Received: 24 January 2017 | Revised: 5 September 2017 | Accepted: 12 September 2017

DOI: 10.1002/rob.21758

REGULAR ARTICLE

WILEY

Surface meshing of underwater maps from highly defective point sets

Ricard Campos | **Rafael Garcia**

Computer Vision and Robotics Institute, University of Girona, Girona, Spain

Correspondence
Ricard Campos, Computer Vision and Robotics Institute, University of Girona, Edifici P-IV, Campus de Montilivi, 17071 Girona, Spain.
Email: rcampos@eia.udg.edu

Abstract

Owing to the many possible errors that may occur during real-world mapping, point set maps often present a huge amount of outliers and large levels of noise. We present two robust surface reconstruction techniques dealing with corrupted point sets without resorting to any prefiltering step. They are based on building an unsigned distance function, discretely evaluated on an adaptive tetrahedral grid, and defined from an outlier-robust splat representation. To extract the surface from this volumetric view, the space is partitioned into two subsets, the surface of interest being at the boundary separating them. While both methods are based on a similar graph definition derived from the above-mentioned grid, they differ in the partitioning procedure. First, we propose a method using S-T cuts to separate the inside and outside of the mapped area. Second, we use a normalized cut approach to partition the volume using only the values of the unsigned distance function. We prove the validity of our methods by applying them to challenging underwater data sets (sonar and image based), and we benchmark their results against the approaches in the state of the art.

1 | INTRODUCTION

Over the past few decades, underwater robotics have opened the door to deep underwater exploration. Knowing the shape and structure of the seafloor at depths otherwise unreacheable by humans is of paramount importance for the scientific community. To faithfully represent these environments, three-dimensional (3D) maps provide rich and straightforward information.

In this context, recent advances in range-scanning technologies have led to the widespread use of point cloud maps. These 3D point sets represent discrete measurements taken at the surface of a rigid scene and are the base data used to compose 3D maps. It is evident that the lack of connectivity between points leads to problems in data processing, as it prevents to easily take new measurements using this representation, and also complicates visualization, since from a given point of view it is difficult to discern whether a given point should be visible or otherwise occluded by other points in the set. Thus, useful 3D maps require recovering a continuous surface representing the scanned object or scene. The surface reconstruction problem deals with the creation of this continuous surface from the discrete measures represented by the points, normally in the form of a triangle mesh. Surface reconstruction is specially relevant to provide robots with spatial awareness capabilities when navigating in complex 3D and unstructured environments.

A relevant issue is that in real-world applications, and regardless of the type of scanner used, the retrieved point sets inherently suffer from some corruption. The data tend to be outliers ridden and contain noise of different magnitude within the same data set. We refer to *noise* as the variations on the measurements mainly caused by the precision and repeatability of the sensor, whereas *outliers* are purely spurious measurements caused by errors during the scanning process and that do not represent samples of the surface. In addition, the problem of data corruption is aggravated by the underwater medium. Underwater exploration mainly relies on two types of sensors for range data collection. On the one hand, we find the acoustic sensors, providing range data based on the time of flight of emitted sound beams. While they are well known on the area, their resolution is only good for large area mapping. On the other hand, we have the imaging sensors. In this case, due to the visibility limitations of the medium (i.e., light attenuation, low contrast, blurring, artificial lighting), these techniques attain a higher detail, at the expenses of small locality. By means of computer vision techniques, correspondent feature points from different points of view of the scene are used to reconstruct a point set model.

Regardless of the acquisition methodology used, it is difficult to obtain reliable data. In this sense, data filtering and/or smoothing may be needed prior to its further processing. However, a method able to directly cope with unprocessed point set data would be preferred to tackle the problem without the need of manual filtering or parameter

tuning. In this paper, we face the problem of surface reconstruction on data that is far from ideal. We use robust statistics methods to allow the elimination of outliers while obtaining manifold surfaces of the surveyed areas. Moreover, the methodology used in both methods avoids the recurrent requirement of known normals at input points. As we will present in later sections, per-point normal computation is an ill-posed problem on itself. In addition, this allows the applicability of the methods to a wider range of inputs other than robotics derived, where the location of the scanning sensor may not be available. For instance, this is the case for the range scanning examples shown in Section 8, for which no additional data than the 3D point cloud were available. Additionally, we face the problem of bounded surfaces, normally obviated by state-of-the-art solutions. Note that when observing a part of the seafloor, we naturally describe a bounded surface, as we have clear boundaries at the parts where we end or stop the survey. We demonstrate the versatility of the proposed method by applying it to both in-lab and underwater data sets.

2 | RELATED WORK

Despite the above-mentioned richness and usefulness of 3D maps, their creation has gained little interest of the underwater mapping community. Owing to the previously mentioned difficulty associated with the processing of unconstrained 3D data, the widespread use of downward-looking sensors on surveying platforms has promoted the assumption of all range data being projectable on a common plane. This conjecture is used both in acoustic[1–4] and image-based mapping,[5–7] as it reduces a 3D problem to a simpler 2.5D. In this case, the map creation is oversimplified because one can benefit from the large set of tools in the robotics community to retrieve smooth elevation maps even if the data contain some level of corruption. It is also easy to then combine both acoustic and image-based maps to reach further detail.[8,9]

However, it is of high relevance to consider the full 3D structure of the surveyed site. The observation of an area from unrestricted viewpoints reveals the arbitrary concavities that a scene may contain, presenting in this way the real shape of the zone to be mapped. As pointed out in Ref. 10, there is a growing necessity for full 3D mapping in robotics applications and recent advances in underwater robotics work toward this unrestricted scenario.[11–13] Thus, methods such as the ones proposed in this article will render necessary in the near future.

The problem at hand, surface reconstruction, has been an extensively studied issue in the computer graphics and computational geometry communities. For the case of almost-ideal data, where the points are supposed to be measured exactly on the surface of the object, interpolation-based approaches are the most commonly used. In this class of methods, all or some of the input points are part of the output surface vertices. On the one hand, procedures may work with a surface-oriented view, by constructing the surface incrementally by joining triplets of points into triangles, where these triangles normally follow some properties.[14–18] On the other hand, we find volume-oriented approaches, where the problem is defined as separating the inner and an outer volume of the object or scene, such that the surface we are interested in is found at the interphase between both. Normally, the space partition is modeled inside a cell decomposition of the space, such as the Delaunay triangulation or its dual, the Voronoi diagram.[19,20]

On the contrary, when the points have some noise (i.e., they cannot be assumed to be measured precisely on the surface), the approximation methods are more adequate. In this case, the problem is normally cast to an implicit formulation. Thus, the surface is defined implicitly using a distance function (signed or unsigned) or an indicator function. From this model, we can gather a triangle mesh of the surface using iso-surface extraction methods such as marching cubes[21] or the restricted Delaunay triangulation (RDT) mesher.[22]

In this direction, most methods create a signed distance function (SDF) from the input points. This can be done by merging local primitives into a global function. In the case of having a normal associated with each of the input points, each oriented point can be considered as a tangent plane to the surface, such that a simple SDF can be defined by computing the mean distance from a query point to a set of nearer planes.[23,24] There are also many methods using the radial basis functions (RBF) mechanism to interpolate the SDF from the input points and normals.[25] Additionally, instead of using directly the RBF to interpolate, they can also be used to weight the merging of different local contributions, as in the multilevel partition of unity (MPU) algorithm,[26] where the local surfaces computed at the leafs of an octree[27] containing the points are then merged using this technique. To alleviate computational costs, RBFs of compact support have also been explored.[28] In a similar way, variational approaches involving the coherency between the gradients of the SDF and the oriented points have been proposed.[29] Other techniques are those of moving least squares (MLS), also known as point set surfaces (PSS), which was first defined as a projection procedure that, given a point, projects it onto the surface defined by the input set. It turns out that this projection can be casted in most cases to an implicit formulation, which can then be used to retrieve the meshed surface.[30,31] Finally, some methods rely on the prior knowledge of 2.5D connectivity in a single range scan and the known positioning of the sensor at the time of data acquisition to attain a merging of the local contributions into a global SDF.[32,33] The popularity of this last method is proven by the large acceptance of variants of these procedures in real-time mapping applications for land robotics.[34,35]

Besides, there are methods based on building an indicator function to extract the surface from. An indicator function is a pure in/out function from where, again, the surface can be extracted at the interphase between the two volumes. In this category, we find methods where the points with their associated normals are seen as samples of a gradient field, such that the indicator function can be recovered by means of applying the fast Fourier transform (FFT),[36] the Poisson equation,[37,38] or wavelets.[39]

Indeed, there is a clear dependence of most of the approximation-based methods on the knowledge of per-point normals, which is an information that may not be provided directly by the scanning mechanism and that may have to be computed from the points themselves. It should be noted that the problem of computing

normals is expected to be as complex as the surface reconstruction problem itself. Indeed, the normal computation problem can be split in two parts: normal direction estimation and normal orientation.

On the one hand, the estimation of the normal direction of a point consists of fitting a plane with data from its vicinity. These data, in some cases, can be computed from a single scan depending on the type of sensor. Imaging depth sensors provide the ability of computing the normal from a single scan since they have enough neighboring spatial information, derived from the base two-dimensional (2D) image grid, for a given point (note that they may face problems when dealing with occlusions and missing data). However, this is not always the case in robotic applications, where single-stripe 3D scanning mechanisms may be used (e.g., multibeam sonar). In these cases, normal direction computation relies on using data from merged scans in time.[14,23,40–43]

On the other hand, achieving a coherent normal orientation throughout the scene is a complex problem. For the case where no additional information is available, a widely used approach is to heuristically propagate the orientation in the points following a minimum spanning tree (MST).[23] In robotic applications, where we have a notion of where the sensor was at the time of taking the samples, we can force the orientation of the normals to be coherent with that information. Thus, normals making an obtuse dihedral angle with the vector joining the point and the scan position are inverted. Still, when normal directions are computed from noisy data, and taking into account that the scanning position may be uncertain, this method will fail in some cases (see a real-world example in Fig. 11).

For these reasons, recent methods in the state of the art try to overcome this limitation by working with raw data. In the case of unknown normals (or unknown global orientation of those), one can use unsigned distance functions (UDF). Note, however, that an UDF cannot be directly meshed, as we require a positive and a negative volume to extract the surface at the zero level set. To deal with this problem, Hornung and Kobbelt[44] diffuse the contribution of each point in space using simple dilate operations to build a pseudodistance function without the sign inside a regular grid. Then, with the assumption of watertightness on the object to reconstruct, they use an S-T cut algorithm to find the minimum cut in the grid and recover back the sign of the implicit function.

As a matter of fact, the case of outlier-ridden data is dealt by very few methods in the state of the art. However, the lack of available methods is in contrast to the above-stated fact of real-world data always containing outliers to some extent. Methods such as those of Ref. 45 or 46 try to overcome this limitation by means of computing a robust UDF which eliminates the effect of outliers in the final representation. Then, using some heuristics, they recover the sign of the function to extract the surface as the zero level set. A recent method[47] deals with large levels of noise and outliers using local surfaces computed to answer the segment intersection queries required by a RDT mesher. This method is based on building local surfaces of small extent named *splats* around each input point, using robust statistic techniques to disregard outliers, and then merges them together using a modified RDT meshing accepting this new input. A similar method, this time computing on-demand of the RDT mesher the local surfaces, is presented in Ref. 48. Going back to interpolation-based approaches but focusing on the outliers problem, the method of Kolluri and coworkers[49] relies on defining a spectral partitioning of a graph derived from the Delaunay triangulation and the Voronoi diagram to divide the inside from the outside volume disregarding outliers. Also based on interpolation, the sensor position can be included to further refine the graph cut in the presence of outliers.[50–52]

The new methods proposed in this paper fall within those approaches in the state of the art finding a cut surface in a UDF representation extracted from the input points. However, they overcome some of the limitations pointed out for the methods reviewed on the state of the art and described in this section. First, they are designed to cope with high levels of both noise and outliers in the data. Second, they do not require any additional information (i.e., normal vectors) to perform this task. Third, they provide a manifold surface. And finally, we successfully address the usually obviated problem of bounded surfaces.

3 | MOTIVATIONS, OVERVIEW, AND CONTRIBUTIONS

We decided to base our method on the splat representation presented in Ref. 47 because of the superior performance this method has shown when dealing with noise and outliers. However, only the first step of the algorithm is used, that is, the splats creation. For each point in the input set, we generate a local surface named splat with a given small extend, by accounting for noise and disregarding outliers. The second part of the method proposed in Ref. 47 tries to generate the surface by directly meshing the splats using a tailored RDT surface mesher.[22]

Given a set of points E on or near a surface, the RDT is a subcomplex of the 3D Delaunay triangulation of E formed by the Delaunay triangles whose dual Voronoi edges intersect the surface. Each RDT triangle has a circumball centered on the surface and empty of all other E points, named surface Delaunay ball. Boissonnat and Oudot[22] proved that if the sampling E of the surface is dense enough with respect to the local feature size of the surface, the RDT provides a good approximation of the Hausdorff distance to the surface as well as a good approximation of its normals, areas, and curvatures. The meshing algorithm iteratively refines an initial 3D Delaunay triangulation until all surface Delaunay balls meet some properties. More specifically, starting from a small set of points on the surface, the method inserts the center of a surface Delaunay ball at each iteration (i.e., the intersection between the Voronoi edge and the surface) until all the balls fulfill the following three criteria: The triangle inside the ball must have all its angles larger than α_a, the ball must have a radius lower than α_r, and the distance between the center of the ball and the circumcenter of the associated RDT triangle must be lower than α_d. Tuning α_a, α_r, and α_d changes the quality of the approximation and of the output surface triangle mesh in terms of sizing and shape of its triangles.

Since the only requirement of RDT meshing is to be able to know when an arbitrary segment intersects a given surface representation, in Ref. 47 the splats were added to an axis-aligned bounding boxes (AABB) tree[53] for rapid intersection query and the contribution of the many splats that may be intersected was merged in a single point. While this surface reconstruction method has been proven useful in a wide range of scenarios, the reconstruction results suffer from some limitations that we describe in the following.

First, each splat is computed inside a Random sample consensus (RANSAC) procedure[54] to account for outliers, but it is difficult to remove outliers that fall too close to the surface with respect to the desired RANSAC distance threshold. The robust computation of splats has proven successful in removing the so called *gross outliers*, that is, outliers far from the true surface, and whose distance to their nearest neighbors is larger and thus are more sparsely distributed. Only in cases where the outliers are close enough to the surface with respect to the RANSAC threshold they are used to generate splats. However, using these near-by outliers leads to a jagged, self-intersecting splat representation. However, if we take the outliers that fall very close to the true surface as part of the noise, the problem translates into a noise reduction issue.

Second, the splats creation is purely local, and thus no coherence is enforced between neighboring splats, which are supposed to represent a smooth continuous surface. Hence, splats may not fit completely with their neighbors. Moreover, in areas of high curvature, the splats are likely to self-intersect. This method, as defined, works with smooth surfaces. However, even in cases where a surface can be considered smooth, highly curved parts may distort the splat representation. Furthermore, the self-intersection problem is aggravated by the naive splat sizing mechanism, which may lead to larger splats in these sharper areas. In both cases, the result is a nonmanifold surface. This is because the method is set to directly mesh this representation using a modified RDT mesher. As stated above, this mesher requires the representation to be meshed (in this case, the splats) to allow for intersection queries with arbitrary segments. However, the smaller the required output resolution of the mesh, the smaller the query segments required, and consequently the lower the possibility that some splats are intersected during a query. This results in nonmanifold surfaces, as depicted in Figure 7. Having a nonmanifold surface is nonrealistic and complicates further processing applied to the resulting mesh.

Consequently, our previous surface reconstruction solutions having a local view[47,48] may require a postprocessing of the resulting surface to recover from the nonmanifold configurations. Considering the review of the state of the art presented in the preceding section, it can be observed how in the case of having the surface represented by local primitives, the common approach is to merge or blend them into a global implicit SDF. Merging different contributions in a global representation leads to significant noise reduction. Note that in our case, we will use the robust splats computation to eliminate the gross outliers, leaving for the surface extraction step the problem of robustly dealing with high noise. Thus, if the contributions of the possibly self-intersecting splats are merged, this should provide a smooth distance function easier to mesh afterwards. It is because of these reasons that in this article we propose a global implicit setup, from which a manifold surface can be extracted from the vicinity between two subvolumes, attaining also further noise reduction.

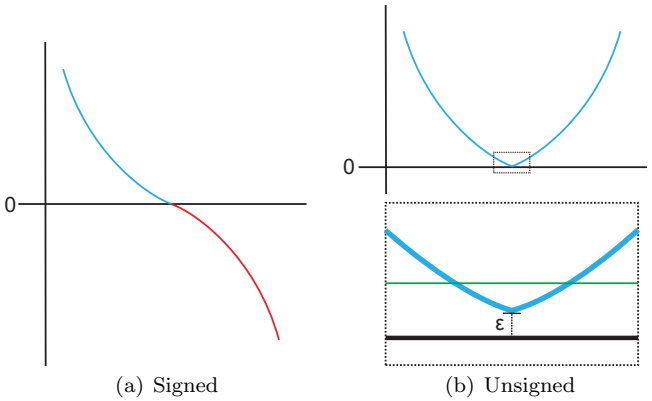


FIGURE 1 The problem of contouring a UDF. Imagine we cut a 2D slice near the surface of the signed (a) and unsigned (b) versions of the distance function defined in Eq. (1). In the signed version (a), there is a clear zero value to isocontour located at the inflection point between positive and negative values; whereas for the unsigned version, due to roundoff errors, an absolute zero value is never found but only a local minima of ϵ. Note, however, that this ϵ changes for different parts in the 3D distance function and thus cannot be fixed. If we try to isocontour the surface at a value close to zero (green line), we will obtain two valid isovalues, defining not just one surface but two, thus capturing not a surface but a *band* of volume

Note, however, that most methods of the state of the art require a priori knowledge of the orientation of the local primitives, usually as known normals at input points, or the knowledge of other additional information such as the sensor position at the time of acquisition. Yet, our splats are not coherently oriented through the surface they define, as we do not take into account their orientation at any step of the splats generation process. This prevents a direct computation for the distance function to be signed.

If we take a close look to how the state-of-the-art approaches deal with this issue, we can see how there is a tendency toward trying to correctly orient the local primitives globally, prior to merge their contributions into an implicit SDF. For this purpose, and up to date, the approaches dealing with this subject are mainly variants of Hoppe's method.[23] Hoppe's method consists of propagating the orientation between neighboring primitives following the MST generated from the input points. In noise-free cases where there are small variations in curvature, Hoppe's method has proven to provide satisfactory results. However, in real-world data, the approach is likely to fail.

Despite this lack of coherence in orientation, we can still use the nonoriented splats to produce a UDF. Unfortunately, we cannot directly extract a surface mesh from a UDF. Figure 1 shows a schematic representation of this phenomenon. Owing to roundoff errors, the exact zero value may never exist, and the surface is defined by local minima on the implicit function. Thus, if we mesh for zero isovalue, we will not obtain one 2D manifold, but two. This is so because we need the isovalue to be a clear inflexion point in the SDF. Thus, we have an unsigned distance field from which we need to extract the

surface. After discarding the orientation propagation in the splats, the best chance is to try to recover the sign of the SDF out of the UDF. In essence, the process consists of dividing the object into its *inside* and *outside* parts.

Note that we can refer to this subdivision as a *clustering* or a *segmentation*. In fact, the problem of inside/outside labeling is a binary segmentation problem. Inspired by the great results achieved by the computer vision community for this task, we propose using a graph formulation and use minimum cut algorithms for solving the inside/outside labeling and, consequently, the surface reconstruction problem.

This paper proposes two new approaches differing in the cutting procedure and the graph construction. However, the computation of the UDF is the same in both cases. For completeness, we start by giving an overall review of the splats creation procedure in Section 4. Then, in Section 5 we present the construction of the UDF. Furthermore, Sections 6 and 7 present the two different methodologies to solve the binary labeling problem, along with the solutions to tackle bounded surfaces. Finally, Section 8 shows the results and discussion on the advantages and disadvantages of both cutting methods, and we finalize this article with Section 9 where we present the conclusions derived from the present work.

4 | SPLATS CREATION

In our case, a splat is defined as a surface of small extent that locally describes the surface around a point. They are constructed using a fixed k-neighborhood around each point. Inside this neighborhood, we try to generate a local jet surface.[55] This jet surface is a least-squares approximation of a bivariate height function referenced in a local frame computed using principal component analysis (PCA). Hence, the accuracy of each splat is defined by the degree of the Taylor expansion approximating the local bivariate function of degree d. Note, however, that throughout this article, we use splats of $d = 2$ at most, that is local bivariate quadrics (LBQ). Using LBQs provides smoother local approximations of the surface than simpler planar primitives, without overfitting to the data as it would be the case when using higher order ones.

To disregard outliers in the computations, the jet surface fit is performed inside a RANSAC procedure.[54] This robust statistics method allows to detect as outliers those points which are not supported by the jet surface of most consensus generated from its neighborhood, or those which did not generate a consensus surface at all. Finally, to give an extent to the splat, we give it a radius equal to the mean distance to the k nearest neighbors. This simple approach generates larger splats in sparse areas and, on the contrary, generates small splats in densely sampled parts.

This method provides two main advantages. First, it removes the so-called *gross* outliers, that is those wrong measurements that are far away from the correct points (even if noisy) and that are easily spottable by a human when observing the data. Second, it provides us with a first noise smoothing step. While this smoothing has proven useful in the case of in-lab range scan data (see the results in the original reference[47]), it is not sufficient for the type of data we expect, for instance, in underwater optical mapping. Optical underwater data suffer from multiple phenomena such as light attenuation, blurring, low contrast, or nonuniform illumination. Thus, the gathered data are noisy, and this noise is translated to any further processing to be applied to these images, such as recovering the observed 3D structure. Thus, when applying 3D optical reconstruction methods, the resulting point sets are far noisier when using underwater images than when the images are captured in-air or in a controlled environment [e.g., see Fig. 14].

5 | UNSIGNED DISTANCE FUNCTION

The purpose of building a distance function is to blend the local contributions into a global distance field. This permits the otherwise noncoherent-related local surfaces to contribute to a consistent global surface represented as the zero isolevel in the distance field. The steps to create a UDF from the splats representation are explained in the following; whereas some of the main concepts are depicted in Figure 2, and the procedure is summarized in Algorithm 1.

We use a variant of the implicit MLS definition[56] to blend the splats together. In the original definition, the contribution of oriented points (seen as planar structures) was blended together to generate a SDF. We update the formula by adding support for splats of degree 2 (i.e., LBQs) and by taking into account the unsigned distance to the local surfaces instead of the signed one. The UDF at a given query point p is then computed as follows:

$$u(p) = \frac{\sum_{s_i \in S} \phi_{s_i}(p) f_{s_i}(p)}{\sum_{s_i \in S} \phi_{s_i}(p)}, \tag{1}$$

where S is the set of splats s_i, $f_{s_i}(p)$ is the algebraic projection of p onto s_i, and ϕ_{s_i} is a Gaussian of the following form:

$$\phi_{s_i}(p) = \frac{e^{-\frac{\|p-c_i\|^2}{\sigma^2}}}{N_s}, \tag{2}$$

where c_i being the center of s_i and N_s is the total number of splats involved in the computation. In fact, for each query point p, $u(p)$ is computed using the splats whose centers fall at a radial neighborhood of size σ. These neighborhood relationships are efficiently obtained through the use of a K-dimensional data structure (KD-tree).[57] Consequently, this distance function has a bounded support, that is only defined on a limited extent of volume around the splats, at a distance of σ at most. Unfortunately, if we want the distance function to be defined in a larger band, increasing σ values leads to an increase of the computational cost. For this reason, we propose constructing a secondary band governed by the parameter σ_2. In this σ_2 band, we compute a coarser approximation of the function defined in Eq. (1) by not taking into account the full radial neighborhood around the query point, but only the first k_σ nearest neighbors. This reduces the computational effort and provides an acceptable approximation. Note that having a coarser approximation of the UDF when far from the input points is

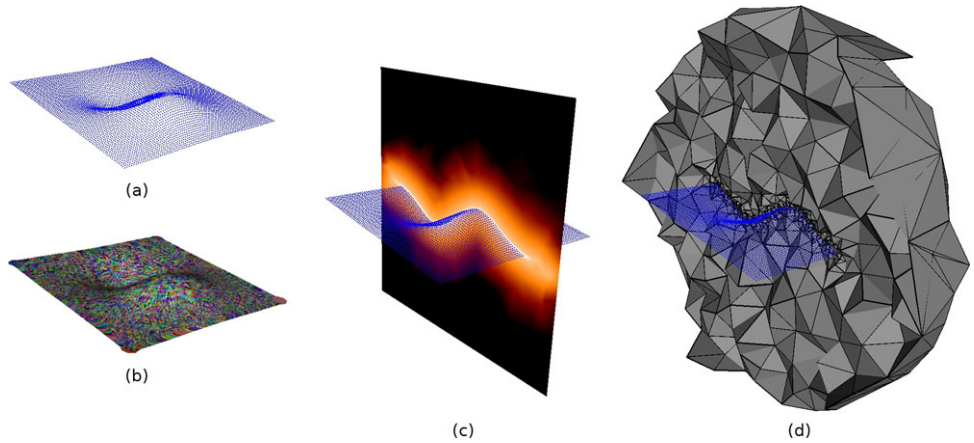


FIGURE 2 UDF creation steps. The original point cloud is shown in (a). From these points, a splats representation is extracted as presented in (b). A slice of the UDF derived from the splats is shown in (c), and in (d) we show the adaptive triangulation containing the function. More precisely, (d) depicts a cut of the volumetric tetrahedralization using the same slicing plane as in (c)

not important in our case, since we are interested in the minimum of the function, and this is located inside the finer σ band. We only use this secondary band because it may be interesting in some cases to use this low-quality version of the distance function, for instance, when a small σ results in holes appearing on the surface and we want our global function to fill them smoothly. In this sense, the larger the band we consider the larger the global implicit function support and, consequently, the larger the reconstructed surface area.

This UDF is a weighted mean of the individual distance contributions of the splats falling near a given query point. It is obvious that the distance function presented does not take into account gross outliers, which are supposed to be eliminated by the splats creation procedure. Note also that we just use the centers of the splats and do not take into account their size in the distance computation. Moreover, we just limit their degree to be of $d = 1$ and $d = 2$, corresponding to points with normals or LBQs, respectively, and they are assumed to be unoriented.

Despite the use of a KD-tree for a rapid query of neighborhood relationships, computing the function at an arbitrary point in space is a costly operation. To alleviate the computational burden of intensively querying this implicit function, which is likely to be required during the surface extraction step, we discretize our domain. At each of the vertices in the partitioned domain, the UDF is computed using Eq. (1). Then, for an arbitrary query point in $\mathbb{R}^3$, the function value is returned using a linear interpolation of the values stored in those vertices. Moreover, instead of following the traditional approach of using a regular grid, we lean toward using an irregular tetrahedral grid that adapts to the density of the input points. This sampling-dependant data structure provides a memory-efficient way of storing the distance function, more precise when closer to the original point set and less precise when away from it.

We use 3D Delaunay refinement[58,59] to obtain the tetrahedral grid discretizing our working space. Starting from a Delaunay triangulation containing, a set of base points, the tetrahedra are refined by interleaving vertex insertion (to modify the connectivity of the mesh) and energy minimization (to optimize the positioning of the vertices) until a given set of user requirements are fulfilled. The triangulation criterion we forced in our case is the ratio between the edge of the minimum length and the circumradius of the tetrahedra to be lower than the threshold α_{re} (also referred as the radius-edge ratio). Note that we are aiming at having a faithful approximation of the distance function near the centers of the splats (which are located near the original points in the input P). By forcing the radius-edge bound to be relatively low, the tetrahedra become larger when far from the points, giving a rough approximation, and smaller when closer to the input data, providing a more precise approximation.

The set of base initial vertices used to trigger the 3D Delaunay refinement step could be, for reasonably small data sets, the centers of the previously computed splats. However, when the number of points exceeds a few thousands of points, including all the centers of the splats generates a too fine-grained tetrahedral mesh. The extraction of a surface of a reasonable resolution does not require the distance function to be extremely precise. Thus, we start by decimating these base vertices to a representative set, able to maintain a proper resolution for the distance function after refinement, but not too dense so as to require plenty of memory resources for the creation of the refined tetrahedral mesh. This is done with a simple octree simplification guided by a depth threshold o.

As previously stated, the values of the distance function are computed at the vertices of the triangulation. Thus, for an arbitrary query point, the tetrahedron containing it is localized and the function value is interpolated using the values at the vertices of the tetrahedra along with the barycentric coordinates of the point. In the case where the query point is outside the band, the value is undefined. We refer to the vertices of this data structure as U and to its refined Delaunay structure as Del(U).

ALGORITHM 1 Create UDF

function UDF(S, o, α_{re})
  # Create data structures
  $C_s \leftarrow S$.getCenters() *# Centers of the Splats*
  $O \leftarrow$ createOctree(C_s, o) *# Octree at depth o*
  $T \leftarrow$ createKDTree(C_s) *# KD-Tree*
  # Get the mean points at the leafs of the octree
  $C_o \leftarrow O$.getLeafCenters()
  # Build the Delaunay triangulation with these centers
  Del(C_o) $\leftarrow$ createDelaunayTriangulation(C_o)
  # Refine the Delaunay triangulation to fulfill the requirements
  Del(U) $\leftarrow$ DelaunayRefinement(Del(C_o), α_{re})
  # Compute the distance function for each vertex in Del(U)
  for $v \in U$ **do**
    $p = v$.point *# The 3D point associated with the vertex*
    $d = T$.getDistanceToNearestPoint(p)
    if $d < \sigma$ **then**
      $S_N = T$.getRadialNN(p) *# All Splats falling in radial search*
      v.distance = udf(p, S_N) *# UDF at p using S_N, see equation 1*
    else if $d < \sigma_2$ **then**
      $S_N = T$.getKNN(p, k) *# K-Nearest Splats only*
      v.distance = udf(p, S_N) *# UDF at p using S_N, see equation 1*
    else
      v.distance = *Undefined*
    end if
  end for
  return Del(U)
end function

6 | S-T CUT

Given the unsigned function described above, our goal is to recover its sign. That is, we want to distinguish between the part of the volume inside the object from the outside part and then simply apply the corresponding sign to each part to convert the original UDF into a SDF. You can see a 2D depiction of the proposed approach in Figure 3 as well as a summary of the method in the form of a pseudocode in Algorithm 2. Certainly, this problem can be considered as a binary labeling problem: We aim at partitioning the vertices *U* in the above-mentioned tetrahedralization into the two labels *in* and *out*. Given the extensive use of the S-T cut methods in binary optimization, our first proposal is to adapt this method to perform the partition on our scenario.

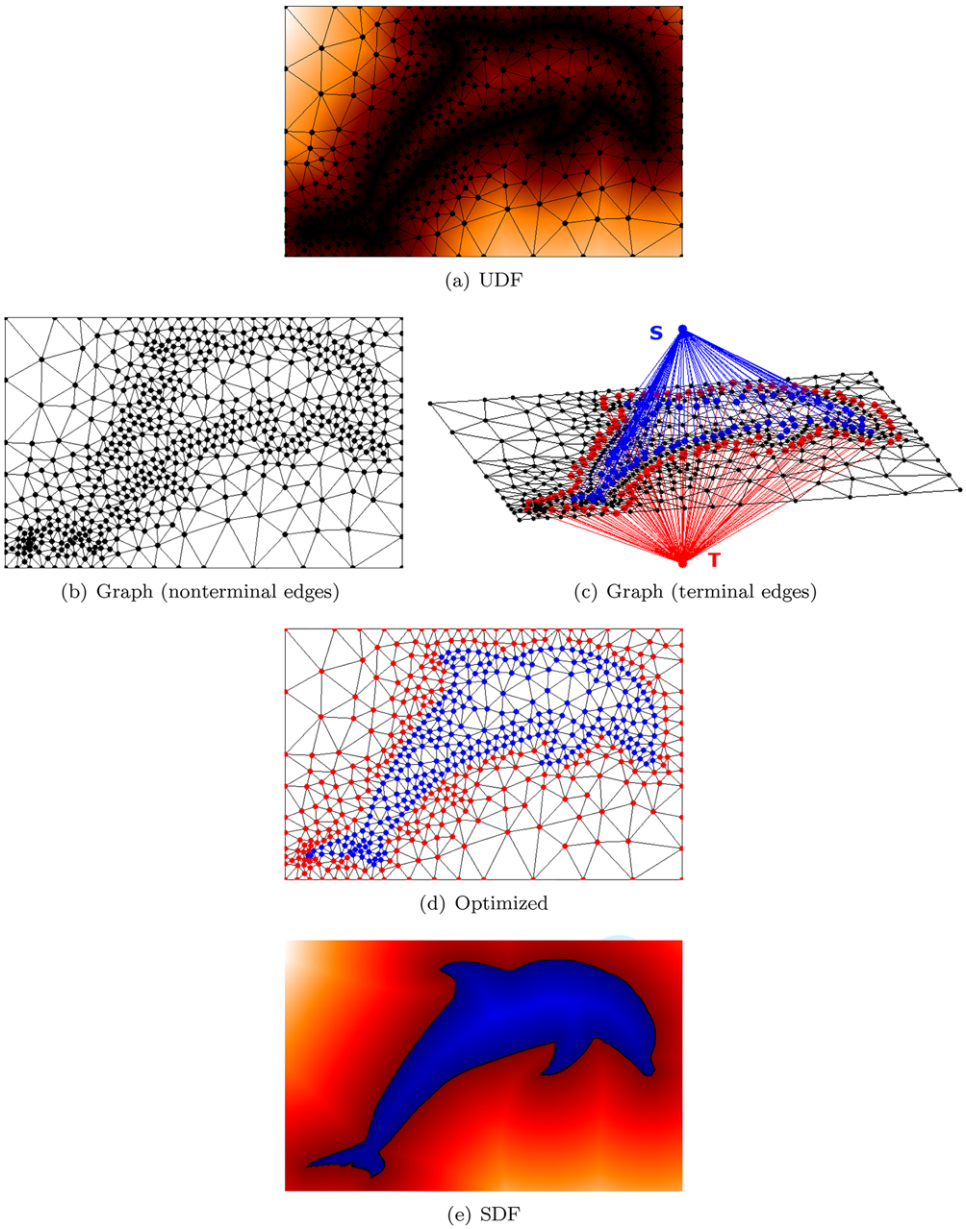
An S-T cut divides a set of nodes in a specific type of graph into two disjointed sets minimizing the cost associated with the removed edges. Commonly referred to in the literature with the generic name of *graph cuts* or, alternatively, the min-cut/max-flow algorithm, this method obtains an exact minimization for binary optimization problems. This technique has been extensively used in a wide range of applications in computer vision and graphics.[44,50,51,60–69]

A graph $G = \langle V, E \rangle$ is composed of a set of vertices V and the set of edges E joining them. In the specific case of S-T cut algorithms, the set of nodes V contains two special nodes, s (source) and t (sink), referred to as the *terminal* nodes, so that $V = P \cup \{s, t\}$, P being the rest of nonterminal nodes. Each edge joining vertices v_i and v_j stores a given weight $w(e_{i,j})$. We differentiate between these edges by calling them terminal edges if they join a terminal vertex with a nonterminal one, or nonterminal edges if they only describe interactions between nonterminal vertices.

An S-T cut of the presented graph is a partitioning of the graph nodes into two subsets, S and T, so that $s \in S$ and $t \in T$. The cost of cutting a graph equals the sum of weights on the severed edges $w(e_{i,j})$, so that $v_i \in S$ and $v_j \in T$. Thus, the problem is then to find the minimum S-T cut from all the possible cuts in the graph. A useful result in combinatorial optimization is that the minimum S-T cut is dual to the problem of finding a *maximum flow* from source s to sink t.[70]

From the binary optimization point of view, the terminal nodes s and t represent our possible labels. Suppose we can define a cost for assigning each node to a given label. Also, assume the labeling problem does not depend only on the nodes themselves but also on the labels of their neighbors. Thus, our problem requires an optimization of the labels for each vertex in the graph, by enforcing spatial coherence between neighbors. This results in the minimization of an energy function composed of a data term and a smoothing term:

$$E(l) = \sum_{p \in P} T_p(l_p) + \sum_{e_{i,j} \in N} V_{i,j}(l_i, l_j), \tag{3}$$



(a) UDF

(b) Graph (nonterminal edges)

(c) Graph (terminal edges)

(d) Optimized

(e) SDF

FIGURE 3 2D depiction of the pipeline followed in the S-T cut method. Starting from an UDF evaluated in an adaptive grid (a), a graph structure is built using the connectivity defined by the triangulation (b). Additionally, for the case of S-T cut, some of the vertices in the graph are connected to two special nodes; s and t (c). After the graph optimization, we obtain a binary partition of the vertices in the graph (d), which is used to sign the original UDF (e). Using the SDF, we can extract the surface, the curve in this case, at its zero-level set. Note that the normalized cut method passes directly from (a) to (d), given that it does not require inside/outside knowledge

where $T_p(l_p)$ is the data penalty term, $V_{i,j}(l_i, l_j)$ is the interaction potential, and l_p is the binary labeling to optimize, defined as follows:

$$l_p = \begin{cases} 0 \text{ if } \mathsf{v}_\mathsf{p} \in \mathsf{S} \\ 1 \text{ if } \mathsf{v}_\mathsf{p} \in \mathsf{T}. \end{cases} \tag{4}$$

Note that, in our case, the 0 and 1 labels correspond to the *inside* and *outside* of the object to be reconstructed.

Related to the above-mentioned interaction potentials, $T_p(l_p)$ indicates per-vertex labeling preferences, whereas $V_{i,j}(l_i, l_j)$ encourages spatial coherence and penalizes discontinuities between neighboring labels. Kolmogorov and Zabin [71] proved that a globally optimal labeling for the energy in Eq. (3) can be found using the minimum cut on an S-T graph. As suggested in many other approaches,[44,50,51,68] we use the algorithm presented by Boykov and Kolmogorov[72] to solve for this minimum cut.

We define our graph G using the same connectivity of Del(U), that is, the vertices P of the graph correspond to U, and the edges in the adaptive structure containing the UDF also define the relationships between vertices in G. Additionally, some of the vertices in the graph have a pair of edges joining them to both the s and t nodes. This leads to the graph definition depicted in Figures 3(b) and 3(c) for the 2D case.

ALGORITHM 2 S-T cut algorithm

```
function STCut(S, Del(U), β, δ_r, N_rays, BoundedSurfaceFlag)
    # Graph structure created from vertices and edges in Del(U)
    V ← U
    E ← Del(U).getEdges()
    G = ⟨V, E⟩
    # Smooth weights matrix W_n
    for e_{i,j} ∈ E do
        W_n(i, j) = w_n(e_{i,j}) = ( (u(p_i)+u(p_j))/2 )^β  # As in equation 6
    end for
    # W_s and W_t vectors of terminal weights
    A(S) = createAABBTree(S) # AABB tree of splats
    # If we want to recover a bounded surface, prepare the spherical cap
    if BoundedSurfaceFlag then
        C = createSphericalCap(S)
    end if
    for p ∈ U do
        i_even = 0
        i_odd = 0
        for i = 0 to N_rays do
            r = createRandomRay(p) # Random ray with origin at p
            n = getNumIntersections(r, A(S))
            if BoundedSurfaceFlag then
                if rayIntersectsSphericalCap(r, C) then
                    n = n + 1
                end if
            end if
            if n mod 2 = 0 then
                i_even = i_even + 1
            else
                i_odd = i_odd + 1
            end if
        end for
        W_s(i) = w_s(p) = i_even/N_rays
        W_t(i) = w_t(p) = i_odd/N_rays
    end for
    # Compute the cut
    labels = STCut( W_n, W_s, W_t )
    # Change the sign of the distances at vertices in Del(U) according to the label
    for l ∈ labels do
        if l = 0 then
            v = vertexCorrespondingToLabel(Del(U), l)
            v.distance = −v.distance
        end if
    end for
    return Del(U)
end function
```

To apply the optimization method to the U nodes in our space partition, two main steps are required. On the one hand, we have to infer a confidence for each $p_i \in U$ to be inside or outside the shape. On the other hand, we need to devise an internodes weighting. Equivalently, we need to define both terminal and smooth weights.

On the one hand, for the terminal weights composing $T_p(l_p)$, we take advantage of our splat representation. Recall that this representation is already a good approximation of the surface of the object. In the original paper,[47] the authors defined a robust intersection method using RANSAC to be able to query for intersection tests given the segments required by a RDT mesher. Consequently, we have a way of inducing an intersection test against the splats approximation.

An intuitive approach to knowing whether an arbitrary point in space is inside or outside an object is to count the number of

intersections between a ray with its origin at this point and the surface of the object. When the number of intersections is *odd*, the point is inside the object, and when *even* outside. Thus, we use a procedure similar to that presented in Ref. 45 but adapted to our representation. We induce the confidence of a point as being inside or outside the object by throwing random rays originating from that point in multiple directions and counting the number of intersections with the splat representation. By throwing multiple rays, we are accounting for the possible unreliability of the ray-intersection procedure due to noisy and/or self-intersecting splat representations, as explained above.

For each vertex/point p in the triangulation, we throw a given number of rays, N_{rays}, with a starting point at p in random directions and count the number of intersections. Given all these intersections, we count those resulting in an even number, i_{even}, and those giving an odd number, i_{odd}. Note that we do not throw a single ray, taking into account that the ray–splats intersection query may fail in some cases. Using all these tests, the confidence for a given point to be part of the inside or the outside, and, consequently, its weight with the s and t nodes is

$$\begin{aligned} w_s &= i_{\text{even}}/N_{\text{rays}} \\ w_t &= i_{\text{odd}}/N_{\text{rays}}. \end{aligned} \tag{5}$$

Note that, in this case, we have fixed the s node, and consequently the S set of the cut, to represent the outside of the object, whereas t (resp. T) represents the inside.

In fact, for the tested data sets, the robust ray–splats intersection query has been empirically proven to work on large scales, that is, far from the splats, but on the contrary is not reliable within small scales, that is, near the splats. Consequently, the vertices in U outside the σ band are more suitable for reliable stochastic ray signing as presented above. Furthermore, despite the fact that the ray–splats intersection query is optimized using AABB trees, its repeated use could lead to a drop in time performance for the method. To alleviate computational complexity, we simply apply the stochastic ray confidence computation to the points at the interphase of the σ band. That is, we just compute the confidence for a vertex if at least one of its adjacent neighbors is inside the band. This means that only a small subset of vertices has a terminal weight, and, for the rest of the vertices, no terminal edge is added. Thus, points with no terminal weights obtain the final label due to the propagation ruled by the smooth weights.

On the other hand, smooth weights are derived directly from the unsigned distance function values. Thus, each $e_{i,j} \in \mathrm{N}$ has a weight $w_n(e_{i,j})$, which corresponds to the direct evaluation of the unsigned distance function along the edge. Since our vertices in the graph coincide with those in Del(U), we set the weight to

$$w_n(e_{i,j}) = \left(\frac{u(p_i) + u(p_j)}{2} \right)^{\beta}, \tag{6}$$

which is the mean value of the function value at the endpoints of the edge, and where the power β is a user parameter allowing the emphasis of the minimum of the distance function as suggested in other unsigned reconstruction approaches.[44,73] Using this definition for the smooth weights enforces the minimum cut to pass through the minimum of the UDF. Note that our distance function is defined just on a narrow *sigma* band near the centers of the splats. Thus, smooth weights $w_n(e_{i,j})$ are only defined inside this band. For the rest of the edges, a default constant value is added, enforcing the propagation of labels from the nearby neighbors in the band.

6.1 | Extension to bounded surfaces

It is obvious that the above-presented S-T cut approach has a clear flaw regarding our application area. As described in Section 1, the 3D reconstruction of a part of the seafloor often corresponds to a bounded surface. We have seen that most of the methods in the state of the art assume the underlying surface described by the input points to be closed (i.e., watertight). This is motivated by the broad application domain, which mainly consist of the range scanning of objects, which can be scanned from arbitrary viewpoints in a laboratory or in-air environment. However, when exploring the seafloor, there are clear boundaries defined in the places where the survey ends, and it is a very rare case to be able to observe and scan a shape from arbitrary viewpoints due to the restrictions in the surveying vehicle, even if observing small-scale structures (e.g., corals). Thus, we cannot assume that there is an *inside* notion and, consequently, the inside/outside confidence procedure as previously presented does not apply. We adapted our method to work in this scenario by *simulating* the surface to be closed.

We start by computing a global plane using PCA with the centers of the splats. Then, we compute the bounding sphere containing these centers and use the previously computed plane to chop it off into two spherical caps. In this way, one of the caps is used as the reference to virtually define an inside/outside part of the shape. To do so, during the inside/outside labeling through stochastic ray throwing, we count any intersection with the selected spherical cap as a valid intersection. This process is intuitively depicted in the 2D schematic in Figure 4.

Note that we do not distinguish which part is which, that is we randomly label one of the two spherical caps resulting from stabbing the bounding sphere by the global plane as inside or outside. Since just the surface is required, the retrieved solution is valid up to a possible global orientation change of the resulting triangle mesh after the surface extraction step. Nevertheless, due to the insertion of this virtual spherical cap into the system, the retrieved surface is closed. Thus, we create some parts of the surface that are, in fact, not part of the real bounded surface. These parts of the surface should be eliminated, using the approach that will be presented for the *normalized cut* case in subsequent sections.

7 | NORMALIZED CUT

We have seen that the S-T cut technique requires providing a notion for some of the points to be inside or outside the shape. This procedure is necessary with several state-of-the-art methods that also try to find the optimal separation of inside/outside volumes. However, in our specific case, this knowledge is just a by-product, since the final goal is to recover the interface between the two volumes, that is, the surface of the object. Given that we are just interested in this separating

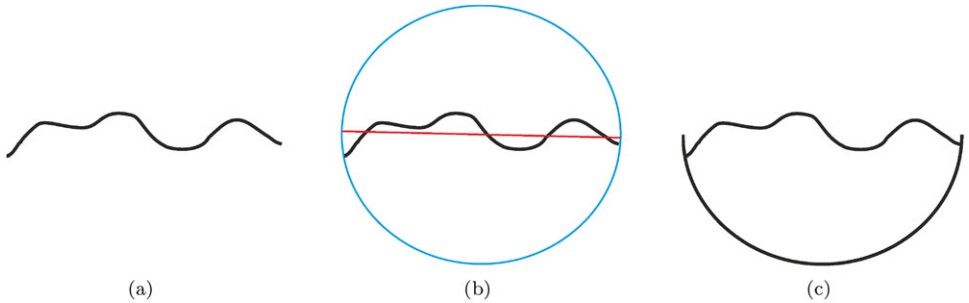


FIGURE 4 2D schematic of the extension for the S-T cut method to handle bounded surfaces. The curve in (a) is not closed, and consequently it does not define an inside or an outside. We compute a global frame using PCA and use it to chop a bounding sphere into two caps, as depicted in (b). One of the spherical caps is used to *virtually close* the surface as presented in (c), so that we can disambiguate the inside/outside computation

surface, in this section we propose a method to partition the volume into two, but disregarding which part is the inside or the outside. For this purpose, we use another commonly referred to graph-based technique applied in solving the binary segmentation problem: the normalized cut.

Spectral methods are extensively used in clustering and segmentation in image processing.[74–79] It is also worth noticing that, even if originally described using graph theory, normalized cuts have a direct formulation as a random walk,[80] or as a separation using hyperplanes (similar to support vector machines),[81] among other interpretations.

Moreover, normalized cut has been used with minor changes in surface reconstruction by Kolluri et al.[49] Note, however, that their method is interpolation based, and thus its definition is completely different from ours. While they were concerned with the outlier rejection problem, we are more concerned with the attenuation of noise. That is, in their approach, the spectral cut was used to disregard outliers, whereas, in our case, we use it to partition a volume from an implicitly defined UDF.

All in all, the question we want to answer is: Can we solve the partitioning problem without having to rely on a specific labeling? Taking a look back at Eq. (3), this results in removing the data term, leaving the energy to minimize as follows:

$$E(l) = \sum_{e_{i,j} \in N} V_{i,j}(l_i, l_j). \tag{7}$$

Thus, we have to deal with the general definition of a minimum cut: The graph has to be partitioned into two groups, regardless of their label, so that the edges of the same group have a high weight, and, on the contrary, edges between different groups have low weights. Since they do not have a specific meaning in this case, we rename our binary regions as A and B, so that the cut can be defined as follows:

$$\text{cut}(A, B) = \sum_{i \in A, j \in B} w(e_{i,j}), \tag{8}$$

which means that the minimum cut corresponds to the one minimizing the total weight of the edges removed.

However, as pointed out by Wu and Leahy,[82] minimizing the cut directly, as described in Eq. (8), favors the cut of a small set of edges, that is, severing a few edges often leads to a minimization of the cut. Conceptually, we want the groups in this partition to be relatively large with respect to the total number of nodes. The normalized cut method tries to overcome this problem by forcing the sum of weights in both parts to be *similar*. This is obtained by normalizing the cost of the cut relative to the cost of all the edges in each region:

$$NCut(A, B) = \frac{\text{cut}(A, B)}{cost(A)} + \frac{\text{cut}(A, B)}{cost(B)}, \tag{9}$$

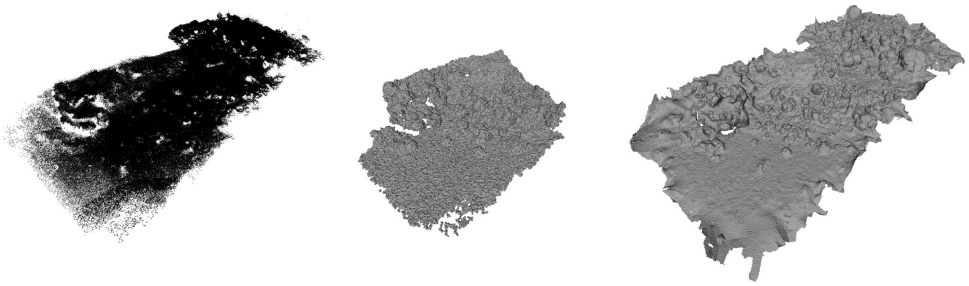
where $cost(X)$ is a sum of the weights of the edges contained in the set X. This definition promotes the two sets A and B to be larger and of similar cost.

As pointed out in the original article by Shi and Malik,[74] solving this problem is NP-hard. However, if we change the labeling from pure binary to continuous, the problem can be reformulated into a minimization that can be solved exactly.

In fact, the underlaying graphs of the two techniques proposed in this article are very similar, just the links to S-T sites are missing in the normalized cuts case. Thus, the filling of the graph and the creation of the weights are exactly the same. In this case, since there is no additional labeling hint, the parameter β, used to stress the UDF minimum, has more relevance than in the case of the S-T cut.

Note, however, that the small σ band used in the S-T cut case may not be sufficient for the present case. If this band is too small, the cut criterion could lead to problems like the one posed in Figure 5. If the σ band is too narrow, the cut that minimizes the cost, and also balances the sum of edge weights in each partition, may separate the edges into two parts that do not necessarily pass through the minimum of $u(x)$. To force the cut to pass through the minimum of the UDF, we need a larger σ band, so that the weights on each side of the band take greater importance in the balancing factor [i.e., the term $cost(X)$ in Eq. (9)].

Hence, in this case, the σ_2 band is more important, as it allows defining a coarsely approximated larger band of the UDF, promoting this way the cut to have large sum of weights on each side and to cut edges that are nearer to the minimum of our UDF. It is also worth mentioning that it may be interesting in some cases to use this low-quality version of the distance function in the S-T cuts case, for instance, when a small σ results in holes appearing on the surface and we want our global function to fill them smoothly. In this sense, the larger the band we consider,



(a) Sample point set (b) Smaller band result (c) Larger band result

FIGURE 5 Relevance of the σ (or σ_2) band size for the normalized cut method, depicted on a data set of a small coral reef. When applied to a narrower band (b), the normalized cut may be minimized by a cut that does not pass through the minimum of the UDF. By enlarging the σ band (c), the weight of the edges on each side of the partition increases, forcing the cut to pass through the minimum in UDF

the larger the global implicit function support and, consequently, the larger the reconstructed surface area.

After obtaining the cut, we flip the sign of one of the regions, *A* or *B* (it does not matter which), and extract the surface using a surface mesher on the resulting SDF. Finally, the results presented in Section 8 demonstrate that the differences between the S-T cut and the normalized cut methods are not that great. What is important to emphasize is that, in this second case, we do not use labeling hints at all. This can be seen in Algorithm 3, which basically mimics the computation of the smooth weights in Algorithm 2 and applies a different partitioning strategy to the graph (which also equals to passing from (a) to (d) directly in Fig. 3). Thus, we prove that solving the surface reconstruction by partitioning the working volume does not require the knowledge of a specific labeling but just the partition itself, that is, swapping the sign in the SDF does not impact the recovered surface.

7.1 | Removing hallucinated triangles

The partition presented above takes place only in the σ band. Thus, we give a sign to a slab of volume and, consequently, the retrieved surface mesh does not contain only the part we are interested in, that is, near the input splats, but also some triangles corresponding to the closing of this volume [see Fig. 6(d)]. Following the nomenclature proposed by Jancosek and Pajdla,[52] we refer to *hallucinated triangles* as those which are part of the reconstructed surface obtained so far, but that do not correspond to the real surface (i.e., they are far from the input splats in this case). To remove these artifacts, we eliminate the hallucinated triangles by using the $u(x)$ value of their vertices in the original UDF. An example of the reconstructed surface before and after removing hallucinated triangles is presented in Figure 6.

We have a large number of vertices having a $u(x)$ close to zero, corresponding to the part of the mesh close to the surface, and another large part having $u(x)$ values that progressively increase the farther away from zero they are, corresponding to the hallucinated part.

ALGORITHM 3 Normalized cut algorithm

function NormalizedCut(Del(U), β)
  # Graph structure created from vertices and edges in Del(U)
  $\mathsf{V} \leftarrow U$
  $\mathsf{E} \leftarrow \mathrm{Del}(U).\mathrm{getEdges}()$
  $\mathsf{G} = \langle \mathsf{V}, \mathsf{E} \rangle$
  # Smooth weights matrix W_n
  for $e_{i,j} \in \mathsf{E}$ **do**
    $W_n(i,j) = w_n(e_{i,j}) = \left(\frac{u(p_i)+u(p_j)}{2}\right)^{\beta}$ *# As in equation X*
  end for
  # Compute the cut
  labels = NormalizedCut(W_n)
  # Change the sign of the distances at vertices in Del(U) *according to the label*
  for $l \in$ *labels* **do**
    if $\mathrm{l} = 0$ **then**
      $v = \mathrm{vertexCorrespToLabel}(\mathrm{Del}(U), l)$
      $v.\mathrm{distance} = -v.\mathrm{distance}$
    end if
  end for
  return Del(U)
end function

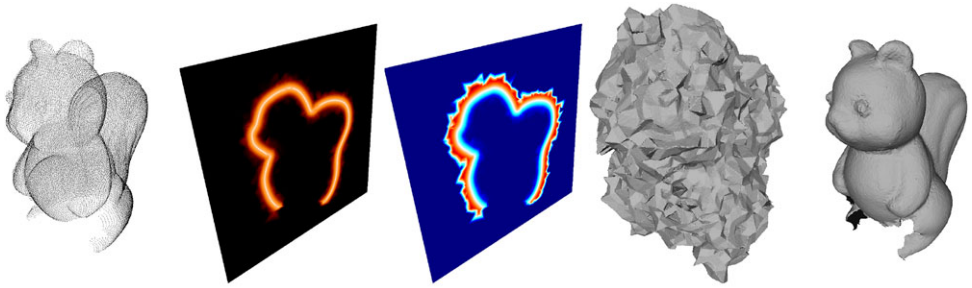


FIGURE 6 The effect of hallucinated triangles on the normalized cut method. Note how the reconstruction in (d) is covered by the meshed part of one of the volume slabs that has been meshed. This is due to our procedure giving a coherent sign only inside the σ_2 band. This phenomenon can be seen in (b) and (c), showing a 2D slice of the unsigned and signed versions of the 3D distance function, respectively. After the automatic hallucinated triangle removal, we can see the underlying surface in (e)

Therefore, we need to infer if a vertex is part of the surface by checking if its $u(x)$ is close enough to zero. However, this notion of *close* is not defined for a given data set. For this purpose, we use the modified selective statistical estimator method (MSSE)[83] to detect such a gap between $u(x)$ values close to zero and the rest.

The MSSE is a scale estimation method that works by detecting a big jump in the sorted residuals of a model. In our case, residuals are defined as the $u(x)$ values evaluated for all the vertices of the resulting mesh. Starting at a low position in this sorted list, fixed to the 10% of the number of values in all the presented cases, we can run over the list and update the scale measure iteratively until we find a *big jump*. When moving through the list of sorted residuals, it is supposed that we will find this big jump when the values of the implicit function get larger and larger, corresponding to the $u(x)$ of vertices in the mesh away from input points. Thus, the scale can be estimated as the first value φ_j in the list not following the inequality below:

$$\frac{\varphi_{j+1}^2}{\varphi_j^2} > \frac{T^2-1}{j-2}, \tag{10}$$

where φ_j is the standard deviation of the $u(x)$ values considered up to the j index in the sorted list and T is a constant factor that we set to 2.5. So, we iteratively move the j-sorted $u(x)$ a position and compare with that in $j-1$ to see if there is a too large step, which would indicate that this value could be considered as from another distribution, that is, from the part away from the input points.

Then, we simply remove the triangles having a vertex whose $u(x)$ is bigger than $2.5\varphi_j$ from the surface mesh. Note that, as previously mentioned, we also apply this method when using the S-T cut method in the case of bounded surfaces. Needless to say, since in this case the labeling only happens in the σ/σ_2 band, this procedure also handles the bounded surfaces problem [e.g., see Fig. 6(e)].

8 | RESULTS

In this section, we present the results obtained using the two graph cutting techniques described above. We list the parameters used to obtain the presented results for each method in Table 1, along with the running times required for each experiment. In this section, we focus on the application of both methods in noisy underwater data sets, using both acoustic and optical data. Finally, we provide a qualitative and quantitative review of the behavior of the methods against the state of the art.

Once the SDF has been retrieved, we can use any contouring algorithm to extract the surface in the form of a triangle mesh. For instance, we could apply the widely used marching cubes. However, given the advantages of RDT meshing[22] in terms of the quality of the triangles (close to regular sized), we use this one for all the results contained in this section.

Note that we are able to obtain manifold surfaces using both methods, which makes them amenable to further postprocessing. Take as example the simple underwater data set consisting of a rocky area in shallow water presented in Figure 7 and note the nonmanifold configurations generated by the original splats mesher method[47] in contrast with the results obtained by our methods and how the differences between the results obtained by both the S-T and normalized cut algorithms are imperceptible.

As pointed out in Section 5, we also open the door to the use of directed points (i.e., points with unoriented normals) in our framework. Many proposals in the literature deal with the problem of finding robust but unoriented normals.[40–43] In each of these cases, the problem is what to do with this directed but unoriented point set to obtain the surface. Using our normalized cut for this purpose, we can retrieve a manifold surface. For the case of the S-T cuts method, we would require splats of $d = 1$ with a limited extension to use this approach. This reduces to using the linear approximation of the directed point as local surface and compute its extent, for instance, as explained in Section 4. The results in Figure 8 were obtained using this kind of linear primitives. We provide in this figure the application of both methods to an in-lab range scanned data set consisting of an amphora.

Regarding range scanning applied underwater, it is well known that acoustic range sensing is the preferred tool for mapping large underwater areas, thanks to their long working distance.[1,84,85] For this reason, we test our methods against some data sets acquired by a DeltaT multibeam echosounder mounted in the Girona500 AUV (see Fig. 9). Problems like reflections and the inherent low resolution of acoustic sensors result in the point cloud retrieved using this technology being

TABLE 1 Details of the parameters used and the running times required to generate the results for the S-T and normalized cut algorithms

				Parameters						Run Times (s)		
	Name	Number of points	Figure	σ	σ_2	k_σ	δ_r	β	α_r/α_d	UDF	SDF	Mesh
S-T	Shallow Water	1,856,271	7(c)	3	0	0	1	4	0.05	85.44	435.62	60.20
	Amphora	829,039	8(c)	3	0	0	0.7	4	0.5	234.73	3177.66	67.14
	Tour Eiffel	1,368,115	13(a)	7	0	0	0.5	4	0.05	361.61	2807.2	55.39
Normalized	Risu3	43,535	6(c)	5	10	5	–	4	0.1	23.74	8.38	33.42
	Coral Reef	1,656,413	5(c)	6	250	25	–	8	0.05	86.22	31.29	218.24
	Shallow Water	1,856,271	7(d)	6	100	25	–	8	0.05	148.44	93.88	49.00
	Amphora	829,039	8(d)	3	25	5	–	4	0.5	247.73	302.13	92.89
	Mound	394,071	10	15	50	50	–	9	0.05	1784.31	71.44	232.77
	Cave	55,876	12(c)	3	20	5	–	8	0.5	58.59	79.45	16.50
	Tour Eiffel	1,368,115	13(b)	6	150	25	–	7	0.05	313.17	99.61	25.71
	La Lune (1)	1,137,820	14(a)	15	50	150	–	8.5	0.05	2361.61	96.33	194.35
	La Lune (2)	832,009	15(c)	15	50	50	–	8	0.05	3884.01	103.25	101.82

Regarding parameters, σ and σ_2 are expressed in terms of the average spacing between points (with $k = 6$), k_σ is the number of nearest neighbors taken into account to compute the UDF in the σ_2 band, δ_r is the RANSAC distance threshold (only used in S-T cut), and α_r/α_d are the meshing parameters goberning the resolution of the output mesh. Some of the parameters presented in the text have been fixed for all the data sets: the octree depth $o = 10$ (except for the Cave data set, for which no octree was used), $\alpha_{re} = 1.5$, $N_{rays} = 25$. All results were generated on a Intel Core i7-3770 CPU with 32 Gb of RAM. Regarding runtimes, UDF, SDF, and Mesh refer to the time spent on the creation of the unsigned distance function, its signing and its meshing, respectively.

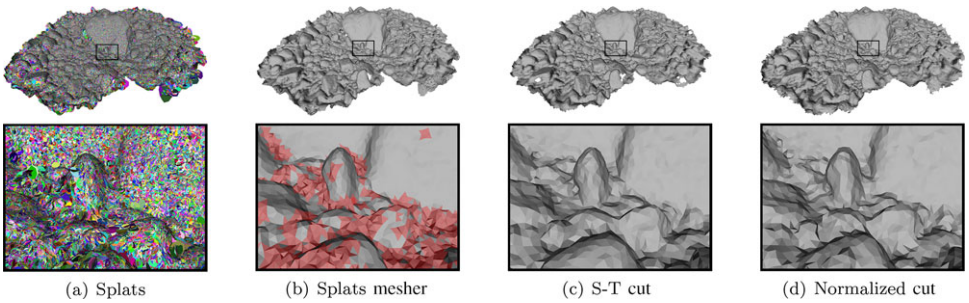


FIGURE 7 Using the same splat representation shown in (a), we present a comparison of the splats mesher[47] (b) and the two methods proposed in this article: (b) for S-T cut and (c) for normalized cut. One of the main problems of the splats mesher method presented in Campos et al.[47] are the nonmanifold configurations when the query segment becomes small enough to not merge contributions of more than a single splat. We marked in red each triangle containing a nonmanifold edge. Since the robust intersection detection this method is based on requires redundancy, that is, more than a single intersection point, to consider an intersection as valid, the intersection becomes invalid when the query segment becomes too small to intersect more than one splat. Note how this happens in areas of high curvature, where splats have less coherence between one another

quite noisy. Additionally, underwater mapping using a multibeam sonar requires very accurate navigation estimates to align all the range scans into a single reference frame. When a lightweight low-cost underwater robot is used, the poor navigation data cause several double contours in unprocessed point sets. Having outliers and double contours generally creates a noisy splats representation with self-intersecting splats not amenable to the RANSAC intersection procedure[47] we use for the S-T cut case. This renders the S-T cut useless, and, consequently, we only test acoustic range data sets using the normalized cut method.

The first acoustic data set presents a point set obtained using a multibeam sonar scanning an underwater Mound (hill) rising at a depth from 40 to 27 m located near the harbor of Sant Feliu de Guíxols, on the Costa Brava of Catalonia, Spain (see Fig. 10). With the sonar set in a slanted orientation toward the hill, the point cloud was automatically obtained with an adaptive replanning strategy that uses stochastic trajectory optimization to reshape the nominal path to cope with the actual target structure perceived in real time during the exploration.[13] As it can be observed in Figure 10, despite the outliers and double contours (i.e., parts of the object that are doubled due to bad positioning of the sensor) present in the final retrieved data, the normalized cut is able to obtain a consistent surface. We will use this data set to observe the behavior of normal computation on a complex real-world example and its implications on surface reconstruction methods requiring per-point normals. Figure 11 presents the two commonly used approaches for normal computation, and the result of using them as input to the well-known Poisson surface reconstruction method.[38] In both cases, the normals were computed using PCA plane estimation on each point given their $K = 100$ neighbors (an empirical value set based on the amount of noise in the data). However, they differ in the orientation procedure. In Figure 11(a), we exploited the known position of the vehicle at the time of capturing the data to reorient the points. As you can observe, the orientation of the points is not coherent in some areas (black spots in the shaded model), which leads to a wrong

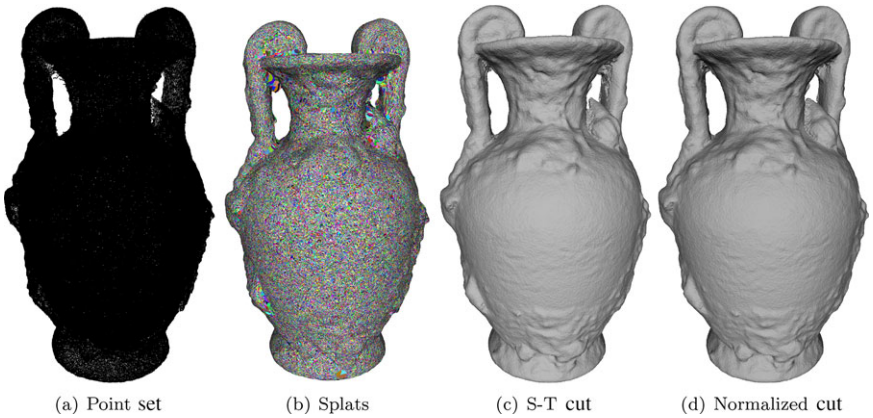


FIGURE 8 Examples of the methods' behavior when applied to a range scan data set. From left to right, input points, splats ($d = 1$), and the results of the S-T cut and normalized cut methods. Note how both methods retrieve similar surfaces

FIGURE 9 Deployment of Girona500 AUV

surface reconstruction in Figure 11(b). The more generic orientation propagation following a MST[23] is used to obtain the results in Figure 11(c). The more coherent orientation leads to an improved surface reconstruction [Fig. 11(d)]. Still, when compared to the results of our method in Figure 10(b), we can observe some off surfaces and unrealistic overhanging parts in the recovered scene. Additionally, note how the irregular sampling of this scanning methodology presents some artifacts in the Poisson reconstruction, resembling the linear stripes of the original individual scans, which are otherwise smoothed out with our approach. These results prove the superior performance of our approach.

The second acoustic data set corresponds to a profiling sonar survey of the interior of an underwater cave located in L'Escala, also on the Costa Brava. During the survey, the monobeam rotating sonar head was positioned orthogonally to the cave, so that a single scan provides a 360° view of its walls. Additionally, in this case the trajectory was optimized a posteriori through a SLAM approach.[86] Figure 12 shows the data set along with two close-up views allowing the understanding of the shape of the interior of the cave. Note that, regardless of the sparse sampling this data set poses, small details, such as the small tunnel visible in the upper part of Figure 12 (c), are faithfully recovered. Additionally, this example proves the usefulness of the methods

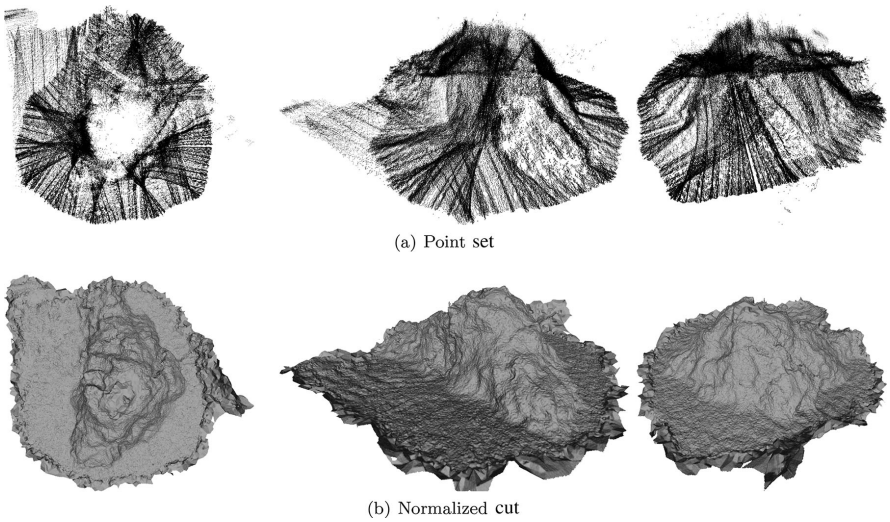


FIGURE 10 Underwater Mound retrieved with a multibeam survey (a) and the reconstructed surface using normalized cuts (b)

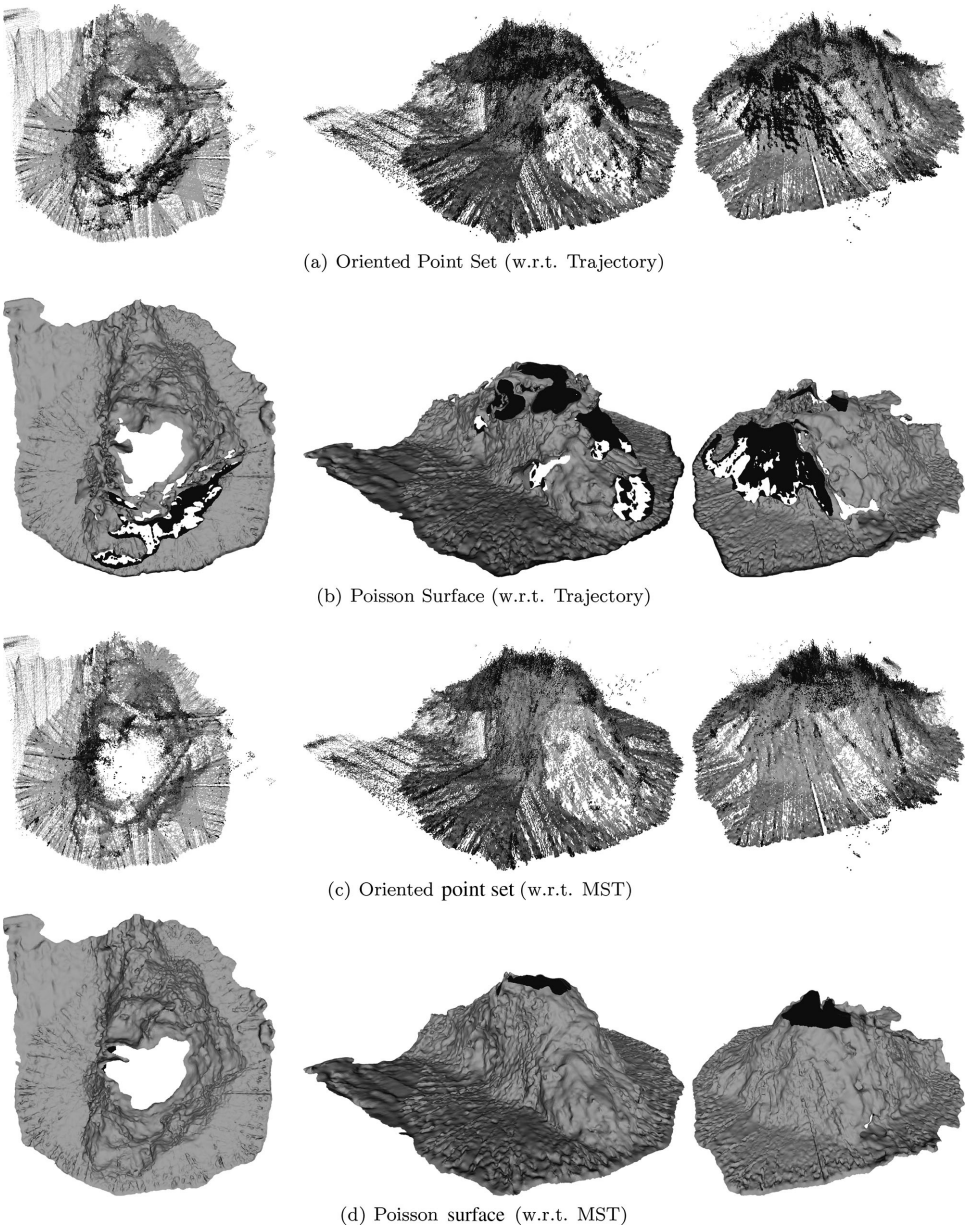


FIGURE 11 Normal estimation methods exploiting the information from the robot trajectory (a) and the MST propagation (c) and their consequences on Poisson surface reconstruction for the underwater Mound data set (b, d). Shadow casting is applied to the point sets to visualize the orientation of the normals, that is, black spots denote wrong orientation

presented with data sampled from confined environments, a hot topic in the underwater robotics community.[11,12]

Regarding optical underwater data sets, Figure 13 illustrates the Tour Eiffel data set acquired by the remotely operated vehicle (ROV) Victor6000 (Ifremer). The Tour Eiffel is an underwater hydrothermal vent located at about 1, 700m depth in the mid-Atlantic ridge, and which has been the objective of many science expeditions in the past decade.[87] Observe in Figure 13 (a) how the corrupted point set leads to a splat representation containing a large number of primitives that intersect with their neighbors. It should be noted that often ROVs rely on local navigation based on the visual feedback that the pilot receives from ROV's cameras; therefore, the navigation sensors carried by the robot are very limited. Even in this case, we are able to retrieve a perfectly manifold surface approximating the shape of this underwater chimney. However, we can see in Figures 13(c) and 13(d) how a larger part of the object is retrieved in the normalized cut case. This is due to the sparse sampling paired with high noise of the left-most part of the input point set, presented in Figure 13(b), which results in splats being more incoherent in this area. Thus, this area is not amenable to the inside/outside guess of the S-T cut method, dooming the global optimization to consider it as part of the outside of the shape and, hence, not reconstructed. Nevertheless, both methods accomplish a high level of detail.

We end the empirical evaluation of the methods with two data sets collected during the survey of La Lune shipwreck.[9] This shipwreck of the 17th century is an interesting archeological site located near the

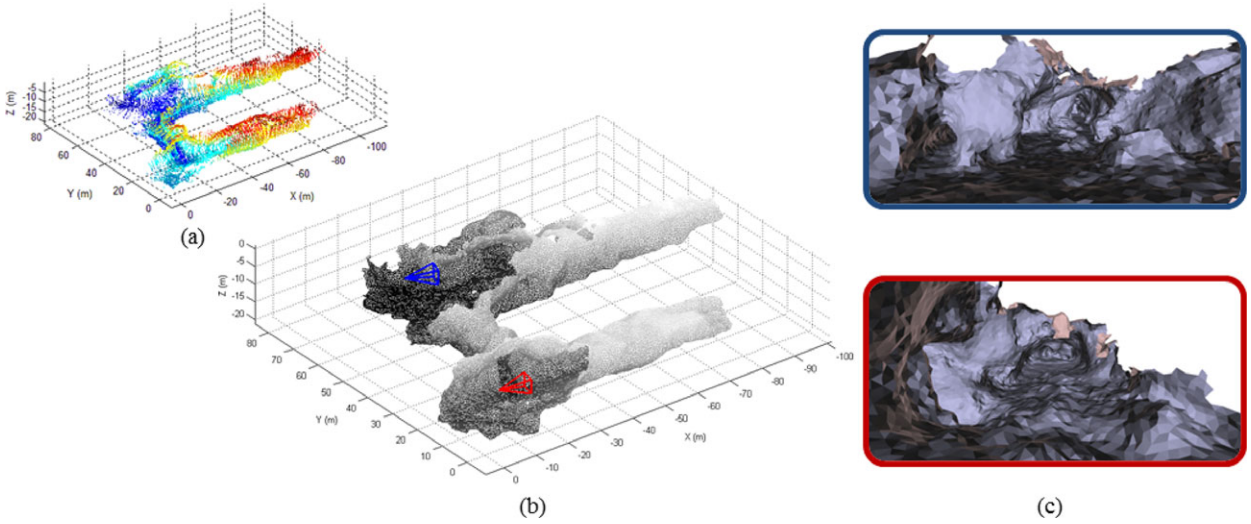


FIGURE 12 Profiling sonar survey of an underwater cave in (a). One can see the reconstruction obtained by the normalized cut algorithm in (b), with close-ups of the two main tunnels highlighted in (c). The view direction is also marked on the right part

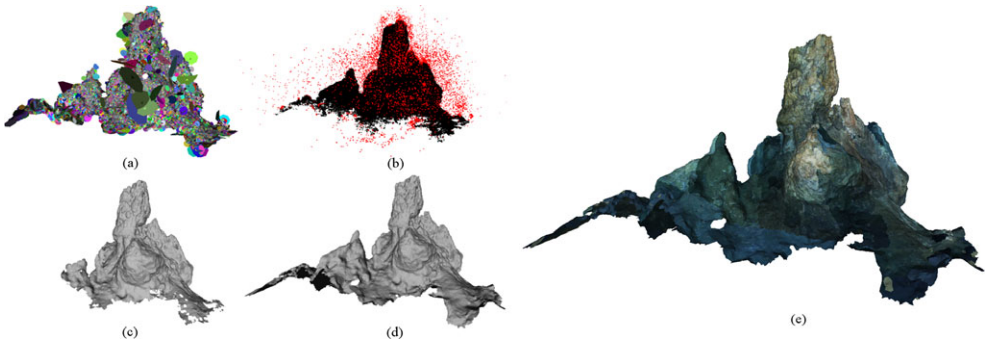


FIGURE 13 Results for the Tour Eiffel data set. In (a) we show the splats representation of the scene, in (c) the results of the S-T cut, and in (d) the normalized cut algorithm, along with its texture-mapped version in (e). Finally, we show in (b) the original point set with the gross outliers with respect to the surface highlighted in red to depict the level of corruption of this data set

coast of Toulon, France. The bad conditions during the original image acquisition (analog, interlaced, low-resolution, grayscale camera) contributed to the reconstructed point set containing a large set of outliers, noise, and also double contours. We present two data sets corresponding to in-detail explorations of small areas in Figures 14 and 15. In both cases, the shapes of the objects in the scene are indistinguishable either in the point set or in the splat representation [two first subfigures in the lower part of Fig. 14, Fig. 15(a), and 15(b)], but they are clearly visible in the surface reconstruction obtained with the normalized cut [third subfigure in lower part of Fig. 14 and Fig. 15(c)]. As previously noted, our robust intersection test fails when applied to highly nonconforming splats, which prevents the use of the S-T cut method in this case. Note also, in Figure 14, that we reach a far finer scale and, consequently, a more detailed model than using the method in Campos et al.,[48] which also tackles this data set.

As previously mentioned, the parameters used for achieving the results presented so far are listed in Table 1. Indeed, it may seem that there are a number of parameters involved in the computation of the methods. Thus, we want to briefly discuss the parameter selection in the presented cases, to provide a notion of appropriate parameter tuning depending on the properties of the data set. On the one hand, parameters σ, δ_r and β are the ones governing the S-T cut. The S-T cut only considers the first σ band, which defines the volume on our grid assigned to the UDF, and, thus, depends on the amount of noise in the data. This is the reason why Shallow Waters and Amphora data sets use a small $\sigma = 3$, whereas the noisier Tour Eiffel uses a larger sigma $= 7$. The δ_r parameter accounting for the RANSAC threshold has to be more precisely tuned depending on the noise in the data. Also, this parameter is defined in the units of the data, so it is easier to tune if the data are metric and we have known reference of the scale of the model. Note that the tested data sets have different scales, so the values listed on the table are just for reference and not directly comparable. Then, as already commented, and as can be seen in the table, the β value is not that relevant in this case and is set to $\beta = 4$ by default as suggested in Hornung and Kobbelt.[73] On the other hand, the normalized cut method uses σ, σ_2, k_σ, and β parameters. The first σ is defined as in the S-T case. Then, the second σ_2 band is just used for balancing the cut, so we want to make it larger but not so large so as to add complexity to the creation of the UDF and the cut. Thus, it is a trade-off between the correct balancing and the computational effort needed for the cut. The number of k_σ required to compute this second σ_2 band depends on the density, sampling rate, and amount of noise in the data. Dense and nicely

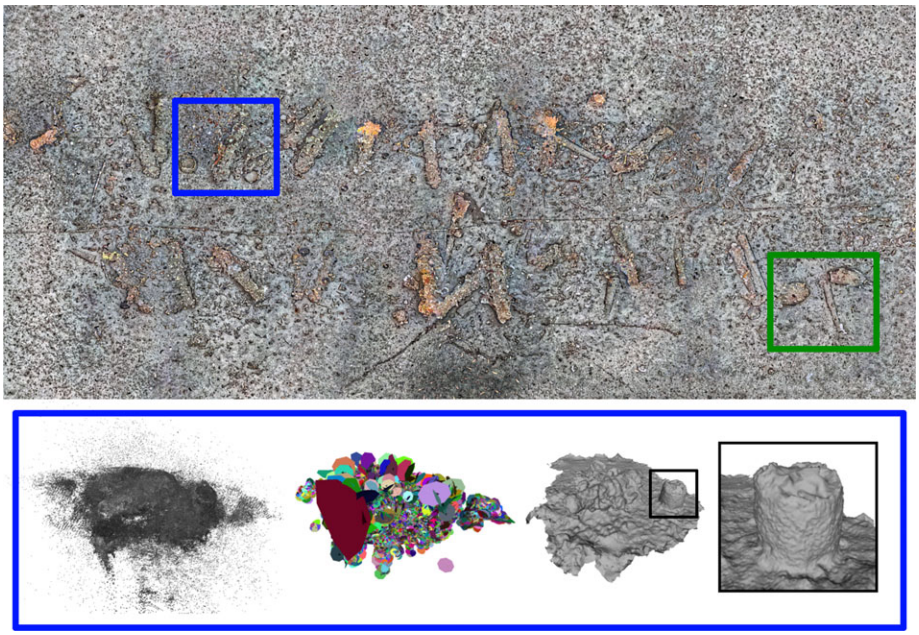


FIGURE 14 La Lune survey. On the top, we can see a mosaic of the shipwreck area,[9] where we marked the two areas surveyed in 3D. In blue, and depicted in the lower part of the figure, we find from left to right: the retrieved point set, the splats representation, and the reconstructed surface (with a close-up of a cauldron). Note that the large amount of noise and outliers poses the splat representation to contain a large set of self-intersecting primitives. Nevertheless, we are able to recover a reliable reconstruction, showing the cannon and two cauldrons, using our normalized cut method

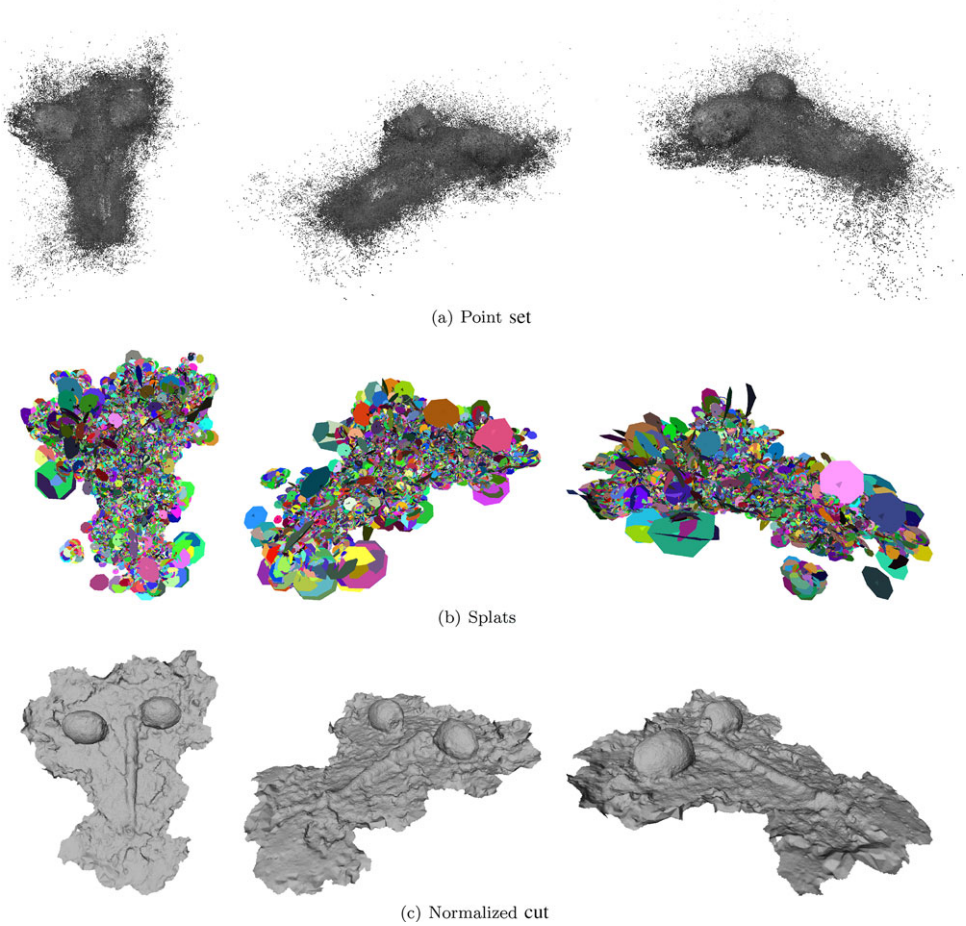


FIGURE 15 La Lune, second data set (marked in green in the upper part of Fig. 14). The point set in (a) and its splat representation in (b) show how this data set reproduces the same level of corruption of the first data set. The normalized cut surface (c) reveals two cauldrons located on either side of a cannon

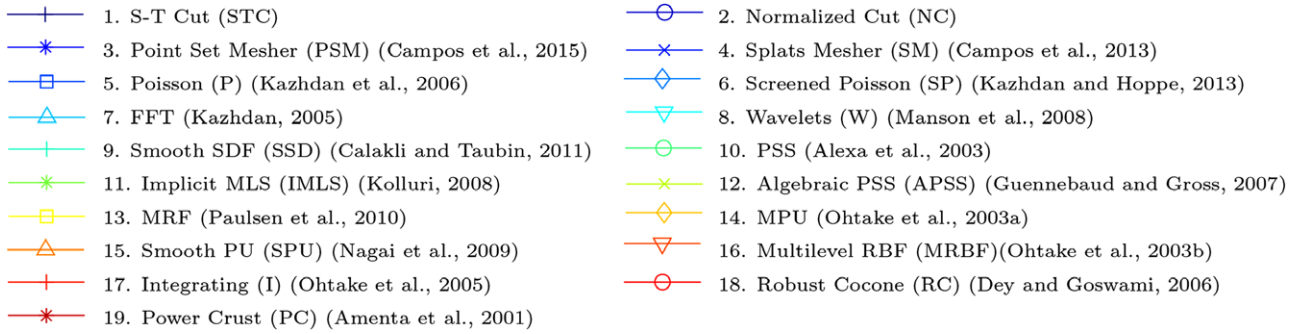


FIGURE 16 List of methods compared in the Results section, depicting for each algorithm its name, marker, color, the acronym used in the text, and its main reference

sampled data sets (e.g., Risu3 or Amphora data sets) use very few k_σ samples to compute the distance. On the contrary, noisier data sets (e.g., La Lune (1) data set) require a larger number of samples to be less sensitive to outliers. Then, in this case the β parameter is more sensitive and related to the noise in the data. To constrain the cut to pass by the minimum of the UDF we need to emphasize this minimum valley, and this is done by increasing the β parameter. The noisier the data set is, the larger this value. Finally, the α_r and α_d parameters are part of the surface mesher step and just tuned based on aesthetic reasons. Recall that, once the SDF has been reconstructed, we can extract a surface with different properties in terms of approximation quality, complexity, and shape regularity of the triangles in the mesh by just tuning these parameters. This is interesting, for instance, to create several meshes at different resolutions for the same SDF for visualization purposes.

Finally, we compare the behavior of our method against the state of the art. First, we apply a representative number of methods of the state of the art to the Shallow Water data set, presented in Figure 17(a). In Figure 16, one can see the list of methods evaluated, along with their acronym and their references. To simplify, when referring a method in the text, we will use the acronym form. Thus, to perform this test, we follow the approach of Kazhdan and Hoppe.[38] We randomly divide the input points into two equally sized sets, so that one of the sets is used for evaluation whereas the second is used for validation. In this way, we use the evaluation set as input to the algorithms and the validation set to compute the distances to this reconstructed surface. Figure 17 enables a qualitative comparison of the different methods in the state of the art, while also shows the mean distances obtained for each evaluated method [see the chart on Fig. 17(u)]. Given the fact that the Shallow Water data set describes a bounded surface, some of the results on Figure 17 present made-up parts as a consequence of the methods trying to find a watertight surface. Nevertheless, all mean distances are very similar. This is caused by the good sampling provided by the point set. The ones obtaining the worst measures are FFT, wavelets, and RC. On the contrary, the best results are obtained by SPoisson and PC. Our methods, STC and NC, compare favorably to the state of the art in this simple test. Note that we are not taking into account the manifoldness of the recovered surface, and, for instance, the SM method obtains a decent mean but the resulting surface is far from manifold [as shown in Fig. 7(b)]. Bear in mind that, since we are using real data, the input point sets contain some noise, and the tests measure how good the reconstruction is when compared to this noisy data. Thus, we evaluate the overall behavior of the method against the input data, but we cannot draw global conclusions as no ground truth is available.

To properly quantify the results provided by the methods proposed against more complex data sets, and with the proper ground truth referencing, we use the benchmark of Berger and colleagues.[88] While the authors provide a significant number of examples to test the algorithms against, we think the level of corruption present in these point sets is more similar to in-lab captures and far from that contained in data acquired in real-world environments, specially in participating media, such as underwater. The authors also provide tools to generate new simulations of scans, using a virtual laser scanner. Thus, these tools were used to generate a new set of 44 shapes by varying some specific parameters we found directly related to noise. More precisely, and following the nomenclature of the original reference, we modify the noise magnitude from 0 to 0.5, with increments of 0.05, and the laser's field of view from 2.5 to 10, with increments of 2.5. Figure 18 presents some examples of the corrupted point sets tested, so that their noise magnitude can be compared to those on Berger and colleagues.[88]

Notice that our evaluation extends that in the original article[88] not just in the level of corruption considered but also in the number of methods tested. It is also worth noting that, following the tendency of most methods in the state of the art, all of the methods in the original evaluation required per-point normals, which is not the case for the methods proposed in this article. In this direction, we also tested some methods not requiring the use of per-point normals. The names of the algorithms tested are presented in Figure 16, and they are separated in Figure 19(a) into those requiring normals, or otherwise working on raw point sets. For all the algorithms, we have respected the parameters recommended by the authors, when available, and tuned them when necessary to achieve a better reconstruction quality, based on visual examination of the results. We further dealt with each point set individually, that is we tuned the parameters to obtain the best result on each case instead of resorting to a fixed parameterization for all the point sets.

Given a reconstruction, the evaluation creates a bidirectional distance map from the recovered surface to a large densely sampled

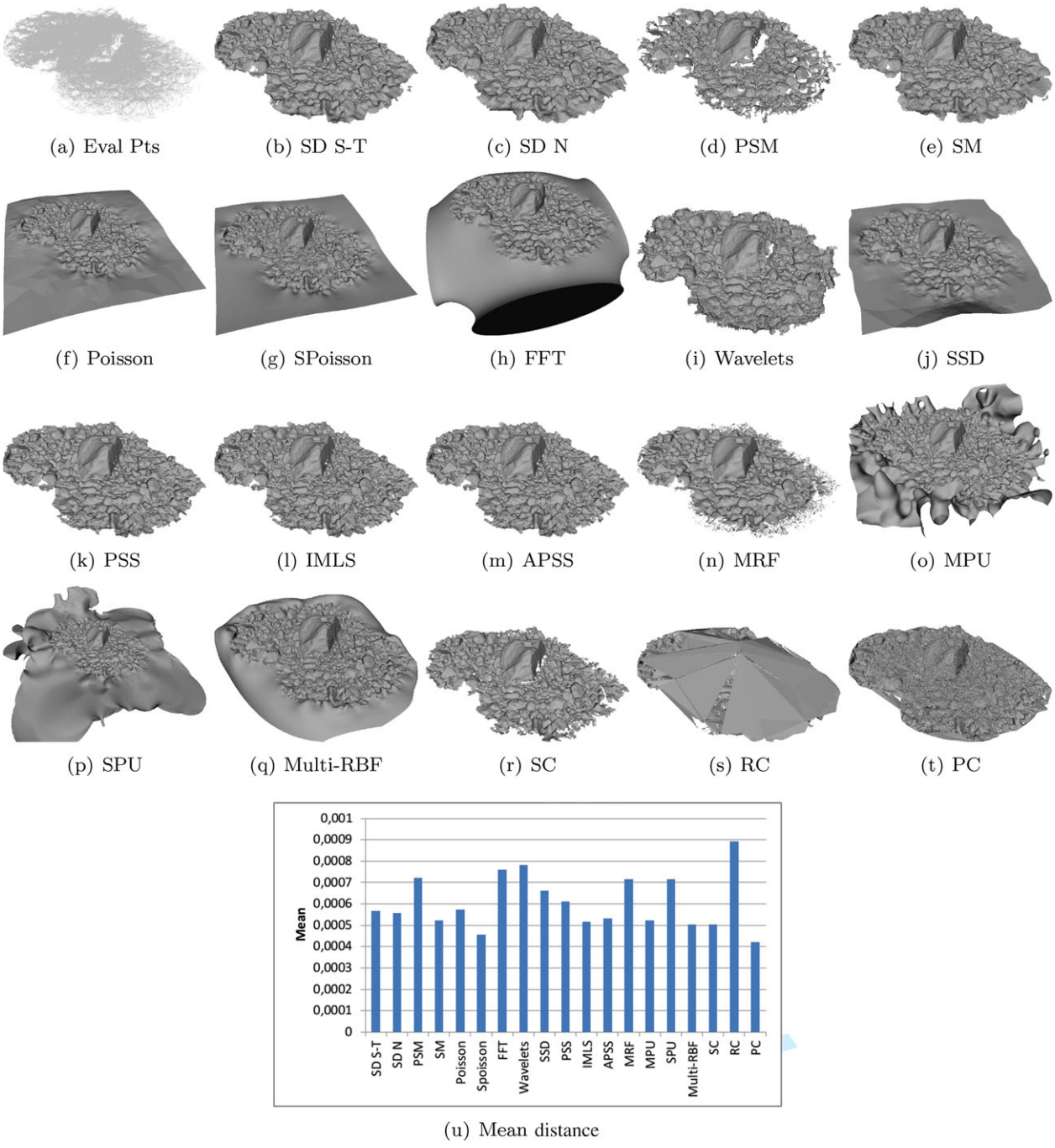
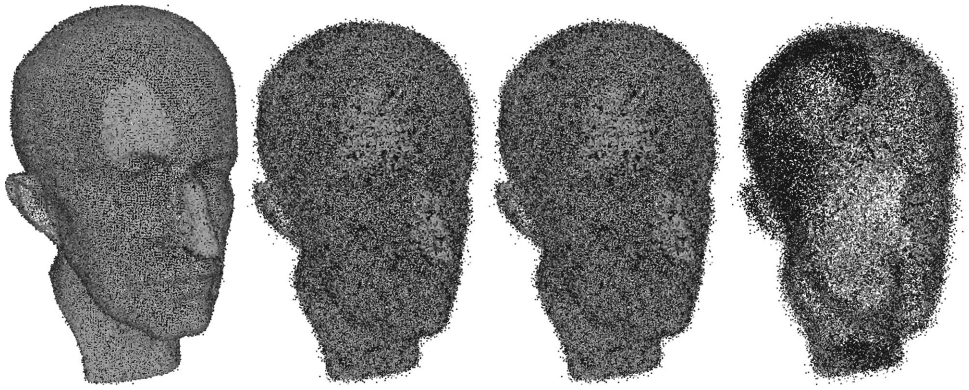


FIGURE 17 State-of-the-art evaluation using the Shallow Water data set (1,856,266 points). The evaluation set (928,133 points) is shown in (a), whereas the results for the different methods tested are shown in (b)–(t). In (u), we depict the mean distance from the validation point set to the reconstructed surfaces, where the values are with respect to the diagonal of the bounding box enclosing the original point set

version of the base reference shape. From this distance map, we compute the mean and maximum (i.e., Haussdorf) distance values. Additionally, also the mean divergence between normals at the surface and normals at the reference are computed. With this information, Figure 19(a) shows these results using box plots. Through these plots named error distribution plots in the original evaluation, one can feel the overall behavior of the methods under varying noise levels. Regarding mean/max errors, the MPU method is clearly the one obtaining the worst results and seems unable to handle large amounts of noise. Nevertheless, the SPU method, which is a broad smoothing on the primitives of the original MPU approach, obtains very good estimations. Additionally, the SC, RC, and PC methods, all working with raw point sets, do not achieve good results. The gradient-based methods (i.e., P, SP, FFT, W, and SSD), as well as the MRBF, all behave acceptably well. However, the SSD method attains a larger variability in this case. Additionally, due to its stiffness to the input points, the SP variant reproduces a box a little wider (i.e., more variable errors) than the original P method, since its known smoothing resolves the



(a) $noise = 0.2$, $fov = 2.5$ (b) $noise = 0.45$, $fov = 5$ (c) $noise = 0$, $fov = 7.5$ (d) $noise = 0.5$, $fov = 10$

FIGURE 18 Four sample point sets of the Max Planck shape, from a total of 44. Under each figure, we detail the level of noise magnitude (*noise*) and laser's field-of-view size (*fov*). Normals are used to apply shadow casting to the model

noise issue better. From the MLS methods (PSS, IMLS, and APSS), the one behaving the worst is the IMLS. Regarding PSM and SM methods, they both behave erratically, obtaining a wide box despite having a low median. On the contrary, our STC and NC methods behave favorably, achieving the best results in terms of variance in the plots, with NC obtaining values a bit larger for the Hausdorff distances. If we then focus on the mean angle deviation, our methods STC and NC, working on raw point sets, obtain an extremely good measure, comparable to those of P, W, PSS, APSS and SPU, all of which work with the additional knowledge of per-point normals. On the contrary, methods achieving larger variation in this error measure are PSM, SM, MPU, SC and PC.

On the other hand, Figure 19(b) plots the errors in a sorted manner, increasing the two parameters related to noise: from left to right for the noise magnitude and from top to bottom for the field of view. Contrary to the previous box plot representation, this sorted plot allows to detect the amount of noise that a method is able to attain. This is depicted with a large step increment of the errors (either mean or maximum), and we leave the larger values out of the graphic for clarity. Using these plots, we can clearly see how an increase in noise progressively degrades the results obtained, which is worsened at each incremental step of the laser's field of view. Surprisingly enough, we can detect how some specific configurations cause some of the reconstruction methods to fail for a given point set, to then obtain more favorable results in even noisier data sets. This is depicted in the peaks visible for some methods in the line charts, and I, FFT or SPU are some examples. For the smaller laser field-of-view values, the decrease in performance, that is, the increase in mean/max error values, tends to increase steadily for most of the methods (disregarding the mentioned peaks). This raising in the curves step matches the previous results for the above-presented error distribution plots: more variable error distribution plots grow faster in these line charts. Finally, another important aspect to observe is the noise level due to which some methods totally fail. This is shown with a large step in the error measures with values that get out of the chart. Nevertheless, we can observe that our methods, STC and NC, obtain the smallest step curves, always at the bottom of the graph and almost undistinguishable at this scale.

9 | CONCLUSIONS AND FUTURE WORK

We have presented two volumetric surface reconstruction methods based on minimum cuts on graphs. By making an analogy with a binary partitioning problem, we use the minimum cut along an unsigned distance function to promote its signing. This UDF is defined from the splat representation presented in a previous article[47] and is discretized in a tetrahedral grid adapted to the density of the input points. Merging the different contributions of the splats in a global view allows the mitigation of spurious splats that may remain near the surface, and extracting the surface as the zero isovalue in a well-defined volume results in the retrieved surface being manifold. Furthermore, and as opposed to most methods in the state of the art, both our proposals are designed to handle bounded surfaces. Additionally, we outperform the state of the art by not requiring any other additional information than the input points to work, which is clearly an advantage when using a lightweight AUV carrying inexpensive navigation sensors.

While sharing the distance function definition, we divided the methods according to their graph cutting technique. In both cases, the base graph representation is derived from the adaptive grid storing the UDF. First, we have presented the S-T cut method, which needs an initial guess for inside/outside for some of the vertices in the graph to then propagate these labels following the smooth weights governed by the UDF. Second, we have introduced the normalized cut method, which only uses the smooth weights to define a minimum cut balancing the cost of the two volumes after the partition. In both cases, we have shown with many examples that the retrieved surfaces are similar.

When comparing both proposals, we have proven that the normalized cut method is more versatile, as it does not require any additional knowledge such as the inside/outside guess for the S-T cut case. We have noticed that when using normalized cut, we can overcome the limitation posed by noisy self-intersecting splats that prevented the method by Campos et al.[47] to provide a coherent surface. However, this type of data is still not solvable using the S-T cut method, as it depends on the same RANSAC-based robust intersection detection procedure than the original splats mesher method did. On the contrary, parameter tuning is more sensitive for normalized cut than for S-T cut (as depicted in Table 1).

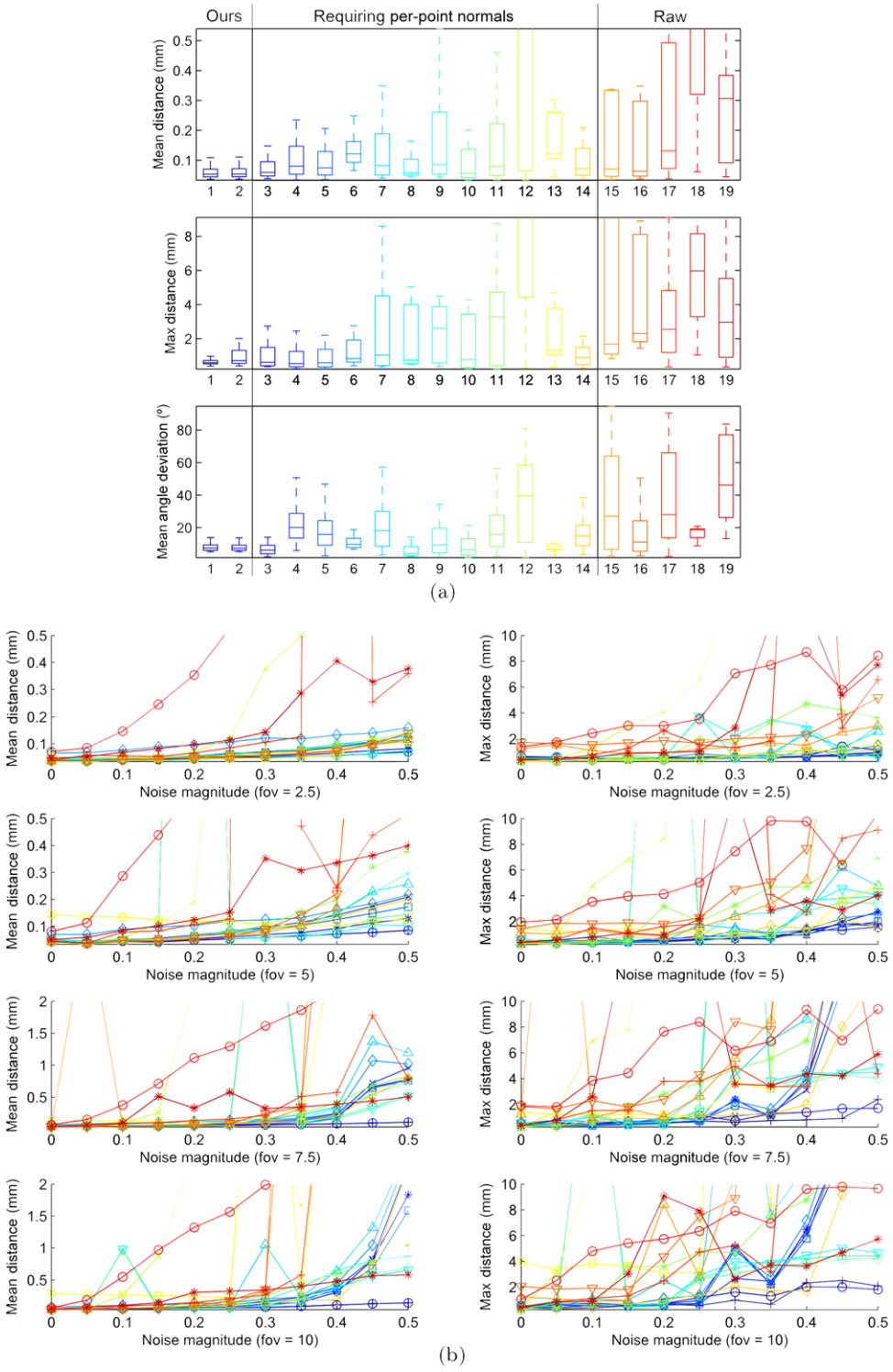


FIGURE 19 Noise test of the Max Planckdata set, using the benchmark of Berger et al.[88] (a) Box plot form of the mean and maximum (i.e., Hausdorff) distances, and the mean angle deviation for each method when applied to 44 synthetic scans with varying noise scales. (b) Results using incremental values for the parameters of the virtual scanner related to the noise

We also performed a quantitative evaluation of the methods against the state of the art. This survey brought into relief the resilience to noise that our methods provide when compared to other algorithms, outperforming even those requiring per-point normals. Additionally, and to the best of our knowledge, this evaluation is the first to consider such large levels of noise.

A drawback of both methods when compared to those in the state of the art is their increase in memory requirements. Despite the fact that we use several data structures to speed up the processing of the intensive parts of the algorithms (adaptive tetrahedral grid, octree, AABB trees, etc.), in both cases the storage of the distance function in a tetrahedralization represents a large amount of memory.

Furthermore, there is an inherent redundancy of splats in our representation, inherited from the redundancy requirements of the algorithm in Campos and co-workers.[47] In our case, redundant splats could be simplified to alleviate the computational cost of computing the UDF. Thus, simplification strategies for the splat representations are left as future work. Moreover, the methods suffer from data-dependant parameter tuning. This should be alleviated in future work by automatically setting them based on measures such as noise levels and data sampling rate or density. Note, however, that this is not trivial, as noise is difficult to quantify in the presence of outliers, and neither noise nor sampling need to be uniform in a given data set.

ACKNOWLEDGMENTS

The authors wish to thank the AIM@SHAPE consortium (the ISTI-CNR Visual Computing Laboratory) for providing the range scanned models used in this paper.

This work was partially funded through MINECO (grant number CTM2013-46718-R) and the EU VII Framework Programme as part of the Eurofleets2 (grant number FP7-INF-2012-312762) and Robocademy (grant number FP7-PEOPLE-2013-ITN-608096) projects.

ORCID


Ricard Campos http://orcid.org/0000-0003-4718-468X

REFERENCES

1. Roman C, Singh H. A self-consistent bathymetric mapping algorithm. *J Field Robot*. 2007;24(1–2):23–50.
2. Johnson-Roberson M, Pizarro O, Willams S. Towards large scale optical and acoustic sensor integration for visualization. In: *OCEANS 2009-EUROPE*. Bremen, Germany: IEEE; 2009:1–4.
3. Yoerger DR, Kelley DS, Delaney JR. Fine-scale three-dimensional mapping of a deep-sea hydrothermal vent site using the jason rov system. *Int J Robot Res*. 2000;19(11):1000–1014.
4. Barkby S, Williams S, Pizarro O, Jakuba M. Bathymetric particle filter SLAM using trajectory maps. *Int J Robot Res*. 2012;31(12):1409–1430.
5. Pizarro O, Eustice R, Singh H. Large area 3-D reconstructions from underwater optical surveys. *IEEE J Ocean Eng*. 2009;34(2):150–169.
6. Nicosevici T, Gracias N, Negahdaripour S, Garcia R. Efficient three-dimensional scene modeling and mosaicing. *J Field Robot*. 2009;26:759–788.
7. Johnson-Roberson M, Pizarro O, Williams SB, Mahon I. Generation and visualization of large-scale three-dimensional reconstructions from underwater robotic surveys. *J Field Robot*. 2010;27(1):21–51.
8. Bryson M, Johnson-Roberson M, Pizarro O, Williams S. Colour-consistent structure-from-motion models using underwater imagery. In: Roy N, Newman P and Srinivasa S, eds. *Robotics: Science and Systems*. Cambridge, Massachusetts: MIT Press; 2013:33–40.
9. Gracias N, Ridao P, Garcia R, et al. Mapping the moon: Using a lightweight AUV to survey the site of the 17th century ship "La Lune." In: *OCEANS – Bergen, 2013 MTS/IEEE*. Bergen, Norway: IEEE; 2013:1–8.
10. Bülow H, Birk A. Spectral registration of noisy sonar data for underwater 3D mapping. *Auton Robot*. 2011;30(3):307–331.
11. am Ende BA. 3D mapping of underwater caves. *IEEE Comput Graph*. 2001;21(2):14–20.
12. Fairfield N, Kantor G, Wettergreen D. Real-time slam with octree evidence grids for exploration in underwater tunnels. *J Field Robot*. 2007;24(1–2):03–21.
13. Galceran E, Campos R, Palomeras N, Ribas D, Carreras M, Ridao P. Coverage path planning with real-time replanning and surface reconstruction for inspection of three-dimensional underwater structures using autonomous underwater vehicles. *J Field Robot*, 2014;32(7):952–983.
14. Amenta N, Bern M. Surface reconstruction by voronoi filtering. In: *14th Annual Symposium on Computational Geometry*, SCG '98. New York, NY: ACM; 1998:39–48.
15. Amenta N, Choi S, Dey TK, Leekha N. A simple algorithm for homeomorphic surface reconstruction. In: *16th Annual Symposium on Computational Geometry*, SCG '00. New York, NY: ACM; 2000:213–222.
16. Bernardini F, Mittleman J, Rushmeier H, Silva C, Taubin G. The ball-pivoting algorithm for surface reconstruction. *IEEE Trans Vis Comput Graph*. 1999;5(4):349–359.
17. Cohen-Steiner D, Da F. A greedy Delaunay-based surface reconstruction algorithm. *Visual Comput: Int J Comput Graphic Arch*. 2004;20(1):4–16.
18. Ohtake Y, Belyaev A, Seidel H-P. An integrating approach to meshing scattered point data. In: *ACM Symposium on Solid and Physical Modeling*, SPM '05. New York, NY: ACM; 2005:61–69.
19. Amenta N, Choi S, Kolluri R. The power crust. In: *6th ACM Symposium on Solid Modeling and Applications*, SMA '01. New York, NY: ACM; 2001:249–266.
20. Dey TK, Goswami S. Provable surface reconstruction from noisy samples. *Comput Geom Theor Appl*. 2006;35(1):124–141.
21. Lorensen WE, Cline HE. Marching cubes: a high resolution 3D surface construction algorithm. *SIGGRAPH Comput Graph*. 1987;21:163–169.
22. Boissonnat J-D, Oudot S. Provably good sampling and meshing of surfaces. *Graph Models*. 2005;67:405–451.
23. Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W. Surface reconstruction from unorganized points. *SIGGRAPH Comput Graph*. 1992;26(2):71–78.
24. Paulsen RR, Baerentzen JA, Larsen R. Markov random field surface reconstruction. *IEEE Trans Vis Comput Graph*. 2010;16(4):636–646.
25. Carr JC, Beatson RK, Cherrie JB, et al. Reconstruction and representation of 3D objects with radial basis functions. In: *28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01. New York, NY: ACM; 2001:67–76.
26. Ohtake Y, Belyaev A, Alexa M, Turk G, Seidel H-P. Multi-level partition of unity implicits. *ACM Trans Graph*. 2003;22(3):463–470.
27. de Berg M, van Kreveld M, Overmars M, Schwarzkopf OC. *Quadtrees*. Berlin: Springer; 2008:307–322.
28. Ohtake Y, Belyaev A, Seidel H-P. A multi-scale approach to 3D scattered data interpolation with compactly supported basis functions. In: *Proceedings of the Shape Modeling International*. Washington, DC: IEEE Computer Society; 2003:153.
29. Calakli F, Taubin G. SSD: Smooth signed distance surface reconstruction. *Comput Graph Forum*. 2011;30(7):1993–2002.
30. Alexa M, Behr J, Cohen-Or D, Fleishman S, Levin D, Silva TC. Computing and rendering point set surfaces. *IEEE Ttans Visual Comput Graph*. 2003;9(1):3–15.
31. Guennebaud G, Gross M. Algebraic point set surfaces. *ACM Trans Graph*. 2007;26(3):23.1–23.9.
32. Curless B, Levoy M. A volumetric method for building complex models from range images. In: *23rd Annual Conference on Computer*

Graphics and Interactive Techniques, SIGGRAPH '96. New York, NY: ACM; 1996:303–312.

33. Fuhrmann S, Goesele M. Fusion of depth maps with multiple scales. *ACM Trans Graph*. 2011;30(6):148:1–148:8.
34. Newcombe RA, Izadi S, Hilliges O, et al. Kinectfusion: real-time dense surface mapping and tracking. In: *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11. Washington, DC: IEEE Computer Society; 2011:127–136.
35. Whelan T, Kaess M, Johannsson H, Fallon M, Leonard JJ, McDonald J. Real-time large-scale dense rgb-d slam with volumetric fusion. *Int J Robot Res*. 2015;34(4-5):598–626.
36. Kazhdan M. Reconstruction of solid models from oriented point sets. In: *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. Aire-la-Ville, Switzerland: Eurographics Association; 2005:73–82.
37. Kazhdan M, Bolitho M, Hoppe H. Poisson surface reconstruction. In: *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '06. Aire-la-Ville, Switzerland: Eurographics Association; 2006:61–70.
38. Kazhdan M, Hoppe H. Screened poisson surface reconstruction. *ACM Ttans Graph*. 2013;32(3):29:1–29:13.
39. Manson J, Petrova G, Schaefer S. Streaming surface reconstruction using wavelets. *Comput Graph Forum (SGP)*. 2008;27(5):1411–1420.
40. Mitra NJ, Nguyen A, Guibas L. Estimating surface normals in noisy point cloud data. In: *Int J Comput Geom Appl*. 2004;14(4-5):261–276.
41. Dey TK, Sun J. An adaptive MLS surface for reconstruction with guarantees. In: *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '05. Aire-la-Ville, Switzerland: Eurographics Association; 2005;43–52.
42. Alliez P, Cohen-Steiner D, Tong Y, Desbrun M. Voronoi-based variational reconstruction of unoriented point sets. In: *5th Eurographics Symposium on Geometry Processing*. Aire-la-Ville, Switzerland: Eurographics Association; 2007:39–48.
43. Li B, Schnabel R, Klein R, Cheng Z, Dang G, Shiyao J. Robust normal estimation for point clouds with sharp features. *Comput Graph*. 2010;34(2):94–106.
44. Hornung A, Kobbelt L. Robust reconstruction of watertight 3D models from non-uniformly sampled point clouds without normal information. In: *4th Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP'06. Aire-la-Ville, Switzerland: Eurographics Association; 2006:41–50.
45. Mullen P, Goes FD, Desbrun M, Cohen-Steiner D, Alliez P. Signing the unsigned: robust surface reconstruction from raw pointsets. *Comput Graph Forum*. 2010;29(5):1733–1741.
46. Giraudot S, Cohen-Steiner D, Alliez P. Noise-adaptive shape reconstruction from raw point sets. *Comput Graph Forum*. 2013;32(5):229–238.
47. Campos R, Garcia R, Alliez P, Yvinec M. Splat-based surface reconstruction from defect-laden point sets. *Graph Models*. 2013;75(6):346–361.
48. Campos R, Garcia R, Alliez P, Yvinec M. A surface reconstruction method for in-detail underwater 3D optical mapping. *Int J Robot Res*. 2015;34(1):64–89.
49. Kolluri R, Shewchuk JR, O'Brien JF. Spectral surface reconstruction from noisy point clouds. In: *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04. New York, NY: ACM; 2004:11–21.
50. Labatut P, Pons J-P, Keriven R. Efficient multi-view reconstruction of large-scale scenes using interest points, Delaunay triangulation and graph cuts. In: *2007 IEEE 11th International Conference on Computer Vision*. Rio de Janeiro, Brazil: IEEE; 2007:504–511.
51. Labatut P, Pons JP, Keriven R. Robust and efficient surface reconstruction from range data. *Comput Graph Forum*. 2009;28(8):2275–2290.
52. Jancosek M, Pajdla T. Hallucination-free multi-view stereo. In: *11th European Conference on Trends and Topics in Computer Vision*, ECCV'10. Berlin: Springer-Verlag; 2012:184–196.
53. Ericson C. *Real-Time Collision Detection*. Boca Raton, FL: CRC Press; 2004.
54. Fischler MA, Bolles RC. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun ACM*. 1981;24(6):381–395.
55. Cazals F, Pouget M. Estimating differential quantities using polynomial fitting of osculating jets. In: *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '03. Aire-la-Ville, Switzerland: Eurographics Association; 2003:177–187.
56. Kolluri R. Provably good moving least squares. *ACM Trans Algorithms*. 2008;4:18:1–18:25.
57. de Berg M, van Kreveld M, Overmars M, Schwarzkopf OC. *Orthogonal Range Searching*. Berlin: Springer; 2008:95–120.
58. Alliez P, Cohen-Steiner D, Yvinec M, Desbrun M. Variational tetrahedral meshing. *ACM Trans Graph*. 2005;24(3):617–625.
59. Tournois J, Wormser C, Alliez P, Desbrun M. Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans Graph*. 2009;28(3):75:1–75:9.
60. Greig DM, Porteous BT, Seheult AH. Exact maximum a posteriori estimation for binary images. *J R Stat Soc B Met*. 1989;51(2):271–279.
61. Roy S, Cox IJ. A maximum-flow formulation of the n-camera stereo correspondence problem. In: *Sixth International Conference on Computer Vision*. Bombay, India: IEEE; 1998:492–499.
62. Kolmogorov V, Zabih R. Computing visual correspondence with occlusions using graph cuts. In: *IEEE International Conference on Computer Vision (ICCV)*, vol. 2. Vancouver, BC, Canada: IEEE; 2001:508–515.
63. Kolmogorov V, Zabih R. Multi-camera scene reconstruction via graph cuts. In: *7th European Conference on Computer Vision (ECCV)*, ECCV '02. London, UK: Springer-Verlag; 2002:82–96.
64. Kwatra V, Schodl A, Essa I, Turk G, Bobick A. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans Graph, SIGGRAPH 2003*. 2003;22(3):277–286.
65. Rother C, Kolmogorov V, Blake A. "GrabCut": interactive foreground extraction using iterated graph cuts. *ACM Trans Graph*. 2004;23(3):309–314.
66. Paris S, Sillion FX, Quan L. A surface reconstruction method using global graph cut optimization. *Int J Comput Vis*. 2006;66(2):141–161.
67. Lempitsky V, Boykov Y. Global optimization for shape fitting. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Minneapolis, MN, USA: IEEE; 2007:1–8.
68. Hiep VH, Keriven R, Labatut P, Pons J-P. Towards high-resolution large-scale multi-view stereo. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Miami, FL, USA: IEEE; 2009:1430–1437.
69. Jancosek M, Pajdla T. Multi-view reconstruction preserving weakly-supported surfaces. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Colorado Springs, CO, USA: IEEE; 2011:3121–3128.
70. Ford L, Fulkerson D. *Flows in networks*. Princeton, NJ: Princeton University Press; 1962.
71. Kolmogorov V, Zabin R. What energy functions can be minimized via graph cuts? *IEEE Trans Pattern Anal*. 2004;26(2):147–159.

72. Boykov Y, Kolmogorov V. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans Pattern Anal (PAMI)*. 2004;26(9):1124–1137.
73. Hornung A, Kobbelt L. Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1. New York, NY, USA: IEEE; 2006:503–510.
74. Shi J, Malik J. Normalized cuts and image segmentation. *IEEE Trans Pattern Anal*. 2000;22(8):888–905.
75. Malik J, Belongie S, Leung T, Shi J. Contour and texture analysis for image segmentation. *Int J Comput Vis*. 2001;43(1):7–27.
76. Cour T, Benezit F, Shi J. Spectral segmentation with multiscale graph decomposition. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2. San Diego, CA, USA: IEEE; 2005:1124–1131.
77. Eriksson A, Olsson C, Kahl F. Normalized cuts revisited: A reformulation for segmentation with linear grouping constraints. In: *11th IEEE International Conference on Computer Vision (ICCV)*. Rio de Janeiro, Brazil: IEEE; 2007:1–8.
78. Kim TH, Lee KM, Lee SU. Learning full pairwise affinities for spectral segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. San Francisco, CA, USA: IEEE; 2010:2101–2108.
79. Maji S, Vishnoi NK, Malik J. Biased normalized cuts. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Colorado Springs, CO, USA: IEEE; 2011;0:2057–2064.
80. Maila M, Shi J. A random walks view of spectral segmentation. In: AI and STATISTICS (AISTATS). 2001.
81. Rahimi A, Recht B. Clustering with normalized cuts is clustering with a hyperplane. In: *ECCV 2004 Workshop on Statistical Learning in Computer Vision*. Prague, Czech Republic: 2004.
82. Wu Z, Leahy R. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Trans Pattern Anal*. 1993;15(11):1101–1113.
83. Bab-Hadiashar A, Suter D. Robust segmentation of visual data using ranked unbiased scale estimate. *Robotica*. 1999;17(6):649–660.
84. de Moustier C. Beyond bathymetry: Mapping acoustic backscattering from the deep seafloor with sea beam. *J Acoust Soc Am*. 1986;79(2):316–331.
85. Mayer LA, Calder BR, Schmidt JS, Malzone C. Providing the third dimension: high-resolution multibeam sonar as a tool for archaeological investigations-an example from the D-day beaches of Normandy. In: *U.S. Hydrographic Conference (US HYDRO)*. Biloxi, MS, USA: The Hydrographic Society of America; 2003:0–16.
86. Mallios A, Ridao P, Ribas D, Hernández E. Scan matching SLAM in underwater environments. *Auton Robot*. 2013;35(1):1–20.
87. Langmuir C, Humphris S, Fornari D, et al. Hydrothermal vents near a mantle hot spot: the lucky strike vent field at 37n on the mid-atlantic ridge. *Earth Planet Sci Lett*. 1997;148(1–2):69–91.
88. Berger M, Levine J, Nonato L, Taubin G, Silva C. A benchmark for surface reconstruction. *ACM Trans Graph*. 2013;32(2):20:1–20:17.

How to cite this article: Campos R, Garcia R. Surface meshing of underwater maps from highly defective point sets. *J Field Robotics*. 2018;35:491–515. https://doi.org/10.1002/rob.21758