IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 5, 2020, accepted February 9, 2021, date of publication February 23, 2021, date of current version March 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3061525

Repairing 3D Models Obtained From Range Sensors

EMILIANO PÉREZ[1], SANTIAGO SALAMANCA[1], PILAR MERCHÁN[1], AND ANTONIO ADÁN[2]

[1]Industrial Engineering School, University of Extremadura, 06006 Badajoz, Spain
[2]Computer Science School, University of Castilla-La Mancha, 13071 Ciudad Real, Spain

Corresponding author: Emiliano Pérez (emilianoph@unex.es)

This work was supported in part by the FEDER Funds (Programa Operativo FEDER de Extremadura 2014-2020) through the grant Ayuda a Grupos de Investigación of Junta de Extremadura under Grant GR18159, and in part by the National Project under Grant PID2019-108271RB-C32 (AEI/FEDER, UE).

ABSTRACT This paper presents a method to restore 3D models created from range sensors. The approach recovers the 3D information that is lost during the scanning process, usually due to inaccessible zones on the object (occluded areas and crevices). The raw and incomplete 3D data are first projected onto a grid to identify the zones to be repaired. Our algorithm calculates a suitable projection plane for each lost area defined and restores the 2D projected image. Finally, the inverse 2D to 3D transformation is carried out, and the new 3D data are merged with the initial mesh model, providing the completely restored surface. The approach has been tested on a database containing a wide variety of models, yielding excellent results. The experimental section shows that our method works for a large diversity of non-sensed and difficult-to-access zones and provides precise restorations.

INDEX TERMS Range sensors, polygonal models, mesh repair.

I. INTRODUCTION

The techniques for 3D range data acquisition were first developed in the early 1970s [1]. Technological advances have produced more precise systems, with a higher resolution and greater velocity, in addition to gradually reducing their cost. This has led to the use of 3D scanners in an increasing number of applications [2]–[5].

There are several ways in which to classify range sensors, one of which depends on the distance between the sensor and the objects to be sensed. In this case, sensors can be classified as short, medium or long-range. Range is also related to the size of the objects to be scanned. Short-range sensors can digitize small and medium-sized objects (in the order of several tens of centimeters). The most used technologies for this type of sensors are either structured light [6], [7] or laser scanners [8]. Medium and long distances are scanned using phase shift and time of flight laser scanners, which permit the scanning of large objects and spaces [9].

The two most common types of data representation provided by these sensors are polygonal meshes and point clouds. The main difference between them is that in meshes, the neighborhood relationships between the points are known, in addition to the list of the 3D points coordinates. Usually, these relationships define triangles, which provide a model of the surface of the object. Although both representations can be used for the data originated from short, medium or long-range sensors, the native format for the data provided by short-range sensors is usually the polygonal mesh, while the point cloud is the native output format for medium and long-range sensors.

The information supplied by polygonal meshes is much richer than that given by point clouds, since they provide an explicit representation of the surface of the objects. This has implications in the treatment of meshes that do not exist in the case of point clouds. Thus, while registration is a common problem in the two representations that can be solved by using very similar algorithms such as the classic ''iterative closest point'' [10], ICP, or some of its variants [11], the generation of complete and well-defined surfaces is a question of the polygonal meshes themselves. In this paper, we propose a fast, robust and efficient method that allows a complete representation by means of a polygonal mesh of the surface of an object sensed with a range sensor. The solution

The associate editor coordinating the review of this manuscript and approving it for publication was Jenny Mahoney.

to this problem may be especially relevant in some areas of application, such as 3D printing in which a complete closed model of the object is required.

There are two main reasons why a mesh may lack surface information. The first is that there has been a loss or an incorrect redefinition of the mesh, such as the definition of a missing triangle, during the registration and/or integration process. This is not a serious problem and the hole is directly filled regardless of the geometrical information around the missing zone. The second is lack of geometrical information owing to occlusions or shadows on objects. In this case, some holes that are usually larger than in the previous case may appear, and a more complex technique is required to repair the mesh.

Moreover, when working with free-form objects in some detail, there may be holes located on complex surfaces (with, for example, large variations of curvature), and this makes the reparation process difficult. It is for this reason that despite many authors' attempts to find a solution to this problem, there is still no general procedure for all possible situations that may arise. User intervention is usually required to assess the problem, and there is no automatic algorithm that can provide a solution. Within the 3D reconstruction process, mesh repairing processes are currently rather an open topic, in which the human factor is decisive and which has aroused great interest from a research point of view.

Therefore, what we propose in this document is a robust algorithm for repairing holes in 3D models, that is, a hole-filling algorithm that is valid for any possible configuration of the problem: different types of objects and holes, resolution of the meshes or sizes of holes (A good collection of criteria to define a robust filling algorithm is proposed in [12]). To this end, we used a Field of Experts (FoE) model which is a robust, well-tested procedure that has already been used in image restoration algorithms with excellent results [13], [14]. As mentioned, the FoE was developed to be applied to images, so this work also presents a procedure that automatically determines the best point of view from which to project the 3D mesh to generate a depth image to be restored. A new transformation from 2D to 3D will then be carried out to obtain the restored mesh. In summary, the pipeline that is followed, and which will be explained in more detail in section III, is:

1) 3D to 2D transformation to obtain a depth image.
2) Application of the FoE.
3) 2D to 3D transformation.

As can be deduced from the previous scheme, the depth image generation procedure (item 1) and the 2D to 3D transformation procedure (item 3), are independent of the image restoration algorithms that can be used (item 2). Therefore, the method presented in this work is easily adaptable for future work in which it is decided to use a pipeline such as the one proposed.

A preliminary study that showed that the FoE algorithm could be used for filling holes was presented in [15]. However, the procedure of the study was not really functional since it was not possible to use it in complete 3D meshes with holes, but only in partial views with holes, so the projection points of view were not determined nor was it possible to fill complex gaps such as those described in this paper. The 2D to 3D transformation procedure has also been redone.

The rest of the document is organized as follows: SectionII reviews previous work in this area. Our method is described in general terms, listing its various stages, in Section III. Sections IV to IX provide the details of each of these stages. Experimentation and results are presented in Section X and, finally, our conclusions are given in Section XI.

II. RELATED WORKS

A great variety of repairing methods or hole-filling algorithms for 3D meshes can be found in the literature. An extensive survey and a comparison is proposed in [16]. In addition, other interesting reviews are available in the works of [17], [18] and [19].

There are algorithms in which the input data are the set of 3D coordinates of the surface points of the object with no topological information associated. These data enable the creation of a new representation (mesh, Bezier, implicit functions ...) which, by definition, has no holes. For example, in [20] and [21] the original data are interpolated using alpha shapes. Similarly, in [22] and [23], crusts are used and in [24] spheres are utilized in the interpolation. As a drawback, it must be mentioned that using continuous shapes to interpolate causes increased error when the input data is noisy. To resolve this issue, in [24], the radio of spheres that fill the holes is also iteratively incremented, smoothing and merging processes in iteration are applied so as to avoid gaps between such spheres.

Sets of radial basis functions (RBF) are fitted to point clouds in [25] and [26]. Basically, they carry out a weighted sum of RBFs to obtain a new global function, which generates the whole surface. Another work that also applies RBFs is [27], where after segmenting the data into 3D uniform grids, they reconstruct the implicit surface by means of a continuous deformation of an initial surface, following the Partial Differential Equation (PDE)-based diffusion model. The evolution of this deformation is driven both by the curvature and the distance (computed using RBF) from the data set.

The approach in [28] performs the reconstruction of the surface by applying a B-Spline surface fitting on the point cloud, using the triangular mesh structure to obtain a good data parametrization.

In contrast to the aforementioned methods, there are studies in which mesh repair is independent of the 3D modeling process, thus allowing more flexible working procedures. For example, in [29] the authors carry out an unfolding process of the mesh using an energy minimization process and perform the filling in 2D. Mingqiang *et al.*, in [30], propose an algorithm that is run in three steps. The first step is a hole triangulation by means of a function that optimizes the triangulation angle. In the second, a subdivision is iteratively

applied to make the size of the faces equal to those of the environment of the hole. Finally, a Laplacian filter is applied to smooth the surface. Another example is the study in [31], where after triangulating the hole with the authors' frontal advance technique, the new vertices are positioned by the resolution of Poisson equations (which are based on the appropriate normals and the hole's boundary).

The proposal in [32] introduces a method that synthesizes the missing geometry by using the information from similar parts of the rest of the mesh. It uses a multi-scale representation of the mesh, coarse-to-fine, to identify every patch with a signature. Then, an iterative refination is applied to the target surface to minimize coherence error. Basically, authors find the most similar patch and then adjust it to the specific circumstances of the holes.

In some approaches, some initial specifications are imposed onto the input, such as those in [33] and [34]. In the former, the focus is on filling digitized CAD models' holes. This method relies on a prior knowledge of the numerical model, which the authors call nominal mesh, before digitization. Then, the holes are identified and the differences between the nominal mesh and the digitized point cloud are calculated. Finally, a deformation is applied to the nominal mesh by a minimization of deformation energy, so that the holes are filled. The latter proposal [34] only works on meshes with strong geometric variations. It handles meshes with relief patterns (near-regular patterns, irregular patterns and stochastic patterns). A multiresolution approach is used to decompose the model into a coarse mesh and a relief mesh. By applying texture synthesis techniques in [35], the relief is transferred to the smoothly filled hole of the coarse mesh.

A popular and widely-adopted method for filling holes in triangular meshes is that presented in [36], which extends the dynamic programming technique of [37] in order to handle 3D polygons with irregular shapes by adopting a dihedral-angle-based weighting scheme. Another work that also follows the programming technique is [38], in which the authors offer MeshWorks software to carry out various operations with meshes. Its tools include a hole-filling technique, which is based on [39]. This technique focuses on filling the holes with non-trivial boundaries. They apply a geometric hashing technique to detect and bridge boundary parts with a similar shape. First, they use a partial curve matching technique in order to identify matching boundary portions. Then, they choose and stitch together a consistent set of matched candidates. After this, they initiate the process with the remaining holes, until the mesh is completely closed. A similar technique is also used in [40].

Apart from these two types of hole-filling techniques, there is a great variety of methods which can be considered as a mix of both types, that is, methods in which the input data is a triangular mesh and where volumetric divisions of point clouds are computed to obtain the points that make up the filled hole which is ultimately to be triangulated. Other methods also use volumetric information to detect meaningful parts that can be used to repair the mesh.

A valuable example within this type of technique is [41], which applies the idea of disjointed internal and external volumes computing an octree grid. The method is divided into four stages: detection of boundary cycle, patching of boundary cycle, generation of sign in octree nodes, and reconstruction of surfaces by contouring. The result is a closed model where the sharp features are kept from the original geometry. Authors developed the free software Polymender, which is useful for repairing polygonal meshes.

Other works that also compute a volumetric division are those in [42], [43] and [44]. Argudo *et al.* [42] applies the concept of bi-harmonic fields in order to approximate the signed distance field on a voxel grid in each hole's neighborhood. Conversely, in the proposal in [43], authors copy a smooth patch from suitably selected valid regions of a surface. They segment the point cloud with an octree and build a local implicit approximation of the shape of an octree cell, which is used as a signature. After selecting the most suitable cell to fill a hole, they copy part of the surface and apply an iterative closest point procedure. An extension of this work is presented in [44], where authors analyze similarities in both shape and appearance using the information from curvature and color of the surface patches.

A proposal that uses an implicit function from the well-known Poisson Reconstruction method ([45]) can be found in [46]. This interpolation method computes a ''surface oracle'' that guides a restricted Delaunay tessellation of the hole patches.

Finally, in [47], the authors present an algorithm that re-samples a low-quality digitized polygonal mesh and converts it into a watertight and manifold mesh. Then, it iteratively removes growing neighborhoods of undesired portions and patches the holes, until the mesh is totally fixed.

Our proposal fits in with the second type of procedure mentioned above, that is, with the mesh repairing process independent of the 3D modeling process.

III. SKETCH OF THE MESH REPAIR PROPOSAL

Our method begins with the creation of a list containing all the zones to be repaired. The restoration algorithm is then applied to one zone at a time. In each of the iterations of the method, we focus on one of the badly acquired zones or holes, whose boundaries will be used as the starting seeds to extract a meaningful portion of mesh to generate a 2D image range. The suitability of the size of this mesh portion is determined by both the characteristics of the mesh and the input limitations of the image algorithm. A detailed explanation of these processes is provided in Section V.

A 3D to 2D transformation is then carried out to obtain the range image. To do so, it is necessary to choose a reference plane on which the mesh portion will be projected, and some specific parameters of that reference plane must be computed, as described in Section VI.

The range image on which to apply the restoration algorithm is now available, but it must first be corrected to

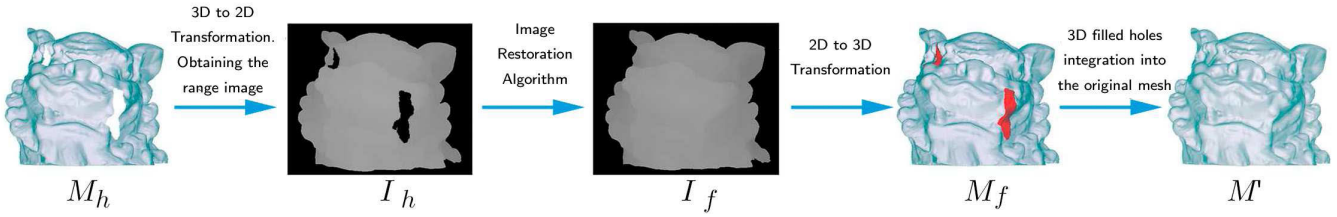

FIGURE 1. Main steps of the method. M_h is the initial mesh with holes. The first stage is the image range generation, I_h corresponding to M_h. In the second stage, an image restoration algorithm is applied to I_h and image I_f is obtained. Finally, a conversion from 2D to 3D representation is performed to generate the mesh M_f (shown in red) which is merged with M_h, resulting in M'.

eliminate any lack of data resulting from the irregularity of the mesh. This is explained in Section VI-D.

After applying an image restoration algorithm, the inverse transformation is carried out and the resulting point cloud is triangulated, as shown in Section VII.

The newly generated mesh portion is ultimately integrated with the whole mesh, as described in Section IX.

Fig. 1 shows a sketch of the repair method in which the different steps outlined above are applied to a lion head mesh.

IV. IDENTIFICATION OF BADLY SENSED AREAS

As has been stated previously, to begin the repairing process it is first necessary to identify the zones to be repaired or holes, and then determine the boundaries of these holes. We consider that input meshes in our algorithm are meshes where every edge is composed of two vertices and is shared by only two triangles (2-manifold), except for those that are in the contour of a hole. Taking into account this consideration, we obtain the boundary of a hole as:

$$B_\rho = \{< v_1, v_2 >, \ldots, < v_k, v_{k+1} >, \ldots, < v_{P-1}, v_P >, < v_P, v_1 >\} \quad (1)$$

The various B_ρ are used to define the set of boundaries of defective areas as $B_m = \cup_i B_\rho$. Once defined, we proceed to repair them, one at a time and in an iterative manner. Fig. 2 shows an example of the identification of four defective areas in a mesh.

V. EXTRACTION OF THE ZONE OF INTEREST

As mentioned above, the input to the restoration algorithm is a range image containing the zone with the missing data. The algorithm uses the information surrounding the badly sensed area to infer the information which does not exist. The size of the area to be projected must be large enough to provide the necessary information. Its extension is defined by the number of concentric rings of faces starting from the hole's boundary, which we have called the neighborhood degree, υ.

The suitable value of υ was empirically computed after carrying out experiments on a set of meshes at different resolutions. Fig. 3 depicts one of the meshes used during experimentation, along with the different values of υ. We sought to find the approximate relationship between the degree of the neighborhood, the resolution and the area of the portion generated. The expression obtained, which proved to be useful with regard to automating our algorithm, is as follows:

$$\upsilon \approx \frac{A_i}{d_m\,(0.106 \cdot \kappa \cdot A_i + 18)} \quad (2)$$

where A_i represents the projected area of the $i-th$ hole on the projection plane, κ is the ratio between the hole size and the size of valid data in the image (number of pixels $\neq 0$), which the restoration algorithm requires to produce adequate results, and, finally, d_m is the average length of the edges of the mesh. The value of κ cannot be computed accurately since it is determined by the quality of the result obtained with the restoration algorithm.

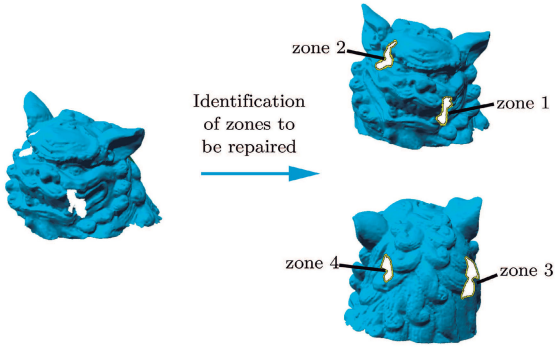

FIGURE 2. Initial identification stage of the h_θ ($\theta = 1, \ldots, m$) holes in the total mesh M_T.

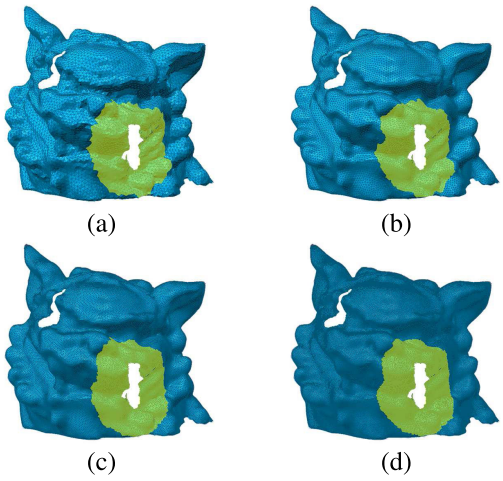

FIGURE 3. Neighborhood grade (υ) required to cover a similar area in the holes of Object 1, $H_i(1)$, depending on the average length of the edges, d_m: (a) $H_1(1)$: $\upsilon = 8$ – $d_m = 0.3406$ *mm*, (b) $H_1(1)$: $\upsilon = 11$ – $d_m = 0.2335$ *mm*, (c) $H_1(1)$: $\upsilon = 18$ – $d_m = 0.1566$ *mm*, (d) $H_1(1)$: $\upsilon = 27$ – $d_m = 0.1043$ *mm*.

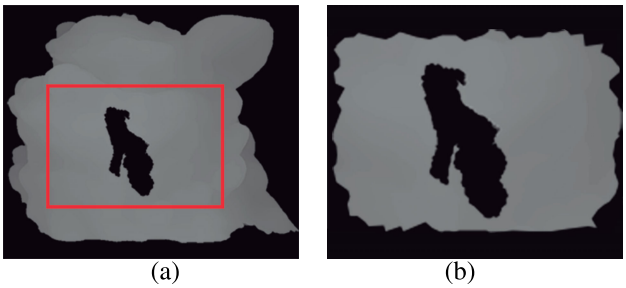

FIGURE 4. Two range images on which the same hole is projected: (a) Image of 257 × 300 pixels, which captures and represents a greater amount of the surface. The hole represents approximately 4% of pixels with information, (b) range image of 120 × 173 for a smaller mesh, equivalent to the area marked in red in (a). The hole covers approximately 12% of pixels in relation to the pixels with information.

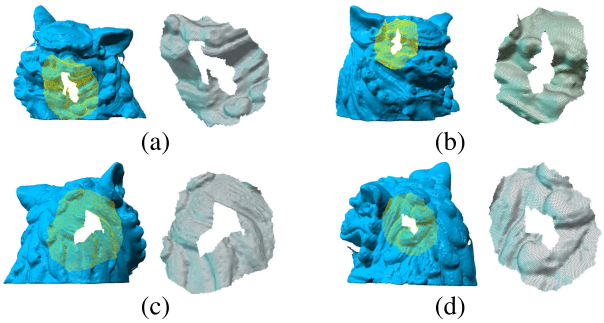

FIGURE 5. Selection of mesh portions for each of the holes appearing in Fig. 2: (a) Zone 1; (b) Zone 2; (c) Zone 3; (d) Zone 4. Different values of υ were chosen in each situation: $\upsilon = 27$; $\upsilon = 21$; $\upsilon = 34$; $\upsilon = 26$, respectively.

As an example, Fig. 4 illustrates the fact that different extensions of the mesh portion give different rates between the area of the mesh and the hole projected onto the corresponding range image. In the case of Fig. 4a, the rate between the number of pixels occupied by the hole and those occupied by the surface is specifically 4 %, whereas in the second figure, Fig. 4b, this value increases to 12 %. The value of κ is conditioned by the performance and quality of the result of the image restoration algorithm. Since a myriad of possible situations can arise, the only way to set a specific value of κ is by means of experiments. We realized that values which generally provide good results are around $\kappa \approx 0.15 - 0.20$, and so we decided to set this value at 0.15. Fig. 5 depicts the portions of meshes extracted for each of the holes shown in Fig. 2.

At this point, it is worth noting that there may be surface topologies in the vicinity of the hole that can potentially lead to errors in the range image generation, as shown in Fig. 6.

Upon observing the figures, it is clearly seen that in all problematic cases the curvature of the surface changes dramatically in some zones near the hole boundary. This curvature can be measured in terms of the normals to the faces in the hole's neighborhood. Substantial changes in the evolution of such normals should be detected and avoided.

We introduced this restriction into the selection process for the zone of interest by developing an algorithm that follows

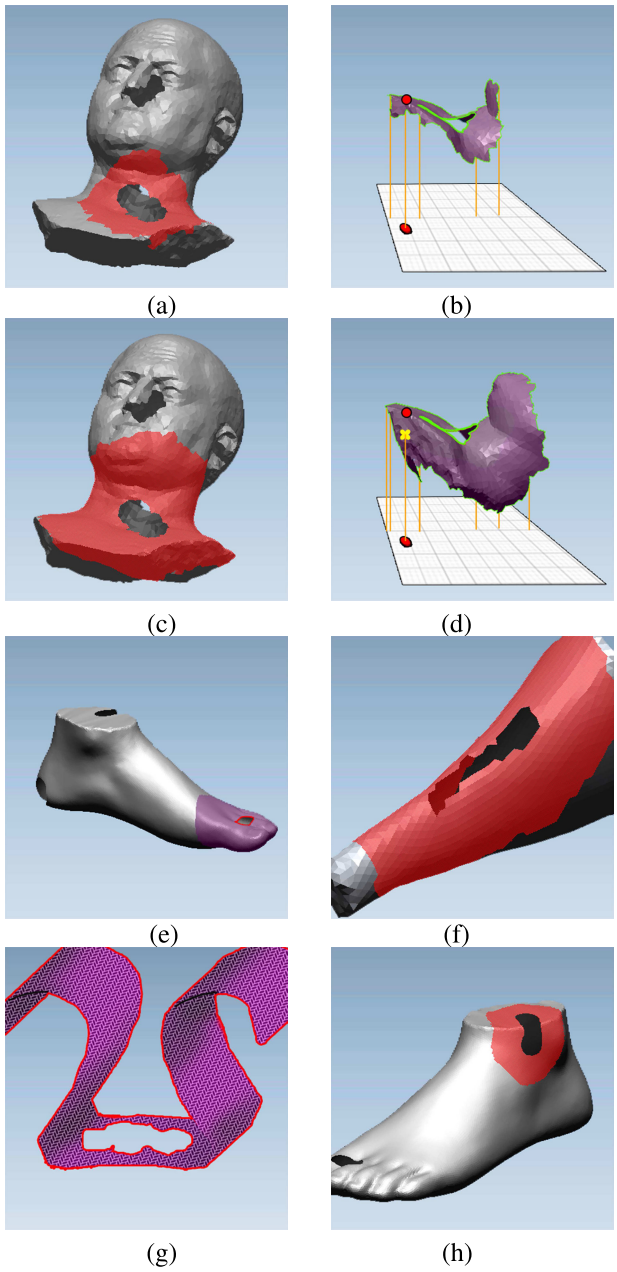

FIGURE 6. (a), (b), (c) and (d) Influence of topology of the hole's surroundings on the error generation in the projection of the mesh portion. In (a) and (b), the projection line of each node does not intersect with the mesh itself, but in (c) and (d) it does, and this must be avoided; (e), (f) and (g) Other similar topologies of the environment of the hole that may give rise to the same error, (h) Topology with a hole located on a sharp change in the surface normals.

the steps described below. Let us consider a mesh composed of a list of vertices $\mathcal{V}$, a list of edges $\mathcal{A}$, a list of faces $\mathcal{T}$, a list of normals $\mathcal{N}$, and the hole B_k.

- We start with the boundary of the hole B_k, from which g nodes that are part of it are extracted:

$$B_k = \{< v_1, v_2 >, < v_2, v_3 >, \ldots, < v_{g-1}, v_g >, < v_g, v_1 >\} \quad (3)$$

where $v_{B_k} = \{v_1, v_2, v_3, \ldots, v_g\}$ is the list of vertices of the hole B_k.

- The list of normals $\mathcal{N}'$ defined is, at first, identical to the list of normals $\mathcal{N}$ associated with each vertex of the mesh. This list is either defined along with the mesh or is calculated before starting the filling process. We shall use the $\mathcal{N}'$ list since it will be dynamically modified during the execution of the algorithm.
- The list of normals corresponding to the hole boundary is obtained, $\mathcal{N}\left(v_{B_k}\right)$.
- The triangle subset $\mathcal{T}_C$, which contains one or two vertices of v_{B_k}, is extracted from v_{B_k} and $\mathcal{T}$. These triangles are candidates for inclusion in the mesh portion. The vertices we shall continue working with (in this case, the vertices v_{B_k}) are called 'search vertices'.
- The normals of the vertices of each triangle $t_i \in \mathcal{T}_C$ are then analyzed. Each t_i is, therefore, composed of three vertices $\left\{v_{B_i}, p_1, p_2\right\}$, one of them being logically a search vertex (belonging to the hole boundary) and the other two being candidates. We compare the normals of the candidate vertices $\overrightarrow{n_{p_1}}$ and $\overrightarrow{n_{p_2}}$ with the normal of the search vertex $\overrightarrow{n_{v_{B_i}}}$. This is done by measuring the angle between these vectors and verifying whether it exceeds a certain threshold value γ. This condition can be formulated using the following expression:

$$\widehat{\overrightarrow{n_{p_j}}\,\overrightarrow{n_{v_{B_i}}}} = \arctan\left(\frac{\left|\overrightarrow{n_{p_j}} \times \overrightarrow{n_{v_{B_i}}}\right|}{\left|\overrightarrow{n_{p_j}} \cdot \overrightarrow{n_{v_{B_i}}}\right|}\right) > \gamma \tag{4}$$

  where j = {1,2}.
  It is necessary to point out here that a triangle can be composed of two search vertices and one candidate vertex. In this situation, two comparisons will be made between the normal of the candidate vertex and the normals of the search vertices.
- If the condition is met for any of the candidate vertices, this triangle is discarded to join the mesh portion. If the condition is not met, the list of normals $\mathcal{N}'$ will be updated by associating a weighted average of its current normal and the normal of the search vertex with each point of the accepted triangle. That is, if p_j is accepted, its new normal is:

$$\overrightarrow{n_{p_j}}' = \frac{\overrightarrow{n_{p_j}} + \alpha \cdot \upsilon_i \cdot \overrightarrow{n_{v_{B_i}}}}{2} \tag{5}$$

  where α is a coefficient that provides the search vertex normal with the greatest weight in the average and υ_i is the degree of the neighborhood at that time, that is, the number of iterations of the process. It is initially 1. After starting from the boundary of the hole, υ_i will, make its normal $\overrightarrow{n_{v_{B_i}}}$ exert a greater influence on the search process close to the boundary of the hole, and it will also ensure that this influence is reduced farther from the boundary. This means that a great variation in the surface curvature in the vicinity of the hole will be not allowed, and the permittivity will increase as we move away from the boundary of the hole. Therefore, considering the equation 5, the process is said to be a diffusion of the boundary normals of the hole.
- After going through the whole $\mathcal{T}_C$ list, a new mesh portion is obtained by integrating new triangles into that list. We then detect the boundary of this new portion, B_{e1}, which will consist wholly or partly of the new search vertices v_{b_k}.
- We continue with the procedure described above with the new search vertices, repeating it as many times as indicated by the degree of neighborhood υ.

This procedure of mesh portion extraction differs from the one which does not consider the normal evolution since, in this case, a set of triangles will be integrated into the mesh portion in each search step, and this set does not always form a ring since there will be areas which do not grow. It is, therefore, also possible to state that the final mesh portion will have fewer triangles (less area) than when the initial procedure was used.

With regard to the coefficients defined for this algorithm, it has been experimentally proven to work correctly for the values $\gamma = 75$ and $\alpha = 3$ for the kind of free form objects used during experimentation. These could be adjusted if the user finds it more convenient to work with other values when using other kinds of objects or holes. Fig. 7 shows various examples of the application of this algorithm during the extraction of mesh portions. Note the difference between the application or non-application of the curvature restriction when extracting the mesh portion. It is worth highlighting the case shown in Fig. 6d in which the hole itself has a very abrupt change of curvature. Although it might be assumed that the algorithm in this situation would limit the expansion of the mesh portion, the results show that, due to radial diffusion of the normal, the growth occurs properly.

A particular situation that may occur is shown in Fig. 7e and 7f, in which a hole appears inside the handle of the pot. As depicted in Fig. 7f, the normal diffusion algorithm limits the growth of the mesh along with the handle itself. However, the mesh portion extends in order to surround the handle. The result is a final portion which has a hole over the area in which growth has been limited, as illustrated in Fig. 7g. This hole will also be represented in the range image. In this situation, this unexpected hole is interpolated, as explained in Section VI-B.

At the end of this stage, a surface portion is available surrounding the hole. This surface provides sufficient information to successfully repair the zone. The next step of our algorithm consists of obtaining the range image of this extracted portion.

VI. GENERATION OF THE RANGE IMAGE

As is known, a range image is a representation method for surfaces on which the height information of the 3D points relative to a point of view or reference plane is coded with a gray level image. The reference plane is chosen in such a way that the area of the zone of interest is maximum (Fig. 8).

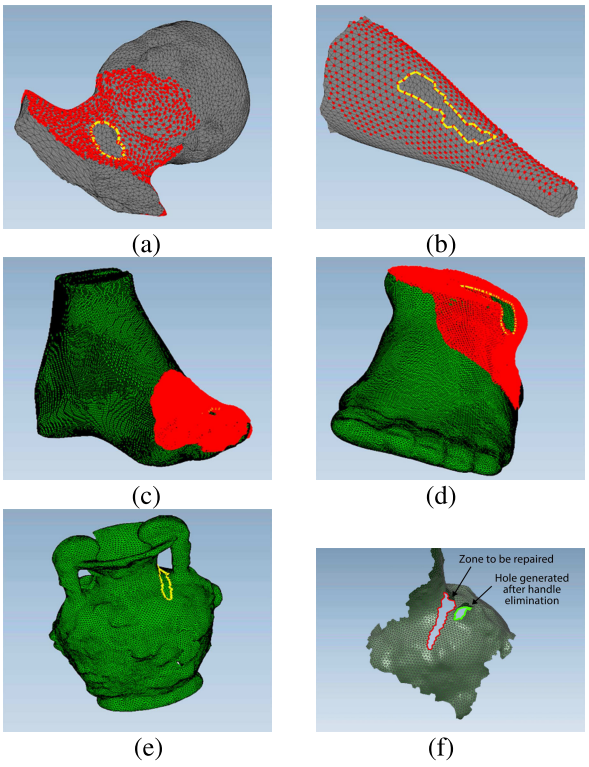

FIGURE 7. (a), (b), (c) y (d) Mesh portions obtained by the normal diffusion algorithm for situations that arise in Fig. 6a, 6c, 6e, 6f and 6h; (e) Object with a hole inside a handle; (f) Mesh portion after applying the algorithm to the hole shown in Fig. 7e, and representation of the new hole, generated after limiting the extension of the mesh portion towards certain areas.

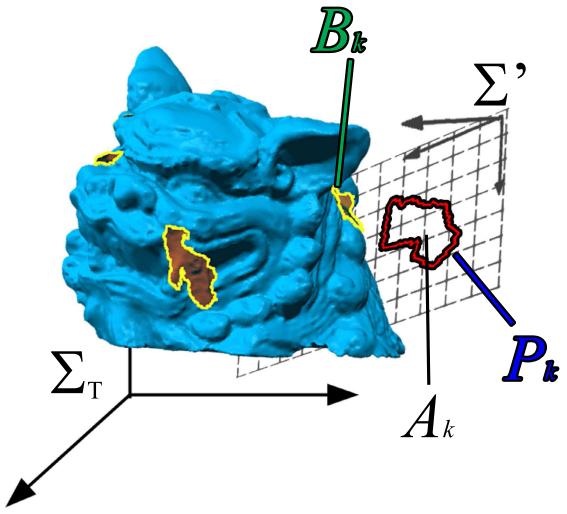

FIGURE 8. Computation of the orientation of S' linked to the reference plane that maximizes the area of P_k, A_k.

The size of the range image defines the rectangular area to be divided into a grid, whose cells correspond to the pixels in image I_{h_i}.

The number of cells in the projection grid sets the resolution and the discretization of the projection of M_{v_i}. Thus, if we choose the width of the grid, denoted as q $(mm/pixels)$, the image resolution is $1/q(pixels/mm)$. The value of q will determine the resolution of the mesh that will be created to fill or restore the areas with holes. For the resolution of the mesh before and after applying the hole-filling algorithm to be similar, we shall define the value of q as:

$$q = \beta \cdot d_m \tag{6}$$

where $0 < \beta \leq 1$ is a proportionality factor and d_m is the mean value of the length of the edges of the mesh before it is repaired. Experimentally it has been determined that, $\beta = 0.87$ provides good results.

A. TRANSLATION AND SCALING OF RANGE IMAGE

Although the range image can be generated for any value of the distance between the partial view and the projection plane, the restoration algorithm used in our proposal is sensitive to that value. In other words, it is sensitive to the range of values stored in the image that it has to process. If the distance is long, the values stored in the range image are high, and more iterations are needed to restore the image. It is, therefore, of interest to move the partial view as close as possible to the reference plane. This can be achieved by performing two consecutive operations on the mesh: a translation and a scaling. These operations are applied only in the Z coordinate of the nodes of the mesh since this is the only coordinate that affects the relative position of the mesh portion with respect to the reference plane. It is thus simpler to apply the operations to the range image, I_h, in such a way that the final range image, I'_h, will be:

$$I'_h = (I_h - d_i) \cdot f_{esc} \tag{7}$$

where d_i signifies a translation in $\min(z_i)$ units or, which is the same, in $\min(I_h)$ units, and, f_{esc} is a scale factor which normalizes the values of I_h to 1, that is:

$$f_{esc} = \frac{1}{\max\left(I_h - \min\left(I_h\right)\right)} \tag{8}$$

These two operations must be taken into consideration after the application of the *image inpainting* algorithm, at the 2D to 3D transformation step described in Section VII.

B. INTERPOLATION IN THE RANGE IMAGE

Most of the meshes are not completely uniform in triangle size, so obtaining a range image with losses is inevitable. To reduce the effect of this lack of information that the range image provides the restoration algorithm with, we apply an interpolation process to the range image, I'_h.

This interpolation of small areas with no data is carried out with an inward diffusion mechanism, using average values of the pixels that are in the vicinity of the hole. The interpolation takes place iteratively, that is, once an area is filled, this new generated image will be used to fill the next zone. The criterion employed to undertake the interpolation of all the areas is the number of pixels they are composed of. We start by filling those areas containing fewest pixels and finish with those with the greatest number of pixels. Fig. 9a shows an example of a range image with some losses or additional

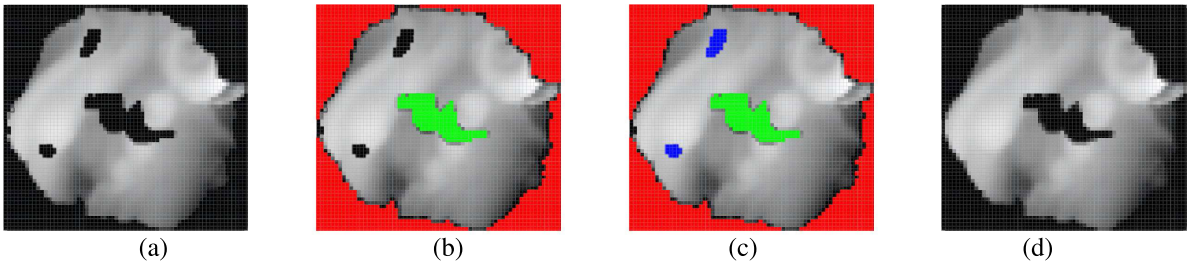

FIGURE 9. Interpolation of small zones in the range image: (a) Range image with some losses or additional small holes; (b) Detection of zone to be restored (green); (c) Detection of holes in the range image that may affect the restoration process (blue); (d) Interpolation of these holes and final range image to which the repair algorithm is applied.

small holes. In Fig. 9b, the external area of the projection of the mesh portion is colored in red and the hole to be filled is colored in green, while in Fig. 9c, the holes to be interpolated are colored in blue. Finally, Fig. 9d shows the range image after the interpolation process.

A more detailed explanation of this process is as follows:

- Empty areas, λ_i, are detected in the range image. One of these areas corresponds to the zone to be repaired, λ_h. There may also be another zone, λ_o, corresponding to the outside of the mesh portion, which does not have to be interpolated, signifying that only the areas to be interpolated are considered: $\lambda_i' = \lambda_i - \{\lambda_h, \lambda_o\}$.
- λ_i' areas are ordered according to their number of pixels, in increasing order.
- We now interpolate the list of zones created in the previous step.
- For a given area λ_k', the pixels to be filled are ordered according to the number of filled neighbors, in decreasing order. If two of them have an equal number of filled neighbors, the order will be established according to the row and column that these pixels occupy in the image. They will thus be ordered in increasing order of rows and columns. The pixels can, therefore, be denominated as follows: $\rho_k\,(r_i, c_i)^{\eta_i}$, where k is the area containing this pixel; r and c are the row and column occupied by this pixel, respectively, and η is the number of filled neighbors. Two consecutive pixels in the list, $\rho_k\,(r_m, c_m)^{\eta_m}$ and $\rho_k\,(r_n, c_n)^{\eta_n}$, which have the same number of filled neighbors, will therefore meet this condition:

$$\begin{aligned} r_m &\leqslant r_n \\ c_m &\leqslant c_n \end{aligned} \tag{9}$$

  with $\eta_m = \eta_n$ and $m < n$.
- We then interpolate the list of pixels created in the previous step, taking into account that the value to be assigned to a pixel is given by the following expression:

$$\rho_k(r_i, ci)^{\eta_n} = \Sigma \frac{\rho_{neighbor}(r_j, cj)}{\eta_n} \tag{10}$$

  where $\rho_{neighbor} \not\subset \lambda_k'$.

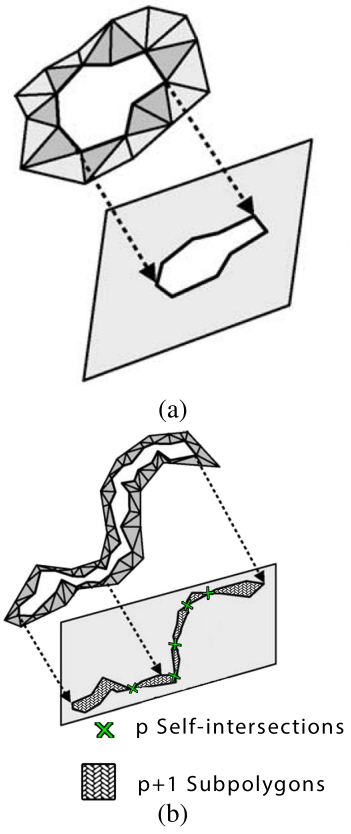

FIGURE 10. Representation of two types of holes, depending on the projection of their boundary on the reference plane: (a) Simple hole: its projection does not have self-intersections; (b) Complex hole: its projection results in a polygon with self-intersections.

C. SIMPLE AND COMPLEX HOLES

Two possible circumstances may occur as a result of the projection of the hole's boundary on the reference plane. The first situation is that in which the boundary of the zone to be repaired becomes a polygon on the plane without any self-intersection. In this case, the hole is said to be simple. The second situation is that in which the projection generates a polygon with self-intersections, whereupon the hole is said to be complex. Both situations are depicted in Fig. 10. The occurrence of a simple or complex hole may condition the continuation of the filling process. More specifically,

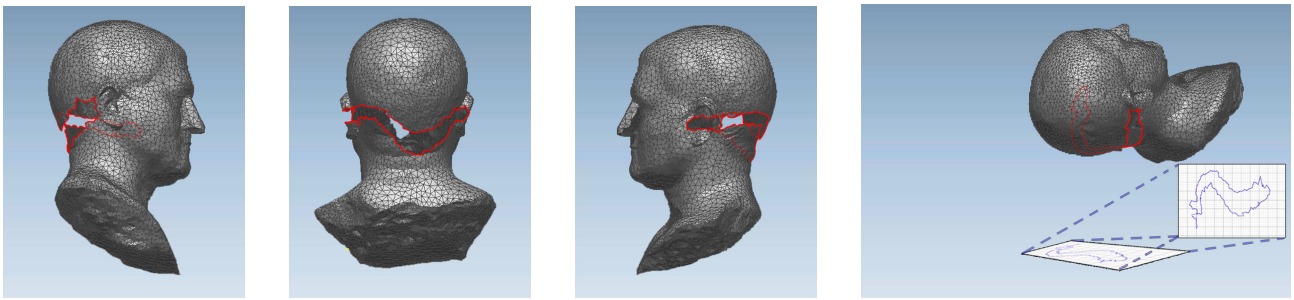

FIGURE 11. (a), (b) y (c) Different views of an elongated hole with a boundary which is complex, (d) Projection of boundary of hole after computing its orientation.

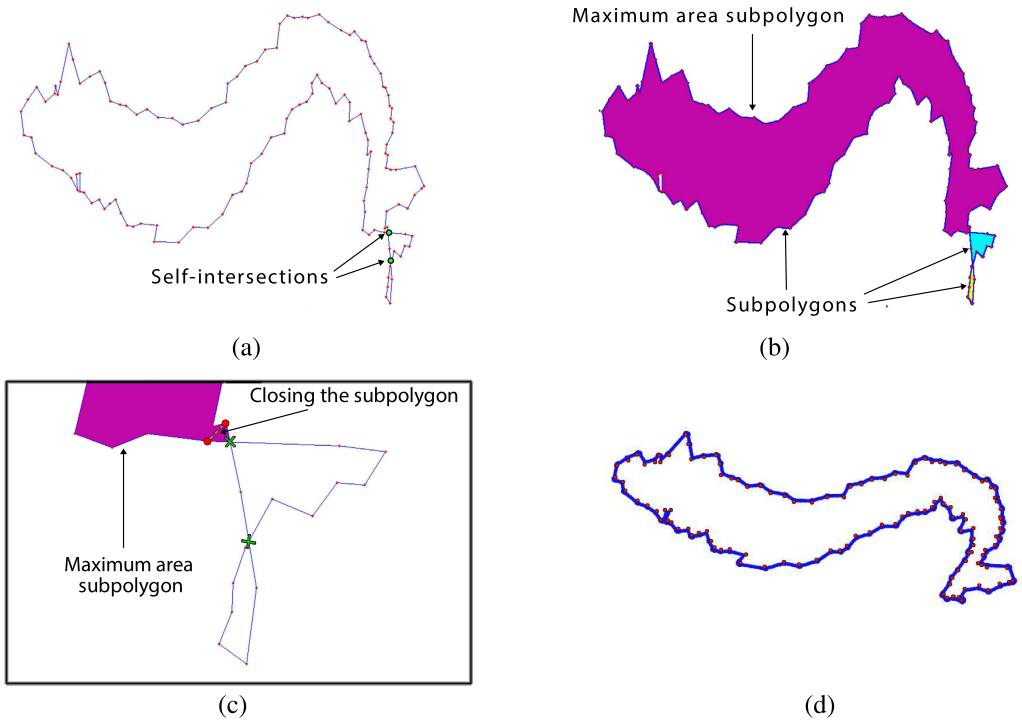

FIGURE 12. (a) Polygon obtained after the projection of the hole shown in Fig. 11. Two self-intersections are computed; (b) Intersection points divide the polygon into three subpolygons. Areas are computed and the maximum is sought; (c) The new boundary will be the subpolygon with maximum area, but is closed by linking the two subpolygon vertices prior to the intersection point; (d) Obtaining the projection of the boundary of the new hole after applying the orientation criteria.

if the hole is complex it will be necessary to modify the general filling process. This modification consists essentially in dividing the complex hole-filling process into multiple branches, as many as the number of self-intersections. The procedure can be schematized in the following steps, which will be illustrated using Figs. 11, 12 and 13 as an example:

- We have the boundary of a hole, B_H, which is projected onto the chosen reference plane, leading to the polygon P_H. Fig. 11 shows an example of a mesh with an elongated hole. This hole is visible from both sides, right and left, and from the rear. This is an indication that the projection onto a plane will be a complex hole.
- The intersections of the edges of the polygon are checked to discover whether P_H has self-intersections. If not, the filling process continues normally. Otherwise, if p intersections are detected, the polygon P_H is divided into $p+1$ subpolygons, P_{If}, whose joints are the p points of intersection. Fig. 12a shows the resulting polygon projection that has two self-intersections.
- The areas of the subpolygons are calculated and the polygon with the largest area, P_{Sk}, is sought. In the case of the hole in our example, one of the subpolygons has a considerably larger area than the other two subpolygons, as shown in Fig. 12b.
- We take all the vertices of P_{Sk} except the intersection points. The polygon is now open in one or two parts; it is closed directly by joining the open ends. Fig. 12c represents the zoomed area in which self-intersections

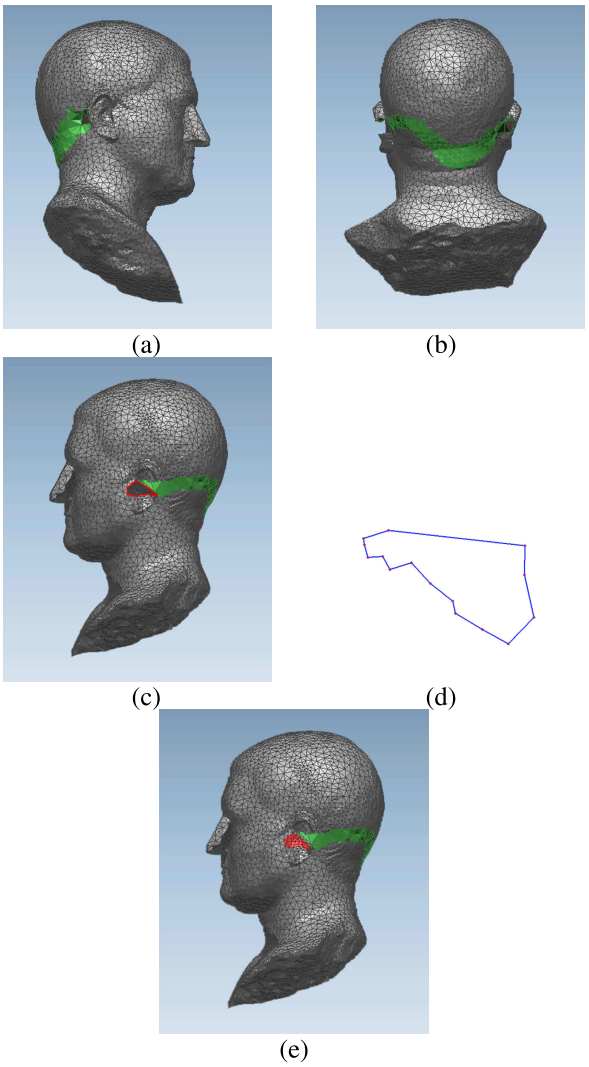

FIGURE 13. (a), (b) and (c) Some views of the partial filling of the hole in Fig. 11, after computing the boundary of the new hole in Fig. 12d; (d) Projection of the boundary of the newly generated hole; (e) Filling of the boundary of the new hole, which completes the filling of the entire initial hole.

appear. Note that the boundary of the new hole is closed using the points of the subpolygon with the greatest area, which are located before the point of intersection.

- We take the vertices of B_H for the polygon that has just been built, and generate a new boundary B_{H1} with which to continue the repairing process. Before proceeding, we must recalculate the orientation plane. Fig. 12d shows the new projection of the new hole, which has no self-intersections.

This operation ensures that at the end of the process we partly fill the hole for B_H (see the result for our example in Fig. 13a, 13b and 13c). We therefore subsequently continue to fill the remaining part of the hole. The orientation of the projection plane must be recalculated for this new hole. After computing this plane, if the hole remains complex, the procedure is the same as in the previous case, that is, the hole is subdivided. Otherwise, we continue with the filling process as we would do for a simple hole. Fig. 13d shows the resulting polygon after the projection of the boundary of the new hole. The final result after filling the entire hole can be observed in Fig. 13e.

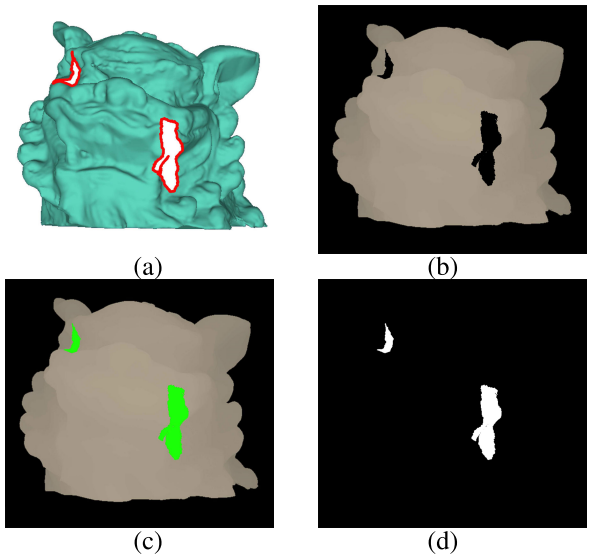

FIGURE 14. Obtaining the selection mask: (a) Partial view with holes to be filled selected; (b) range image; (c) selection of areas to fill in the range image; (d) selection mask.

D. APPLICATION OF RESTORATION ALGORITHMS

At this stage, the image inpainting algorithm is applied to the image I_{h_i} obtained in the step explained in section VI (Fig. 14). Specifically, we used the Roth and Black [13] algorithm. This algorithm exploits the idea of image coding in order to learn the parameters of Markov Random Fields (MRF). The method employed is FoE and models the probability of an image in terms of a random field of overlapping patches whose potential functions are represented as a Product of Experts (PoE) [48].

The objective of the PoE is to model the probability of high-dimensional distribution (image patches) through the product of several expert distributions, in which each expert is working on a low dimensional subspace that is easier to model. The application of a linear filter to an image patch results in a one-dimensional subspace to be modeled. The probability density of an image patch can, therefore, be defined as the product of the distributions produced by the linear filters.

In general, these algorithms do not automatically select the pixels on which the filling process should be executed. The application of the algorithm to the range image, I_{h_i}, is therefore limited by a selection mask, Q_h. This mask is a binary image that has the same size as the range image, (n, m) pixels, and in which the pixels that we intend to fill are active, as shown in Fig. 14d.

Once the image inpainting algorithm, $\Omega : \Re^2 \rightarrow \Re^2$, has been applied, a restored range image is available:

$$I_R = \Omega(I_{h_i}, Q_h) \tag{11}$$

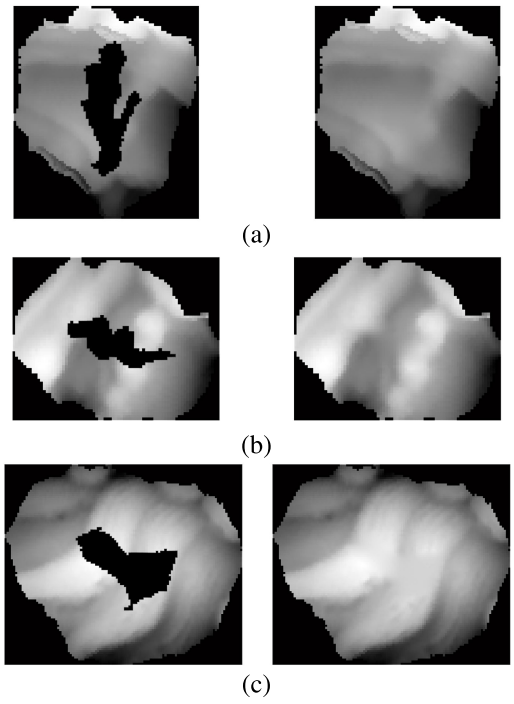

FIGURE 15. Representation of range images taken from the mesh portions $M_(v_i)$ (left) and the range image obtained after application of the restoration algorithm to holes 1, 2 and 3 in (a), (b) y (c), respectively.

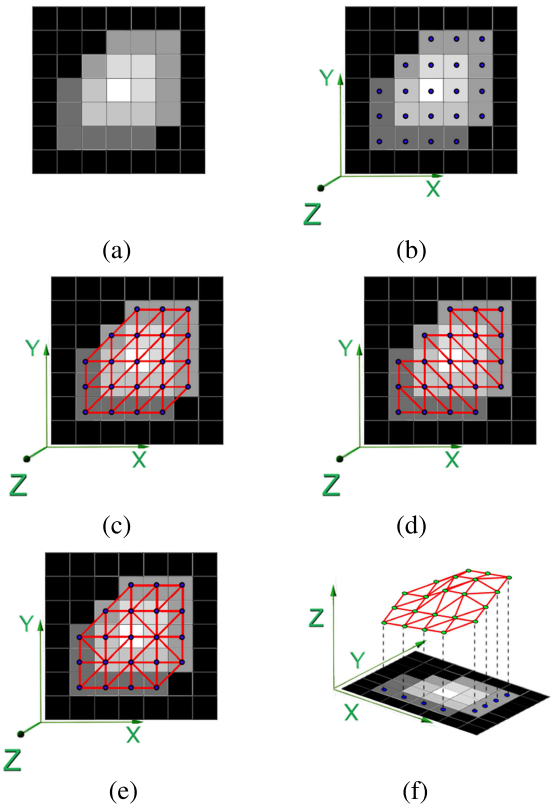

FIGURE 16. Example of the triangulation of filling points in an image range: (a) range image; (b) points corresponding to pixels; (c), (d) and (e) different triangulations between points, taking into account the neighborhood relations; (f) translating the triangulation to the equivalent point cloud.

As a result, the initially empty zones corresponding to the holes in the generated image now store new values, as can be observed in Fig. 15. It is possible to extract them applying the aforementioned mask. Their expression is, therefore:

$$I'_R = Q_h \circ I_R \tag{12}$$

where the operator $\circ$ represents the element-wise multiplication between matrices.

VII. 2D TO 3D TRANSFORMATION

In order to achieve the fully repaired mesh, it is necessary to transform the 2D data obtained from the previous stage into 3D data. It will be necessary only to apply a transformation to new data that have been generated after the execution of the restoration algorithm, i.e. to I'_R.

The 2D to 3D transformation results in the 3D coordinates corresponding to the filled pixels. Concerning the process of creating the range image, that the value stored in each pixel corresponds to the distance from the 3D point and whose orthogonal projection onto the projection plane falls on that cell or pixel. Consequently, the value that each pixel stores determines the distance from the 3D point to the image plane. With respect to the reference system Σ', this corresponds to the coordinate z'. Coordinates x' and y' will be defined by the position of that pixel on the image plane. Thus, if the range image represents values between (x_{min}, y_{min}) and (x_{max}, y_{max}), and a resolution of q pixels per millimeter is chosen, given a pixel (f, k) that stores the value r, the 3D coordinate corresponding to that pixel will be:

$$(x', y', z') = (x_{min} + k \cdot q,\ y_{max} - f \cdot q,\quad \frac{r}{f_{esc}} + d_i) \tag{13}$$

where f_{esc} is the scale factor applied in equation (7). Note that in the above equation it has been considered that the coordinates of the pixels are defined by the row and column f and k, occupied in the image matrix. It is for this reason that the coordinate x depends on the second coordinate of the image plane, k, and coordinate y depends on the first, f.

After calculating all the coordinates, we obtain a 3D point cloud, L'_i, corresponding to the hole that has been filled. This point cloud is expressed in the coordinate system Σ', attached to the reference plane, and an inverse transformation T^{-1} must, therefore, be applied in order to obtain its representation in the initial reference system Σ_i, L_i.

The filled surfaces must now be defined by triangulating this point cloud. Points forming part of hole boundaries, B_h, will be added to these nodes in order to calculate the triangulation. Although these points belong to the initial mesh, M_T, they will serve as a joint or sewing line between the initial mesh with holes, M_T, and the filling mesh, H_c.

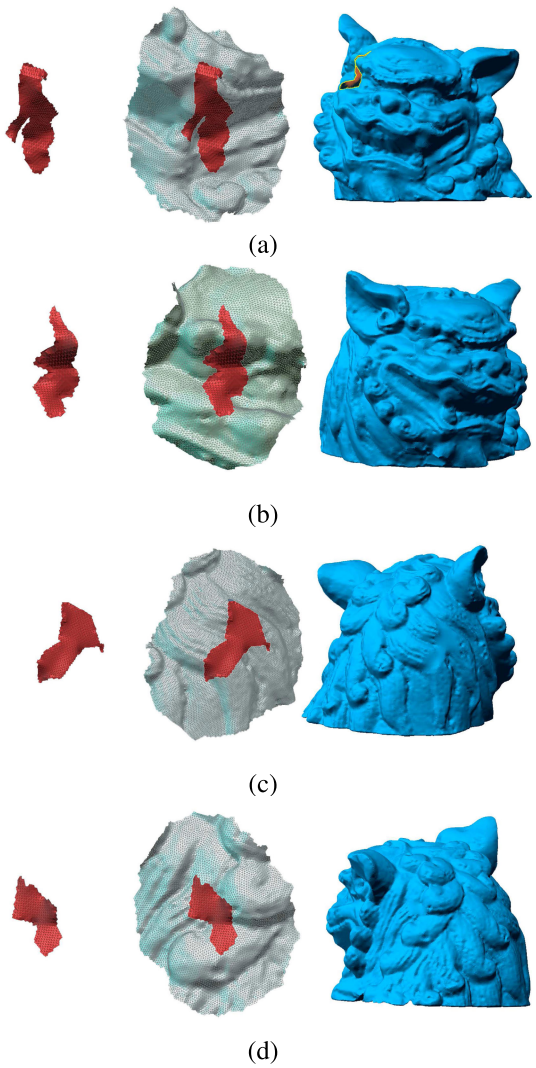

FIGURE 17. Integration of portion meshes that were filled by applying the restoration algorithm to the whole mesh M_T depicted in Fig. 2. Zones 1, 2, 3 and 4 in (*a*), (*b*), (c) and (*d*), respectively.

VIII. TRIANGULATION

The triangulation process is divided in two: T_1, which is the triangulation of the point cloud L_i, and T_2 which is the triangulation between points on the mesh boundary H_i, denominated as B_{1_i}, and those of the boundary of hole h_i, denominated as B_{H_i} (which belong to M_T).

It is possible to generate triangulation T_1 easily by taking into account the origin of each L_i point. Since these points were obtained after a 2D to 3D transformation from the restored pixels in the image, this triangulation can be built by establishing a connection based on the neighborhood relations in the image. This signifies that, if we start from a restored pixel $p_{(f,c)}$, located in row r and column c, its eight neighbors are:

$$\begin{aligned}\aleph_8(p_{(f,c)}) = \{&p_{(f-1,c-1)},\ p_{(f-1,c)},\\ &p_{(f-1,c+1)},\ p_{(f,c-1)},\ p_{(f,c+1)},\\ &p_{(f+1,c-1)},\ p_{(f+1,c)},\ p_{(f+1,c+1)}\}\end{aligned} \quad (14)$$

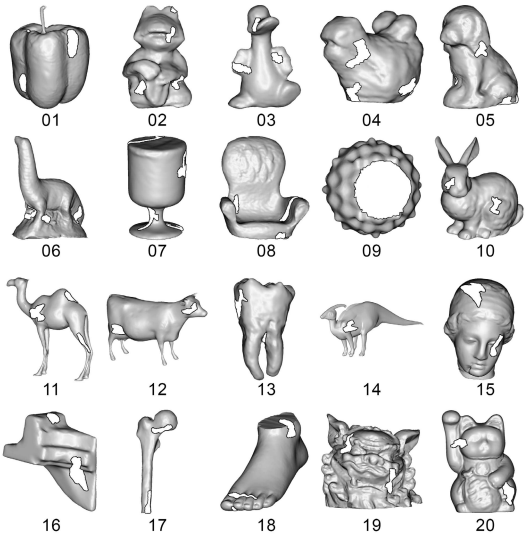

FIGURE 18. Set of twenty meshes with artificially generated holes that were used to test our algorithm.

If we denote $v(p_{(f,c)})$ as the point equivalent to that pixel $p_{(f,c)}$, once the transformation has been carried out, the triangulation of the point group corresponding to the neighbors of $p_{(f,c)}$ along with $v(p_{(f,c)})$, is generated as follows:

$$\begin{aligned}&T(\aleph_8(p_{(f,c)}) \cup p_{(f,c)})\\ &= \begin{Bmatrix} v(p_{(f-1,c-1)}) & v(p_{(f,c-1)}) & v(p_{(f-1,c)})\\ v(p_{(f-1,c)}) & v(p_{(f,c-1)}) & v(p_{(f,c)})\\ v(p_{(f,-1c)}) & v(p_{(f,c)}) & v(p_{(f-1,c+1)})\\ v(p_{(f-1,c+1)}) & v(p_{(f,c)}) & v(p_{(f,c+1)})\\ v(p_{(f,c-1)}) & v(p_{(f+1,c-1)}) & v(p_{(f,c)})\\ v(p_{(f,c)}) & v(p_{(f+1,c-1)}) & v(p_{(f+1,c)})\\ v(p_{(f,c)}) & v(p_{(f+1,c)}) & v(p_{(f,c+1)})\\ v(p_{(f,c+1)}) & v(p_{(f+1,c)}) & v(p_{(f+1,c+1)}) \end{Bmatrix}\end{aligned} \quad (15)$$

Taking into account the neighborhood relations in the image, the triangulation is built by creating edges between the points whose pixels have a horizontal or vertical neighborhood. If these connections are projected onto the image plane, we attain a regular grid. It is then necessary to create unions corresponding to the diagonals of the grid. Only one of the two possible diagonals must be chosen to split the square into two triangles. Different triangulations will be obtained depending on the diagonals chosen. Therefore, it is advisable to establish a criterion to choose the same direction for all diagonals.

Fig. 16 shows an example of T_1 triangulation. Figs. 16c, 16d and 16e depict the triangulation of various possibilities depending on the orientation of the diagonals. Finally, as shown in Fig. 16f, the triangulation established for the image points is translated to the point cloud L_i.

With regard to triangulation T_2, it is computed for the point cloud $B_{1_i} \cup B_{H_i}$. In this situation, there is no prior knowledge of the relations between the points as in T_1. The method employed to perform this triangulation is that proposed in [49]. This method, which is based on the Delaunay triangulation ([50]) and has been extended to 2.5 D, starts from an

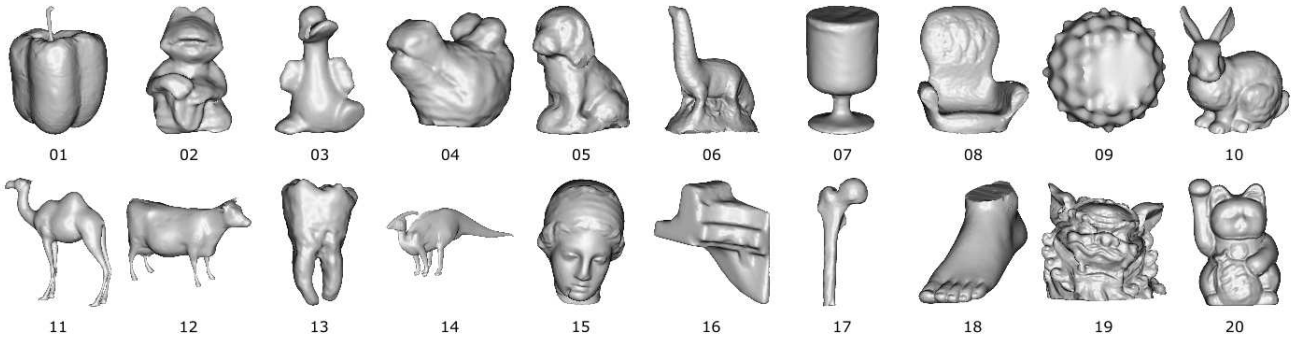

FIGURE 19. Results of the application of the hole-filling algorithm.

initial triangle that acts as the seed of the process from which the triangulation is extended to the whole point cloud.

IX. INTEGRATION OF MESH PORTIONS

For the integration of the filled mesh, the tuple that characterizes the mesh portion corresponding to the hole h_i, $\{\mathcal{V}_i,\mathcal{A}_i,\mathcal{T}_i\}$, will be attached to the tuple of the whole mesh M_T, which results in the tuple in 16, that represents M'_T.

$$\left\{\begin{bmatrix}\mathcal{V}_T\\ \mathcal{V}_i\end{bmatrix}, \quad \begin{bmatrix}\mathcal{A}_T\\ \mathcal{A}_i\end{bmatrix}, \quad \begin{bmatrix}\mathcal{T}_T\\ \mathcal{T}_i\end{bmatrix}\right\} \tag{16}$$

The new global mesh M'_T will be used in the next iteration to fill the next hole h_{i+1}. It is particularly important to ensure that the parts belonging to the original mesh and those which have been artificially generated during the restoration process are always identified. To do this, we have defined the 'restoration Matrix', $M\rho$, which has three columns and the same number of rows as processed holes. Cardinal of vertices, edges and triangles sets that are being added at each iteration of the process are stored in the three columns:

$$M\rho = \begin{bmatrix} card\,(\mathcal{V}_T) & card\,(\mathcal{A}_T) & card\,(\mathcal{T}_T)\\ card\,(\mathcal{V}_1) & card\,(\mathcal{A}_1) & card\,(\mathcal{T}_1)\\ card\,(\mathcal{V}_2) & card\,(\mathcal{A}_2) & card\,(\mathcal{T}_2)\\ \vdots & \vdots & \vdots\\ card\,(\mathcal{V}_m) & card\,(\mathcal{A}_m) & card\,(\mathcal{T}_m)\end{bmatrix} \tag{17}$$

Fig. 17 shows an example of the triangulation and integration of four zones (a), (b), (c) and (d), of the mesh shown in Fig. 2. The filling process is executed iteratively with the aim of generating the filling meshes for all the holes h_i that were initially detected. Upon completion, and to ensure that a closed mesh is obtained, the hole-detection process is executed again, since it is possible that, due to the various sub-processes of the method, some holes may be generated.

Once the mesh has been filled, a global remeshing can be applied to ensure a uniform distribution of the resulting mesh.

X. EXPERIMENTAL RESULTS

To experimentally validate the hole-filling method proposed in this paper, several tests were performed. The algorithms were programmed in Matlab, running on an Intel i7-3770 3.4 GHz PC with 8 GB of RAM.

Twenty manifold and closed 3D meshes were used (Fig. 18). Some of them belong to the AIM @ SHAPE repository [51] and the rest were obtained by our research group with a Minolta VIVID laser scanner. The selected set of meshes is composed by a great variety of free form objects whose surfaces do not have deep crevices or handles that could give rise to hidden holes not seen from the outside. In each of the 3D models, some holes were artificially generated (from 2 in mesh 8 to 8 in mesh 5). The total number of holes is 83. These holes are of different shapes and sizes and they are on surfaces with a varied topology. The holes were placed in areas with different degrees of curvature and, in some cases, in areas with linear features. Fig. 18 also shows the holes that were created. Next, we will show the qualitative results of applying the hole-filling algorithm to the meshes indicated above in sub-section X-A. In sub-section X-B, a quantitative comparison between the results obtained with the algorithm proposed in this work and the results obtained with other algorithms will be made.

A. RESULTS OF THE HOLE-FILLING METHOD

The results of applying our hole-filling algorithm to all the meshes of Fig. 18 are showed in Fig. 19. Some interesting results are discussed in detail below. Fig. 20a shows mesh 13, which presents a hole located in an inner zone or slit. This is a type of region in which holes usually appear in the digitization of meshes. Due to the fact that the proposed method is based on the projection of a part of the 3D mesh on a plane, these holes, in areas of high curvature surface, would theoretically be difficult to fill. However, Fig. 20b shows that the algorithm fills this type of hole without problems.

Another interesting case is the one presented in mesh 11 (Fig. 20i). It might seem that it is necessary for there to be a large area around the hole for the filling to be correct. However, as can be seen in Fig. 20j, the algorithm can reconstruct the surface satisfactorily.

In general, zones with geometric or curvature discontinuities are zones where hole-filling algorithms generate worse results. Figs. 20c and 20d show mesh 15 before and after

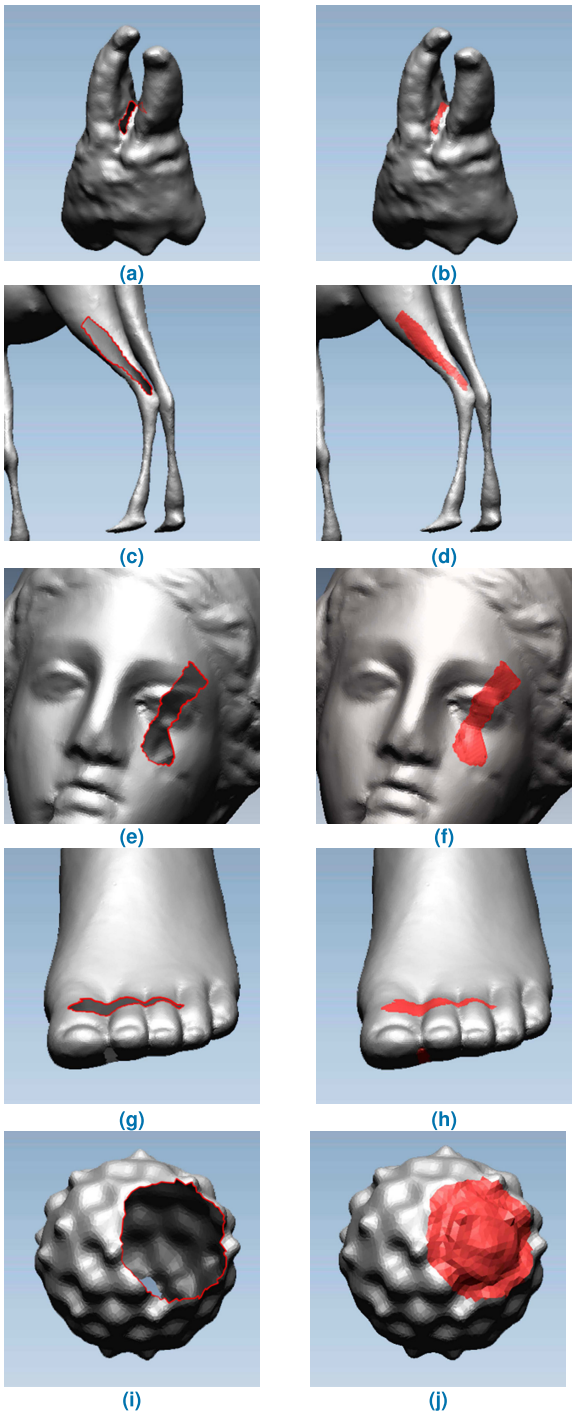

FIGURE 20. Details of the results after the application of the hole-filling algorithm to several meshes.

applying our proposal. As is clear, the result of our algorithm is also quite good in this type of zone.

Another outstanding feature of our hole-filling procedure is the ability to follow linear structures (for example, an eyebrow on a face or the edge of a polyhedron), as occurs in the upper part of mesh 15, previously analyzed (Figs. 20c and 20d). A similar example occurs in mesh 18, shown in Figs. 20k and 20l.

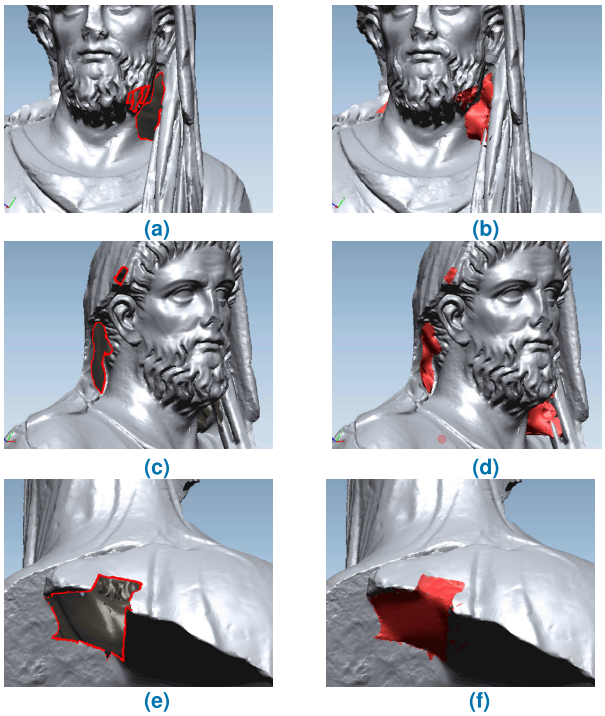

FIGURE 21. Restoration of the 3D models of the Aeneas sculptures from the National Museum of Roman Art (Mérida, Spain) after the appearance of various holes in the scanning process.

Conversely, for cases of repeated non-linear structures in a mesh, this method does not provide satisfactory results. This is a common situation for non-exemplar based hole-filling algorithms (e.g. [43]), such as our proposal. An example can be seen in Figs. 20g and 20h.

In addition, our method was applied to various meshes obtained after the digitization of sculptures at the National Museum of Roman Art in Mérida (Spain), where holes appeared owing to some limitations or errors in the acquisition process. Fig. 21 shows the results obtained for one of the sculptures.

B. COMPARISON WITH OTHER APPROACHES

In order to validate the proposed hole-filling method, we compared our results with those provided by three widely used algorithms: Liepa's method [36], Ju's method [41] and Wang's method [38]. All these methods are currently being used in different 3D mesh processing software. Specifically, Liepa's method is used in MeshLab [52] and ReMESH [53] (we have used the former, in much more common use than the latter); Ju's method is used in PolyMender [54]; and finally, Wang's method is used in MeshWorks [55].

A comparison is made to analyze two aspects: error between ground truth and the portion of mesh generated by each method, and the resolution of the 3D models before and after applying the filling algorithms.

Error between the solution proposed by each method and ground truth is measured by calculating the Hausdorff distance [56] between them. As ground truth we consider the

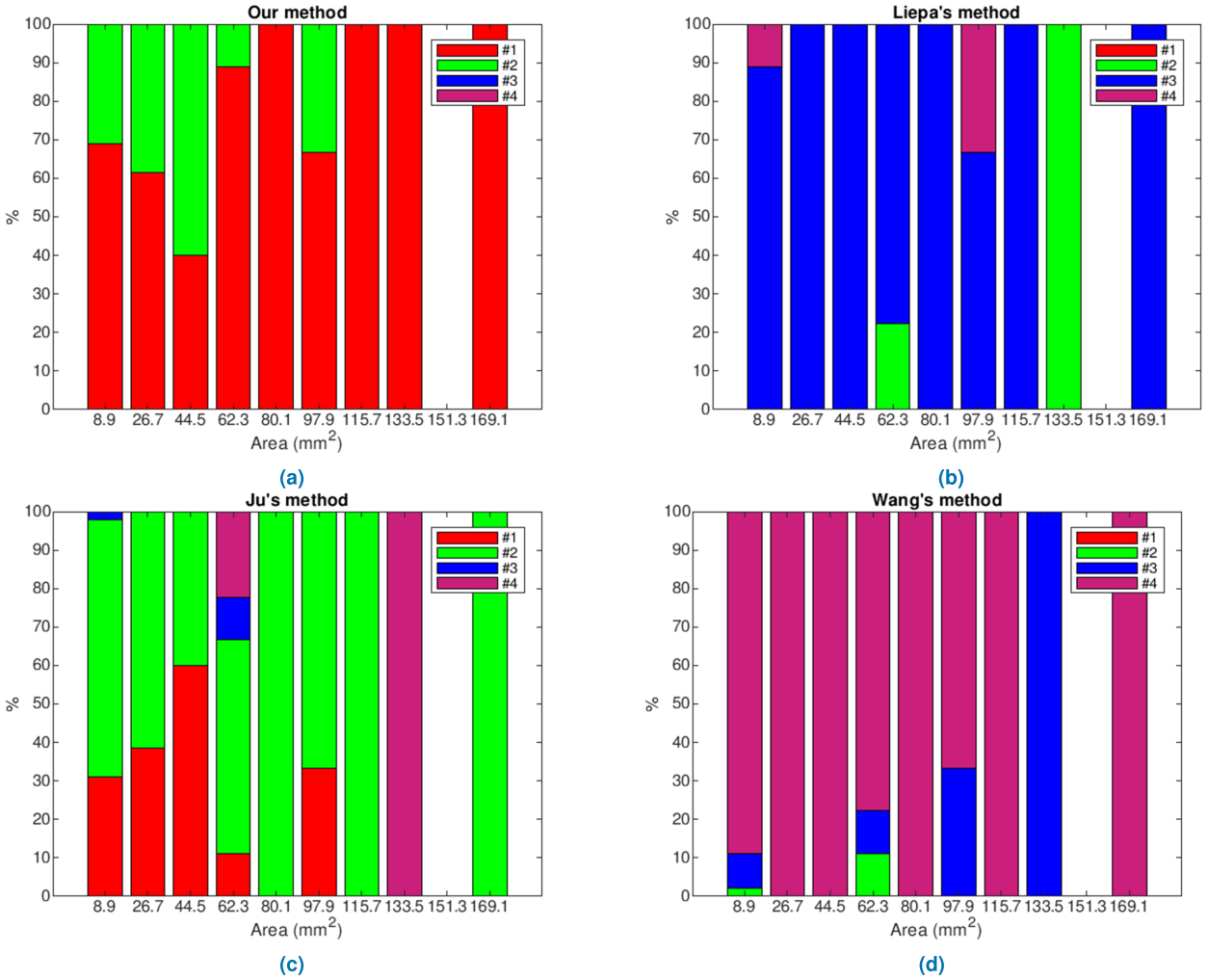

FIGURE 22. Effectiveness ranking by classes for (a) our method, (b) Liepa's method, (c) Ju's method and (d) Wang's method.

portions of the mesh removed from the 3D models to create the artificial holes. Tables 1 and 2 show all this information.

We also determined the effectiveness of the four algorithms when filling the 83 holes under test. This effectiveness is obtained by ordering the errors computed for the four methods in each hole. This results in a vector of 83 elements associated to each method, whose $i-th$ element $\in 1, 2, 3, 4$ represents the position obtained by the method when filling the $i-th$ hole. The first position means the lowest error and the fourth position the highest one. Effectiveness is then ranked by calculating the percentage of times that each method achieves each position. Table 3 shows the effectiveness ranking obtained in terms of Hausdorff distance. As can be seen, our method is in first place in 71.08 % of the cases and in second place in the remaining cases.

The next analysis performed is related to the sizes of the holes. This is important because a filling method must be able to fill holes of any size to be considered robust. As a first step, the Pearson correlation coefficient (PCC), $\rho_{d_H,a}$, between the vector of Hausdorff distances, d_H, and the areas of the holes, a, was determined. The closer to 1 $\rho_{d_H,a}$, the greater the positive linear relationship between the analyzed variables. If the value is close to 0, there will be no relation between them. As can be seen in Table 4, the method with the lowest $\rho_{d_H,a}$ value is ours. This means that, although there is a positive relationship between the variables (the greater the area of the hole, the greater the Hausdorff distance), our method is the most robust of the four analyzed, since its performance is the least affected by the sizes of the holes.

To confirm the robustness of the proposed method, a more in-depth analysis of the influence of holes size on the final result was carried out. We defined 10 evenly spaced classes from the smallest to the largest hole area. The effectiveness ranking of each class was then determined. The results are shown in Table 5 and in Fig. 22.

The effectiveness of each class is represented (in %) with a color code against the area associated with each class (in mm^2). From analysis of the data, we can conclude that ours is the algorithm with the best behavior for all the classes. In addition, for the largest holes (from 115.7 mm^2),

TABLE 1. Hausdorff distance for all the holes of the 3D meshes in Figs. 18 and 19. The lower the value, the better the result.

		Hausdorff distance (mm)			
Mesh number	Hole number	Ours	Liepa's [36]	Ju's [41]	Wang's [38]
1	1	1.412	5.059	2.078	5.059
	2	0.753	5.872	2.501	5.872
	3	1.872	3.039	2.333	3.039
	4	0.813	5.416	0.819	5.416
2	1	0.881	3.666	0.959	3.666
	2	0.763	1.701	1.514	1.701
	3	1.025	1.940	1.508	1.940
	4	1.361	2.516	1.856	2.516
	5	1.135	3.601	1.682	3.601
	6	0.695	2.624	0.577	2.624
3	1	2.225	2.470	1.116	2.470
	2	0.511	2.173	0.247	2.173
	3	0.584	3.804	1.436	3.804
	4	2.891	2.891	0.838	2.891
	5	0.491	1.941	0.466	1.941
	6	0.543	3.027	0.327	3.027
4	1	0.935	2.574	1.137	2.574
	2	1.798	3.097	2.034	3.097
	3	0.773	2.757	1.016	2.757
	4	0.640	3.325	0.820	3.325
	5	0.484	1.757	1.239	1.757
	6	0.674	2.625	1.104	2.625
5	1	1.999	4.720	2.462	4.720
	2	5.640	5.640	5.636	5.640
	3	0.934	1.911	1.304	1.911
	4	2.463	4.211	2.488	4.211
	5	0.945	2.385	1.421	2.385
	6	0.440	2.384	1.028	2.384
	7	0.550	3.426	0.447	3.426
	8	0.425	2.598	0.271	2.598
6	1	2.188	2.734	2.885	2.363
	2	5.651	5.651	5.895	5.651
	3	1.796	3.407	1.769	3.407
	4	3.515	3.515	3.521	3.659
7	1	0.660	0.660	0.629	0.660
	2	3.021	3.021	3.336	3.136
	3	0.435	3.654	0.550	3.654
	4	2.238	2.769	1.139	2.769
	5	1.917	3.115	2.539	3.115
	6	0.574	0.574	0.544	0.574
	7	0.601	1.684	0.329	1.684
8	1	1.898	3.764	2.280	3.764
	2	2.572	2.572	1.591	2.572
	3	1.361	3.964	2.583	3.964
	4	1.091	3.073	1.514	3.073
	5	0.803	3.816	0.509	3.816
9	1	0.566	1.848	0.907	1.848
	2	2.810	6.234	3.124	4.599

our approach always provides the best results. All of this corroborates the fact that the robustness of our proposal surpasses that of the rest of the methods analyzed. The method

TABLE 2. Hausdorff distance for all the holes of the 3D meshes in Figs. 18 and 19 (Cont.). The lower the value, the better the result.

		Hausdorff distance (mm)			
Mesh number	Hole number	Ours	Liepa's [36]	Ju's [41]	Wang's [38]
10	1	0.002	0.005	0.004	0.005
	2	0.001	0.005	0.002	0.005
	3	0.001	0.005	0.001	0.005
11	1	0.021	0.068	0.013	0.068
	2	0.012	0.064	0.023	0.064
	3	0.011	0.028	0.005	0.028
12	1	0.093	0.469	0.202	0.469
	2	0.034	0.207	0.060	0.207
	3	0.082	0.474	0.076	0.474
13	1	0.229	0.358	0.195	0.358
	2	0.137	0.951	0.159	0.951
	3	0.234	0.961	0.390	0.961
14	1	0.018	0.027	0.009	0.027
	2	0.005	0.034	0.008	0.034
	3	0.003	0.011	0.003	0.011
15	1	0.006	0.031	0.014	0.031
	2	0.018	0.069	0.021	0.069
	3	0.009	0.066	0.012	0.066
16	1	0.191	0.371	0.253	0.359
	2	0.034	0.304	0.077	0.304
	3	0.029	0.158	0.024	0.158
17	1	0.116	0.423	0.158	0.423
	2	0.247	0.366	0.247	0.366
	3	0.205	0.373	0.351	0.373
18	1	0.011	0.068	0.021	0.068
	2	0.009	0.028	0.013	0.028
	3	0.036	0.074	0.066	0.074
	4	0.051	0.135	0.110	0.093
19	1	0.227	0.650	0.316	0.650
	2	0.267	1.112	0.297	0.884
	3	0.230	0.697	0.426	0.689
	4	0.442	0.594	0.539	0.589
20	1	0.249	0.949	0.230	0.949
	2	0.352	1.206	0.392	1.206
	3	0.215	0.751	0.448	0.751

TABLE 3. Effectiveness ranking percentages. The higher the value, the better the result.

Position	$1st$	$2nd$	$3rd$	$4th$
Ours	71.08 %	28.92 %	0 %	0 %
Liepa's [36]	0 %	3.61 %	89.16 %	7.23 %
Ju's [41]	28.92 %	65.06 %	2.41 %	3.61 %
Wang's [38]	0 %	2.41 %	8.43 %	89.16 %

TABLE 4. Pearson correlation coefficients (PCC). The lower the value, the better the result.

Method	Ours	Liepa's [36]	Ju's [41]	Wang's [38]
PCC	0.52	0.86	0.62	0.85

that follows ours in the ranking is Ju's, which is in second position in most of the classes. This is followed by Liepa's method. At the end of the ranking is Wang's method.

As mentioned in Section I, robustness in a hole-filling algorithm means that it must be valid for any configuration of

TABLE 5. Effectiveness ranking by classes. The higher the value, the better the result.

		1st	2nd	3rd	4th
Class 1	Ours	68.89 %	31.11 %	0.00 %	0.00 %
	Liepa's [36]	0.00 %	0.00 %	88.89 %	11.11 %
	Ju's [41]	31.11 %	66.67 %	2.22 %	0.00 %
	Wang's [38]	0.00 %	2.22 %	8.89 %	88.89 %
Class 2	Ours	61.54 %	38.46 %	0.00 %	0.00 %
	Liepa's [36]	0.00 %	0.00 %	100.00 %	0.00 %
	Ju's [41]	38.46 %	61.54 %	0.00 %	0.00 %
	Wang's [38]	0.00 %	0.00 %	0.00 %	100.00 %
Class 3	Ours	40.00 %	60.00 %	0.00 %	0.00 %
	Liepa's [36]	0.00 %	0.00 %	100.00 %	0.00 %
	Ju's [41]	60.00 %	40.00 %	0.00 %	0.00 %
	Wang's [38]	0.00 %	0.00 %	0.00 %	100.00 %
Class 4	Ours	88.89 %	11.11 %	0.00 %	0.00 %
	Liepa's [36]	0.00 %	22.22 %	77.78 %	0.00 %
	Ju's [41]	11.11 %	55.56 %	11.11 %	22.22 %
	Wang's [38]	0.00 %	11.11 %	11.11 %	77.78 %
Class 5	Ours	100.00 %	0.00 %	0.00 %	0.00 %
	Liepa's [36]	0.00 %	0.00 %	100.00 %	0.00 %
	Ju's [41]	0.00 %	100.00 %	0.00 %	0.00 %
	Wang's [38]	0.00 %	0.00 %	0.00 %	100.00 %
Class 6	Ours	66.67 %	33.33 %	0.00 %	0.00 %
	Liepa's [36]	0.00 %	0.00 %	66.67 %	33.33 %
	Ju's [41]	33.33 %	66.67 %	0.00 %	0.00 %
	Wang's [38]	0.00 %	0.00 %	33.33 %	66.67 %
Class 7	Ours	100.00 %	0.00 %	0.00 %	0.00 %
	Liepa's [36]	0.00 %	0.00 %	100.00 %	0.00 %
	Ju's [41]	0.00 %	100.00 %	0.00 %	0.00 %
	Wang's [38]	0.00 %	0.00 %	0.00 %	100.00 %
Class 8	Ours	100.00 %	0.00 %	0.00 %	0.00 %
	Liepa's [36]	0.00 %	100.00 %	0.00 %	0.00 %
	Ju's [41]	0.00 %	0.00 %	0.00 %	100.00 %
	Wang's [38]	0.00 %	0.00 %	100.00 %	0.00 %
Class 9	Ours	0.00 %	0.00 %	0.00 %	0.00 %
	Liepa's [36]	0.00 %	0.00 %	0.00 %	0.00 %
	Ju's [41]	0.00 %	0.00 %	0.00 %	0.00 %
	Wang's [38]	0.00 %	0.00 %	0.00 %	0.00 %
Class 10	Ours	100.00 %	0.00 %	0.00 %	0.00 %
	Liepa's [36]	0.00 %	0.00 %	100.00 %	0.00 %
	Ju's [41]	0.00 %	100.00 %	0.00 %	0.00 %
	Wang's [38]	0.00 %	0.00 %	0.00 %	100.00 %

TABLE 6. Number and percentages of holes belonging to each of the established categories to compare the performance of all methods after the curvature analysis.

	Number of holes	% holes
Low curvature areas	28	34%
Medium curvature areas	36	43%
High curvature areas	19	23%

the problem, that is, it should fill holes of different sizes and different types of 3D objects. To validate the robustness of the algorithms when filling any type of 3D free form objects, we propose to analyze the curvature of the surfaces where the holes are located, specifically the maximum curvature. After this analysis, we provide two different classifications of holes. On the one hand, we establish a division of three ranges in the variations of curvature. Thus, we distinguish between holes located in areas with low (e.g. some holes in meshes 1, 13,...- Fig. 23a), medium (e.g. some holes in meshes 2, 8, 11, 17,...- Fig. 23b) and high maximum curvature (e.g. some holes in meshes 4, 7, 8, 18,...- Fig. 23c). Table 6 shows the number and percentages of holes belonging to each of those categories. On the other hand, with this analysis some linear

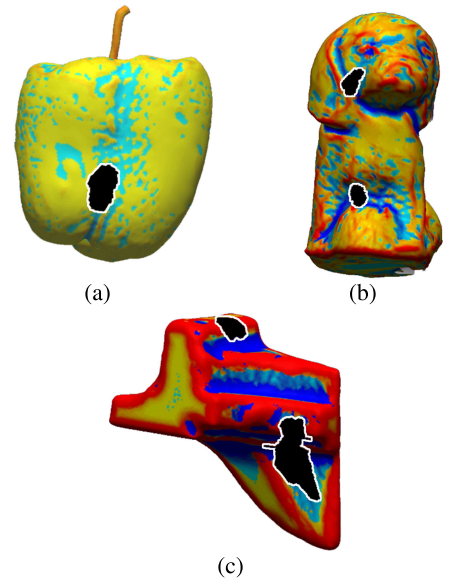

FIGURE 23. Example of representation of the maximum curvature analysis for meshes with holes belonging to the three established ranges: mesh 1 (hole in low curvature area), mesh 5 (holes in medium curvature area) and mesh 16 (holes in high curvature areas).

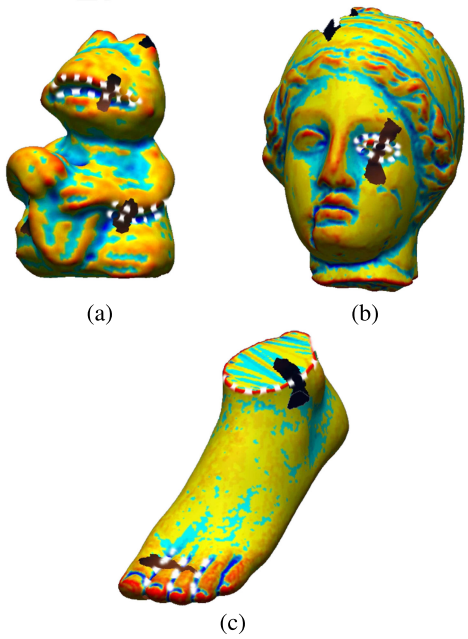

FIGURE 24. Example of representation of the maximum curvature analysis where some paths can be traced by following similar colors (values) on the curvature map. These paths correspond to linear features of the mesh and are marked with a white dashed line: (a) mesh 2; (b) mesh 15 and; (c) mesh 18.

features of the meshes can be easily detected by grouping areas of extreme values of curvature, such as, for example, the mouth in mesh 4, the limits between basis or caps in mesh 7, morphological characteristics of faces in meshes 15 and 19, abrupt changes of surface in mesh 16. Therefore, the second classification distinguishes between holes located in areas with and without linear features. Specifically, this is the case

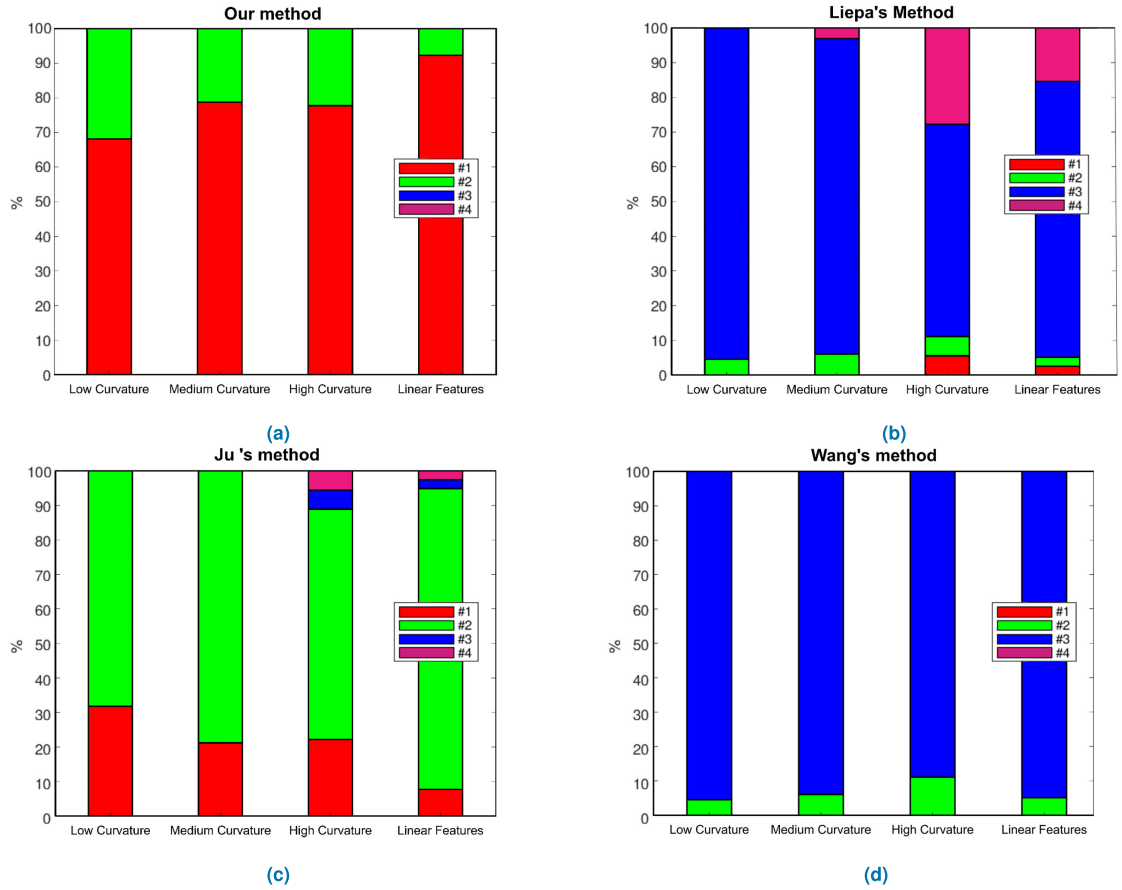

FIGURE 25. Classification of performances of all methods when filling holes located in high, medium and low curvature areas and when filling holes located in areas with linear features: (a) our method, (b) Liepa's method, (c) Ju's method and (d) Wang's method.

in 47% of the set of 83 holes in our study. Some meshes with some linear features are represented in Fig. 24.

Following the analysis of the results, the performance of our method stands out among the others. Concerning the curvature ranges, our method achieved first position when filling 68%, 79% and 78% of holes with low, medium and high curvatures, respectively. The second position percentages for our method are 32%, 21% and 22%. As for the performance of our method with holes in areas with linear features, it is remarkable that it achieves first position when filling 92% of this type of holes. Fig. 25 summarizes the performance of the four methods in both types of classifications described above.

The second aspect to be analyzed, as stated at the beginning of the section, concerns the resolution of the model. Of the methods analyzed in this section, Liepa's, Wang's and ours are independent of the modeling process (see section III), that is, the mesh generated to repair the hole is integrated into the original 3D model. In the case of Ju's method, the mesh is completely rebuilt from a volumetric grid on the original data, so the resolution of the final mesh will be constant. For the other three methods, this does not happen. We analyzed this aspect and the results are in Table 7. In this table, column 1 is for the mesh number; column 2 shows the mean edge value for the original mesh, $\overline{d_o}$; columns 3, 5 and 7 have the mean edges for the portions of meshes generated by each of the three methods, $\overline{d_h}$; and columns 4, 6 and 8 show the ratio between these two mean edge values, $r = \overline{d_h}/\overline{d_o}$. Thus, r greater than 1 means that the resolution of the mesh generated to fill the hole is lower than the resolution of the original mesh. The closer to 1 this ration is, the better the mesh obtained.

TABLE 7. Study of the resolution of the original models and the meshes obtained by the different algorithms to fill the holes.

		Our Method		Liepa's [36]		Wang's [38]	
Mesh Number	$\overline{d_o}$ (mm)	$\overline{d_h}$ (mm)	r	$\overline{d_h}(mm)$	$\overline{d_o}/\overline{d_h}$	$\overline{d_h}(mm)$	$\overline{d_o}/\overline{d_h}$
1	1.48	1.06	0.72	4.45	3.01	4.24	2.88
2	1.54	1.13	0.73	3.67	2.38	3.22	2.09
3	1.18	1.06	0.89	3.75	3.17	3.35	2.83
4	1.13	0.92	0.82	3.08	2.73	2.67	2.36
5	1.41	0.57	0.41	3.13	2.22	2.51	1.78
6	1.29	0.98	0.76	3.48	2.69	1.40	1.08
7	0.65	0.89	1.39	3.05	4.72	2.22	3.44
8	1.19	1.23	1.04	5.28	4.45	3.55	2.99
9	0.58	0.67	1.16	2.67	4.62	1.92	3.33
10	$0.8{\cdot}10^{-3}$	$0.9{\cdot}10^{-3}$	1.19	$0.42{\cdot}10^{-2}$	5.46	$0.38{\cdot}10^{-2}$	4.91
11	0.01	0.01	1.21	0.04	5.78	0.04	5.14
12	0.03	0.04	1.17	0.25	7.65	0.23	7.04
13	0.19	0.23	1.20	0.75	3.96	0.65	3.47
14	$0.33{\cdot}10^{-2}$	$0.43{\cdot}10^{-2}$	1.32	0.02	6.37	0.02	6.18
15	0.01	0.01	1.15	0.04	6.52	0.03	5.60
16	0.03	0.04	1.21	0.19	6.05	0.17	5.29
17	0.06	0.07	1.19	0.37	6.04	0.32	5.23
18	0.01	0.01	1.20	0.06	5.47	0.05	4.86
19	0.11	0.13	1.21	0.63	5.83	0.51	4.76
20	0.11	0.13	1.18	0.67	6.25	0.56	5.20

TABLE 8. Mean ratio between the mean edges for the portion of meshes and the mean edge value for the original mesh for each of the methods ($r = \overline{d_h}/\overline{d_o}$). The closer to 1, the better the result.

Method	Ours	Liepa's [36]	Wang's [38]
r	1.06 ± 0.25	4.77 ± 1.62	4.02 ± 1.61

As shown in Table 8, our method presents the mean ratio value closest to 1, meaning that the meshes obtained with it are more similar to the original meshes than those offered by the other methods.

Although all methods are sensitive to the resolution of the input mesh, we observed that Liepa's and Wang's were affected to a much greater extent by this parameter. Therefore, considering the set meshes of our study, these two methods only carry out a triangulation of the nodes of the holes' boundaries, when filling the lower resolution meshes of the set. However, Ju's method (for the reasons stated above) and our's always insert a mesh portion (with additional points besides the boundary points) to fill every hole.

XI. CONCLUSION

This paper presents an original method to restore incomplete 3D mesh models. The algorithm presented is based on the use of image restoration techniques, traditionally applied to photographs, and adapted here to 3D data. To do so, we first developed a process to optimize the conversion of 3D data to an image format. An inpainting algorithm is then applied to the image. Finally, the inverse 2D to 3D transformation is performed to achieve filling of the hole.

We present the results offered by our method when filling a set of twenty meshes. These results provide proof of good performance and robustness for a great variety of hole sizes and configurations. Its ability to fill holes in corners and areas with abrupt changes of curvature is noteworthy. Additionally, the algorithm also responds correctly in linear structures that appear on surface linear patterns.

We compared our results with those obtained after applying three different popular restoration algorithms. This comparison was made taking into account two aspects: error between ground truth and the portion of mesh generated by each method, and the ratio between the resolution of these portions before and after applying the filling algorithms.

It has been demonstrated that our method produces the lowest errors in most of the holes filled and that is the one with the best behavior on increasing the areas of the holes. It has also been shown that the resolution of the meshes generated by our proposal are the most similar to the original ones.

For future works, we plan to apply our proposed method to other areas of interest such as medical imaging or microscopy where, due to physical or technological restrictions, it almost impossible to directly obtain 3D models without gaps or holes, and, as such, the hole-filling techniques are very helpful to generate a complete 3D model. Furthermore, another novel area where this method could be applied is in 3D printing where a closed 3D model is required as input to the 3D printing system.

REFERENCES

[1] R. A. Jarvis, "A perspective on range finding techniques for computer vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-5, no. 2, pp. 122–139, Mar. 1983.
[2] G. Sansoni, M. Trebeschi, and F. Docchio, "State-of-the-art and applications of 3D imaging sensors in industry, cultural heritage, medicine, and criminal investigation," *Sensors*, vol. 9, no. 1, pp. 568–601, Jan. 2009.
[3] V. Amaral, F. Marques, A. Lourenço, J. Barata, and P. Santana, "Laser-based obstacle detection at railway level crossings," *J. Sensors*, vol. 2016, pp. 1–11, Jan. 2016.
[4] P. Olivka, M. Krumnikl, P. Moravec, and D. Seidl, "Calibration of short range 2D laser range finder for 3D SLAM usage," *J. Sensors*, vol. 2016, pp. 1–13, Dec. 2016.
[5] A. Adán, S. Salamanca, and P. Merchán, "A hybrid human–computer approach for recovering incomplete cultural heritage pieces," *Comput. Graph.*, vol. 36, no. 1, pp. 1–15, Feb. 2012.
[6] J. Geng, "Structured-light 3D surface imaging: A tutorial," *Adv. Opt. Photon.*, vol. 3, no. 2, p. 128, Jun. 2011.
[7] A. Adan, F. Molina, A. S. Vazquez, and L. Morena, "3D feature tracking using a dynamic structured light system," in *Proc. 2nd Can. Conf. Comput. Robot Vis. (CRV)*, May 2005, pp. 168–175.
[8] S. Son, H. Park, and K. H. Lee, "Automated laser scanning system for reverse engineering and inspection," *Int. J. Mach. Tools Manuf.*, vol. 42, no. 8, pp. 889–897, Jun. 2002.
[9] C. Frohlich and M. Mettenleiter, "Terrestrial laser scanning–new perspectives in 3D surveying," *Proc. Work. Group VIII/2, Laser-Scanners Forest Landscape Assessment (ISPRS)*, 2004, pp. 7–13.
[10] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.
[11] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets," *Auto. Robots*, vol. 34, no. 3, pp. 133–148, Apr. 2013.
[12] J. Podolak and S. Rusinkiewicz, "Atomic volumes for mesh completion," in *Proc. 3rd Eurographics Symp. Geometry Process. (SGP)*. Aire-la-Ville, Suiza: Eurographics Association, 2005, p. 33.
[13] S. Roth and M. J. Black, "Fields of experts," *Int. J. Comput. Vis.*, vol. 82, no. 2, pp. 205–229, Jan. 2009, doi: 10.1007/s11263-008-0197-6.
[14] Y. Li, H. Wu, L. Li, Y. Yang, C. Zhang, and R. Bie, "Improved field of experts model for image completion," *Optik*, vol. 142, pp. 174–182, Aug. 2017, doi: 10.1016/j.ijleo.2017.05.077.
[15] E. Pérez, S. Salamanca, P. Merchán, A. Adán, C. Cerrada, and I. Cambero, "A robust method for filling holes in 3d meshes based on image restoration," in *Proc. 10th Int. Conf. Adv. Concepts Intell. Vis. Syst. (ACIVS)*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 742–751.
[16] E. Pérez, S. Salamanca, P. Merchán, and A. Adán, "A comparison of hole-filling methods in 3D," *Int. J. Appl. Math. Comput. Sci.*, vol. 26, no. 4, pp. 885–903, Dec. 2016, doi: 10.1515/amcs-2016-0063.
[17] M. Attene, M. Campen, and L. Kobbelt, "Polygon mesh repairing: An application perspective," *ACM Comput. Surv.*, vol. 45, pp. 15:1–15:33, Mar. 2013.
[18] T. Ju, "Fixing geometric errors on polygonal models: A survey," *J. Comput. Sci. Technol.*, vol. 24, no. 1, pp. 19–29, Jan. 2009.
[19] X. Guo, J. Xiao, and Y. Wang, "A survey on algorithms of hole filling in 3D surface reconstruction," *Vis. Comput.*, vol. 34, no. 1, pp. 93–103, Jan. 2018, doi: 10.1007/s00371-016-1316-y.
[20] C. L. Bajaj, F. Bernardini, and G. Xu, "Automatic reconstruction of surfaces and scalar fields from 3D scans," in *Proc. 22nd Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, New York, NY, USA, 1995, pp. 109–118.
[21] H. Edelsbrunner and E. P. Mücke, "Three-dimensional alpha shapes," *ACM Trans. Graph.*, vol. 13, no. 1, pp. 43–72, Jan. 1994.
[22] N. Amenta, M. Bern, and M. Kamvysselis, "A new Voronoi-based surface reconstruction algorithm," in *Proc. 25th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, New York, NY, USA, 1998, pp. 415–421.
[23] T. K. Dey, J. Giesen, and J. Hudson, "Delaunay based shape reconstruction from large data," in *Proc. IEEE Symp. Parallel Large-Data Vis. Graph.* Piscataway, NJ, USA: IEEE Press, 2001, pp. 19–27.

[24] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Trans. Vis. Comput. Graphics*, vol. 5, no. 4, pp. 349–359, Oct. 1999.
[25] H. Quynh Dinh, G. Turk, and G. Slabaugh, "Reconstructing surfaces using anisotropic basis functions," in *Proc. 8th IEEE Int. Conf. Comput. Vis. (ICCV)*, Jul. 2001, pp. 606–613.
[26] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3D objects with radial basis functions," in *Proc. 28th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*. Cham, Switzerland: Springer, 2001, pp. 67–76.
[27] E. Franchini, "Implicit shape reconstruction of unorganized points using PDE-based deformable 3D manifolds," *Numer. Math., Theory, Methods Appl.*, vol. 3, no. 4, pp. 405–430, Jun. 2010.
[28] L. S. Tekumalla and E. Cohen, "Reverse engineering point clouds to fit tensor product b-spline surfaces by blending local fits," 2014, *arXiv:1411.5993*. [Online]. Available: https://arxiv.org/abs/1411.5993
[29] A. Brunton, S. Wuhrer, C. Shu, P. Bose, and E. D. Demaine, "Filling holes in triangular meshes by curve unfolding," in *Proc. IEEE Int. Conf. Shape Model. Appl.*, Jun. 2009, pp. 66–72.
[30] M. Wei, J. Wu, and M. Pang, "An integrated approach to filling holes in meshes," in *Proc. Int. Conf. Artif. Intell. Comput. Intell.* Washington, DC, USA: IEEE Computer Society, Oct. 2010, pp. 306–310.
[31] W. Zhao, S. Gao, and H. Lin, "A robust hole-filling algorithm for triangular mesh," *Vis. Comput.*, vol. 23, no. 12, pp. 987–997, Nov. 2007.
[32] G. Harary, A. Tal, and E. Grinspun, "Context-based coherent surface completion," *ACM Trans. Graph.*, vol. 33, no. 1, pp. 5:1–5:12, Jan. 2014.
[33] Y. Quinsat and C. Lartigue, "Filling holes in digitized point cloud using a morphing-based approach to preserve volume characteristics," *Int. J. Adv. Manuf. Technol.*, vol. 81, nos. 1–4, pp. 411–421, Oct. 2015.
[34] P. Vichitvejpaisal and P. Kanongchaiyos, "Surface completion using Laplacian transform," *Eng. J.*, vol. 18, no. 1, pp. 129–144, Jan. 2014.
[35] L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk, "State of the art in example-based texture synthesis," Eurographics, Eurographics Assoc., State Art Rep. EG-STAR, 2009.
[36] P. Liepa, "Filling holes in meshes," in *Proc. Eurographics/ACM SIGGRAPH Symp. Geometry Process. (SGP)*. Aire-la-Ville, Switzerland: Eurographics Association, 2003, pp. 200–205.
[37] G. Klincsek, "Minimal triangulations of polygonal domains," in *Combinatorics 79* (Annals of Discrete Mathematics), vol. 9, P. L. Hammer, Ed. Amsterdam, The Netherlands: Elsevier, 1980, pp. 121–123.
[38] C. C. L. Wang and D. Manocha, "Efficient boundary extraction of BSP solids based on clipping operations," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 1, pp. 16–29, Jan. 2013.
[39] G. Barequet and M. Sharir, "Filling gaps in the boundary of a polyhedron," *Comput. Aided Geometric Design*, vol. 12, no. 2, pp. 207–229, Mar. 1995.
[40] M. Kazhdan, A. Klein, K. Dalal, and H. Hoppe, "Unconstrained isosurface extraction on arbitrary octrees," in *Proc. 5th Eurographics Symp. Geometry Process. (SGP)*. Aire-la-Ville, Switzerland: Eurographics Association, 2007, pp. 125–133.
[41] T. Ju, "Robust repair of polygonal models," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 888–895, Aug. 2004.
[42] O. Argudo, P. Brunet, A. Chica, and À. Vinacua, "Biharmonic fields and mesh completion," *Graph. Models*, vol. 82, pp. 137–148, Nov. 2015.
[43] A. Sharf, M. Alexa, and D. Cohen-Or, "Context-based surface completion," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 878–887, Aug. 2004.
[44] S. Park, X. Guo, H. Shin, and H. Qin, "Surface completion for shape and appearance," *Vis. Comput.*, vol. 22, no. 3, pp. 168–180, Mar. 2006.
[45] M. Kazhdan and H. Hoppe, "Screened Poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 3, pp. 29:1–29:13, 2013.
[46] M. Centin, N. Pezzotti, and A. Signoroni, "Poisson-driven seamless completion of triangular meshes," *Comput. Aided Geometric Des.*, vols. 35–36, pp. 42–55, May 2015.
[47] M. Attene, "A lightweight approach to repairing digitized polygon meshes," *Vis. Comput.*, vol. 26, no. 11, pp. 1393–1406, Nov. 2010.
[48] G. E. Hinton, "Products of experts," in *Proc. 9th Int. Conf. Artif. Neural Netw. (ICANN)*, 1999, pp. 1–6.
[49] L. Di Angelo, P. Di Stefano, and L. Giaccari, "A new mesh-growing algorithm for fast surface reconstruction," *Comput.-Aided Des.*, vol. 43, no. 6, pp. 639–650, Jun. 2011.
[50] J. O'Rourke, *Computational Geometry in C*. Cambridge, U.K.: Cambridge Univ. Press, 1994.
[51] AIM@SHAPE, Digital Shape Workbench, Marzo. (2007). *Aim@shape Shape Repository*. Accessed: Feb. 1, 2021. [Online]. Available: http://visionair.ge.imati.cnr.it/ontologies/shapes/
[52] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: An open-source mesh processing tool," in *Proc. Eurographics Italian Chapter Conf.*, V. Scarano, R. D. Chiara, and U. Erra, Eds., The Eurographics Association, 2008, pp. 129–136.
[53] M. Attene and B. Falcidieno, "ReMESH: An interactive environment to edit and repair triangle meshes," in *Proc. IEEE Int. Conf. Shape Model. Appl. (SMI)*, Jun. 2006, p. 41.
[54] T. Ju. *Polymender*. Accessed: Feb. 1, 2021. [Online]. Available: https://www.cse.wustl.edu/~taoju/code/polymender.htm
[55] C. C. L. Wang. *MeshWorks*. Accessed: Feb. 1, 2021. [Online]. Available: https://mewangcl.github.io/Projects/projMeshWorks.html
[56] N. Aspert, D. Santa-Cruz, and T. Ebrahimi, "MESH: Measuring errors between surfaces using the Hausdorff distance," in *Proc. IEEE Int. Conf. Multimedia Expo*, Aug. 2002, pp. 705–708.

EMILIANO PÉREZ received the Ph.D. degree in software engineering and computer systems from the Universidad Nacional de Educación a Distancia (UNED), Spain, in 2011. He is currently an Adjunct Professor with the Universidad de Extremadura, Badajoz, Spain. He has worked as a Researcher on five research and development projects. He has made more than 20 contributions to conferences and journals. He focuses his research on 3-D computer vision and virtual reality, pattern recognition, modeling, representation and reconstruction of 3-D objects, 3-D sensors applied on cultural heritage, and virtual simulation.

SANTIAGO SALAMANCA received the M.Sc. degree in physics from the Universidad Complutense de Madrid, Madrid, Spain, in 1995, and the Ph.D. degree in industrial engineering from the Universidad Nacional de Educación a Distancia (UNED), Madrid, in 2005. Since 2002, he has been an Associate Professor of systems engineering and automation with the Escuela de Ingenierías Industriales, Universidad de Extremadura, Badajoz, Spain. He has made more than 60 international technical contributions to prestigious journals and conferences/workshops. His research interests include pattern recognition, 3-D object modeling and representation, and 3-D sensors.

PILAR MERCHÁN received the Ph.D. degree in industrial engineering from the Universidad de Extremadura, Spain, in 2007. Since 2000, she has been working as an Assistant Professor with the Universidad de Extremadura, where she has been an Associate Professor, since 2012. She has worked as a researcher on more than 30 research and development projects. She has generated about 70 technical contributions to prestigious journals and conferences. Her research interests include 3D computer vision: sensory systems for 3D vision, complex scenes segmentation and retrieval, 3D scene modeling and representation, and their application to cultural heritage.

ANTONIO ADÁN is currently a Full Professor with the University of Castilla-La Mancha (UCLM), Spain, in 2018. He is the Leader of the 3D Visual Computing & Robotics Laboratory, UCLM. He has made more than 120 international technical contributions to prestigious journals and conferences. His research interests include 3D object representation and recognition, 3D data processing, 3D sensors, automatic BIM with scanners, and robot interaction in complex scenes.