

UNIVERSIDADE FEDERAL FLUMINENSE

LIZETH JOSELINE FUENTES PEREZ

**Dictionary Learning-based Inpainting on Triangular
Meshes**

NITERÓI

2017

UNIVERSIDADE FEDERAL FLUMINENSE

LIZETH JOSELINE FUENTES PEREZ

Dictionary Learning-based Inpainting on Triangular Meshes

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Computação Visual

Orientador:

Prof. Anselmo Antunes Montenegro

NITERÓI

2017

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

F954 Fuentes Perez, Lizeth Joseline
Dictionary learning-based inpainting on triangular meshes / Lizeth
Joseline Fuentes Perez. – Niterói, RJ : [s.n.], 2017.
49 f.

Dissertação (Mestrado em Computação) - Universidade Federal
Fluminense, 2017.
Orientador: Anselmo Antunes Montenegro.

1. Computação visual. 2. Imagem tridimensional. 3. Aprendizado
de máquina. 4. Equação de Poisson. I. Título.

CDD 006

Lizeth Joseline Fuentes Perez

Dictionary Learning-based Inpainting on Triangular Meshes

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Computação Visual

Aprovada em Fevereiro de 2017.

BANCA EXAMINADORA

Prof. Anselmo Antunes Montenegro - Orientador, UFF

Prof. Marcos Lage, UFF

Prof. Ricardo Marroquim, UFRJ

Prof. Cristian López del Alamo, UNSA

Niterói

2017

*This work is dedicated to my beloved mother.
“I always feel you within my soul, guiding my life wisely”.*

Acknowledgements

Every person has his own color, not all persons are sensitive to this, but in life, there are some moments when the colors are merged between determined persons. So far, the canvas of my life has been painted with the most incredible colors I've ever seen and it still has plenty of room for learning new things. This work is a little picture of that canvas and it wouldn't be possible without the invaluable help of all these persons.

I would like to thank my friend Cristian López, who believed in me from the beginning when I was an undergraduate student and for helping me to discover the research world. I would like to say how lucky I was when I met Alex Bronstein in 2015 because since then, I was motivated to research shape analysis area and he kindly shared his expertise with me, providing an important guidance for this work. I would like to express my gratitude to my advisor and friend Anselmo Montenegro, who patiently guided me and lectured me during these two years and who taught me the importance of doing things with a sincere dedication and willing. I am also grateful to my friend and collaborator Luciano Romero, for encouraging me to get over the obstacles with a quiet attitude and to accept the difficulties as part of the learning process. I would also like to thank professors Marcos Lage and Ricardo Marroquim for accepting to be part of the jury. This work was supported by the CAPES, which I would like to thank.

At last but not least I would like to thank my family, for supporting me all these years.

Resumo

O problema de inpainting consiste no preenchimento de regiões perdidas ou deterioradas em imagens e vídeos, de tal forma que o padrão de preenchimento não produza artefatos que destoem do dado original. Além da restauração de dados perdidos, a técnica de inpainting também pode ser usada para remoção de objetos indesejados. A possibilidade de aquisição de dados geométricos a partir da popularização de dispositivos de aquisição de geometria 3D tornou relevante a questão de inpainting em malhas. Malhas são geradas a partir de nuvens de pontos que são obtidas dos dispositivos de aquisição de forma 3D. Após os processos de digitalização e triangulação, as malhas apresentam frequentemente orifícios devido a defeitos na nuvem original. Uma das causas mais comuns é a oclusão.

Na presente dissertação, abordamos o problema de inpainting em superfícies através de um novo método baseado em aprendizado de dicionários e codificação esparsa. O uso da técnica de aprendizado de dicionários para resolver o problema de inpainting em imagens têm obtido resultados bastante satisfatórios; por outro lado, sua aplicação em superfícies descritas por malhas de triângulos ainda é um desafio. Nosso método aprende o dicionário através da subdivisão da malha em retalhos (*patches*) e reconstrói a malha através de um método de reconstrução inspirado no método Non-local Means sobre os códigos esparsos. Nosso método tem como vantagens a capacidade de preencher as regiões perdidas ao mesmo tempo em que remove ruído e realça características importantes da malha. Além disso, é globalmente coerente, uma vez que a representação esparsa baseada nos dicionários captura toda a informação geométrica no domínio transformado. Apresentamos duas variações do método: uma direta na qual o modelo é reconstruído e restaurado diretamente da representação no domínio transformado e uma segunda, adaptativa, na qual as regiões perdidas são recriadas iterativamente através da propagação sucessiva do código esparsos, computado no bordo dos furos, que guia reconstruções locais. O segundo método produz melhores resultados para regiões extensas porque os códigos esparsos dos novos retalhos são adaptados de acordo com os códigos esparsos dos retalhos do bordo.

Finalmente, apresentamos e analisamos resultados experimentais que demonstram o desempenho de nosso método comparado com os da literatura.

Palavras-chave: Inpainting em superfícies, aprendizagem de dicionários, codificação esparsa, Non-local means, equação de Poisson, métodos preenchimento de buracos, malhas triangulares.

Abstract

The problem of inpainting consists of filling missing or damaged regions in images and videos in such a way that the filling pattern does not produce artifacts that deviate from the original data. In addition to restoring the missing data, the inpainting technique can also be used to remove undesired objects. The search for surface inpainting solutions became more relevant when the 3D geometry acquisition devices came into mainstream. Meshes are typically generated from point clouds that are obtained from 3D shape acquisition devices. After the scanning and triangulation processes, the meshes often have holes due to defects in the original cloud. One of the most common causes is occlusion.

In this dissertation, we address the problem of inpainting on surfaces through a new method based on dictionary learning and sparse coding. The use of the dictionary learning techniques to solve the inpainting problem in images has obtained successful results; on the other hand, its application on surfaces described by triangle meshes is still a challenge.

Our method learns the dictionary through the subdivision of the mesh into patches and rebuilds the mesh via a method of reconstruction inspired by the Non-local Means method on the computed sparse codes. One of the advantages of our method is that it is capable of filling the missing regions and simultaneously removes noise and enhances important features of the mesh. Moreover, the inpainting result is globally coherent as the representation based on the dictionaries captures all the geometric information in the transformed domain. We present two variations of the method: a direct one, in which the model is reconstructed and restored directly from the representation in the transformed domain and a second one, adaptive, in which the missing regions are recreated iteratively through the successive propagation of the sparse code computed in the hole boundaries, which guides the local reconstructions. The second method produces better results for large regions because the sparse codes of the patches are adapted according to the sparse codes of the boundary patches. Finally, we present and analyze experimental results that demonstrate the performance of our method compared to the literature.

Keywords: Surface Inpainting, Dictionary learning, Sparse coding, Non-local means, Poisson equation, Hole-filling methods, Triangular meshes.

List of Figures

4.1	Surface Inpainting problem. 7.4a Set of known vertices. 4.1b Set of unknown vertices. 7.4b All vertices.	8
5.1	Image denoising via Dictionary Learning. 5.1a Barbara image with Gaussian noise; 5.1b Barbara image recovered; 5.1c Barbara original image; 5.1d Dictionary learned from the Barbara image with Gaussian noise.	13
5.2	The Principal curvatures over an elliptic surface (left) and a hyperbolic surface (right). In an elliptic surface both principal curvatures are positive, conversely, in the hyperbolic surface the principal curvatures have different signs [7].	16
6.1	Surface inpainting via Dictionary Learning proposed method.	17
6.2	Dictionary Learning for triangular meshes method.	18
6.3	Maximum Principal curvature direction over a triangular mesh at the point v	21
6.4	Hole triangulation and fairing process.	23
6.5	Rules for the Create new triangles (algorithm) 3. The first case, when $\theta_i \leq 75^\circ$ is shown in Figure 6.5a. The second case, when $75^\circ < \theta \leq 135^\circ$ is shown in Figure 6.5b. Third case, when $\theta_i > 135^\circ$ is shown in Figure 6.5c	24
6.6	Growing regions R_i for the Iterative Inpainting algorithm. 6.6a Initial hole triangulation; 6.6b The growing regions are R_0 (red), R_1 (yellow) and R_2 (green). The number of levels to cover the hole is $r = 3$	26
7.1	Cossine 7.1a and Gaussian 7.1b basis for Dictionary Learning algorithm.	29
7.2	Dictionaries atoms learned from shapes. 7.2a Dictionary atoms learned from the bunny model, the overlapping factor is $\sigma = 2.5$; 7.2b Dictionary atoms learned from the Armadillo model, the overlapping factor is $\sigma = 1.5$. Both dictionaries has 16 basis functions as shown in Figure 7.1.	30

7.3	Inpainting existing holes on the Bunny model. 7.3a, 7.3d Original Bunny model with holes; 7.3b, 7.3e Hole-filling result using the method of [38]; 7.3c, 7.3f Hole filling result using our Adaptative Inpainting method.	30
7.4	7.4a Original Sphere model with a hole. 7.4b Hole-filling result using the proposed method.	31
7.5	Inpainting existing holes on Armadillo model. 7.5a, Original Armadillo model; 7.5b, Original Armadillo model with holes; 7.5c Hole filling result using our the proposed method considering all vertices as seeds.	31
7.6	Inpainting existing small holes in Armadillo shape	32
7.7	Inpainting existing holes on Bunny shape with noise. 7.7a Original Bunny model with noise and small holes. 7.7b Hole-filling result using our Direct inpainting method	32
7.8	Inpainting existing large hole on Wolf shape. 7.7a Original Wolf model with a large hole. 7.7b Hole-filling result using our Iterative inpainting method .	32
7.9	Inpainting existing large hole in Armadillo shape. 7.9a Inpainting result using our Iterative Inpainting method (left). The input Armadillo shape with holes (right). 7.9b Difference between the inpainting result (left) and the original Armadillo model without holes (right).	33

Contents

1	Introduction	1
1.1	Main Contribution	1
2	Introdução	3
2.1	Contribuição principal	4
3	Related Works	5
3.1	Geometry-based approaches	5
3.2	Texture-based approaches	6
4	Surface Inpainting Problem	8
5	Background	10
5.1	Sparse Coding	10
5.2	Dictionary Learning	11
5.2.1	The K-SVD algorithm	12
5.3	Non-local-means	12
5.4	Poisson Equation	15
5.5	Principal Curvatures	15
6	Surface Inpainting with Dictionary learning	17
6.0.1	Overview of the method	18
6.1	Dictionary Learning on Triangular meshes	18
6.1.1	Patch Creation	19

6.1.2	Patch Normalization	20
6.1.3	Learning a continuous dictionary on surfaces	21
6.2	Hole Identification	22
6.3	Hole Triangulation and Fairing	22
6.4	Sparse Coding approaches	25
6.4.1	Direct Sparse Coding approach	25
6.4.2	Adaptative Sparse Coding approach	25
6.5	Mesh reconstruction	27
7	Experiments and results	29
8	Conclusions	34
	References	36

Chapter 1

Introduction

The surface inpainting problem can be defined as the task of in filling missing regions so that these regions are not highly noticeable with respect to the surrounding mesh. This problem is also known as hole filling and mesh completion [2]. The main goal of inpainting is to restore damaged parts of the surface, but it can also be used to remove some not desired objects, present in the scenario to be scanned (e.g. a house behind an occluding tree).

3D shape acquisition devices produce point clouds that are typically converted into mesh representations before any kind of geometric processing. The produced meshes often present holes because of imperfections in the original points cloud, which can be due to several reasons. The most common is occlusion, but low reflectance, constraints in the scanner placements and lack of enough coverage of the object by the scanner, in case of scanning some art pieces, are also common causes [11].

Scanned 3D objects have become a primary asset, in many applications domains (medicine, manufacturing, art, cultural heritage, 3D printing, architecture, and construction, entertainment industry, etc) [12]. Because of this, it is quite important to inpaint the missing information present in the acquired models as a post-processing task.

1.1 Main Contribution

In this dissertation, we present a new method to address the surface inpainting problem via dictionary learning and sparse coding.

Our method is quite different from most methods found in the literature because it tackles the problem in a transformed domain instead of working directly in the mesh

domain, considering the whole mesh information and not only the information of the hole surroundings. The proposed method is inspired by the inpainting methods based on dictionary learning technique, which have been successfully applied to images. However, adapting these techniques is quite challenging because of the non-uniform sampling of the considered surfaces.

Our method learns the dictionary through the subdivision of the mesh into patches and rebuilds the mesh via a method of reconstruction inspired by the Non-local Means method on the computed sparse codes.

The main contributions of the present dissertation are:

- We introduce a new surface inpainting method capable of filling the missing regions and simultaneously remove noise and enhance important features of the mesh.
- The inpainting result is globally coherent due to the fact that the proposed method uses the whole remaining shape.

We present two variations of the method: a direct one, in which the model is reconstructed and restored directly from the representation in the transformed domain and a second one, adaptive, in which the missing regions are recreated iteratively through the successive propagation of the sparse code computed in the hole boundaries, which guides the local reconstructions. The second method produces better results for large regions because the sparse codes of the patches are adapted according to the sparse codes of the boundary patches.

Chapter 2

Introdução

O problema de inpainting em superfícies pode ser definido como a tarefa de preencher regiões, de tal forma que estas regiões não destoem das características das regiões circundantes na malha. Este problema é também conhecido como preenchimento de buracos [2]. O objetivo principal do inpainting é restaurar partes danificadas da superfície, mas também pode ser usado para remover alguns objetos não desejados, presentes no cenário a ser digitalizado (por exemplo, remover parte de uma árvore que esconde parte de uma casa).

Dispositivos de aquisição de forma 3D produzem nuvens de pontos, as quais são tipicamente convertidas em representações baseadas em malhas, antes que qualquer tipo de processamento geométrico possa ser realizado. As malhas produzidas freqüentemente apresentam buracos devido a imperfeições nas nuvens de pontos originais, o que pode ser explicado por diferentes razões. As mais comuns incluem oclusão parcial, baixa reflectância e restrições no posicionamento do scanner. No caso peças de arte, a falta de cobertura do objeto pelo scanner também é uma causa frequente [11].

Objetos 3D digitalizados tornaram-se um recurso primário em muitos domínios de aplicações (medicina, fabricação, arte, patrimônio cultural, impressão 3D, arquitetura e construção, Indústria do entretenimento, etc) [12]. Devido a isso, é muito importante fazer o inpainting das informações ausentes nos modelos adquiridos, como uma tarefa de pós-processamento.

2.1 Contribuição principal

Nesta dissertação, apresentamos um novo método para abordar o problema de inpainting em superfícies, baseado em aprendizagem de dicionários e codificação esparsa. Nosso método é bastante diferente da maioria dos métodos encontrados na literatura porque aborda o problema num domínio transformado, em vez de trabalhar diretamente no domínio da malha, considerando a informação da malha inteira e não somente da informação na vizinhança do buraco. O método proposto é inspirado nos métodos de inpainting baseados na técnica de aprendizagem de dicionários, as quais foram aplicadas com sucesso em imagens. No entanto, a adaptação destas técnicas é desafiador devido à amostragem não uniforme das superfícies. Nosso método aprende o dicionário através da subdivisão da malha em retalhos e reconstrói a malha através de um método de reconstrução inspirado no método Non-local Means aplicado ao vetor de códigos esparsos.

As principais contribuições da presente dissertação são:

- Introduzimos um novo método para inpainting de superfícies capaz de preencher as regiões ausentes e simultaneamente eliminar o ruído, além de melhorar as características importantes da malha.
- O resultado do processo de inpainting é globalmente coerente, devido ao fato de que o método proposto usa a informação global da malha e não somente as informações na vizinhança dos buracos.

Apresentamos duas variantes do método: uma direta, na qual o modelo é reconstruído e restaurado diretamente a partir da representação no domínio transformado e uma segunda, adaptável, em que as regiões que faltam são recriadas iterativamente através da sucessiva propagação do código esparsos calculado nos limites dos buracos, que guia as reconstruções locais. O segundo método produz melhores resultados para regiões grandes porque os códigos esparsos são adaptados de acordo com os códigos esparsos dos retalhos do bordo.

Chapter 3

Related Works

In the literature, there exists many different approaches used to solve the surface inpainting problem. For simplicity, we can divide the inpainting approaches into main two groups. The *geometry-based methods* and *texture-based methods*.

3.1 Geometry-based approaches

These methods are also called structural because they preserve smoothness between the inpainted patch and the remaining mesh. The smoothness can be measured by a certain degree of curvature. Geometry-based approaches also can be divided into two categories: the *voxel-based* and the *triangle-based* methods. The main difference between them is the representation.

In the case of a voxel-based approach, it is necessary to first convert the mesh representation into a voxelized representation. Curless and Levoy [13] proposed a volumetric method in voxel space. David [14] uses a volumetric diffusion technique from the hole boundary to the interior based on a space carving information. These approaches are suitable for complex holes but also may generate incorrect topology in some cases [38].

In triangle-based approaches, there also exists also different kind of approaches. We refer readers to the survey in [32], which presents a quantitative comparison for hole filling methods. The interpolating-based methods are one of the most simple approaches; they create smooth and continuous patches across the boundaries. Wang and Oliveira [35] proposed a method that creates an interpolating patch using moving least squares to fit polynomial functions. Pfeifle and Seidel [31] create an interpolating patch using triangular B-splines and Branch [6] uses a radial basis functions interpolator. Although

these approaches generate good results for disk-shaped holes, they are not appropriate to deal with complex holes in regions of high curvature.

One of the most popular approaches to deal with complex holes is the method proposed by Liepa [25], which is a complete method for filling holes. It uses a dynamic programming algorithm for the mesh triangulation, which takes into account dihedral angles and areas. After the initial triangulation, the algorithm performs a refinement step, based on the Umbrella operator, which is similar to the Laplacian. The main drawback of this algorithm is the high complexity of the mesh triangulation algorithm, which is $O(|E|^3)$, where $|E|$ is the number of edges in the mesh.

Jun [21] developed a method that divides a complex hole into pieces. Then, each sub-hole is filled with a planar triangulation. Finally, the generated mesh is improved using sub-division and refinement.

Zhao [38] proposed a robust hole-filling method that uses an advancing front strategy for the mesh generation. In a subsequent step, the positions of the vertices are optimized by solving the Poisson equation.

A method that minimizes the variational curvature between the inpainted patch and its surroundings was proposed by Caselles [11].

Brunton [8] developed a method for filling holes in meshes by curve unfolding.

Although the geometry-based methods produce a smooth inpainting result, the texture is still missing. In order to overcome this issue, the texture-based approach were developed.

3.2 Texture-based approaches

Image inpainting methods work similarly to methods applied to texture synthesis. Generally, these methods progressively propagate the texture patches until they cover the missing regions [2].

Geometry and texture-based methods present different analysis and implementation tools. We believe that sparse signal recovery methods on surfaces are capable of introducing some texture, as shown in the work proposed by Zhong [39]. As far as we are concerned, this is the first work that addresses the surface inpainting problem using sparsity constraints. It represented the shape in a transformed domain, using the eigenfunctions of the Laplacian as atoms of the fixed dictionary. The main difference between

our method is that we learn a dictionary from the input mesh instead of using the Laplacian eigenfunctions as dictionary atoms, which produces a more adaptative inpainting result.

Chapter 4

Surface Inpainting Problem

The problem of surface inpainting can be stated as follows: Let a triangle mesh $T = (V, E, F)$, where V , E and F are the set of vertices, edges and faces. We can define a function f over each vertex, such that $f : V \rightarrow \mathbb{R}^3$, which denotes the mesh coordinates for each vertex.

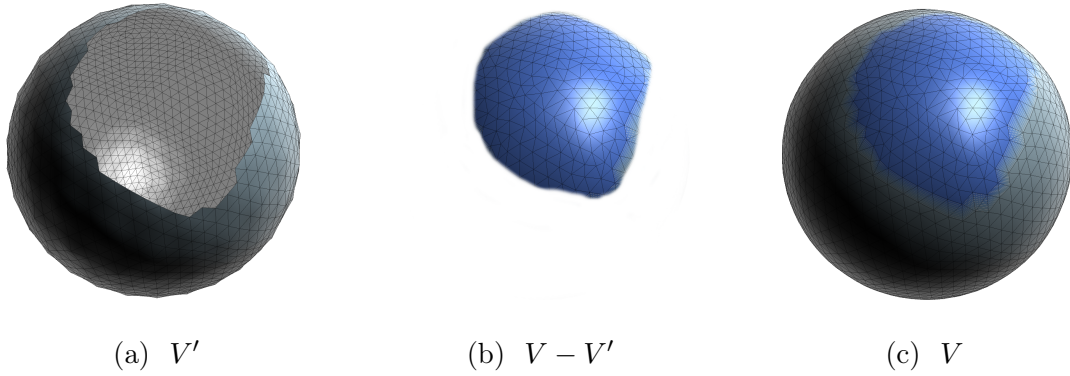


Figure 4.1: Surface Inpainting problem. [7.4a](#) Set of known vertices. [4.1b](#) Set of unknown vertices. [7.4b](#) All vertices.

We assume that $V' \subset V$ are the known vertices and $V - V'$ correspond to the unknown surface hole vertices. We define a mask M as a projection matrix with dimensions $|V'| \times |V|$ as follows:

$$M(i, j) = \begin{cases} 1 & \text{if } v_i \text{ is the } j\text{th vertex} \in V' \\ 0 & \text{otherwise} \end{cases}$$

Denote $x' = f(V')$ as the signal over the visible vertices of $x = f(V)$, such that $x' = Mx$. Generally, the inpainting problem can be formulated as a constrained optimization

problem:

$$\hat{x} = \arg \min_x Pr(x) \text{ s.t. } \| Mx - x' \|_2^2 < \epsilon \quad (4.1)$$

where the data term $\| Mx - x' \|_2^2$ guarantees the closeness between the known information and the resultant information. The term $Pr(\hat{x})$ is a prior that regularizes certain properties of the reconstructed signal.

The proposed method computes the approximate signal in a transformed domain instead of computing it directly in the mesh domain. Consider a dictionary $D = (d_1, \dots, d_m)$, where d_i is a dictionary atom. We represent x as a linear combination of the dictionary atoms.

$$x = D\alpha = \sum_{i=1}^m \alpha_i d_i$$

where $\alpha = (\alpha_1, \dots, \alpha_m)^T$ represents the coefficients of x with respect to the dictionary D .

Finally, in order to estimate the coefficients α we impose the sparsity of the coefficients as a prior. Thus, the surface inpainting problem formulations is as follows:

$$\hat{\alpha} = \arg \min_{\alpha} \| \alpha \|_0 \text{ s.t. } \| MD\alpha - x' \|_2^2 < \epsilon \quad (4.2)$$

The pseudo-norm l_0 counts the non-zero elements of α and D is the dictionary that will be learn as part of the proposed method.

Chapter 5

Background

5.1 Sparse Coding

Sparse coding is an operation to obtain the codification (coefficients) as sparse as possible to construct a sparse representation. These concepts belong to the *Sparse-Land* model, which is a way for synthesizing signals according to a prior defined by a transform [16]. A sparse representation is *based* on the idea that a signal can be *decomposed* as a sparse linear combination of atoms, which are understood in a base called dictionary $y \approx D\hat{\alpha}$. The minimization problem of sparse coding is formulated as follows:

$$\hat{\alpha} = \arg \min_{\alpha} \| y - D\alpha \|_2 \quad s.t. \quad \| \alpha \|_0 \leq L \quad (5.1)$$

Note that in this formulation, the dictionary is given. The representation is guided by the sparse codes α , where α are the sparse coefficients to approximate the signal y as sparse as possible. This implies that only the meaningful atoms are considered [16]. The term $\| \alpha \|_0$ measures the sparsity of the decomposition and can be understood as the number of non-zero coefficients in α , this is controlled by the regularization term L .

The solution to the equation with the norm l_0 is an NP-hard problem. Fortunately, it is possible to relax the norm under certain conditions and find out an approximate solution. The class of greedy methods can yield a good enough approximate solution for the problem. One example is the Orthogonal Matching Pursuit (OMP) algorithm, which is an iterative greedy algorithm that chooses the best matching projections of a multidimensional data onto a dictionary D [10]. The OMP algorithm [16] is summarized in Algorithm 1.

Algorithm 1 Orthogonal Matching Pursuit (OMP) algorithm

Require: Dictionary D , sparsity constraint L **Ensure:** Sparse coding vector α 1: $\Gamma = \emptyset$ 2: **for** $i = 1, \dots, L$ **do**

3: Select the atom which most reduces the objective function

4:

$$i' \leftarrow \arg \min_{i \in \Gamma^C} \{ \min_{\alpha'} \|x - D_{\Gamma \cup \{i\}} \alpha'\|_2^2 \}$$

5: Update the active set: $\Gamma \rightarrow \Gamma \cup \{i'\}$

6: Update the residual (orthogonal projection)

7:

$$r \rightarrow (I - D_\Gamma (D_\Gamma^T D_\Gamma)^{-1} D_\Gamma^T) x$$

8: Update the coefficients

9:

$$\alpha_\Gamma \rightarrow (D_\Gamma^T D_\Gamma)^{-1} D_\Gamma^T x$$

10: **end for**11: **return** α

5.2 Dictionary Learning

One important component of the sparse coding optimization problem is the dictionary. A wise choice of a proper dictionary is a relevant issue because it has significant impact in the performance of the sparse decomposition problem. Most of the dictionaries used are pre-constructed dictionaries such as those based on wavelets, contourlets, curvelets and more. However, these dictionaries present some limitations in the proficiency to make sparse the signals and in most cases they are restricted to signals of a certain type [16]. This gave origin to another approach, from the learning point-of view, for obtaining dictionaries that helps to approximate the signal as sparse as possible.

The dictionary learning approach builds empirically a dictionary from a training database of signal instances. Because of this it is able to adapt to any type of signals that complies with the Sparse-Land model [16]. However, this approach present some drawbacks. The computational load is higher compared to the pre-constructed dictionaries and the training methodology is restricted to low dimensional signals. Thus, dictionary learning techniques usually require the signal to be divided into small patches[1].

Formally, the dictionary learning problem can be formulated as:

$$\min_D \|Y - D\alpha\|_2^2 \quad s.t. \quad \|\alpha\|_0 \leq L \quad (5.2)$$

Given signal Y , the solution to Equation 5.2 learns a dictionary and its sparsest representation α over the unknown dictionary D , updating D until the proper dictionary is found, considering the sparsity constraints for α , in this formulation L is the sparsity regularization term.

In the literature, it is possible to find approaches to solve this problem for images. For instance, the Method of Optimal Directions (MOD) applied to speech and electrocardiogram (ECG) signals, proposed by Egan et al. [18] and the K-SVD algorithm proposed by Aharon et al. [1].

Dictionary learning has been applied to image processing tasks. For instance, denoising, inpainting, compressed sensing and demosaicing are examples of image processing problems that can be successfully solved via dictionary learning[20]. An image denoised via Dictionary Learning is shown in Figure 5.1.

5.2.1 The K-SVD algorithm

K-SVD is an iterative algorithm used to build a dictionary from a set of input signals. It was proposed by Aharon [1] and is comprised of two main stages. The sparse coding stage and dictionary learning stage. In the dictionary learning stage, the dictionary atoms are updated using Single Value Decomposition (SVD).

Given the set of signals Y , a dictionary D is built to sparsely represent the data by approximating the to make sparse the signals solution to Equation 5.2.

The general algorithm for K-SVD algorithm is described in Algorithm 2. The parameter ϵ ensures the similarity between the input signal and the approximate $D\alpha$ signal.

5.3 Non-local-means

The non-local means is a method proposed by Buades [9] to solve the image denoising problem. It has its spirit based on the non-local averaging of all pixels in the image. The

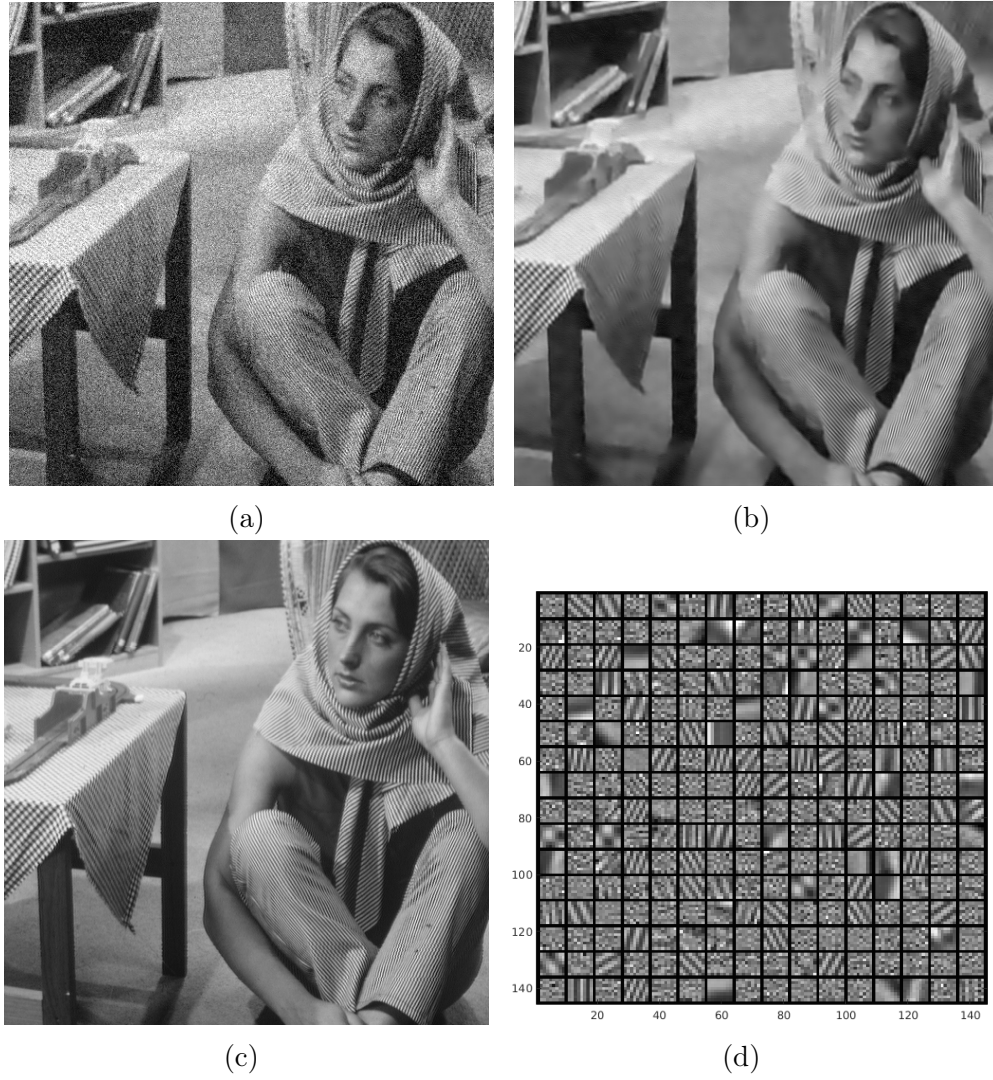


Figure 5.1: Image denoising via Dictionary Learning. 5.1a Barbara image with Gaussian noise; 5.1b Barbara image recovered; 5.1c Barbara original image; 5.1d Dictionary learned from the Barbara image with Gaussian noise.

non local value of a pixel i of an image I is:

$$NL[v](i) = \sum_{j \in I} w(i, j) v(j) \quad (5.3)$$

The discrete noisy image is $v = v(i) | i \in I$, the weights $w(i, j)$ depend on the similarity between neighborhoods of the pixels $v(N_i)$ and $v(N_j)$. The weights are defined as:

$$w(i, j) = \frac{1}{Z(i)} e^{-\frac{\|v(N_i) - v(N_j)\|_{2, \sigma}}{h^2}} \quad (5.4)$$

where the standard deviation of the Gaussian kernel is $\sigma > 0$.

Algorithm 2 The K-SVD dictionary-learning algorithm [16]

Require: Set of training signals Y

Ensure: Trained dictionary D

- 1: **Initialize Dictionary:** Build $D_{(0)} \in \mathbb{R}^{n \times m}$, either by using random entries, or using m randomly chosen examples.
- 2: **while** $\|Y - D\alpha\|_2^2 > \epsilon$ **do**
- 3: **Sparse Coding Stage:** Use a pursuit algorithm to approximate the solution of

$$\alpha'_i = \arg \min_{\alpha} \|y_i - D_{(k-1)}\alpha\|_2^2 \text{ s.t. } \|\alpha\|_0 \leq k_0$$

obtaining sparse representations α'_i for $1 \leq i \leq M$. These form the matrix $X_{(k)}$

- 4: **K-SVD Dictionary-Update Stage:**
- 5: **for** $j_0 = 1, 2, \dots, m$ **do**
- 6: Update the columns of the dictionary and obtain $D_{(k)}$
- 7: Define the group of examples that use the atom a_{j_0} ,

$$\omega_{j_0} = \{i | 1 \leq i \leq M, X_{(k)}[j_0, i] \neq 0\}$$

- 8: Compute the residual matrix

$$E_{j_0} = Y - \sum_{j \neq j_0} d_j \alpha_j^T$$

, where x_j are the j 'th rows in the matrix $X_{(k)}$.

- 9: Restrict E_{j_0} by choosing only the columns corresponding to ω_{j_0} , and obtain $E_{j_0}^R$.
 - 10: Apply SVD decomposition $E_{j_0}^R = U\Delta V^T$
 - 11: Update the dictionary atom $d_{j_0} = u_1$, where u_1 is the higher value of matrix U , obtained from SVD decomposition.
 - 12: **end for**
 - 13: **end while**
 - 14: **return** D
-

Finally, the term $Z(i)$ is the normalizing constant:

$$Z(i) = \sum_j e^{-\frac{\|v(N_i) - v(N_j)\|_{2,\sigma}}{h^2}} \quad (5.5)$$

where the parameter h acts as a degree of filtering and controls the decay of the exponential function.

Arias [2] proposed a variational framework for the Non-local (NL) image inpainting. They derived three different inpainting schemes. The patch NL-means, patch NL-medians and patch-NL Poisson.

5.4 Poisson Equation

Poisson Equation is widely used for many geometry processing applications. For instance, fairing, mesh editing [37] and mesh completion [22]. The discrete solution of the Poisson PDE on a triangular mesh, is formulated as follows [5]:

$$\Delta^k f = b \quad (5.6)$$

The function $f : S \rightarrow \mathbb{R}$ is defined by the values at the mesh vertices $f_i = f(v_i)$. The Laplace Beltrami Δf can be discretized using the cotangents formula obtaining the Laplacian matrix L . Therefore, the discretization of the Equation 5.7 leads to the linear system:

$$L^k x = b \quad (5.7)$$

The boundary conditions are represented as b and k is the order of the Laplacian. In this work, we compute the bi-Laplacian matrix L^2 to compute an initial reconstruction of the mesh geometry for large holes. This yields the minimum curvature surface patch that satisfies the boundary conditions.

5.5 Principal Curvatures

The curvature on surfaces measures the deviation of a surface from a flat plane. It can be understood as the rate of change of the normal along the surface [7].

Consider a surface X , a trajectory described by $\Gamma : [0, L] \rightarrow X$ and the acceleration vector $\ddot{\Gamma}$ in a direction $v \in T_x X$ where $T_x X$ is the tangent plane at a point x .

A point $x \in X$ has multiple curvatures because for each direction $v \in T_x X$, a curve Γ passing through $\Gamma(0) = x$ in the direction $\dot{\Gamma}(0) = v$ may have a different normal curvature kn . There exists two special curvatures, the principal curvatures which can be defined as:

- Minimum Principal curvature:

$$k_1 = \min_{v \in T_x X} \langle N, \ddot{\Gamma} \rangle$$

- Maximum Principal curvature:

$$k_2 = \max_{v \in T_x X} \langle N, \ddot{\Gamma} \rangle$$

The principal directions are defined as follows:

- Minimum Principal direction:

$$T_1 = \arg \min_{v \in T_x X} \langle N, \ddot{\Gamma} \rangle$$

- Maximum Principal direction:

$$T_2 = \arg \max_{v \in T_x X} \langle N, \ddot{\Gamma} \rangle$$

Principal curvatures over an elliptic and a hyperbolic surfaces are depicted in Figure 5.2.

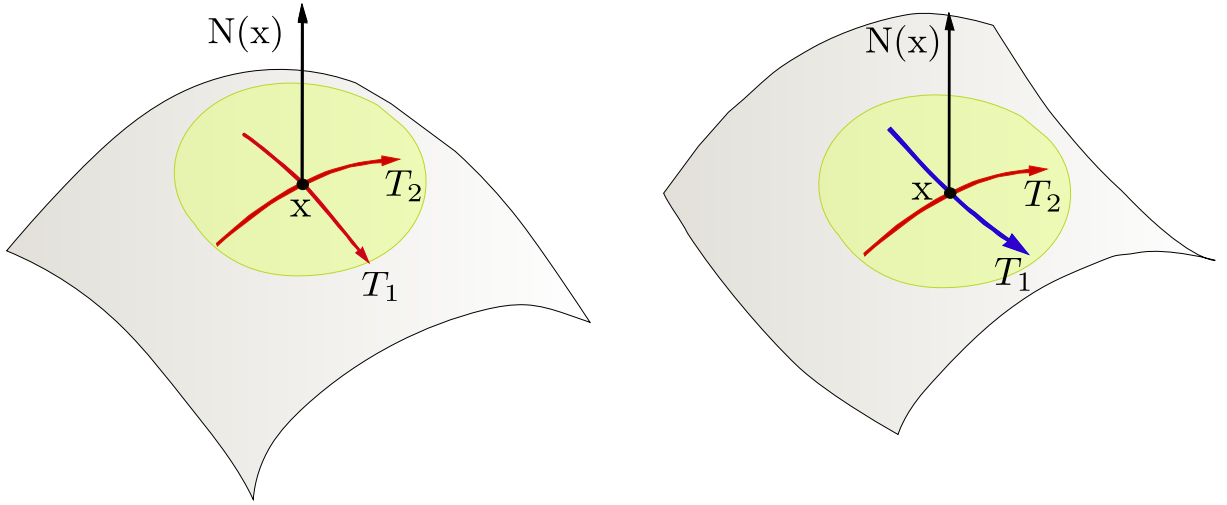


Figure 5.2: The Principal curvatures over an elliptic surface (left) and a hyperbolic surface (right). In an elliptic surface both principal curvatures are positive, conversely, in the hyperbolic surface the principal curvatures have different signs [7].

Chapter 6

Surface Inpainting with Dictionary learning

In this chapter, we present the method we propose to solve the problem of inpainting in surfaces described by meshes. In the first section, we describe an overview of the method and point out each main step that is part of our pipeline. In the following sections, each step is presented in detail.

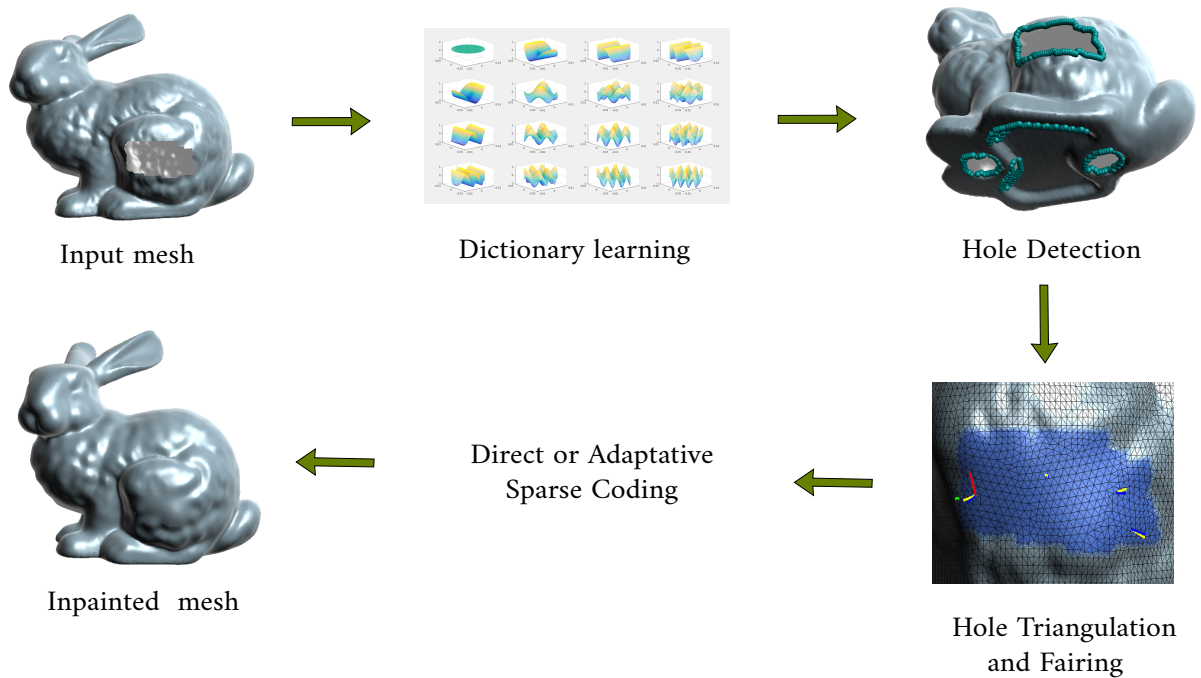


Figure 6.1: Surface inpainting via Dictionary Learning proposed method.

6.0.1 Overview of the method

An overview of the proposed method is depicted in Figure 6.1. The first step is to learn the dictionary from the input data. This stage is composed by the following substeps: patch creation, patch normalization and the building construction of the continuous dictionary. The next step is responsible for building the topology of the missing regions. In order to achieve this, a hole triangulation stage is performed. This stage has two steps: hole detection and hole triangulation. For large holes, it is necessary to apply a fairing step to the triangulation and initial topology, making the surface smoothly blend with the holes' contours. Finally, we apply the sparse coding algorithm using the trained dictionary to finally reconstruct the patches for the missing regions. We present here two approaches: a direct approach which is appropriate for small holes and an adaptative approach which is suitable for large holes, where it is necessary to deduce the sparse codes in patches that did not exist before in the mesh.

6.1 Dictionary Learning on Triangular meshes

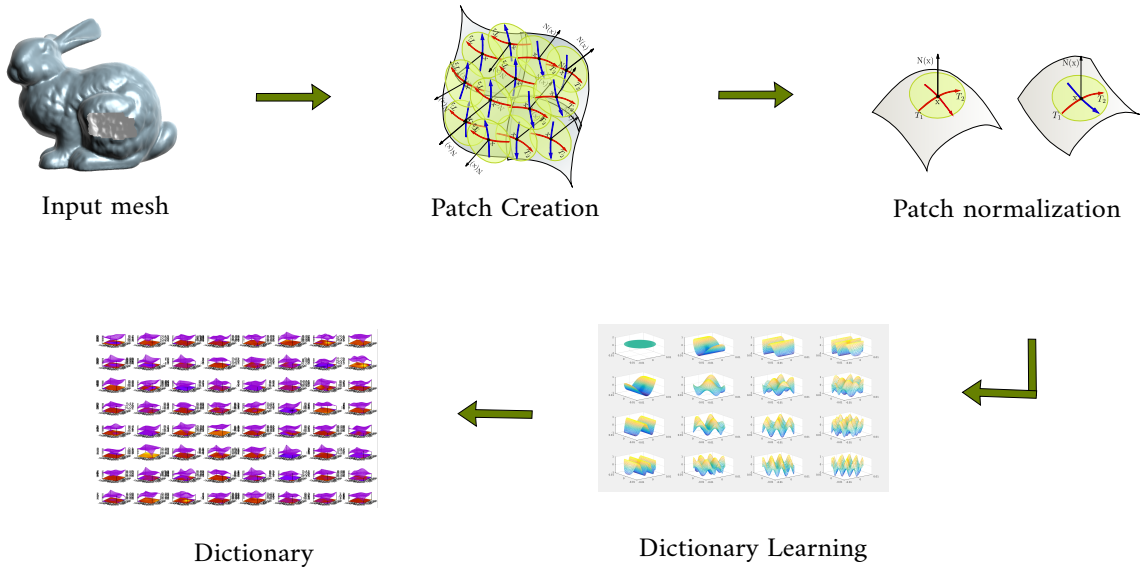


Figure 6.2: Dictionary Learning for triangular meshes method.

Previously, in the background section, we introduced the dictionary learning method applied to images. The main difference between image and surface dictionary learning is the non-uniform sampling in the latter. In the next subsections we describe some methods proposed in the literature to deal with this problem.

The process used to train the dictionary for surfaces is depicted in Figure 6.2. In an

analogous way to image inpainting, the whole surface must be divided into small patches because the training methodology is restricted to low dimensional signals. However, the surface patches present different orientations (normals), and for that reason, it is necessary to re-orient them. Then, a tangent plane is fitted over the points to create a height map for each patch. After this, the signal for the training step is represented by the values of the constructed height map instead of the patches coordinates in \mathbb{R}^3 .

6.1.1 Patch Creation

The subdivision of an image into patches is simple because of the uniform sampling. Nevertheless, in surfaces, it is a non-trivial task. We define a surface patch as a growing region, which has its origin at a center point that we denoted by *seed*. The region is expanded according to a given radius by computing the minimal geodesics from the seeds.

Formally a patch is defined as follows:

Definition 6.1.1 (Patch). Let $T = (V, E, F)$ be a triangular mesh. A patch p is a tuple (s_p, V_p) where $s_p \in V$ and $V_p \subseteq V$ is a set of vertices inside a geodesic ball with radius r :

$$V_p = \{v \in V : d(s_p, v) \leq r\}$$

where d is the geodesic distance function $d : V \times V \rightarrow \mathbb{R}$ and s_p is the center point or seed.

In this formulation, the radius is a very important factor, because it defines the patch size, which has a strong influence in the mesh' reconstruction result. First, because it defines the degree of overlapping between the patches, which serves to create redundancy and allows a better representation of the signal. Second, because when the patches' size is too large, the results tend to present oversmoothing artifacts. This can be explained by the fact that dictionary learning is usually solved as an optimization problem and in many cases the solution is trapped in local minima [28]. Therefore, a good policy is to reduce the patches' size to obtain the finest details as possible.

The radius r is scaled by a factor σ , called *overlapping factor* and can be set by the user to handle the degree of smoothness in the reconstruction. The minimum radius is a lower bound and it is coherent with the mesh geometry. It is computed as follows:

- **Considering all vertices as seed points:** A straightforward way to compute the patches's radius is by averaging the lengths of the edges of the mesh.

- **Considering a subset of vertices seed points:** Using all vertices is computationally intensive. Therefore, one important step is to perform a sampling to define the seeds to be used in the subsequent steps. The radius is computed according to Equation 6.1:

$$r = \max_{v \in V} d_v(v, S) \quad (6.1)$$

where $d_v(v, S)$ is the minimal geodesics distance from the vertex v to the set of seed points S .

We consider two criteria for a good sampling [7]:

- **r-covering:** The sampling must guarantee the covering of the whole surface.
- **r-separated:** The sampling must contain a subset of samples that are well separated.

The **r-covering** property can be understood as the *point-to-set distance* as follows:

$$d_v(v, S) \leq r \quad (6.2)$$

where the distance $d_v(v, S)$ is the minimum *point-to-set distance*.

$$d_v(v, S) = \inf_{s \in S} d_v(v, s) \quad (6.3)$$

These properties are guaranteed by using the Farthest Point Sampling algorithm in the sampling step of our method.

6.1.2 Patch Normalization

Before solving the dictionary learning problem stated in Equation 5.2. We represent each patch geometry $G_p = (X_p, Y_p, Z_p)$ as a height map Z_p defined as a function $Z_p : (X_p, Y_p) \rightarrow \mathbb{R}$ defined in a local coordinate systems. For each patch, we construct a local coordinate system based on a frame defined by the normal N_p and the direction of maximum curvature T_p at the point associated to the patches' seed s_p . Each coordinate system can be modeled as a linear transformation matrix E_p . Then, a plane is fitted over the set of points which are projected onto it. Thus, the height map Z_p is defined for each

patch in a local coordinate system permitting an easier manipulation of the geometry information and the definition of correspondences between the geometry of overlapping regions among neighboring patches.

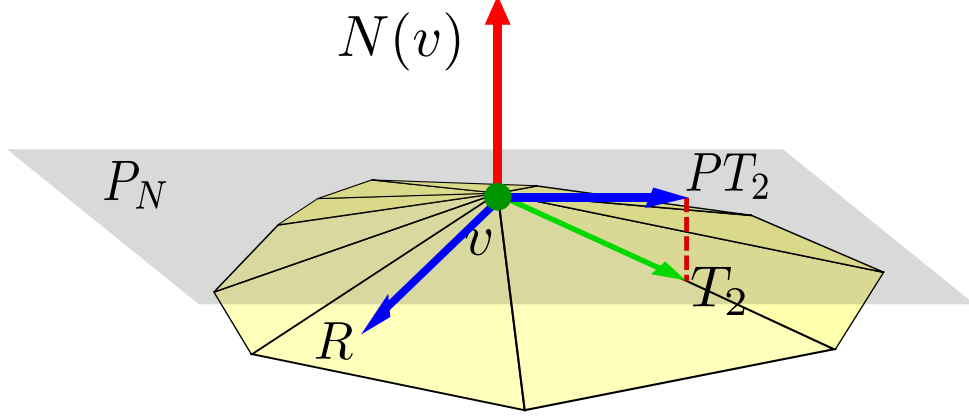


Figure 6.3: Maximum Principal curvature direction over a triangular mesh at the point v .

In this work, we consider only the Maximum Principal Direction to define a local system of coordinates. In the discrete domain, we consider the multiple directions as the vectors that form part of the ring of level 1 of the vertex v . This is depicted in Figure 6.3.

Once the maximum direction is obtained, we perform a projection of the vector T_2 onto the normal plane P_N . Finally the last vector \vec{R} is obtained via the vectorial product between the normal vector and the projection vector PT_2 .

$$\vec{R} = \vec{N} \times P\vec{T}_2$$

6.1.3 Learning a continuous dictionary on surfaces

When an image is considered as the input signal, a dictionary can be learned from a set of overlapping patches which describe the features of the whole image. Differently, when the input signal is a surface, the division into patches does not present a uniform distribution of the points, i.e., it means each patch contains different number of points and they are not located on regular grids as in images. Therefore, it is not possible to use the standard sparse coding algorithms to train the dictionary [26].

Digne et al.[15] proposed an approach to learn a dictionary from point clouds as input data. This method resamples the surface on a radial grid, introducing interpolation steps, which affects the performance of the dictionary learning methods.

A more elegant solution to this problem was proposed by Tal and Bronstein [26], where

a continuous dictionary is learned instead of a discrete one. The dictionary is represented by a combination of a discrete basis Φ_x which is obtained by a continuous basis Φ and its coefficients A .

Then, the dictionary learning problem is defined as:

$$x \approx \Phi_x A \alpha \quad (6.4)$$

In this formulation A is the set of coefficients to be learned. The atoms of the dictionary are defined as a linear combination of the continuous functions:

$$D = \Phi_x A \quad (6.5)$$

$$d_i = \sum_j a_{ij} \Phi_j \quad (6.6)$$

6.2 Hole Identification

The first step to surface inpainting is the detection of the holes in the mesh. It can be performed by looking for the boundary edges. In order to do this in an efficient way, the mesh data must be represented by a topological data structure; in our implementation, we used the Compact Half Edge (CHE) data structure [24]. Before performing the hole detection step, it is necessary to consider minimal requirements on meshes. We assume that the mesh is an oriented manifold. A pre-processing step perform mesh repair operations if such requirements are not satisfied. Some types of flaws and defects that can occur in polygon meshes are: Isolated and dangling elements, singular vertices and edges, topological noise, inconsistent orientation, holes with islands, gaps with partial overlap, self-intersections, and others. We refer the readers to the mesh repairing survey [3], which analyses and compares numerous mesh repairing methods.

6.3 Hole Triangulation and Fairing

Our method is directly influenced by the hole triangulation step; for that reason, it is important to establish the interior mesh connectivity (topology). A proper hole triangulation must ensure a compatible vertex density between the inserted mesh and the surrounding

mesh.

In our pipeline we can adopt any method proposed in the literature for hole triangulation and refinement. For instance the methods of Liepa [25] and [38]. However, here we propose our own way to solve this partial problem.

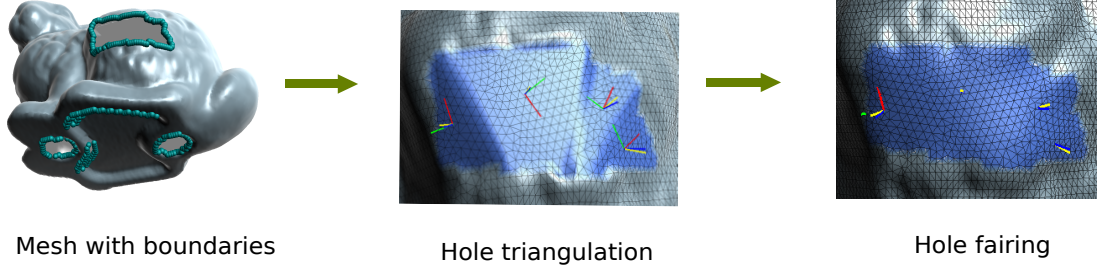


Figure 6.4: Hole triangulation and fairing process.

Our method for hole triangulation is based on a piece-wise scheme, because it divides the complex hole into sub-holes in a similar way to the method proposed by Jun [21]. However, once the complex hole is divided, each sub-hole is filled with an advancing front method, which is similar to the method presented in [38], to create the initial triangulation. Finally, if the hole is large, we perform a fairing step based on the Poisson equation, in order to make the surface smooth. The complete process for this stage is depicted in Figure 6.4.

The proposed hole-filling algorithm is described in 4. The advancing front method consists in iteratively adding new triangles until the hole is covered; the new triangles are created according to the rules depicted in Figure 6.5. The size of the edges is chosen by averaging the sizes of the boundary edges, to enforce the coherence with the size of the edges of the surrounding mesh. This method is more efficient and robust than the traditional 3D polygon triangulation methods because it is always able to patch the hole whatever its shape is [38].

In this approach, we create an inpainted patch starting from the boundary edges of the hole. Then, the advancing front method reduces the boundary edges until rests only a small hole (a triangle). Finally, we merge the inpainted patch and the initial mesh to get the new mesh without holes.

Finally, if the hole is large, the coordinates of the mesh vertices are optimized by solving the Poisson equation (5.7), achieving a smoothing surface.

Algorithm 3 Create new triangles**Require:** Triangular mesh $T = (V, E, F)$, input vertex $v_i \in V$, angle θ **Ensure:** Triangular mesh $T = (V, E, F)$ including new triangles.

```

1: if  $\theta \leq 75^\circ$  then
2:   Add the triangle composed by  $f = v_{i-1}, v_i, v_{i+1}$ 
3: else if  $\theta > 75^\circ$  and  $\theta \leq 135^\circ$  then
4:   Create a new vertex  $v_n$  between  $v_{i-1}$  and  $v_{i+1}$ .
5:   Add two triangles:  $f_1 = \{v_{i-1}, v_i, v_n\}$  and  $f_2 = \{v_n, v_i, v_{i+1}\}$ 
6: else if  $\theta > 135^\circ$  then
7:   Create two new vertices  $v_{n1}$  and  $v_{n2}$  between  $v_{i-1}$  and  $v_{i+1}$ 
8:   Add three new triangles:  $f_1 = \{v_{i-1}, v_i, v_{n1}\}$ ,  $f_2 = \{v_{n1}, v_i, v_{n2}\}$  and  $f_3 = \{v_{n2}, v_i, v_{i+1}\}$ 
9: end if
10: return  $T$ 

```

Algorithm 4 Hole-filling**Require:** Triangular mesh $T = (V, E, F)$, boundary vertices $B \in V$.**Ensure:** Triangular mesh without holes.

```

1: Project the vertices  $b \in B$  to the plane adjusted to the boundary using PCA.
2:  $Q$  is a priority queue.
3: for all  $b \in B$  do
4:   Calculate the angle  $\theta_b$  between two adjacent boundary vertices to vertex  $b$ .
5:    $Q \leftarrow Q \cup \{\theta_b\}$ 
6: end for
7: while  $Q \neq \emptyset$  do
8:    $\theta_m = \min Q$ 
9:   Create the new triangles using Algorithm 3 with the vertex  $m$  and the angle  $\theta_m$ .
10:  Update  $Q$  with the angles of the new vertices.
11: end while
12: return  $T$ 

```

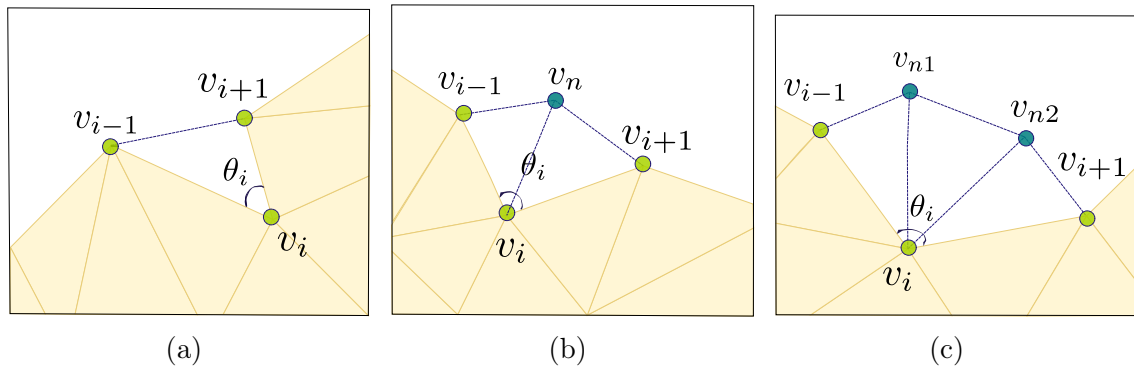


Figure 6.5: Rules for the Create new triangles (algorithm) 3. The first case, when $\theta_i \leq 75^\circ$ is shown in Figure 6.5a. The second case, when $75^\circ < \theta \leq 135^\circ$ is shown in Figure 6.5b. Third case, when $\theta_i > 135^\circ$ is shown in Figure 6.5c

6.4 Sparse Coding approaches

6.4.1 Direct Sparse Coding approach

This approach adapts the original dictionary learning techniques for images to surfaces described by meshes. The inpainting result is good as long as the holes are not bigger than the patches size. For large holes it is necessary to use iterative, multiscale or texture synthesis methods [27].

Considering these restrictions, in this method the sparse codes α are computed by solving the optimization problem (5.1), using the dictionary learned in the previous stage.

Algorithm 5 Direct Inpainting Algorithm

Require: Triangular mesh (V, E, F)

Ensure: Triangular mesh without holes (V, E, F)

- 1: Train the dictionary
 - 2: Fill the holes with the proposed Filling Holes Algorithm 4
 - 3: Generate new patches to cover the holes.
 - 4: Compute the α vector for the new patches using the OMP algorithm
 - 5: Reconstruct the surface according the mesh reconstruction step.
 - 6: **return** T
-

Algorithm 5 summarizes the proposed Surface Inpainting algorithm with the Direct Sparse Coding approach.

6.4.2 Adaptative Sparse Coding approach

We believe that the borders of the holes contain the most compatible information with the interior of the holes. Following this idea, we developed an algorithm which performs a successive propagation of the patch boundary sparse codes toward the interior hole patches. So that, the boundary patches are updated until they cover the whole hole.

This strategy was thought that way, because for large holes, the information is quite limited, especially for the hole patches that are more distant to the hole boundaries.

Consider the following sets: P , V are the initial set of patches and set of vertices respectively, P' and V' are the extended set of patches and set of vertices after the hole-triangulation stage.

Definition 6.4.1. Let $B \subseteq V$ be the set of border patches and $s(p)$ a function that yields

the seed (center point) of patch $p \in P$. We define $R_i \subseteq P'$:

$$R_i = \{p \in P' : s(p) \in V'(R_{i-1})\} \quad (6.7)$$

$$R_0 = \{p \in P : \exists v \in V'(p) \cap B\} \quad (6.8)$$

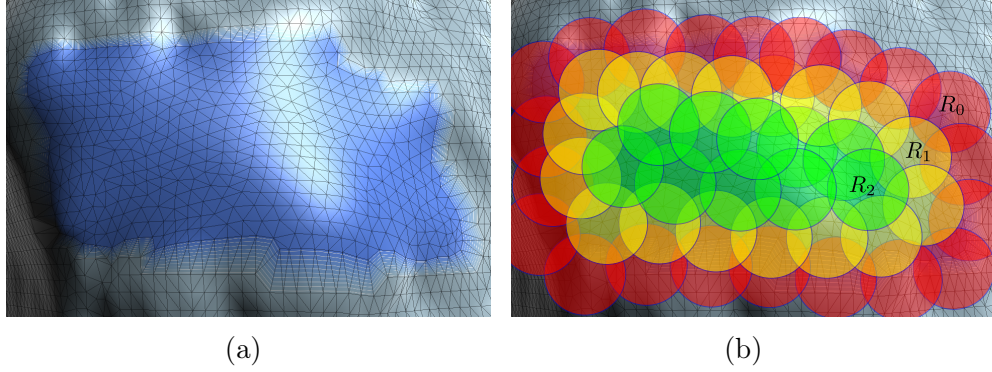


Figure 6.6: Growing regions R_i for the Iterative Inpainting algorithm. [6.6a](#) Initial hole triangulation; [6.6b](#) The growing regions are R_0 (red), R_1 (yellow) and R_2 (green). The number of levels to cover the hole is $r = 3$.

In this definition R_0 is the initial set of patches that has an intersection with the border patches. The sets R_i represent the growing regions in each iteration. The number of levels needed to cover the hole is denoted as r . An illustration of the growing regions is depicted in [6.6](#). The sparse codes of a patch p are obtained by a weighted average of the sparse codes of its neighboring patches denoted as \mathcal{N}_p .

$$\mathcal{N}_p = \{p' \in \bigcup_{i < r_p} R_i : |V_p \cap V_{p'}| > 0\} \quad (6.9)$$

where r_p is the level of the patch p . The weight for a patch p' is defined as follows:

$$w(p') = \frac{|V_p \cap V_{p'}|}{W_p} \quad (6.10)$$

where W_p is defined as:

$$W_p = \sum_{p' \in \mathcal{N}_p} |V_p \cap V_{p'}| \quad (6.11)$$

Algorithm [6](#) summarizes the proposed Surface Inpainting algorithm with the Adaptive Sparse Coding approach.

Algorithm 6 Surface Inpainting with the Adaptive Sparse Coding approach.

Require: Triangular mesh (V, E, F) **Ensure:** Triangular mesh without holes (V, E, F)

- 1: Train the dictionary
 - 2: Fill the holes with the proposed Filling Holes Algorithm 4
 - 3: Compute the poisson Equation 5.7, updating the new coordinates.
 - 4: Generate new patches $p \in P'$ to cover the holes.
 - 5: Compute the α vector for the new patches using the OMP algorithm
 - 6: **for** $i \leftarrow 1$ to r **do**
 - 7: **for all** $p \in R_i$ **do**
 - 8: $\alpha(p) = \sum_{p' \in \mathcal{N}_p} w(p')\alpha(p')$
 - 9: **end for**
 - 10: **end for**
 - 11: Reconstruct the surface according the mesh reconstruction step.
 - 12: **return** T
-

6.5 Mesh reconstruction

Once the dictionary D and the sparse codes α have been learned, we compute the new height map Z_p for each patch $p \in P'$:

$$Z_p = \Phi_p A \alpha_p \quad (6.12)$$

The patches have to return to their original position in the mesh by using the inverse of the transformation matrix E_p , since E_p is a orthonormal basis, we can affirm that $E_p^{-1} = E_p^T$.

So far, each vertex v has m_v estimates (for each patch that contains v). Notice that the parameter m_v is different for each v because each patch contains a different number of vertices. The set of patches that contain v is denoted as P_v and $|P_v| = m_v$.

The final position $G(v)$ for each vertex $v \in V'$ is computed by averaging its m_v estimates:

$$G(v) = \frac{1}{m_v} \sum_{p \in P_v} G_p(v) \quad (6.13)$$

In this work, we perform a better way to averaging the estimates of each vertex, using a weighted average based on the Non-local means method (see Section 5.3).

We are not aware whether there exists a technique that applies the Non-local means

method to surfaces, so we adapted the Non-local means method of images to surfaces, considering the degree of similarity as weights to perform the averaging.

Thus, the final position $G(v)$ is computed as follows:

$$G(v) = \sum_{p \in P_v} w_{vp} G_p(v) \quad (6.14)$$

where w_{vp} and $Z(v)$ are denoted as:

$$w_{vp} = \frac{1}{Z(v)} e^{-\frac{\|\alpha_v - \alpha_p\|}{h^2}} \quad (6.15)$$

$$Z(v) = \sum_{p \in P_v} e^{-\frac{\|\alpha_v - \alpha_p\|}{h^2}} \quad (6.16)$$

In this formula, the parameter h acts as a degree of filtering and controls the decay of the exponential function. The magnitude of the h value controls the influence of the patches in the reconstruction. When h is large, the weights are even more attenuated when the samples are distant from the point of reconstruction. This leads to a more local behavior in the reconstruction process. Conversely, when h is small, the weights are less attenuated and the distant samples contribute more for the reconstruction, producing a smoother effect [9].

Chapter 7

Experiments and results

In this section, we demonstrate the performance of our methods in filling surface holes.

First, we show the basis that we considered to build the dictionaries, the Gaussian basis and the Cosine basis, which are depicted in Figures 7.1a. Some example of dictionaries learned from the Bunny and Armadillo models are depicted in Figures 7.2a and 7.2b respectively.

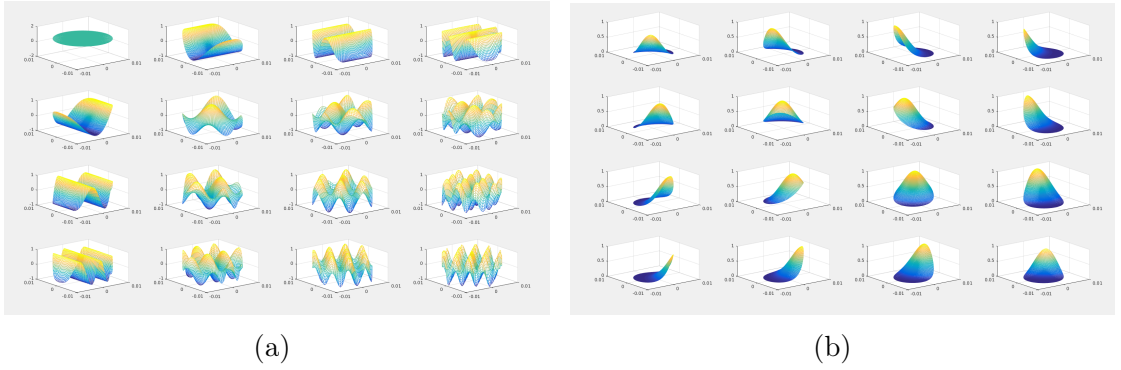


Figure 7.1: Cosine 7.1a and Gaussian 7.1b basis for Dictionary Learning algorithm.

The proposed method, based on dictionary learning, refines the hole filling result, as soon as the topology is added in the hole triangulation step. Figure 7.3 shows the inpainting result of the Bunny model with a large hole. This is a complex example because the whole mesh is not smooth and presents some finer details that we call as texture. Our results are compared with the hole-filling method of [38]. The inpainted Bunny shape with our method tends to be more coherent with the whole remaining shape. Furthermore, the inpainting result presents an enhancement compared to the original Bunny shape, highlighting the features and removing noise.

An example with a large and curved hole is depicted in Figure 7.4. We can observe the

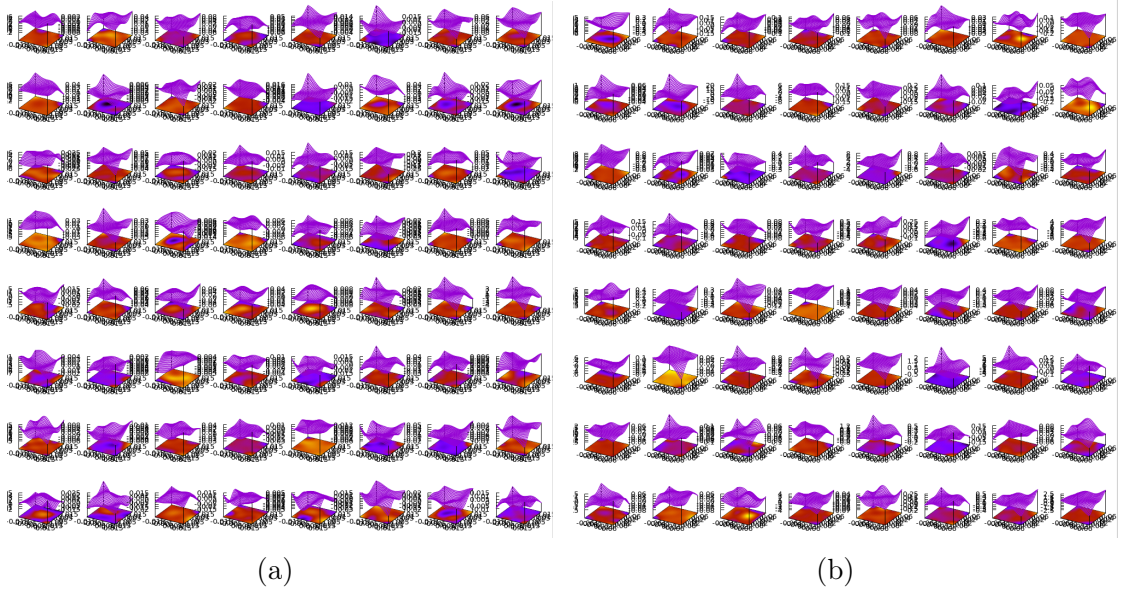


Figure 7.2: Dictionary atoms learned from shapes. 7.2a Dictionary atoms learned from the bunny model, the overlapping factor is $\sigma = 2.5$; 7.2b Dictionary atoms learned from the Armadillo model, the overlapping factor is $\sigma = 1.5$. Both dictionaries has 16 basis functions as shown in Figure 7.1.

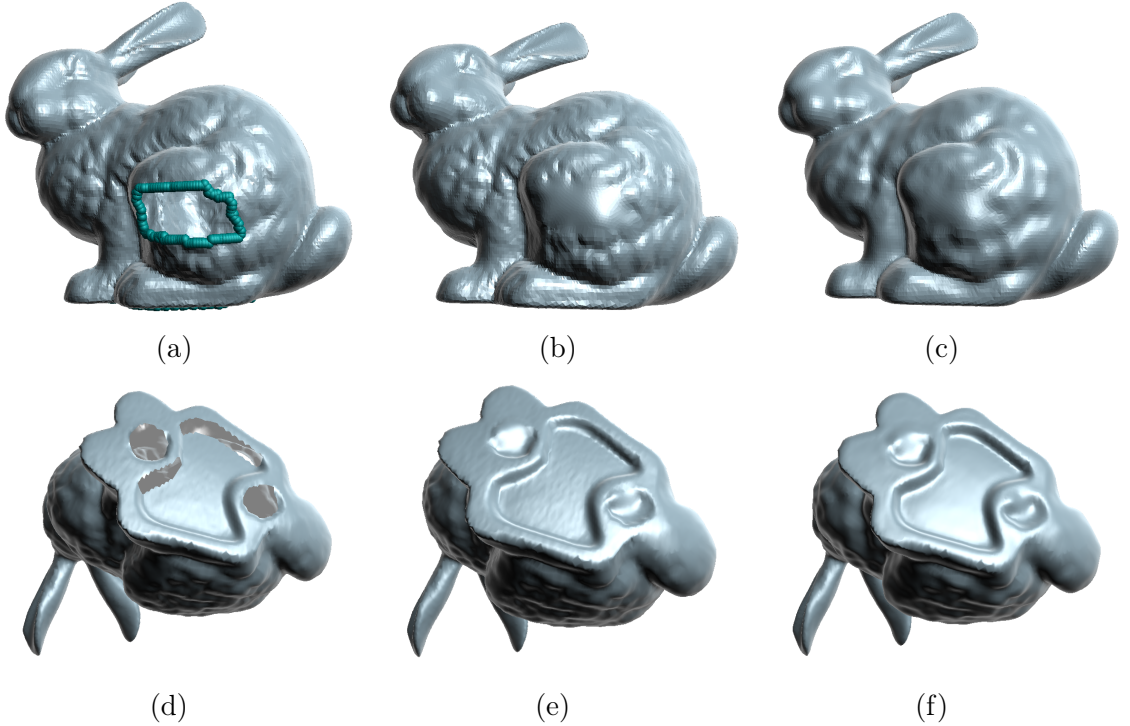


Figure 7.3: Inpainting existing holes on the Bunny model. 7.3a, 7.3d Original Bunny model with holes; 7.3b, 7.3e Hole-filling result using the method of [38]; 7.3c, 7.3f Hole filling result using our Adaptive Inpainting method.

mesh triangulation introduced by our method has good quality and approximates well the missing geometry. Also, the final inpainting result performs a smoothing over the whole

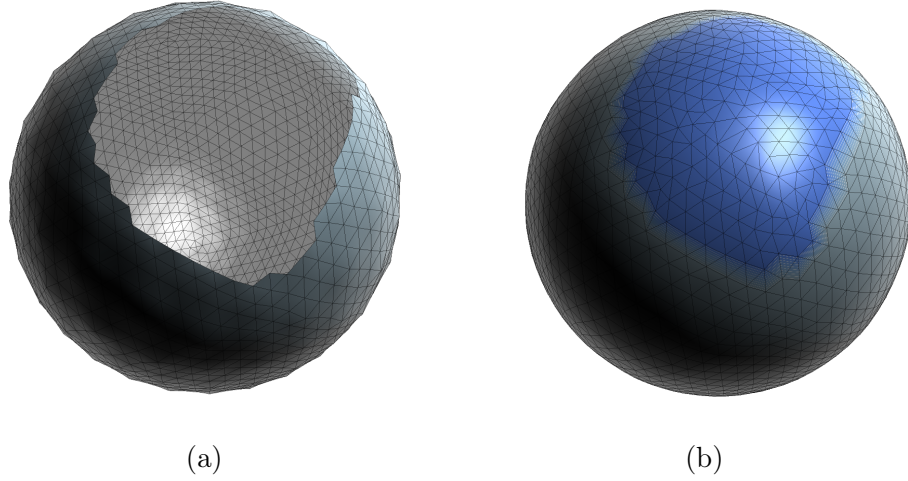


Figure 7.4: 7.4a Original Sphere model with a hole. 7.4b Hole-filling result using the proposed method.

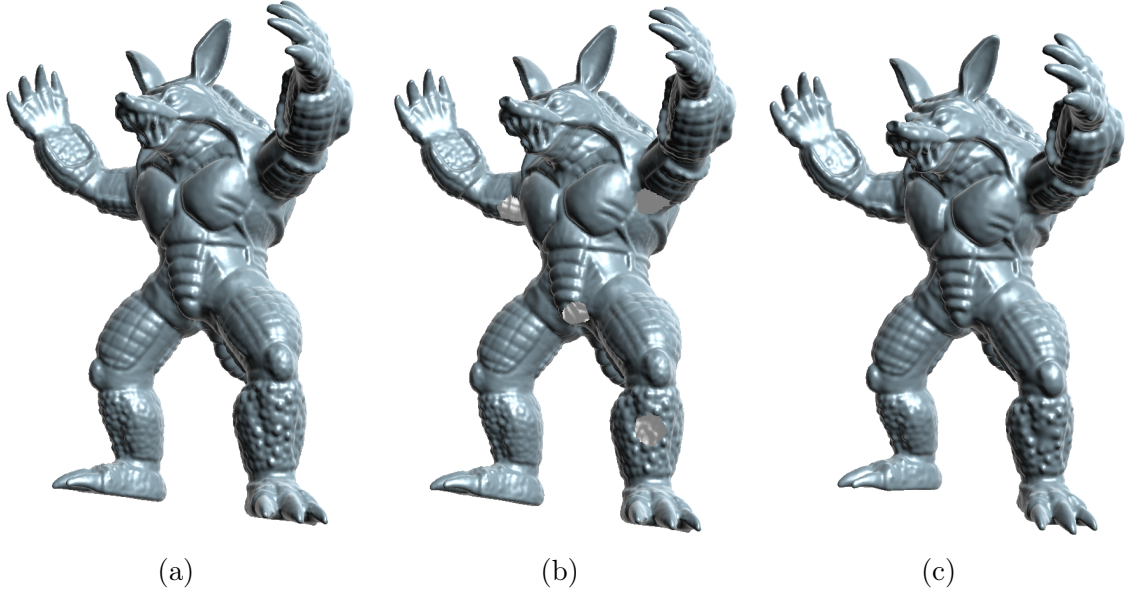


Figure 7.5: Inpainting existing holes on Armadillo model. 7.5a, Original Armadillo model; 7.5b, Original Armadillo model with holes; 7.5c Hole filling result using our the proposed method considering all vertices as seeds.

surface, improving the quality of the input mesh. Another example is shown in Figure 7.8, the resultant triangulation of the Wolf shape is coherent with its surroundings.

In Figure 7.5, we can observe the inpainting result of the Armadillo shape. The result recovers the parts adding some texture as we can observe in the Armadillo's leg. Another example can be appreciated in Figure 7.6, where the Armadillo's arm and forearm are successfully recovered.

The proposed method is robust to noise. In Figure 7.7a, we can observe a noise

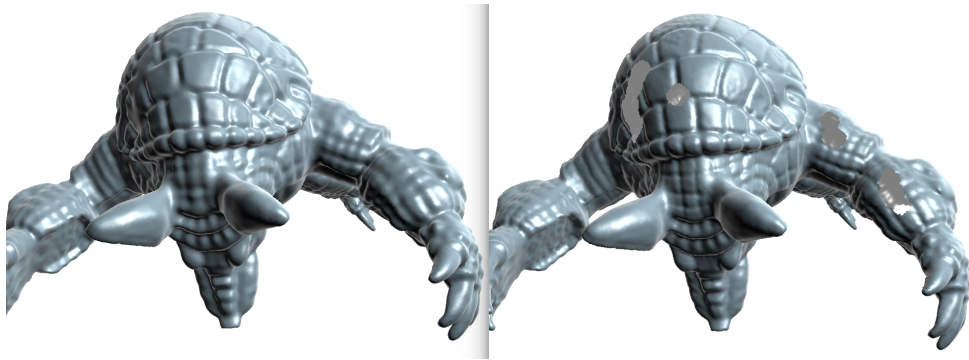


Figure 7.6: Inpainting existing small holes in Armadillo shape

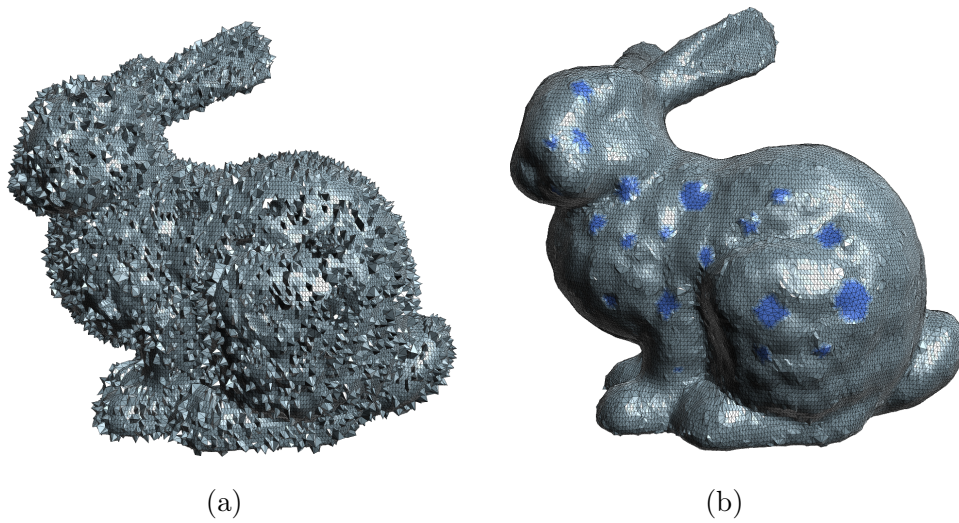


Figure 7.7: Inpainting existing holes on Bunny shape with noise. [7.7a](#) Original Bunny model with noise and small holes. [7.7b](#) Hole-filling result using our Direct inpainting method .

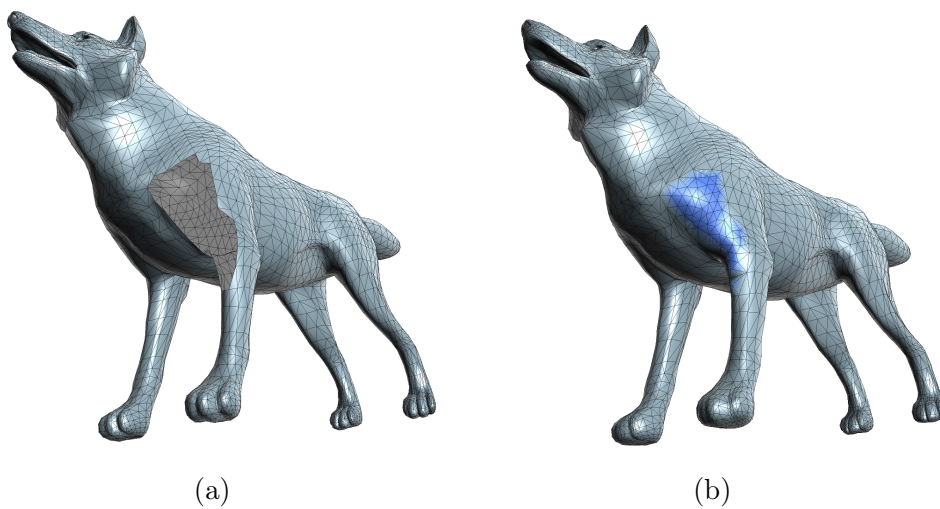


Figure 7.8: Inpainting existing large hole on Wolf shape. [7.7a](#) Original Wolf model with a large hole. [7.7b](#) Hole-filling result using our Iterative inpainting method .

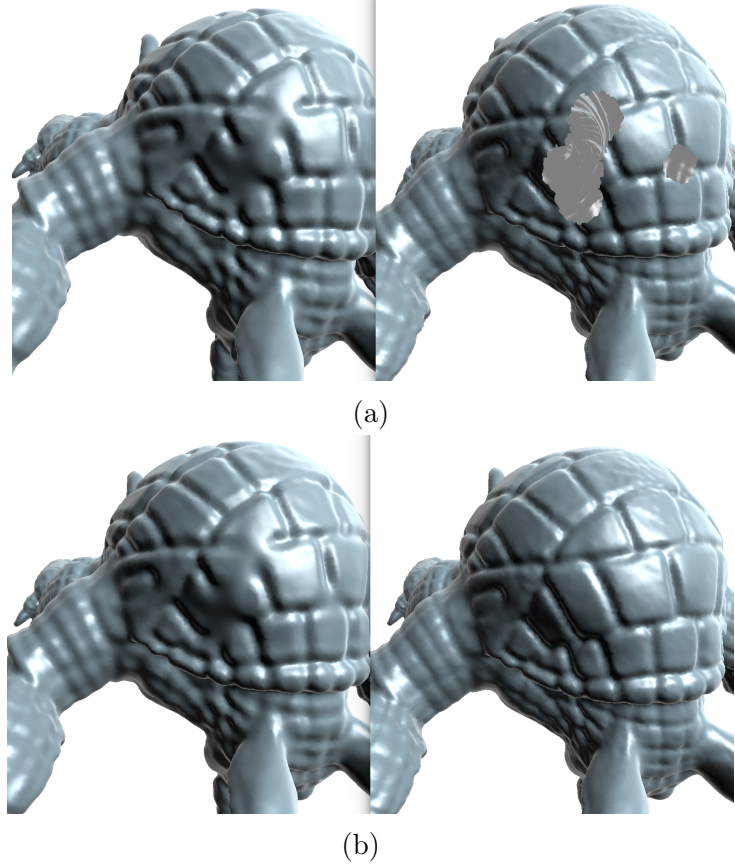


Figure 7.9: Inpainting existing large hole in Armadillo shape. 7.9a Inpainting result using our Iterative Inpainting method (left). The input Armadillo shape with holes (right). 7.9b Difference between the inpainting result (left) and the original Armadillo model without holes (right).

Bunny shape with small holes. The inpainting result with our Direct Inpainting method is shown in Figure 7.7b, where the model enhances the Bunny features and remove the noise considerably.

A main shortcoming of the proposed method is the way it deals with scale. As long as the holes are not significant larger than the patches' size, the proposed method works well. Otherwise, the functions learned from the patches can not approximate well the features for large missing regions. An example of this is shown in Figures 7.9 where a large hole in the Armadillo shape is inpainted with the Iterative proposed method. We can observe that there exist a significant difference compared to the original Armadillo model in the back hard plates of the Armadillo shape. A solution to solve this issue can be addressed by using a multiscale dictionary so that the details levels can be handled in a hierarchical data structure. These ideas remain as a future work.

Chapter 8

Conclusions

In this work, we proposed a new method for the surface inpainting problem based on dictionary learning and sparse coding techniques.

The proposed method works in a transformed domain instead of directly working on the mesh geometry and topology. The surface is approximated by a sparse linear combination of atoms of a dictionary and the sparse codes. Once the dictionary is learnt we can recover the missing information, by estimating the sparse codes in a patch-wise scheme.

We have demonstrated experimentally the performance of our method for surface inpainting. Even for complex holes and surfaces with noise, the inpainting results are quite good. The measurement of the quality of the inpainted result is a subjective task. However, we assume that from the point of view of an observer, who doesn't know the original surface; the inpainting result is good as long as the observer is not able to notice whether the inpainted result is the original or not.

The main advantages of the proposed method are:

- The inpainting result is globally coherent with the observably parts of the surface.
- The proposed method enhances the whole surface and is robust to noise.
- The inpainting result introduces some texture, especially when the holes are small.

For the future work we plan to design a method to overcome the issues related to the scale, because for large holes the atoms of the dictionary are not able to learn the texture patterns of the missing regions. One idea to deal with the scale problem is to learn a hierarchical dictionary. We are also interested in improve the mesh reconstruction stage

to avoid an undesired oversmooth result.

References

- [1] AHARON, M.; ELAD, M.; BRUCKSTEIN, A. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *Trans. Sig. Proc.* 54, 11 (Nov. 2006), 4311–4322. [11](#), [12](#)
- [2] ARIAS, P.; CASELLES, V.; SAPIRO, G. *A Variational Framework for Non-local Image Inpainting*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 345–358. [1](#), [3](#), [6](#), [14](#)
- [3] ATTENE, M.; CAMPEN, M.; KOBELT, L. Polygon mesh repairing: An application perspective. *ACM Comput. Surv.* 45, 2 (Mar. 2013), 15:1–15:33. [22](#)
- [4] BAC, A.; TRAN, N.-V.; DANIEL, M. A multistep approach to restoration of locally undersampled meshes. In *Proceedings of the 5th International Conference on Advances in Geometric Modeling and Processing* (Berlin, Heidelberg, 2008), GMP’08, Springer-Verlag, pp. 272–289.
- [5] BOTSCH, M.; ALLIEZ, P.; KOBELT, L.; PAULY, M.; LÉVY, B. *Polygon mesh processing*. Natick, Mass. A K Peters, 2010. [15](#)
- [6] BRANCH, J.; PRIETO, F.; BOULANGER, P. Automatic hole-filling of triangular meshes using local radial basis function. In *Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT’06)* (Washington, DC, USA, 2006), 3DPVT ’06, IEEE Computer Society, pp. 727–734. [5](#)
- [7] BRONSTEIN, A.; BRONSTEIN, M.; KIMMEL, R. *Numerical Geometry of Non-Rigid Shapes*, 1 ed. Springer Publishing Company, Incorporated, 2008. [vi](#), [15](#), [16](#), [20](#)
- [8] BRUNTON, A. A.; WUHRER, S. S.; SHU, C. C.; BOSE, P.; DEMAINE, E. E. D. Filling holes in triangular meshes by curve unfolding. [6](#)
- [9] BUADES, A.; COLL, B.; MOREL, J. M. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)* (June 2005), vol. 2, pp. 60–65 vol. 2. [12](#), [28](#)
- [10] CAI, T.; WANG, L. Orthogonal matching pursuit for sparse signal recovery with noise. *Information Theory, IEEE Transactions on* 57, 7 (July 2011), 4680–4688. [10](#)
- [11] CASELLES, V.; HARO, G.; SAPIRO, G.; VERDERA, J. On geometric variational models for inpainting surface holes. *Comput. Vis. Image Underst.* 111, 3 (Sept. 2008), 351–373. [1](#), [3](#), [6](#)
- [12] CENTIN, M.; PEZZOTTI, N.; SIGNORONI, A. Poisson-driven seamless completion of triangular meshes. *Comput. Aided Geom. Des.* 35, C (May 2015), 42–55. [1](#), [3](#)

- [13] CURLESS, B.; LEVOY, M. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 303–312. [5](#)
- [14] DAVIS, J.; MARSCHNER, S. R.; GARR, M.; LEVOY, M. Filling holes in complex surfaces using volumetric diffusion. In *3DPVT* (2002), IEEE Computer Society, pp. 428–438. [5](#)
- [15] DIGNE, J.; CHAINE, R.; VALETTE, S. Self-similarity for accurate compression of point sampled surfaces. *Comput. Graph. Forum* *33*, 2 (2014), 155–164. [21](#)
- [16] ELAD, M. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, 1st ed. Springer Publishing Company, Incorporated, 2010. [10](#), [11](#), [14](#)
- [17] ELAD, M.; STARCK, J.-L.; QUERRE, P.; DONOHO, D. Simultaneous cartoon and texture image inpainting using morphological component analysis (mca). *Applied and Computational Harmonic Analysis* *19*, 3 (2005), 340 – 358. Computational Harmonic Analysis - Part 1.
- [18] ENGAN, K.; AASE, S.; HAKON HUSOY, J. Method of optimal directions for frame design. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on* (1999), vol. 5, pp. 2443–2446 vol.5. [12](#)
- [19] GEORGE, P. L.; SEVENO, E. The advancing-front mesh generation method revisited. *International Journal for Numerical Methods in Engineering* *37*, 21 (1994), 3605–3619.
- [20] GIRYES, R.; ELAD, M.; BRUCKSTEIN, A. M. Sparsity based methods for over-parametrized variational problems. *CoRR abs/1405.4969* (2014). [12](#)
- [21] JUN, Y. A piecewise hole filling algorithm in reverse engineering. *Computer-Aided Design* *37*, 2 (2005), 263 – 270. [6](#), [23](#)
- [22] KAZHDAN, M.; BOLITHO, M.; HOPPE, H. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 61–70. [15](#)
- [23] KIM, H. S.; CHOI, H. K.; LEE, K. H. Feature detection of triangular meshes based on tensor voting theory. *Computer-Aided Design* *41*, 1 (2009), 47 – 58.
- [24] LAGE, M.; LEWINER, T.; LOPES, H.; VELHO, L. Che: A scalable topological data structure for triangular meshes. Preprint MAT 13/05, PUC-Rio, May 2005. [22](#)
- [25] LIEPA, P. Filling holes in meshes. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2003), SGP '03, Eurographics Association, pp. 200–205. [6](#), [23](#)
- [26] LITANY, O.; REMEZ, T.; BRONSTEIN, A. Cloud dictionary: Sparse coding and modeling for point clouds. *ArXiv e-prints* (Dec. 2016). [21](#)

- [27] MAIRAL, J.; ELAD, M.; SAPIRO, G. Sparse representation for color image restoration. *Trans. Img. Proc.* 17, 1 (Jan. 2008), 53–69. [25](#)
- [28] MAIRAL, J.; SAPIRO, G.; ELAD, M. Learning multiscale sparse representations for image and video restoration. *Multiscale Modeling & Simulation* 7, 1 (2008), 214–241. [19](#)
- [29] NGO, H. T. M.; LEE, W.-S. *Feature-First Hole Filling Strategy for 3D Meshes*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 53–68.
- [30] PEYRÉ, G. Manifold models for signals and images. *Comput. Vis. Image Underst.* 113, 2 (Feb. 2009), 249–260.
- [31] PFEIFLE, R.; SEIDEL, H.-P. Triangular b-splines for blending and filling of polygonal holes. In *Proceedings of the Conference on Graphics Interface '96* (Toronto, Ont., Canada, Canada, 1996), GI '96, Canadian Information Processing Society, pp. 186–193. [5](#)
- [32] ROJAS, M. A.; SUKNO, F. M.; WADDINGTON, J. L.; WHELAN, P. F. Quantitative comparison of hole filling methods for 3d object search. In *Proceedings of the 7th Eurographics Workshop on 3D Object Retrieval* (Aire-la-Ville, Switzerland, Switzerland, 2014), 3DOR '15, Eurographics Association, pp. 25–32. [5](#)
- [33] SAHAY, P.; RAJAGOPALAN, A. N. Geometric inpainting of 3d structures. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (June 2015), pp. 1–7.
- [34] VAN WITT, F. A. C. Inpaintings, 2017. [Online; accessed January 28, 2017].
- [35] WANG, J.; OLIVEIRA, M. M. Filling holes on locally smooth surfaces reconstructed from point clouds. *Image and Vision Computing* 25, 1 (2007), 103 – 113. [5](#)
- [36] WANG, X.; LIU, X.; LU, L.; LI, B.; CAO, J.; YIN, B.; SHI, X. Automatic hole-filling of {CAD} models with feature-preserving. *Computers & Graphics* 36, 2 (2012), 101 – 110. Virtual Reality in Brazil 2011.
- [37] YU, Y.; ZHOU, K.; XU, D.; SHI, X.; BAO, H.; GUO, B.; SHUM, H.-Y. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 644–651. [15](#)
- [38] ZHAO, W.; GAO, S.; LIN, H. A robust hole-filling algorithm for triangular mesh. *The Visual Computer* 23, 12 (2007), 987–997. [vii](#), [5](#), [6](#), [23](#), [29](#), [30](#)
- [39] ZHONG, M.; QIN, H. Surface inpainting with sparsity constraints. *Computer Aided Geometric Design* 41 (2016), 23 – 35. [6](#)