 Dictionary Learning-based Inpainting on Triangular Meshes

Lizeth J. Fuentes Perez\textsuperscript{a}, Luciano A. Romero Calla\textsuperscript{a}, Anselmo A. Montenegro\textsuperscript{b}

\textsuperscript{a}Universidad La Salle, Arequipa - Perú
\textsuperscript{b}Universidade Federal Fluminense, Niterói RJ - Brazil

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. In this work, we address the problem of inpainting on surfaces through a new method based on dictionary learning and sparse coding. 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, Poison equation, Hole-filling methods, Triangular meshes

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 [1]. 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 [2].

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) [3]. 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 work, 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 paper 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.

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.

2. 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 [4] proposed a volumetric method in voxel space. David [5] 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 [6].

In triangle-based approaches, there also exists also different kind of approaches. We refer readers to the survey in [7], 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 [8] proposed a method that creates an interpolating patch using moving least squares to fit polynomial functions. Pfeifle and Seidel [9] create an interpolating patch using triangular B-splines and Branch [10] 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 [11], 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 [12] 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 [6] 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 [2].

Brunton [13] 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. 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 [1].

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 [14]. 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 adaptive inpainting result.

4. 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 [15]. 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 $y \approx D\hat{a}$. The minimization problem of sparse coding is formulated as follows:
\[ \hat{\alpha} = \arg \min_{\alpha} \| y - D\alpha \|_2 \quad s.t. \quad \| \alpha \|_0 \leq L \] (1)

Note that in this formulation, the dictionary is given. The representation is guided by the sparse codes \( \alpha \), where \( \alpha \) are the sparse coefficients to approximate the signal \( y \) as sparse as possible. This implies that only the meaningful atoms are considered [15]. The term \( \| \alpha \|_0 \) measures the sparsity of the decomposition and can be understood as the number of non-zero coefficients in \( \alpha \), 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 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 \( D \). The OMP algorithm [15] is summarized in Algorithm 4.1.

**Algorithm 4.1 Orthogonal Matching Pursuit (OMP) algorithm**

**Require:** Dictionary \( D \), sparsity constraint \( L \)

**Ensure:** Sparse coding vector \( \alpha \)

1. \( \Gamma = \emptyset \)
2. for \( i = 1, \ldots, L \) do
3. Select the atom which most reduces the objective function
   \[ i' \leftarrow \arg \min_{i \in \Gamma^c} \{ \min_{\alpha'} \| x - D_{\Gamma \cup \{i\}} \alpha' \|_2^2 \} \]
4. Update the active set: \( \Gamma \rightarrow \Gamma \cup \{i'\} \)
5. Update the residual (orthogonal projection)
   \[ r \rightarrow (I - D_{\Gamma}(D_{\Gamma}^T D_{\Gamma})^{-1} D_{\Gamma}^T) x \]
6. Update the coefficients \( \alpha \)
   \[ \alpha_{\Gamma} \rightarrow (D_{\Gamma}^T D_{\Gamma})^{-1} D_{\Gamma}^T x \]
7. end for
8. return \( \alpha \)

### 5. 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 [15]. 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 [15]. 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[17].

Formally, the dictionary learning problem can be formulated as:

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

Given signal \( Y \), the solution to Equation 2 learns a dictionary and its sparsest representation \( \alpha \) over the unknown dictionary \( D \), updating \( D \) until the proper dictionary is found, considering the sparsity constraints for \( \alpha \), 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. [17].

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[19]. An image denoised via Dictionary Learning is shown in Figure 1.

#### 5.1. 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 [17] 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 make sparse the signals solution to Equation 2.

The general algorithm for K-SVD algorithm is described in Algorithm 5.1. The parameter \( \epsilon \) ensures the similarity between the input signal and the approximate \( D\alpha \) signal.

### 6. Non-local-means

The non-local-means is a method proposed by Buades [20] to solve the image denoising problem. It has its spirit
Figure 1: Image denoising via Dictionary Learning. 1a Barbara image with Gaussian noise; 1b Barbara image recovered; 1c Barbara original image; 1d Dictionary learned from the Barbara image with Gaussian noise.

based on the non-local averaging of all pixels in the image. The non local value of a pixel \( i \) of an image \( I \) is:

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

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:

\[
w(i,j) = \frac{1}{Z(i)} e^{-\frac{\|v(N_i) - v(N_j)\|_2^2}{\sigma h^2}}
\]  

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

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

\[
Z(i) = \sum_j e^{-\frac{\|v(N_i) - v(N_j)\|_2^2}{\sigma h^2}}
\]  

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

Arias [1] proposed a variational framework for the Nonlocal (NL) image inpainting. They derived three different inpainting schemes. The patch NL-means, patch NL-medians and patch-NL Poisson.

7. Poisson Equation

Poisson Equation is widely used for many geometry processing applications. For instance, fairing, mesh editing [21] and mesh completion [22]. The discrete solution of the Poisson PDE on a triangular mesh, is formulated as follows [23]:

\[
\Delta^k f = b
\]  

The function \( f : S \rightarrow \mathbb{R} \) is defined by the values at the mesh vertices \( f_i = f(v_i) \). The Laplace Beltrami \( \Delta f \) can be discretized using the cotangents formula obtaining the Laplacian matrix \( L \). Therefore, the discretization of the Equation 7 leads to the linear system:

\[
L^k x = b
\]  

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.

8. 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 [24].

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 \( \Gamma \) passing through \( \Gamma(0) = x \) in the direction \( \dot{\Gamma}(0) = v \) may have a different normal curvature \( k_n \). 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 2.

Algorithm 5.1 The K-SVD dictionary-learning algorithm [15]

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 $\alpha'_i$ for $1 \leq i \leq M$.
These form the matrix $X(k)$
4: K-SVD Dictionary-Update Stage:
5: for $j_0 = 1, 2, ..., m$ 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 $\omega_{j_0}$, 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 $D$

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.

8.1. Overview of the method

An overview of the proposed method is depicted in Figure 3. The first step is to learn the dictionary from the input data. This stage is composed by the following sub-steps: 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 adaptive 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.

9. Dictionary Learning on Triangular meshes

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 4. 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-oriented 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 $\mathbb{R}^3$.

9.1. 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 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 over-smoothing 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 [25]. 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 $\sigma$, 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 8:

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

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 [24]:

- **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:

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

\[
d_v(v, S) = \inf_{s \in S} d_v(v, s) \tag{10}
\]

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

### 9.2. Patch Normalization

Before solving the dictionary learning problem stated in Equation 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 5: Maximum Principal curvature direction over a triangular mesh at the point v.](image)

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 5.

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 \( \bar{R} \) is obtained via the vectorial product between the normal vector and the projection vector \( PT_2 \).

\[
\bar{R} = \bar{N} \times PT_2
\]

### 9.3. 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.[27] 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 \( \Phi \) which is obtained by a continuous basis \( \Phi \) and its coefficients \( A \).

Then, the dictionary learning problem is defined as:

\[
x \approx \Phi_x A \alpha \tag{11}
\]

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:

\[
D = \Phi_x A \tag{12}
\]

\[
d_i = \sum_j a_{ij} \Phi_j \tag{13}
\]

### 10. 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 [28]. 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 [29], which analyses and compares numerous mesh repairing methods.

### 11. 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 [11] and [6]. However, here we propose our own way to solve this partial problem.
Our method for hole triangulation is based on a piecewise scheme, because it divides the complex hole into subholes in a similar way to the method proposed by Jun [12]. However, once the complex hole is divided, each subhole is filled with an advancing front method, which is similar to the method presented in [6], 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.

The proposed hole-filling algorithm is described in 11.2. 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 7. 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 [6].

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 (7), achieving a smoothing surface.

12. Sparse Coding approaches

12.1. 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 [30].

Considering these restrictions, in this method the sparse codes $a$ are computed by solving the optimization problem (4), using the dictionary learned in the previous stage.

Algorithm 12.1 summarizes the proposed Surface Inpainting algorithm with the Direct Sparse Coding approach.
forms 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 2.** 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 $R_0 \subseteq P'$:

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

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

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 8. The sparse codes of a patch $p$ are obtained by a weighted average of the sparse codes of its neighboring patches denoted as $N_p$.

**Algorithm 12.2** summarizes the proposed Surface Inpainting algorithm with the Adaptative Sparse Coding approach.

13. Mesh reconstruction

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

$$Z_p = \Phi_p A \alpha_p$$

The patches have to return to their original position in the mesh by using the inverse of the transformation
Algorithm 12.2 Surface Inpainting with the Adaptative Sparse Coding approach.

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

\begin{enumerate}
\item Train the dictionary
\item Fill the holes with the proposed Filling Holes Algorithm 11.2
\item Compute the poisson Equation 7, updating the new coordinates.
\item Generate new patches $p \in P'$ to cover the holes.
\item Compute the $\alpha$ vector for the new patches using the OMP algorithm
\item for $i \leftarrow 1$ to $r$ do
\item \hspace{1em} for all $p \in R_i$ do
\item \hspace{2em} $\alpha(p) = \sum_{p' \in N_p} w(p')\alpha(p')$
\item \hspace{1em} end for
\item \hspace{1em} end for
\item Reconstruct the surface according the mesh reconstruction step.
\item return $T$
\end{enumerate}

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:

$$G(v) = \frac{1}{m_v} \sum_{p \in P_v} G_p(v)$$ (20)

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 6).

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:

$$G(v) = \sum_{p \in P_v} w_{vp}G_p(v)$$ (21)

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

$$w_{vp} = \frac{1}{Z(v)} e^{-\frac{|\alpha_v - \alpha_p|}{h^2}}$$ (22)

$$Z(v) = \sum_{p \in P_v} e^{-\frac{|\alpha_v - \alpha_p|}{h^2}}$$ (23)

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 [20].

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 9a. Some example of dictionaries learned from the Bunny and Armadillo models are depicted in Figures 10a and 10b respectively.

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 11 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 [6]. 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 12. We can observe the 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 surface, improving the quality of the input mesh. Another example is shown in Figure 16, the resultant triangulation of the Wolf shape is coherent with its surroundings.

In Figure 13, 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 14, where the Armadillo’s arm and forearms are successfully recovered.

The proposed method is robust to noise. In Figure 15a, we can observe a noise Bunny shape with small holes. The inpainting result with our Direct Inpainting method is shown in Figure 15b, 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 significative 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 17 where a large hole in the Armadillo shape is inpainted with the Iterative proposed method. We can observe that there exist a significative 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.

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.
Figure 11: Inpainting existing holes on the Bunny model. 11a, 11d Original Bunny model with holes; 11b, 11e Hole-filling result using the method of [6]; 11c, 11f Hole filling result using our Adaptative Inpainting method.

Figure 12: 12a Original Sphere model with a hole. 12b Hole-filling result using the proposed method.

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.

14. Acknowledgements

I would like to thank professor Alex Bronstein, who provided important guidance for this work.

References

[1] P. Arias, V. Caselles, G. Sapiro, A Variational Framework for Non-local Image Inpainting, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 345–358. doi:10.1007/
Figure 13: Inpainting existing holes on Armadillo model. 13a, Original Armadillo model; 13b, Original Armadillo model with holes; 13c Hole filling result using our the proposed method considering all vertices as seeds.

Figure 14: Inpainting existing small holes in Armadillo shape

---

978-3-642-03641-5_26.
URL http://dx.doi.org/10.1007/978-3-642-03641-5_26

[2] V. Caselles, G. Haro, G. Sapiro, J. Verdera, On geometric variational models for inpainting surface holes, Comput. Vis. Image Underst. 111 (3) (2008) 351–373. doi:10.1016/j.cviu.2008.01.002.
URL http://dx.doi.org/10.1016/j.cviu.2008.01.002

[3] M. Centin, N. Pezzotti, A. Signoroni, Poisson-driven seamless completion of triangular meshes, Comput. Aided Geom. Des. 35 (C) (2015) 42–55. doi:10.1016/j.cagd.2015.03.006.
URL http://dx.doi.org/10.1016/j.cagd.2015.03.006

[4] B. Curless, M. Levoy, A volumetric method for building complex models from range images, in: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH ’96, ACM, New York, NY, USA, 1996, pp. 303–312. doi:10.1145/237170.237269.
URL http://doi.acm.org/10.1145/237170.237269

[5] J. Davis, S. R. Marschner, M. Garr, M. Levoy, Filling holes in complex surfaces using volumetric diffusion., in: 3DPVT, IEEE Computer Society, 2002, pp. 428–438.

[6] W. Zhao, S. Gao, H. Lin, A robust hole-filling algorithm for triangular mesh, The Visual Computer 23 (12) (2007) 987–997. doi:10.1007/s00371-007-0167-y.
URL http://dx.doi.org/10.1007/s00371-007-0167-y

[7] M. A. Rojas, F. M. Sukno, J. L. Waddington, P. F. Whelan, Quantitative comparison of hole filling methods for 3D object search, in: Proceedings of the 7th Eurographics Workshop on 3D Object Retrieval, 3DOR ’15, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2014, pp. 25–32. doi:10.2312/3dor.20141046.
URL http://dx.doi.org/10.2312/3dor.20141046

[8] J. Wang, M. M. Oliveira, Filling holes on locally smooth surfaces reconstructed from point clouds, Image and Vision Computing 25 (1) (2007) 103 – 113.

[9] R. Pfeifle, H.-P. Seidel, Triangular b-splines for blending and filling of polygonal holes, in: Proceedings of the Conference on Graphics Interface ’96, GI ’96, Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 1996, pp. 186–193.
URL http://dl.acm.org/citation.cfm?id=241020.241073

[10] J. Branch, F. Prieto, P. Boulanger, Automatic hole-filling of
Figure 15: Inpainting existing holes on Bunny shape with noise. 15a Original Bunny model with noise and small holes. 15b Hole-filling result using our Direct inpainting method.

Figure 16: Inpainting existing large hole on Wolf shape. 15a Original Wolf model with a large hole. 15b Hole-filling result using our Iterative inpainting method.

Figure 17: Inpainting existing large hole in Armadillo shape. 17a Inpainting result using our Iterative Inpainting method (left). The input Armadillo shape with holes (right). 17b Difference between the inpainting result (left) and the original Armadillo model without holes (right).

[10] triangular meshes using local radial basis function, in: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06), 3DPVT '06, IEEE Computer Society, Washington, DC, USA, 2006, pp. 727–734. doi:10.1109/3DPVT.2006.33. URL http://dx.doi.org/10.1109/3DPVT.2006.33

[11] P. Liepa, Filling holes in meshes, in: Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, SGP '03, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2003, pp. 200–205. URL http://dl.acm.org/citation.cfm?id=882370.882397

[12] Y. Jun, A piecewise hole filling algorithm in reverse engineering, Computer-Aided Design 37 (2) (2005) 263 – 270. doi:http://dx.doi.org/10.1016/j.cad.2004.06.012. URL http://www.sciencedirect.com/science/article/pii/S0010448504001320

[13] A. A. Brunton, S. S. Wuhrer, C. C. Shu, P. Bose, E. E. Demaine, Filling holes in triangular meshes by curve unfold-
[14] M Zhong, H Qin, Surface inpainting with sparsity constraints, Computer Aided Geometric Design 41 (2016) 23–35. doi: http://dx.doi.org/10.1016/j.cagd.2015.10.003.
URL http://www.sciencedirect.com/science/article/pii/S0167839615001211.

[15] M Elad, Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing, 1st Edition, Springer Publishing Company, Incorporated, 2010.

[16] T Cai, L Wang, Orthogonal matching pursuit for sparse signal recovery with noise, Information Theory, IEEE Transactions on 57 (7) (2011) 4680–4688. doi:10.1109/TIT.2011.2146090.

[17] M Aharon, M Elad, A Bruckstein, K-svd: An algorithm for designing overcomplete dictionaries for sparse representation, Trans. Sig. Proc. 54 (11) (2006) 4311–4322. doi:10.1109/TSP.2006.881199.
URL http://dx.doi.org/10.1109/TSP.2006.881199.

[18] K Engan, S Aase, J Hakon Husoy, Method of optimal directions for frame design, in: Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on, Vol. 5, 1999, pp. 2443–2446 vol.5. doi:10.1109/ICASSP.1999.760624.

[19] R Giryes, M Elad, A M. Bruckstein, Sparsity based methods for overparametrized variational problems, CoRR abs/1405.4969.
URL http://arxiv.org/abs/1405.4969.

[20] A Buades, B Coll, J. M Morel, A non-local algorithm for image denoising, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), Vol. 2, 2005, pp. 60–65 vol. 2. doi:10.1109/CVPR.2005.38.

[21] Y Yu, K Zhou, D Xu, X Shi, H Bao, B Guo, H-Y Shum, Mesh editing with poisson-based gradient field manipulation, ACM Trans. Graph. 23 (3) (2004) 644–651. doi:10.1145/1015706.1015774.
URL http://doi.acm.org/10.1145/1015706.1015774.

[22] M Kazhdan, M Bolitho, H Hoppe, Poisson surface reconstruction, in: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, SGP ’06, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2006, pp. 61–70. URL http://dl.acm.org/citation.cfm?id=1281957.1281965.

[23] M Botsch, P Alliez, L Kobelt, M Pauly, B Lévy, Polygon mesh processing, Natick, Mass. A K Peters, 2010.

[24] A Bronstein, M Bronstein, R Kimmel, Numerical Geometry of Non-Rigid Shapes, 1st Edition, Springer Publishing Company, Incorporated, 2008.

[25] J Mairal, G Sapiro, M Elad, Learning multiscale sparse representations for image and video restoration, Multiscale Modeling & Simulation 7 (1) (2008) 214–241. arXiv:http://dx.doi.org/10.1137/070697653, doi:10.1137/070697653.
URL http://dx.doi.org/10.1137/070697653.

[26] O Litany, T Remez, A Bronstein, Cloud dictionary: Sparse coding and modeling for point clouds, ArXiv e-prints arXiv:1612.04956.

[27] J. Digne, R. Chaine, S. Valette, Self-similarity for accurate compression of point sampled surfaces, Comput. Graph. Forum 33 (2) (2014) 155–164. doi:10.1111/cgf.12305.
URL http://dx.doi.org/10.1111/cgf.12305.

[28] M Lage, T Lewiner, H Lopes, L Velho, Che: A scalable topological data structure for triangular meshes, Preprint MAT 13/05, PUC-Rio (May 2005).

[29] M Attene, M Campen, L Kobelt, Polygon mesh repairing: An application perspective, ACM Comput. Surv. 45 (2) (2013) 15:1–15:33. doi:10.1145/2431211.2431214.
URL http://dx.doi.org/10.1145/2431211.2431214.

[30] J Mairal, M Elad, G Sapiro, Sparse representation for color image restoration, Trans. Img. Proc. 17 (1) (2008) 53–69. doi:10.1109/TIP.2007.911828.
URL http://dx.doi.org/10.1109/TIP.2007.911828.