017.
- [BBL<sup>+</sup>17] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.
- [BCA<sup>+</sup>14] Silvia Biasotti, Andrea Cerri, Mostafa Abdelrahman, Masaki Aono, A. Ben Hamza, Magdly El-Melegy, Aaly Farag, Valeria Garro, Aandrea Giachetti, Daniela Giorgi, Afzal Godil, Chen-Feng Li, Yong-Jin Liu, Hero Martono, Chika Sanada, Atsushi Tatsuma, Santiago Velasco Forero, and Cx Xu. Retrieval and Classification on Textured 3D Models. In Benjamin Bustos, Hedi Tabia, Jean-Philippe Vandeborre, and Remco Veltkamp, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2014.
- [BCA<sup>+</sup>16] Silvia Biasotti, Andrea Cerri, Masaki Aono, A. Ben Hamza, Valeria Garro, Andrea Giachetti, Daniela Giorgi, Afzal Godil, Cheng-Feng Li, Chika Sanada, Michela Spagnuolo, Atsushi Tatsuma, and Santiago Velasco Forero. Retrieval and classification methods for textured 3D models: a comparative study. *The Visual Computer*, 32(2):217–241, 2016.
- [BCBB16] Silvia Biasotti, Andrea Cerri, Alexander Bronstein, and Michael Bronstein. Recent trends, applications, and perspectives in 3D shape similarity assessment. *Computer Graphics Forum*, 35(6):87–119, 2016.
- [BCFG11] Silvia Biasotti, Andrea Cerri, Patrizio Frosini, and Daniela Giorgi. A new algorithm for computing the 2-dimensional matching distance between size functions. *Pattern Recogn. Lett.*, 32(14):1735–1746, 2011.
- [BCFS15] Silvia Biasotti, Andrea Cerri, Bianca Falcidieno, and Michela Spagnuolo. 3D artifacts similarity based on the concurrent evaluation of heterogeneous properties. *J. Comput. Cult. Herit.*, 8(4):19:1–19:19, August 2015.
- [BCGS13] Silvia Biasotti, Andrea Cerri, Daniela Giorgi, and Michela Spagnuolo. Phog: Photometric and geometric functions for textured shape retrieval. *Computer Graphics Forum*, 32(5):1383–1392, 2013.
- [BDC18] Yohann Béarzi, Julie Digne, and Raphaëlle Chaine. Wavejets: A local frequency framework for shape details amplification. *Computer Graphics Forum, Proc. Eurographics 2018*, 2018.

- [BJNL14] Sheryl Brahnam, Lakhmi Jain, Loris Nanni, and Alessandra Lumini. Local binary patterns: New variants and applications. *Studies in Computational Intelligence*, 506, 09 2014.
- [BK04] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, September 2004.
- [BKS<sup>+</sup>05] Benjamin Bustos, Daniel A. Keim, D. Saupe, Tobias Schreck, and Dejan V. Vranić. Feature-based similarity search in 3D object databases. *ACM Comput. Surv.*, 37(4):345–387, 2005.
- [BMM<sup>+</sup>15] Davide Boscaini, Jonathan Masci, Simone Melzi, Michael Bronstein, Umberto Castellani, and Pierre Vandergheynst. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. *Computer Graphics Forum*, 34(5):13–23, 2015.
- [BMTA<sup>+</sup>17] Silvia Biasotti, Elia Moscoso Thompson, Masaki Aono, Ben Hamza, Benjamin Bustos, Shuilong Dong, Bowen Du, Amin Fehri, Haisheng Li, Frederico Limberger, Majid Masoumi, Mahsa Rezaei, Ivan Sipiran, Li Sun, Atsushi Tatsuma, Santiago Velasco Forero, Richard Wilson, Yan Wu, Junjie Zhang, Tianyu Zhao, Francesco Fornasa, and Aandrea Giachetti. SHREC’17: Retrieval of Surfaces with Similar Relief Patterns. In I. Pratikakis, F. Dupont, and M. Ovsjanikov, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2017.
- [BMTB<sup>+</sup>18] Silvia Biasotti, Elia Moscoso Thompson, Loic Barthe, Stefano Berretti, Andrea Giachetti, Thibault Lejembre, Nicolas Mellado, Konstantinos Moustakas, Iason Manolas, Dimitrios Dimou, Claudio Tortorici, Santiago Velasco Forero, Naoufel Werghi, Martina Polig, Giusi Sorrentino, and Sorin Hermon. Recognition of Geometric Patterns Over 3D Models. In Alex Telea, Theoharis Theoharis, and Remco Veltkamp, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2018.
- [BMTS18] Silvia Biasotti, Elia Moscoso Thompson, and Michela Spagnuolo. Experimental Similarity Assessment for a Collection of Fragmented Artifacts. In Alex Telea, Theoharis Theoharis, and Remco Veltkamp, editors, *Eurographics Workshop on 3D Object Retrieval*, pages 103–110. The Eurographics Association, 2018.
- [Bre01] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.
- [BUS09] Christian Beecks, Merih Seran Uysal, and Thomas Seidl. Signature quadratic form distances for content-based similarity. In *Proceedings of the 17th ACM Interna-*

*tional Conference on Multimedia*, MM '09, page 697–700, New York, NY, USA, 2009. Association for Computing Machinery.

- [BVZ01] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, November 2001.
- [BYRN99] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
- [BZM07] Anna Bosch, Andrew Zisserman, and X. Muñoz. Image classification using random forests and ferns. *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007.
- [CBA<sup>+</sup>13] Andrea Cerri, Silvia Biasotti, Mostafa Abdelrahman, Jesus Angulo, K. Berger, Louis Chevallier, Magdy El-Melegy, Aly Farag, F. Lefebvre, Andrea Giachetti, Hassane Guermoud, Yong-Jin Liu, Santiago Velasco Forero, Jean-Ronan Vigouroux, Cx Xu, and Jun Zhang. SHREC'13 Track: Retrieval on Textured 3D Models. In *3DOR'13*, EG 3DOR'13, pages 73–80, Girona, Spain, 2013. Eurographics Association.
- [CCB16] Alessio Cislighi, Enrico Antonio Chiaradia, and Gian Battista Bischetti. A comparison between different methods for determining grain distribution in coarse channel beds. *International Journal of Sediment Research*, 31(2):97–109, 2016.
- [CCC<sup>+</sup>08] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *EGIT'08*, pages 129–136. The Eurographics Association, 2008.
- [CCS12] Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno. Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Transactions on Visualization and Computer Graphics*, 18(6):914–924, 2012.
- [CGK03] Antonio Cardone, Satyandra K. Gupta, and Mukul Karnik. A survey of shape similarity assessment algorithms for product design and manufacturing applications. *Journal of Computing and Information Science in Engineering*, 3:109 – 118, 2003/06/00/ 2003.
- [CH67] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.
- [CHL05] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the 2005*



*IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, page 539–546, USA, 2005. IEEE Computer Society.

- [Chu01] Moo K. Chung. Statistical morphometry in computational neuroanatomy. Master's thesis, McGill University, Montreal, 2001.
- [CP03] Frederic Cazals and Marc Pouget. Estimating differential quantities using polynomial fitting of osculating jets. In *Proc. of SGP'03*, pages 177–187. Eurographics Association, 2003.
- [CRC<sup>+</sup>02] Jhonathan Corney, Heather Rea, Doug Clark, John Pritchard, Michael Breaks, and Roddy Macleod. Coarse filters for shape matching. *IEEE Comput. Graph. Appl.*, 22(3):65–74, May 2002.
- [CSB<sup>+</sup>17] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-CMU-Berkeley dataset for robotic manipulation research. *Int. J. Rob. Res.*, 36(3):261–268, March 2017.
- [CSM03] David Cohen-Steiner and Jean-Marie Morvan. Restricted Delaunay triangulations and normal cycle. In *Proc. of the 9<sup>th</sup> Ann. Symp. on Computational Geometry, SCG '03*, pages 312–321, New York, NY, USA, 2003. ACM.
- [CSM06] David Cohen-Steiner and Jean-Marie Morvan. Second fundamental measure of geometric sets and local approximation of curvatures. *J. Differential Geom.*, 74(3):363–394, 11 2006.
- [CSSB10] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(36):1109–1135, 2010.
- [CVHS20] Chiara Eva Catalano, Valentina Vassallo, Sorin Hermon, and Michela Spagnuolo. Representing quantitative documentation of 3d cultural heritage artefacts with CIDOC crmdig. *Int. J. Digit. Libr.*, 21(4):359–373, 2020.
- [CWS<sup>+</sup>15] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. Benchmarking in manipulation research: Using the Yale-CMU-Berkeley object and model set. *IEEE Robotics Automation Magazine*, 22(3):36–52, Sept 2015.
- [DD09] Michel Marie Deza and Elena Deza. *Encyclopedia of Distances*. Springer Berlin Heidelberg, 2009.

- [DDS<sup>+</sup>09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
- [DFG99] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Rev.*, 41(4):637–676, December 1999.
- [DTC<sup>+</sup>17] Teddy Debroutelle, Sylvie Treuillet, Aladine Chetouani, Matthieu Exbrayat, Lionel Martin, and Sebastien Jesset. Automatic classification of ceramic sherds with relief motifs. *Journal of Electronic Imaging*, 26(2):1 – 14, 2017.
- [DVC18] Julie Digne, S. Valette, and Raphaëlle Chaine. Sparse geometric representation through local shape probing. *IEEE Transactions on Visualization and Computer Graphics*, 24:2238–2250, 2018.
- [DW05] Chen-shi Dong and Guo-zhao Wang. Curvatures estimation on triangular mesh. *J. of Zhejiang University-SCIENCE A*, 6(1):128–136, 2005.
- [EEVG<sup>+</sup>15] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jan 2015.
- [EGHP<sup>+</sup>02] Efrat, Guibas, Sarel Har-Peled, Mitchell, and Murali. New similarity measures between polylines with applications to morphing and polygon sweeping. *Discrete & Computational Geometry*, 28(4):535–569, Nov 2002.
- [EK SX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD’96, page 226–231. AAAI Press, 1996.
- [FBF77] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, September 1977.
- [FH05] Michael S. Floater and Kai Hormann. Surface parameterization: a tutorial and survey. In Neil A. Dodgson, Michael S. Floater, and Malcolm A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 157–186, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [FMN<sup>+</sup>19] Linjing Fang, Fred Monroe, Sammy Weiser Novak, Lyndsey Kirk, Cara R. Schiavon, Seungyoon B. Yu, Tong Zhang, Melissa Wu, Kyle Kastner, Yoshiyuki Kubota, Zhao Zhang, Gulcin Pekkurnaz, John Mendenhall, Kristen Harris, Jeremy

- Howard, and Uri Manor. Deep learning-based point-scanning super-resolution imaging. *bioRxiv*, 2019.
- [GBMT<sup>+</sup>20] Andrea Giachetti, Silvia Biasotti, Elia Moscoso Thompson, Luigi Fraccarollo, Quang Nguyen, Hai-Dang Nguyen, Minh-Triet Tran, Gerasimos Arvanitis, Ioannis Romanelis, Vlasios Fotis, Konstantinos Moustakas, Claudio Tortorici, Naoufel Werghi, and Stefano Berretti. SHREC 2020 Track: River Gravel Characterization. In Tobias Schreck, Theoharis Theoharis, Ioannis Pratikakis, Michela Spagnuolo, and Remco C. Veltkamp, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2020.
- [GBP07] Daniela Giorgi, Silvia Biasotti, and Laura Paraboschi. Watertight models track. Research Report 09, IMATI, Genova, 2007.
- [GCL<sup>+</sup>15] Lin Gao, Yan-Pei Cao, Yu-Kun Lai, Hao-Zhi Huang, Leif Kobbelt, and Shi-Min Hu. Active exploration of large 3d model repositories. *IEEE Transactions on Visualization and Computer Graphics*, 99(PrePrints):1, 2015.
- [GCO06] Ran Gal and Daniel Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Transactions on Graphics (TOG)*, 25(1):130–150, 2006.
- [GFF<sup>+</sup>15] Andrea Giachetti, Francesco Farina, Francesco Fornasa, Atsushi Tatsuma, Chika Sanada, Masaki Aono, Silvia Biasotti, Andrea Cerri, and Sungbin Choi. Retrieval of Non-rigid (textured) Shapes Using Low Quality 3D Models. In I. Pratikakis, M. Spagnuolo, T. Theoharis, L. Van Gool, and R. Veltkamp, editors, *3DOR'15*. The Eurographics Association, 2015.
- [GFSF10] Daniela Giorgi, Patrizio Frosini, Michela Spagnuolo, and Bianca Falcidieno. 3D relevance feedback via multilevel relevance judgements. *The Visual Computer*, 26(10):1321–1338, 2010.
- [GG06] Timothy Gatzke and Cindy M. Grimm. Estimating curvature on triangular meshes. *International Journal of Shape Modeling*, 12(1):1–28, 2006.
- [GG07] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. *ACM Trans. Graph.*, 26(3), July 2007.
- [GG16] Valeria Garro and Andrea Giachetti. Scale space graph representation and kernel matching for non rigid and textured 3D shape retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(6):1258–1271, 2016.
- [GGG08] Gaël Guennebaud, Marcel Germann, and Markus Gross. Dynamic sampling and rendering of algebraic point set surfaces. *CG Forum*, 27(2):653–662, 2008.

- [GI04] Jack Goldfeather and Victoria Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Trans. Graph.*, 23(1):45–63, January 2004.
- [Gia18] Andrea Giachetti. Effective characterization of relief patterns. *Computer Graphics Forum*, 37(5):83–92, 2018.
- [GKM<sup>+</sup>02] Elamir S. Gadelmawla, Monir M. Koura, Talal M.A. Maksoud, Ibrahiem M. Elewa, and Hassan H. Soliman. Roughness parameters. *Journal of Materials Processing Technology*, 123(1):133 – 145, 2002.
- [GMGP05] Natasha Gelfand, Niloy J. Mitra, Leonidas J. Guibas, and Helmut Pottmann. Robust global registration. In *Symposium on Geometry Processing*, pages 197–206, 2005.
- [GSM<sup>+</sup>20] Pablo Gainza, Freyr Sverrisson, Federico Monti, Emanuele Rodola, Davide Boscaini, Michael Bronstein, and Bruno Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *NATURE METHODS*, 17:184–+, 2020.
- [HCDC17] Azzouz Hamdi-Cherif, Julie Digne, and Raphaëlle Chaine. Super-resolution of point set surfaces using local similarities. *Computer Graphics Forum*, 37, 06 2017.
- [Hea11] Marti A. Hearst. Natural search user interfaces. *Commun. ACM*, 54(11):60–67, November 2011.
- [HG20] Jeremy Howard and Sylvain Gugger. Fastai: A layered api for deep learning. *Information*, 11(2):108, Feb 2020.
- [HLP<sup>+</sup>19] Gao Huang, Zhuang Liu, Geoff Pleiss, Laurens Van Der Maaten, and Kilian Weinberger. Convolutional networks with dense connectivity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
- [HP04] Klaus Hildebrandt and Konrad Polthier. Anisotropic Filtering of Non-Linear Surface Features. *Computer Graphics Forum*, 2004.
- [HSS<sup>+</sup>13] Shi-Sheng Huang, Ariel Shamir, Chao-Hui Shen, Hao Zhang, Alla Sheffer, Shi-Min Hu, and Daniel Cohen-Or. Qualitative organization of collections of shapes via quartet analysis. *ACM Trans. Graph.*, 32(4):71:1–71:10, July 2013.
- [HT10] Gur Harary and Ayellet Tal. 3d euler spirals for 3d curve completion. In *Proceedings of the Twenty-Sixth Annual Symposium on Computational Geometry*, SoCG ’10, page 393–402, New York, NY, USA, 2010. Association for Computing Machinery.

- [HT11] Gur Harary and A. Tal. The natural 3d spiral. *Computer Graphics Forum*, 30, 2011.
- [HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [IJL<sup>+</sup>05] Natraj Iyer, Subramaniam Jayanti, Kuiyang Lou, Yagnanarayanan Kalyanaraman, and Karthik Ramani. Three-dimensional shape searching: state-of-the-art review and future trends. *Computer-Aided Design*, 37(5):509 – 530, 2005. Geometric Modeling and Processing 2004.
- [Ins] The Cyprus Institute. STARC repository. <http://public.cyi.ac.cy/starcRepo/>.
- [IT11] Arik Itskovich and Ayellet Tal. Surface partial matching and application to archaeology. *Computers & Graphics*, 35(2):334 – 341, 2011.
- [KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.
- [KBC06] Manesh Kokare, Prabir K. Biswas, and Biswa Nath Chatterji. Rotation-invariant texture image retrieval using rotated complex wavelet filters. *IEEE Trans. on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(6):1273–1282, Dec 2006.
- [KC11] Anestis Koutsoudis and Christodoulos Chamzas. 3D pottery shape matching using depth map images. *Journal of Cultural Heritage*, 12(2):128 – 133, 2011.
- [KFLCO13] Yanir Kleiman, Noa Fish, Joel Lanir, and Daniel Cohen-Or. Dynamic maps for exploring and browsing shapes. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing, SGP '13*, pages 187–196. Eurographics Association, 2013.
- [KFR03] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *EG/ACM SIGGRAPH Symposium on Geometry Processing*, pages 156–164. Eurographics Association, 2003.
- [KKF91] Vassos Karageorghis, Jacqueline Karageorghis, and A.G. Leventis Foundation. *The coroplastic art of ancient Cyprus*. Nicosia : A.G. Leventis Foundation, 1991. At head of title: A.G. Leventis Foundation.
- [KLM<sup>+</sup>12] Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Stephen DiVerdi, and Thomas Funkhouser. Exploring collections of 3D models using fuzzy correspondences. *ACM Trans. Graph.*, 31(4):54:1–54:11, July 2012.

- [KPL<sup>+</sup>10] Anestis Koutsoudis, George Pavlidis, Vassiliki Liami, Despoina Tsiafakis, and Christodoulos Chamzas. 3d pottery content-based retrieval based on pose normalisation and segmentation. *Journal of Cultural Heritage*, 11(3):329 – 338, 2010.
- [KSNS07] Evangelos Kalogerakis, Patricio Simari, Derek Nowrouzezahrai, and Karan Singh. Robust statistical estimation of curvature on discretized surfaces. In *Proc of SGP '07*, pages 13–22. EG Association, 2007.
- [KvD92] Jan J Koenderink and Andrea J van Doorn. Surface shape and curvature scales. *IVC*, 10(8):557–564, 1992.
- [KZ04] Vladimir Kolmogorov and Ramin Zabini. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.
- [LBS07] Torsten Langer, Alexander Belyaev, and Hans-Peter Seidel. Exact and interpolatory quadratures for curvature tensor estimation. *Comput. Aided Geom. Des.*, 24(8-9):443–463, November 2007.
- [LBZ<sup>+</sup>13] Zhen-Bao Liu, Shu-Hui Bu, Kun Zhou, Shu-Ming Gao, Jun-Wei Han, and Jun Wu. A survey on partial retrieval of 3D shapes. *J. Comput. Sci. Technol.*, 28(5):836–851, 2013.
- [LHGM05] Yu-Kun. Lai, Shi-Min Hu, David Xianfeng Gu, and Ralph Robert Martin. Geometric texture synthesis and transfer via geometry images. In *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling*, SPM '05, page 15–26, New York, NY, USA, 2005. Association for Computing Machinery.
- [LJH<sup>+</sup>19] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond, 2019.
- [LKF10] Yann LeCun, Koray Kavukcuoglu, and Clement Faret. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256, 2010.
- [LMT05] George Leifman, Ron Meir, and Ayellet Tal. Semantic-oriented 3D shape retrieval using relevance feedback. *The Visual Computer*, 21(8-10):865–875, 2005.
- [Low04] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–, 11 2004.
- [LSZU15] Lei Liu, Yun Sheng, Guixu Zhang, and Hassan Ugail. Graph cut based mesh segmentation using feature points and geodesic distance. In *2015 International Conference on Cyberworlds (CW)*, pages 115–120, 2015.

- [MBBV15] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 832–840, 2015.
- [MBG<sup>+</sup>20] Elia Moscoso Thompson, Silvia Biasotti, Andrea Giachetti, Claudio Tortorici, Naoufel Werghi, Ahmad Shaker Obeid, Stefano Berretti, Hoang-Phuc Nguyen-Dinh, Minh-Quan Le, Hai-Dang Nguyen, Minh-Triet Tran, Leonardo Gigli, Santiago Velasco Forero, Beatriz Marcotegui, Ivan Sipiran, Benjamin Bustos, Ioannis Romanelis, Vlassis Fotis, Gerasimos Arvanitis, Konstantinos Moustakas, Ekpo Otu, Reyer Zwiggelaar, David Hunter, Yonghuai Liu, Yoko Arteaga, and Ramamoorthy Luxman. Shrec 2020: Retrieval of digital surfaces with similar geometric reliefs. *Computers & Graphics*, 91:199 – 218, 2020.
- [MDSB03] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In Hans-Christian Hege and Konrad Polthier, editors, *Visualization and Mathematics III*, pages 35–57, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- [MF97] Andrea Marion and Luigi Fraccarollo. New conversion model for areal sampling of fluvial sediments. *Journal of Hydraulic Engineering*, 123(12):1148–1151, 1997.
- [MGB<sup>+</sup>12] Nicolas Mellado, Gaël Guennebaud, Pascal Barla, Patrick Reuter, and Christophe Schlick. Growing least squares for the analysis of manifolds in scale-space. *Computer Graphics Forum*, 31(5):1691–1701, 2012.
- [MP43] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943.
- [MP17] Marvin Minsky and Seymour A. Papert. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 2017.
- [MPS<sup>+</sup>04] Michela Mortara, Giuseppe Patané, Michela Spagnuolo, Bianca Falcidieno, and Jarek Rossignac. Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. *Algorithmica*, 38(1):227–248, Jan 2004.
- [MPS17] Michela Mortara, Corrado Pizzi, and Michela Spagnuolo. Streamlining the Preparation of Scanned 3D Artifacts to Support Digital Analysis and Processing: the GRAVITATE Case Study. In Tobias Schreck, Tim Weyrich, Robert Sablatnig, and Benjamin Stular, editors, *Eurographics Workshop on Graphics and Cultural Heritage*. The Eurographics Association, 2017.

- [MSR07] Evgeni Magid, Octavian Soldea, and Ehud Rivlin. A comparison of Gaussian and mean curvature estimation methods on triangular meshes of range image data. *CVIU*, 107(3):139 – 159, 2007.
- [MTAM<sup>+</sup>19] Elia Moscoso Thompson, Gerasimos Arvanitis, Konstantinos Moustakas, Nhi Hoang-Xuan, E Ro Nguyen, Minh-Triet Tran, Thibault Lejemble, Loic Barthe, Nicolas Mellado, Chiara Romanengo, Silvia Biasotti, and Bianca Falcidieno. SHREC’19 track: Feature Curve Extraction on Triangle Meshes. In *12th EG Workshop 3D Object Retrieval 2019*, pages 1 – 8, Gênes, Italy, May 2019.
- [MTB18] Elia Moscoso Thompson and Silvia Biasotti. Edge-based lbp description of surfaces with colorimetric patterns. In *Proceedings of the 11th Eurographics Workshop on 3D Object Retrieval, 3DOR ’18*, page 1–8, Goslar, DEU, 2018. Eurographics Association.
- [MTB19] Elia Moscoso Thompson and Silvia Biasotti. Retrieving color patterns on surface meshes using edgelbp descriptors. *Computers & Graphics*, 79:46 – 57, 2019.
- [MTBS<sup>+</sup>18] Elia Moscoso Thompson, Silvia Biasotti, Giusi Sorrentino, Martina Polig, and Sorin Hermon. Towards an Automatic 3D Patterns Classification: the GRAVITATE Use Case. In Robert Sablatnig and Michael Wimmer, editors, *Eurographics Workshop on Graphics and Cultural Heritage*. The Eurographics Association, 2018.
- [MTW<sup>+</sup>18] Elia Moscoso Thompson, Claudio Tortorici, Naoufel Werghi, Stefano Berretti, Santiago Velasco Forero, and Silvia Biasotti. Retrieval of Gray Patterns Depicted on 3D Models. In Alex Telea, Theoharis Theoharis, and Remco Veltkamp, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2018.
- [Mus17] British Museum. British Museum - Naukratis: An introduction, 2017.
- [Nib85] Wayne Niblack. *An Introduction to Digital Image Processing*. Strandberg Publishing Company, DNK, 1985.
- [OBCS<sup>+</sup>12] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: A flexible representation of maps between shapes. *ACM Trans. Graph.*, 31(4):30:1–30:11, July 2012.
- [OFCD02] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Trans. Graph.*, 21(4):807–832, October 2002.
- [OPH96] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, 1996.



- [OPM02] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE T. Pattern Anal. Mach. Intell.*, 24(7):971–987, 2002.
- [OVSP13] Ahlem Othmani, Lew FC Low Yan Voon, Christophe Stolz, and Alexandre Pi-boule. Single tree species classification from terrestrial laser scanning data for forest inventory. *Pattern Recognition Letters*, 34(16):2144–2150, 2013.
- [Pey] Gabriel Peyre. Toolbox graph - A toolbox to process graph and triangulated meshes. <http://www.ceremade.dauphine.fr/~peyre/matlab/graph/content.html>.
- [PSM10] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *Proceedings of the 11th European Conference on Computer Vision: Part IV, ECCV’10*, page 143–156, Berlin, Heidelberg, 2010. Springer-Verlag.
- [PT96] Flip Phillips and James Todd. Perception of local three-dimensional shape. *Journal of experimental psychology. Human perception and performance*, 22 4:930–44, 1996.
- [PV18] M. Prantl and L. Váša. Estimation of differential quantities using hermite rbf interpolation. *The Visual Computer*, 34(12):1645–1659, Dec 2018.
- [PZC13] Giuliano Pasqualotto, Pietro Zanuttigh, and Guido M. Cortelazzo. Combining color and shape descriptors for 3D model retrieval. *Signal Process-Image*, 28(6):608 – 623, 2013.
- [QYSG17] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5105–5114, 2017.
- [RBB09] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3D registration. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation, ICRA’09*, pages 1848–1853, Piscataway, NJ, USA, 2009. IEEE Press.
- [RD06] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Proceedings of the 9th European Conference on Computer Vision - Volume Part I, ECCV’06*, page 430–443, Berlin, Heidelberg, 2006. Springer-Verlag.
- [Rij79] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979.

- [Rob20] Rob Tuytel. Texture Haven. <https://texturehaven.com/>, 2020. Accessed: 2020-04-23.
- [Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.
- [RTG00] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The Earth Mover’s Distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, Nov 2000.
- [Rus04] Szymon Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004.*, pages 486–493, Sept 2004.
- [RZE08] Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit. *CS Technion*, 40, 01 2008.
- [SA16] M. Stamatopoulos and C. Anagnostopoulos. The thickness profile method: A new digital 3D approach for reassembling unpainted archaeological ceramic pottery. In *IMEKO International Conference on Methodology for Archaeology and Cultural Heritage*, 2016.
- [Shi95] Eugene V Shikin. *Handbook and atlas of curves*. CRC, 1995.
- [SJ99] Simone Santini and Ramesh Jain. Similarity measures. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(9):871–883, September 1999.
- [SKVS13] Dirk Smeets, Johannes Keustermans, Dirk Vandermeulen, and Paul Suetens. meshSIFT: Local surface features for 3D face recognition under expression variations and partial data. *Computer Vision and Image Understanding*, 117(2):158 – 169, 2013.
- [SMKF04] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. In *Shape modeling applications, 2004. Proceedings*, pages 167–178. IEEE, 2004.
- [SPS16] Michalis A. Savelonas, Ioannis Pratikakis, and Konstantinos Sfikas. Fisher encoding of differential fast point feature histograms for partial 3D object retrieval. *Pattern Recognition*, 55:114 – 124, 2016.
- [SSB<sup>+</sup>17] Adriana Schulz, Ariel Shamir, Ilya Baran, David I. W. Levin, Pitchaya Sittthi-Amorn, and Wojciech Matusik. Retrieval on parametric shape collections. *ACM Trans. Graph.*, 36(4), January 2017.

- [SSCO08] Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *Vis. Comput.*, 24(4):249–259, 2008.
- [Suz01] Motofumi T. Suzuki. A Web-based retrieval system for 3D polygonal models. In *IFSA World Congress and 20th NAFIPS International Conference. Joint 9th*, volume 4, pages 2271–2276, 2001.
- [SWS<sup>+</sup>00] Arnold W. M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, December 2000.
- [SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
- [SZ19] M. Seidl and M. Zeppelzauer. Towards distinction of rock art pecking styles with a hybrid 2d/3d approach. In *2019 International Conference on Content-Based Multimedia Indexing (CBMI)*, pages 1–4, 2019.
- [Tau95a] Gabriel Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proceedings of IEEE International Conference on Computer Vision*, pages 902–907, June 1995.
- [Tau95b] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '95*, pages 351–358, New York, NY, USA, 1995. ACM.
- [TBF18] Maria-Laura Torrente, Silvia Biasotti, and Bianca Falcidieno. Recognition of feature curves on 3D shapes using an algebraic approach to hough transforms. *Pattern Recognition*, 73:111 – 130, 2018.
- [TCL<sup>+</sup>13] G.K.L. Tam, Zhi-Quan Cheng, Yu-Kun Lai, F.C. Langbein, Yonghuai Liu, D. Marshall, R.R. Martin, Xian-Fang Sun, and P.L. Rosin. Registration of 3D point clouds and meshes: A survey from rigid to nonrigid. *IEEE T. Vis. Comput. Gr.*, 19(7):1199–1217, 2013.
- [TH15] Abdel Aziz Taha and Allan Hanbury. Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool. In *BMC Medical Imaging*, 2015.
- [TL19] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.
- [TSDS11] Federico Tombari, Samuele Salti, and Luigi Di Stefano. A combined texture-shape descriptor for enhanced 3D feature matching. In *Image Processing (ICIP), 2011 IEEE International Conference on*, pages 809–812, 2011.

- [TV04] J. W. H. Tangelder and R. C. Veltkamp. A survey of content based 3d shape retrieval methods. In *Proceedings Shape Modeling Applications, 2004.*, pages 145–156, 2004.
- [TV08] Johan W. H. Tangelder and Remco C. Veltkamp. A survey of content based 3D shape retrieval methods. *Multimedia Tools Appl.*, 39(3):441–471, 2008.
- [TWB15] C. Tortorici, N. Werghi, and S. Berretti. Boosting 3D LBP-based face recognition by fusing shape and texture descriptors on the mesh. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 2670–2674, Sept 2015.
- [TWB19a] Claudio Tortorici, Naoufel Werghi, and Stefano Berretti. Defining mesh-lbp variants for 3d relief patterns classification. In Liming Chen, Boulbaba Ben Amor, and Faouzi Ghorbel, editors, *Representations, Analysis and Recognition of Shape and Motion from Imaging Data*, pages 151–166, Cham, 2019. Springer International Publishing.
- [TWB19b] Claudio Tortorici, Naoufel Werghi, and Stefano Berretti. Extending lbp and convolution-like operations on the mesh. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4479–4483, 2019.
- [UUTCIC<sup>+</sup>] IT Innovation Centre (UK), British Museum (UK), Consiglio Nazionale delle Ricerche Institute of Applied Mathematics The Cyprus Institute (Cyprus), Information Technologies (Italy), University of Amsterdam (Netherlands), Technion – Israel Institute of Technology (Israel), and University of Haifa (Israel). GRAVITATE: Discovering relationships between artefacts using 3D and semantic data. EU H2020 REFLECTIVE project.
- [VRS<sup>+</sup>06] R. Veltkamp, R. Ruijsenaars, M. Spagnuolo, R. V. Zwol, and F. T. Haar. Shrec2006: 3d shape retrieval contest. Technical report, Department of Information and Computing Sciences, Utrecht University, 2006.
- [VVP<sup>+</sup>16] Libor Váša, Petr Vaněček, Martin Prantl, Věra Skorkovská, Petr Martínek, and Ivana Kolingerová. Mesh Statistics for Robust Curvature Estimation. *CG Forum*, 35(5):271–280, 2016.
- [WAvK<sup>+</sup>12] Yunhai Wang, Shmulik Asafi, Oliver van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Active co-analysis of a set of shapes. *ACM Trans. Graph.*, 31(6):165:1–165:10, November 2012.
- [WBB15] Naoufel Werghi, Stefano Berretti, and Alberto Del Bimbo. The mesh-LBP: A framework for extracting local binary patterns from discrete manifolds. *IEEE Trans. Image Processing*, 24(1):220–235, 2015.

- [WCL<sup>+</sup>08] Changchang Wu, B. Clipp, Xiaowei Li, J.-M. Frahm, and M. Pollefeys. 3D model matching with Viewpoint-Invariant Patches (VIP). In *Computer Vision and Pattern Recognition (CVPR), 2008 IEEE Conference on*, pages 1–8, 2008.
- [WLT16] Peng-Shuai Wang, Yang Liu, and Xin Tong. Mesh denoising via cascaded normal regression. *ACM Trans. Graph.*, 35(6):232:1–232:12, November 2016.
- [Wol54] M Gordon Wolman. A method of sampling coarse river-bed material. *EOS, Transactions American Geophysical Union*, 35(6):951–956, 1954.
- [WPS05] Kilian Q. Weinberger, Ben Packer, and Lawrence Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *AISTATS*, 2005.
- [WSK<sup>+</sup>15] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, June 2015.
- [WTBB16] Naoufel Werghi, Claudio Tortorici, Stefano Berretti, and Alberto Del Bimbo. Boosting 3D LBP-based face recognition by fusing shape and texture descriptors on the mesh. *IEEE Trans. Information Forensics and Security*, 11(5):964–979, 2016.
- [WTBdB15] Naoufel Werghi, Claudio Tortorici, Stefano Berretti, and Alberto del Bimbo. Local binary patterns on triangular meshes: Concept and applications. *Computer Vision and Image Understanding*, 139:161 – 177, 2015.
- [YM12] Faliu Yi and Inkyu Moon. Image segmentation: A survey of graph-cut methods. In *2012 International Conference on Systems and Informatics (ICSAI2012)*, pages 1936–1941, 2012.
- [YSLH03] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. Faceted metadata for image search and browsing. In *SIGCHI Conference on Human Factors in Computing Systems, CHI '03*, pages 401–408, New York, NY, USA, 2003. ACM.
- [YZC17] Jiaqi Yang, Qian Zhang, and Zhiguo Cao. The effect of spatial information characterization on 3D local feature descriptors: A quantitative evaluation. *Pattern Recognition*, 66(Supplement C):375 – 391, 2017.
- [ZBH12] Andrei Zaharescu, Edmond Boyer, and Radu Horaud. Keypoints and local descriptors of scalar functions on 2D manifolds. *Int. J. Comput. Vision*, 100(1):78–98, 2012.

- [ZGYL11] Mao Zhihong, Cao Guo, Ma Yanzhao, and Kunwoo Lee. Curvature estimation for meshes based on vertex normal triangles. *Computer-Aided Design*, 43(12):1561 – 1566, 2011.
- [ZLHB19] Michael R. Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. Lookahead optimizer: k steps forward, 1 step back, 2019.
- [ZPS<sup>+</sup>16] Matthias Zeppelzauer, Georg Poier, Markus Seidl, Christian Reinbacher, Samuel Schulter, Christian Breiteneder, and Horst Bischof. Interactive 3D segmentation of rock-art by enhanced depth maps and gradient preserving regularization. *J. Comput. Cult. Herit.*, 9(4):19:1–19:30, September 2016.