Surface reconstruction methods for the recovery of 3D models from underwater interest areas

Ricard Campos, Rafael Garcia and Tudor Nicosevici
Computer Vision and Robotics Group
University of Girona, 17071, Spain
Email: {rcampos, rafa, tudor}@eia.udg.edu

***Abstract*—3D models of seafloor interest areas having a strong 3D relief are a rich source of information for scientists. In the recent years, the computer vision community has been developing techniques to achieve the discrete reconstruction of these interest areas from a set of images. However, 3D reconstruction methods output a 3D point cloud as a representation of the shape of the observed object/area. This representation lacks connectivity information, and a surface that describes the underlying object is needed for both performing computations on the object and to achieve its correct visualization. In this article we aim to survey the State-of-the-Art of the problem of reconstructing the surface of an object represented by a cloud of points. This set of points is assumed to be the result of a Computer Vision based 3D reconstruction pipeline.**

I. Introduction

Underwater imagery is a very useful source of scientific data. For this reason, 2D stitching of several images into a photo-mosaic is an effective tool that helps in understanding the overall appearance of the seafloor. However, most of the areas of interest for scientists are located in zones of the seabed with 3D relief. Therefore, a more meaningful representation for those specific interest areas is needed to provide scientists with useful data. A usual approach to obtain a 3D representation of the seabed is the use of acoustic sensors, i.e. multibeam or sidescan sonars, but although this kind of tools may provide high-resolution models (at high frequencies), their resolution is lower than that obtained by optical cameras. Moreover, their result is usually a 2.5D height map. When locally observing specific areas, it is interesting to obtain a high resolution 3D model of the scene that captures fine details, being able to represent concavities. As an alternative, cameras are inexpensive and reliable sensors for retrieving this sort of information.

In the last years, the computer vision community has proposed several methods to deal with 3D reconstruction of an object from different views, most of them based on texture features that can be extracted from the images. However, their output is a cloud of 3D points, and only a few methods in this community are intended to address the surface reconstruction problem. The surface reconstruction problem consists in estimating the unknown surface that a set of 3D points describes, in the form of a triangle mesh. This representation provides connectivity information between points, and is more suitable for visualization purposes (e.g. textured mesh), allowing further processing or inspection to be applied on the object (e.g. to carry out measurements of length, area or volume).

On the other hand, the computer graphics community has been dealing with the problem of reconstructing a surface (mesh) from a set of 3D points that (theoretically) lay on a surface. Over the last few years, such points were mostly obtained from a laser scanner. Most of the meshing methods available in the literature appear to be generic enough to deal with any kind of 3D point clouds, provided that they present some properties. We will see in this article that point clouds obtained by a computer vision 3D reconstruction pipeline may not fulfill these properties, which will cause difficulties in the surface reconstruction process. Nevertheless, the output of the 3D reconstruction pipeline, as well as the input images, provide more information than just the point cloud (e.g., the camera positions at each view or the texture information contained in the image). This additional information could be helpful if added to the surface reconstruction pipeline, and few methods in the computer vision community have started working on this direction [1].

This article is organized as follows: In section II we perform a quick overview of the object modeling pipeline that we will assume, where we differentiate between the problems of estimation of a 3D point cloud and surface reconstruction. Secondly, we review in section III the different families of methods that exist in the State-of-the-Art to deal with the surface reconstruction problem. Next, section IV discusses the problems that arise when using the 3D points obtained with a computer vision reconstruction pipeline as the input of a surface reconstruction algorithm from the State-of-the-Art, as well as some guidelines on which method to use depending on our purposes and the properties of our point cloud. In section V we present and discuss some results obtained with State-of-the-Art surface reconstruction methods applied to 3D reconstructions obtained from underwater datasets. Finally, in section VI we will discuss the conclusions of this work.

II. Automatic Object Modeling

As previously discussed, in this article we will focus on having the images as the primary (and possibly unique) sensor used to obtain the final 3D model of a local seafloor interest area. The processing pipeline that we are assuming is two-fold: first, the input images are processed using a computer vision-based 3D reconstruction algorithm to obtain a point cloud representing the shape of the observed object/area, then, a surface reconstruction method is needed to recover the surface of the object in the form of a triangle mesh.

The problem of recovering the shape of an object from a set of images has been discussed in the computer vision community for many years, and nowadays continues being a hot research topic. There exist a wide variety of strategies to address the problem, but the main stream research lines can be divided according to whether they use Multiple-View Stereo or Structure from Motion methods. Multiple-View Stereo techniques rely on the previous knowledge of the relative positioning and orientation of each camera at each frame (by means of a calibration method) and where a point has been seen in different images (correspondence problem) to reconstruct its 3D coordinates. The 3D point is computed as the intersection of the viewing rays that emanate from each camera center and pass through the images projections of the point. Moreover, Structure from Motion approaches follow more or less the same pipeline, but they do not assume the cameras poses to be known, and they are computed as another step of the method. As stated, in both cases the results are in the form of a 3D point cloud.

On the other side, the Surface Reconstruction problem has been mainly addressed in the past by the computer graphics community. More precisely, this topic is known as *Surface Reconstruction from Unorganized Points*, since the required input is an unstructured 3D point cloud where the points are supposed to have been measured on the surface we want to recover. In this sense, the existing algorithms tend to be as generic as possible, i. e., they use the input points as the only source of information required in the reconstruction process. This assumption has both advantages and disadvantages. On one hand, this makes the methods theoretically suitable for new types of inputs, like the one we are supposing in this article. On the other hand, given a set of 3D points the surface that wraps them is not unique, and some conditions or properties have to be supposed on both the input points and the final surface in order to obtain a unique result. These properties/restrictions will cause problems with our 3D points, given that the majority of them cannot be easily assumed on the kind of data we are dealing with. These topics will be discussed in section IV.

III. Classification of Surface Reconstruction Methods

As stated above, the problem of computing the underlying surface from a set of unorganized 3D points was historically treated by the computer graphics community, and it is still a hot research topic. The methods proposed by this community can be classified by different criteria depending on their methodology, and this classification has been defined in different ways depending on the author. We will group the existing methods based on common characteristics and we will present in more detail the most representative algorithms for each section. Furthermore, in the very recent years some algorithms from the computer vision literature are including additional information from the 3D reconstruction pipeline into the surface reconstruction process, so we will also review some of them.

Therefore, the proposed classification is organized as follows:

- **Implicit Surfaces Methods:** These methods divide the working space regularly into disjoint cells called *voxels*, and try to find a surface defined in implicit form on this discretized space. Taking into account the way the methodology required to get this implicit surface, these methods can be further classified into: *global approximants* or *local approximants*.
- **Delaunay Methods:** These methods use the structure and/or the properties of the Delaunay Triangulation [2], and its dual, the Voronoi Diagram, to recover the surface. The methodologies followed by this methods can be further divided into four sub-groups [3]: *Restricted Delaunay*, *In/Out Labeling*, *Empty Balls* and *Tangent Planes*.
- **Computer Vision Aided Methods:** By adapting/upgrading some computer graphics techniques, these algorithms use the additional information provided by the computer vision 3D reconstruction pipeline to improve the results of the surface reconstruction process.

A. Implicit Surfaces Methods

This family of methods rely on the extraction of some implicit function on a voxel grid defined by the 3D input point samples. This implicit function defines the surface all around our working space, in the form $f(x, y, z) = d$, where d normally indicates the distance from the voxel (x, y, z) to the surface. Obviously, this is not the surface representation we require, so, as a final step, all these methods rely on an isosurface extractor method (like Marching Cubes [4]) to extract the surface in the form of a triangle mesh from its implicit representation. As stated, these methods can be classified according to whether they are *global approximants* or *local approximants*.

1) Global Approximants: These approaches are based on defining some kind of distance field, the zero-set of which is the surface we want to compute. This distance field is computed by directly using the input points, and some preliminary structure that can be derived from them, as an approximation of the surface.

This kind of methods was pioneered by Hoppe *et al.* in [5]. Hoppe's method computes a tangent plane for each point, using Principal Component Analysis (PCA) on a predefined k-neighborhood, and uses these local planes to compute the distance from each voxel to the nearest plane as the approximation of the surface. The tricky part of the algorithm is to achieve a coherent orientation for all the planes.

Nevertheless, another method has been widely used: the method of Curless and Levoy [6]. Given a set of scanned range images, they take one image at a time, triangulate it and place it into a volume grid. A distance function for this range image is obtained by casting rays from the position of the sensor (assumed to be known) to the triangulated range image, storing distance values for each voxel. Furthermore, distances are weighted taking into account the reliability of the scanned points that form the triangle intersected by the ray. The individual distance functions are merged together following an addition scheme to obtain the final distance function. This method has been used with range scans coming from a laser range-finder sensor, but it has also been used with the output from computer vision techniques [7] [8]. In the later cases, the depth maps are not as dense as the range finder ones, but they are still usable.

2) Local Approximants: These methods are based on computing the implicit function representing the surface by using local approximation functions at a set of points. Obviously, these functions are defined by the input points and their neighbors, and normally they have the form of a Radial Basis Function (RBF). Then, the local contributions are blended together to form a unique implicit surface representation. Methods in this class mainly differ in the number and placement of the local functions, their kind, and their final blending.

Most of the methods that fall in this category use a set of RBF as local approximations that, when merged together, reveal the implicit surface. The method of Carr *et al.* is a good example of such procedure [9]. On the other hand, other methods exist that compute the neighbors of a point and fit a local surface without using RFB, like the Multilevel Partition of Unity (MPU) algorithm by Ohtake *et al.* [10].

Nevertheless, the most relevant approach in this class of methods is the Poisson surface reconstructor by Kazhdan *et al.* [11]. In this case, instead of a distance function, the implicit function we want to reconstruct is an indicator function, i. e., a function that is 0 if inside the object, and 1 if it is outside. The sample points with normals (assumed to be known) are seen as samples of the *gradient* of the indicator function. Given this notion, the problem consists in the computation of the inverse of the gradient, that is, find the scalar function X whose gradient best approximates a vector field $\vec{V}$ defined by the samples:

$$min \left\| \nabla X - \vec{V} \right\|$$

In order to get a full $\vec{V}$, the input points are splatted in our working space, using Gaussian propagation of the contribution of each point. Then, by means of applying the divergence operator, this problem is transformed to a Poisson problem: find the scalar function X whose Laplacian (divergence of gradients) equals the divergence of the vector field $\vec{V}$:

$$\Delta X \equiv \nabla \nabla X = \Delta \vec{V}$$

B. Delaunay Methods

As stated above, these methods are based on the 2D or 3D Delaunay triangulation. Not all these methods necessarily require to build this structure, since some of them are only based on some of the different properties that this space subdivision has. In some cases the Voronoi diagram, which is the dual of a Delaunay triangulation, is also used. We will group these methods following the classification proposed by Cazals and Giesen in [3]: Restricted Delaunay, In/Out Labeling, Empty Balls and Tangent Planes.

1) Restricted Delaunay: Restricted Delaunay methods rely on directly choosing from the Delaunay triangulation those triangles that are restricted to some subset that is a good approximation of the surface and that can be efficiently computed from the input point cloud. Two classical methods of this kind are the Crust method [12], proposed by Amenta and Bern, and the Cocone method [13], proposed by Amenta *et al.*

On one hand, the Crust method is based on a curve reconstruction algorithm from Amenta *et al.* [14]. Being P the set of input points, and V the vertices of the Voronoi diagram of P, the Crust is defined as the set of edges of the Delaunay triangulation of $P \cup V$ with both endpoints from P. By doing this, only the edges whose circumcircle is empty of Voronoi vertices are part of the Crust. The main idea behind this reasoning is that the vertices V of the Voronoi diagram are approximations of the medial axis of the surface S of the object, and the Voronoi disks of $P \cup V$ approximate empty circles between S and its medial axis. The medial axis of an object is the set of points that have more than one closest points on the boundary of the object, and can be seen as its *skeleton*. The implementation in 2D is also very simple, the only thing that has to be done is the triangulation of $P \cup V$, and retain from it the edges that have both endpoints on P.

Nevertheless, the extension to 3D is not that straightforward. The problem is that, in 3D, vertices of the Voronoi diagram may fall very close to the surface, resulting in holes in the Crust. To avoid this behavior, in substitution to the whole Voronoi vertices, we only take into account the poles of each Voronoi cell. The two poles q^+ and q^- of a Voronoi cell are defined as the two Voronoi vertices that are at a farthest distance from the input point p that has generated the cell, where the two form an angle greater than $\pi/2$. Then, similar to the 2D procedure, once we have the set of poles Q computed, we can generate the Crust as the set of triangles of the Delaunay triangulation of $P \cup Q$ that have all its vertices contained in P. This first crust is just a set of triangles (lacks correct connectivity), and a filtering based on normals of the points compared to the direction of the poles and an greedy procedure to ensure the manifoldness of the triangles is needed to recover the final surface.

The Cocone algorithm is as an extension of the Crust. In this case, the direction from a point p to its first pole q^+ is used as an estimation of the normal at that point. Then, the cocone is defined as the cone-complement of the double cone with apex at p that makes an angle of $\pi/2 - \theta$ with the axis defined by the normal estimation at p. Edges in the Voronoi diagram that are intersected by the three cones of the three Voronoi regions that formed that edge are selected. Finally, the candidate triangles for being part of the surface are the complements of these Voronoi edges. Again, a manifold mesh extraction step is needed to get the final topologically correct surface.

2) In/Out Labeling: These methods use a strong assumption on the input data points: the underlying surface they are describing must be watertight (i.e., without boundaries, closed). It is easy to realize that from a watertight surface you can always define an inside and an outside. This fact is exploited by these methods to recover the surface as the border between the inside and the outside of the unknown object/shape.

It is worth noticing that the assumption of a closed surface without boundaries is very common among computer graphics methods. In this context, a well known method is the Power Crust of Amenta et. al. [15]. It uses the Medial Axis Transform (MAT) to define the object as a union of balls centered at the inner medial axis. The algorithm computes the Voronoi diagram of the input point cloud P and its poles. Then, a Power diagram from the poles is computed. A Power diagram is a kind of weighted Voronoi diagram, where the definition of *distance* that defines the region/cell is weighted, in this case by the radius of a Voronoi ball centered at each of the poles. Next, each pole is labeled as inside or outside following a greedy procedure, which starts with the outermost poles (the ones generated from sample points located on a bounding box that encloses the input point cloud). The algorithm iteratively visits the poles following a priority defined by two values named *in* and *out* that are assigned depending on the angle defined by the two poles of a sample and the intersection angle of the polar balls. The power crust is then extracted as the boundary between the power diagram cells belonging to inner poles from the ones belonging to outer poles. Furthermore, this algorithm gives an approximation of the MAT as a secondary output, which is constructed using weighted Delaunay triangulation faces which connect interior poles.

Another method that falls in this category is the presented by Boissonat and Cazals in [16]. They use the notion of *natural neighbors*, which are defined as the points in the input set P whose Voronoi cells V are divided when we insert a new point x to the set. A natural region NR_{x,p_i} is the part from V_{p_i} that is extracted by $V_x \cap V_{p_i}$. Then, a natural coordinate λ_{p_i} is defined as follows:

$$\lambda_{p_i} = \frac{Area(NR_{x,p_i})}{\sum_j Area(NR_{x,p_j})}$$

Then, as in the methods from the implicit surfaces category, we use the natural neighbors and coordinates to define an interpolatory signed distance function:

$$f(x) = \sum_i \lambda_{p_i}(x) \cdot (p_i - x) \cdot n(p_i)$$

Where $n(p_i)$ is the normal at p_i. In other words, given a point x, its value on the distance function is the sum of the distances from x to each of the tangent planes of Natural Neighbor points p_i, weighted by their natural coordinate. In this case, normals of the points (and consequently, local tangent planes) are supposed to be known. In case they are not, they can be computed using the poles as in [15], or PCA using neighboring vertices as in [5]. Instead of using the implicit approach and extract the surface using marching cubes as in the methods on section III-A, the surface is extracted from the Delaunay triangulation, using the Voronoi diagram. A triangle in Delaunay is part of the surface if, when evaluating the distance function, its dual Voronoi edge has one endpoint with a positive value and the other with a negative one.

3) Empty Balls: This family of methods take advantage of the notion of the empty balls to define the surface locally, that is, each triangle of the surface is supposed to have a ball that does not contain any other point (or simplex) from the set.

A well known method in this class, for its very low computational complexity (since it does not compute the Delaunay triangulation itself), is the Ball Pivoting algorithm of Bernardini *et al.* [17]. This algorithm is closely related to the α-shapes concept, since it tries to compute a surface subset of an α-shape of the set of points. Recalling the definition of the α-shapes, a triangle is part of the α-shape of the input point cloud P if there exists a ball of radius α that touches the three triangle points and that does not contain any other point of P in its interior. Given this definition, and once α is fixed (in the original article, named ρ), the algorithm starts constructing the surface with a seed triangle that accomplishes the previously commented property. Then, the ball is pivoted around the edges of the surface (i.e., it revolves around the edge while keeping in contact with the edges endpoints) until it touches another point in P, forming another triangle. This process continues until all reachable edges have been processed. Then, the process continues with a new seed triangle until all points in P have been considered.

Also, in this context we find the Regular Interpolants approach of Petitjean and Boyer [18], which uses the notion of Gabriel triangles to define the surface in terms of the properties of the point set itself. A Gabriel triangle is such that a circumsphere defined by the radius of its circumcircle is empty of other points from the initial set. The algorithm is based on two measures on the input point set. The first one is the *granularity* $g(p)$, that given a sample point p is defined as the radius of the largest ball circumscribing a triangle incident to p. The second one is the *discrete feature size* $f(p)$, which is the minimum distance from p to the discrete medial axis. A discrete medial axis is the Voronoi diagram without the cells whose duals are triangles in the interpolant. With this two measures, a regular interpolant must hold that $g(p) < f(p)$ for all sample points p. The input point cloud is called regular if it admits at least one regular interpolant. For regular point sets, the incremental procedure to get the final interpolant is based on select from the set of Gabriel triangles (3D Gabriel graph) the ones minimizing granularity. Since the input set of points does not have to be regular, the algorithm needs to be extended in order to deal with this special cases. The authors proposed an extension that follows the heuristic of minimize granularity and force the interpolant triangles to be on the Gabriel graph. That results in skipping some points that minimize granularity because they are not Gabriel.

4) Tangent Planes: The algorithms that fall in this class assume that the surface is smooth enough to be approximated by tangent planes placed at each point. By analogy, this also means that we can estimate these planes by computing the normals of each point.

Gopi et. al. presented in [19] a method that computes the tangent planes at each point in a way very similar to the method of Hoppe, then projects the neighbors of each point to its corresponding tangent plane and a local 2D Delaunay triangulation is computed from these projections. Three sample points p, q and r form a triangle in the reconstruction if they all are mutually contained in their Delaunay neighbors.

On the other hand, the Greedy algorithm presented by Cohen-Steiner and Da in [20] incrementally reconstructs an oriented surface S by selecting triangles out of a Delaunay triangulation of the input 3D points P and stitching them to S. The selection of the new triangles to incorporate to S at each step is easy: the less ambiguous triangles are stitched first. Following this scheme, the difficult decisions are delayed, and easy ones are executed first. Doing so, ambiguous decisions could be overcome by the advancing front coming from other directions. Starting from a seed triangle, new triangles have to be added incrementally to the surface. In order to preserve manifoldness, the possibilities of candidate triangles to stitch to the surface are discretized to four cases. Then, the selection of the next triangle to insert in the surface is twofold: first, choose a candidate triangle (among valid ones) for each of the edges forming part of the boundary of the current surface, and second, choose a triangle among all candidates. The choice of candidates for each edge is performed by comparing the triangles according to their circumradius. The triangles having smallest circumradius are chosen first, since they are more likely to be part of the surface (under more or less high and uniform sampling assumption). In order to deal with sliver tetrahedra, valid triangles are discarded if their angle with the current surface is smaller than a constant. A sliver tetrahedron is a tetrahedron with bad aspect ratio, most likely defined by nearly coplanar points placed equally spaced around the equator of a sphere. Now, from the entire set of valid triangles, we need to choose one. This selection is performed by ranking the triangles according to the dihedral angle they form with the current surface and choosing at each iteration the one having smallest value.

C. Computer Vision Aided Methods

Despite the great effort of the computer graphics community to obtain a generic algorithm, some information inferred by the way the data is acquired could be useful if added to the reconstruction process. Historically, computer vision algorithms based on texture patch matching have not taken into account the final surface reconstruction step after reconstruct a 3D point cloud. This is because the algorithm is based on the computation of discrete 3D points, so the final results and error computations must be in this direction. However, a very new family of algorithms that deals with the surface reconstruction problem from a computer vision viewpoint is starting to grow on the last years.

Taking into account that all the work done in computer graphics can be used as a solid base for the construction of this type of algorithms, authors try to use new sources of information, such as restrictions introduced by the camera position (visibility constraints) or texture information (photo-consistency measures) to improve the results.

Methods like the one from Salman and Yvinec [21] are based on gathering all the sets of possible triangles, computed on the 2D image plane. Then, they need to filter the resulting triangle soup by means of different filtering steps. Obviously, these methods need a post-processing step to get a manifold mesh from the set of non-eliminated triangles.

Other methods are usually formulated as an energy minimization problem, and, like in the two main computer graphics approaches, they mainly differ in the subdivision method of the working space.

Methods using an implicit formulation are not typically based on point clouds, but they directly use the voxelized space to recover the object by *removing* the voxels that do not conform with the object (e.g. [22]). In other words, they work by defining an *inside* and an *outside* of the object. Then, as usual, the final surface is obtained by using an isosurface extractor method. This kind of methods are out of the scope of this paper, since they are not using the data representation we are supposing, and we will not comment them. Furthermore, they usually rely on constraints on the camera configurations when capturing the images which are hard to assume when acquiring an underwater sequence.

Alternatively, some methods use a Delaunay triangulation of the space to work. For example, the method of Labatut *et al.* [1] builds a graph having the tetrahedra as nodes and the triangles as edges, and applies a graph cut to this structure in order to minimize a given energy. This energy is based on three terms. The first one is accounting for the visibility of the points, since it penalizes triangles that are intersected by a Line of Sight, that is the line going from the camera center to a 3D point seen from the corresponding view. The second one is based on the photoconsistency of the triangles when projected to the different images. Finally, a third weight penalizing the triangles according to their area is added to promote smoothness on the resulting surface. You can notice that the two first weights are based on information gathered from the computer vision reconstruction pipeline. Nevertheless, it is worth commenting that the photoconsistency term becomes useless for very dense point clouds, since the triangles projected to the image plane are too small and do not contain enough texture information. Furthermore, in a later article by Hiep *et al.* [23] continuing this work, the photoconsistency term is substituted by a surface quality term.

IV. Point Clouds and Surface Reconstruction

Computer vision techniques rely on texture information contained in the input images to get the 3D reconstruction, so the points forming the resulting cloud can be non-uniformly distributed depending on the richness of the textures. Also, the density of a point cloud, which is defining the discretization frequency of the final surface and, consequently, its level of detail, depends also on the texture of the surface. Furthermore, since the 3D reconstruction procedure is not perfect, the final point cloud may contain outliers, which are wrongly estimated 3D points that do not belong to the surface that we aim to reconstruct.

By observing the State-of-the-Art techniques and the restrictions that apply to our kind of data, we can discuss the theoretical advantages and disadvantages of the different classes of methods and their viability to be applied to the data obtained from a computer vision based 3D reconstruction pipeline.

First, the implicit methods have the advantage of achieving smooth surfaces. Nevertheless, global approximant methods are very sensitive to noise in the data. For instance, Hoppe's algorithm needs to have all the points lying on the surface, because if this condition does not hold, the nearest-neighbors selection will yield a wrong tangent plane and a wrong distance function. On the other hand, although giving better results in cases of a noisy input, local approximant methods present complex systems of equations to solve. This is caused by the fact that local approximations are coupled when building the final implicit surface, which causes that small changes in input points resulting in different global coefficients for the local functions.

On the other hand, the computer graphics algorithms based on Delaunay/Voronoi structures fulfill the requirement of the input data points being on the recovered surface, since the structures are formed from them. However, each of the subclasses have also both advantages and disadvantages. First, the restricted Delaunay methods provide an easy solution to the problem that only rely on properties in the underlying structures of the points, nevertheless, this restriction does not achieve good results, and they usually fail on special cases and specially when dealing with sliver tetrahedra. Secondly, in/out labeling methods provide a new restriction that can be used to deal with the special cases in which the previous method could not, which is that the surface has to be watertight (i.e. not bounded). However, this assumption is very restrictive for our needs, since our data can be retrieved not only from one object but, for example, from a generic part of the seafloor where there may be many disjoint objects described and, therefore, we need to define a boundary for each of them. Third, the empty balls methods only achieve good results under the assumption of having a more or less regular sampling density along the data, which can not be assumed in our case, since texture in the images is what determines the final number of reconstructed points (areas with more complex texture will yield more reconstructed points). Finally, the tangent plane methods work under the assumption of having a smooth surface, so they also rely on a dense sampling, along with a more or less uniform sampling density. In the case of noise in the data, the success of this methods depends on whether they are designed to *ignore* some of the data points for probably non being part of the surface, which is not assumed by all the methods.

Notice that the main difference between using a method from the Delaunay or the Implicit group is that the first ones **interpolate** and the second ones **approximate** our data. This means that Delaunay methods provide a final surface that passes through the input points (although does not have to be all of them), while the implicit surfaces approach is finding a surface in the discretized space by using an isosurface method, so the surface does not necessarily has to contain the input points. This property is important depending on the post-processing to be applied to the data. Since we come from a computer vision reconstruction pipeline, all the points have important information associated to it: the corresponding camera poses and views (images) that have been used to generate it. If we decide to use an implicit surface method, this information will be lost, since the resulting surface will be made from different vertices. On the other hand, if we use a Delaunay based approach, the (visual) quality of the output surface will highly depend on the quality of our points, normally resulting in a *bumpy* surface if the error in the location of the points is high. Normally, a smoothing step is needed before using this mesh for visualization.

As a conclusion, if we only want the mesh for visualization purposes, implicit methods will fit our needs. On the other hand, if we decide to use a Delaunay based method, we will need to perform a pre-processing on the input points depending on their precision given their sensitivity to noise, and also possibly apply a smoothing method to the surface if we use it for visualization purposes.

Finally, we can conclude that the methodologies in the computer vision aided group are closely related to the main ideas from the algorithms used in the computer graphics community, since they can also be grouped according to the space subdivision they do of the working space. These methods are more suited to our needs, given that they know their limitations when working with a point cloud obtained with a computer vision 3D reconstruction software. Nevertheless, if we pay attention to the publication date of the approaches we presented in section III-C, we will realize that these approaches are very recent.

V. Experiments and Results

In this section we will perform some experimentation using the methods discussed above, applied to underwater 3D reconstructions obtained by a computer vision system [24] [25]. In order to check the viability of the different families of methods, we will check the behavior of a representative method for each of the groups presented in section III:

- **Implicit Surfaces**:
  - Global Approximants: Hoppe method.
  - Local Approximants: Poisson method.
- **Delaunay**:
  - Restricted Delaunay: Cocone method.
  - In/Out Labeling: Power Crust method.
  - Empty Spheres: Ball-Pivoting method.
  - Tangent Planes: Greedy method.

- **Computer Vision Aided**
    - Graph Cuts method.

More precisely, we will discuss the performance of the methods presented above to deal with different sampling density on the surface using a real dataset.

A. Sparse Dataset

In Fig. 3 we can see the results obtained for the sparse dataset. This sparse 3D reconstruction has been obtained from the Structure From Motion approach described in [24] using a sequence of 324 images. A subset of these input images is shown in Fig. 1, and the obtained cloud of points in 3D is presented in Fig. 2(a). The surface reconstructions shown here are the best ones, obtained by empirically tuning the parameters on which the different methods depend on.

If we look at the surface reconstruction results, we can see that the majority of them have problems when reconstructing the surface. We can see that Hoppe method is not providing good results probably because the tangent planes of some points are not defining the underlying surface consistently enough. We can also see that it generates inexistent off-surface blobs, showing that the tangent planes and, consecuently, the implicit surface has not been correctly estimated. Notice also that Poisson assumes the normals to be known, and since we do not have them for this dataset, we estimated them using the method used by Hoppe in his algorithm. Furthermore, the result shown in Fig. 3(b) is not the direct output of the Poisson method, since it is returning a closed surface. The surface shown is a part of this closed surface, extracted by eliminating from the original surface the triangles having an edge bigger than a threshold (triangles tend to increase size in non-sampled parts). We can see that the lack of sampling creates an oversmoothed surface which lacks details. On the other hand, the in/out labeling is failing for Power Crust, since the method is trying to find a closed surface and our example is a bounded object. That makes the method close parts that contain valid input points. Furthermore, the poor sampling rate makes methods such as Cocone, Ball Pivoting and Greedy fail. In addition to obtain a partially reconstructed surface, we can see that these methods get wrong orientations for some of the triangles in the surface. In the case of Cocone, it stops after finding some inconsistencies in the recovered shape, due to the fact that this method only has guarantees based on sampling conditions that do not hold for this dataset. The Ball Pivoting algorithm provides a quick algorithm to build the surface, but this surface is highly dependent on the unique parameter the algorithm needs (the radius of the ball that is incrementally building the surface). That makes the resulting surface lack of fine details when this parameter is high and, on the contrary, results in a surface with holes on it if the parameter is too low. On the other hand, the Greedy method also returns a non-manifold mesh because the surface is not densely enough sampled to resemble the smooth surface this method is assuming. If we pay attention, the best result for this dataset (although not perfect) is obtained by the Graph Cuts method. As expected, the addition of other sources of information to the reconstruction process reduces the ambiguity and allows to get better results even when reducing the sampling density.

With all this observations, we can conclude that a good sampling density is a critical requirement to get a correct reconstruction.

B. Dense dataset

On the other hand, in Fig. 4 we can see the results obtained for the densely sampled version of the same dataset (see Fig. 2(b)). This version has been obtained by using the software of [26], described in [25], applied to the camera parameters we had already computed from the sparse dataset. A remarkable property of this algorithm is that it also gives an estimation of the normals at each point. Despite this dataset presents well and highly sampled areas, it does not contain information for some parts of the scene that were not reconstructed, which results in abrupt transitions in the surface.

As we can see, most of the algorithms offer better results in this case. The Poisson method obtains visually pleasant and smooth results in this case. However, in the case of the Hoppe method, the computation of the global implicit function failed, resulting in off-surfaces. This behaviour is caused by the abrupt transitions in the surface, where the neighborhood of the points is causing a wrong tangent plane computation. Nevertheless, the Cocone method recovers a larger part of the surface than in the previous case, although the orientations of the triangles are not consistent, and we find non-reconstructed zones. Moreover, the Power Crust method still fails for the reason discussed in the previous section: the fact of trying to obtain a closed surface makes the method avoid part of the sample points. In the case of the Ball Pivoting algorithm, it keeps having the same properties than before: although obtaining a quick reconstruction, its quality depends on a parameter that has to be set up manually. However, in this case the density of the point cloud allows to set a smaller value for this parameter, which results in a finer reconstruction (although some triangles remain incorrectly oriented). The Greedy algorithm is obtaining a more detailed surface than in the sparse case, but still gets wrong orientations, mainly caused by the discontinuities in non sampled areas. Finally, the Graph Cuts method gives also good results in this case.

Also, the previously commented difference between the two main classes of computer graphics surface reconstruction methods can be observed: the Implicit Surfaces methods smooth the data and gives visually pleasant results, while Delaunay-based methods, although retaining the input points positions, provide irregular surfaces induced by the noise on the 3D estimation of the original point cloud.

VI. Conclusions

We have reviewed the State-of-the-Art methods for reconstructing a surface from a cloud of 3D points. We focused on the kind of data we are expecting to work with, i. e., a cloud of points obtained by some texture-based computer vision technique.

We also noticed that the main properties of the point cloud used in the surface reconstruction process have to be taken into account when selecting the adequate processing. These properties are density, regularity on the distribution of the points over the surface to reconstruct and the possible measurement error of the points. Also the nature of the underlying surface we are trying to reconstruct has to be taken into account, since special methods exist for the case where the surface is closed.

Taking into account both the theoretical properties of the algorithms and the observed experimental results, we can state which method to use according to the properties of our data. First, we have to consider whether we need the input points to be part of the final surface or not. In other words, we have to decide whether we want to interpolate the surface or approximate it. If an approximation suffices our needs, methods like the listed as part of the Implicit Surfaces group will give smooth surface as result. However, if we need to interpolate the input points, we need to use a method from the Delaunay group, which will provide a surface whose smoothness depends on the measurement error of the input points.

Secondly, we have to think whether the surface we want to reconstruct is a closed surface or, on the contrary, a surface with boundaries. This will refine the selection of possible candidates to be used for reconstructing it, since methods like Poisson and Power Crust are designed to work with this kind of surfaces, and will therefore give better results in this cases. On the other hand, more generic methods, like the Greedy algorithm, the Cocone or the Ball Pivoting are more suitable when the surface to reconstruct is open.

Finally, we can observe that the method from the Computer Aided group is the one that gives a more reliable reconstruction. We can expect that methods exploiting additional information coming from the gathering of the data need to be developed in the future in order to obtain good surfaces from noisy and non-uniformly sampled input points like the ones we are dealing here with.

Acknowledgments

This work was partially funded through the Spanish Ministry of Education and Science (MCINN) under grant CTM2010-15216 and the EU under grant FP7-ICT-2009-248497. We would also like to thank Dr. Tamal K. Dey for provide us with his variants of the Cocone software.

References

[1] P. Labatut, J.-P. Pons, and R. Keriven, "Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts," *2007 Ieee 11th International Conference on Computer Vision, Vols 1-6*, pp. 504–511, 2007.

[2] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 1st ed. Springer, July 1997.

[3] F. Cazals and J. Giesen, "Delaunay triangulation based surface reconstruction: Ideas and algorithms," in *EFFECTIVE COMPUTATIONAL GEOMETRY FOR CURVES AND SURFACES*. Springer, 2006, pp. 231–273.

[4] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, pp. 163–169, August 1987.

[5] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1992, pp. 71–78.

[6] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 303–312.

[7] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, "Visual modeling with a hand-held camera," *International Journal of Computer Vision*, vol. 59, pp. 207–232, September 2004.

[8] M. Johnson-Roberson, O. Pizarro, S. B. Williams, and I. Mahon, "Generation and visualization of large-scale three-dimensional reconstructions from underwater robotic surveys," *Journal of Field Robotics*, vol. 27, no. 1, pp. 21–51, JAN-FEB 2010.

[9] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3d objects with radial basis functions," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '01. New York, NY, USA: ACM, 2001, pp. 67–76.

[10] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel, "Multi-level partition of unity implicits," *ACM Trans.Graph.*, vol. 22, no. 3, pp. 463–470, July 2003.

[11] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, ser. SGP '06. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2006, pp. 61–70.

[12] N. Amenta and M. Bern, "Surface reconstruction by voronoi filtering," in *Proceedings of the fourteenth annual symposium on Computational geometry*, ser. SCG '98. New York, NY, USA: ACM, 1998, pp. 39–48.

[13] N. Amenta, S. Choi, T. K. Dey, and N. Leekha, "A simple algorithm for homeomorphic surface reconstruction," in *Proceedings of the sixteenth annual symposium on Computational geometry*, ser. SCG '00. New York, NY, USA: ACM, 2000, pp. 213–222.

[14] N. Amenta, M. Bern, and D. Eppstein, "The crust and the beta-skeleton: Combinatorial curve reconstruction," in *Graphical Models and Image Processing*, 1998, pp. 125–135.

[15] N. Amenta, S. Choi, and R. K. Kolluri, "The power crust," in *Proceedings of the sixth ACM symposium on Solid modeling and applications*, ser. SMA '01. New York, NY, USA: ACM, 2001, pp. 249–266.

[16] J.-D. Boissonnat and F. Cazals, "Smooth surface reconstruction via natural neighbour interpolation of distance functions," in *Proceedings of the sixteenth annual symposium on Computational geometry*, ser. SCG '00. New York, NY, USA: ACM, 2000, pp. 223–232.

[17] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 349–359, October 1999.

[18] S. Petitjean and E. Boyer, "Regular and non-regular point sets: Properties and reconstruction," *Computational Geometry*, vol. 19, no. 2-3, pp. 101–126, 7 2001.

[19] M. Gopi, S. Krishnan, and C. T. Silva, "Surface reconstruction based on lower dimensional localized delaunay triangulation," *Computer Graphics Forum*, vol. 19, no. 3, 2000.

[20] D. Cohen-Steiner and F. Da, "A greedy delaunay-based surface reconstruction algorithm," *The Visual Computer: International Journal of Computer Graphics archive*, vol. 20, no. 1, pp. 4–16, April 2004.

[21] N. Salman and M. Yvinec, "Surface reconstruction from multi-view stereo," *Proceedings of the 9th Asian Conference on Computer Vision*, 2009.

[22] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," pp. 199–218, 2000.

[23] V. H. Hiep, R. Keriven, P. Labatut, and J.-P. Pons, "Towards high-resolution large-scale multi-view stereo," *Cvpr: 2009 Ieee Conference on Computer Vision and Pattern Recognition, Vols 1-4*, pp. 1430–1437, 2009.

[24] T. Nicosevici, N. Gracias, S. Negahdaripour, and R. Garcia, "Efficient three-dimensional scene modeling and mosaicing," *Journal of Field Robotics*, vol. 26, no. 10, pp. 759–788, 2009.

[25] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multi-view stereopsis," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2009.

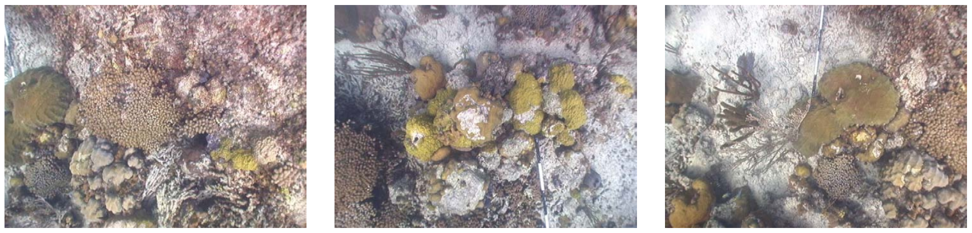
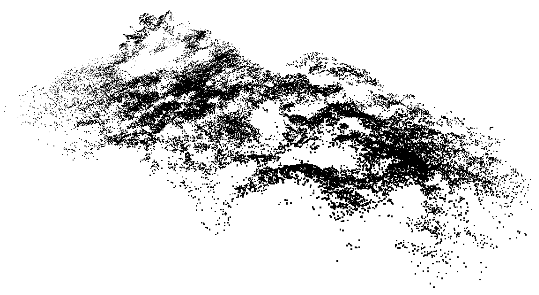
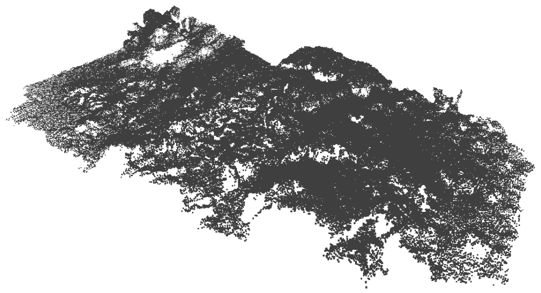


Fig. 1. Three of the input images used to get the 3D cloud of points presented in section V.

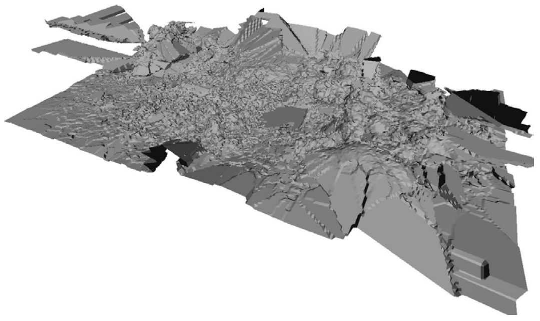


(a) Sparse Input 3D Points

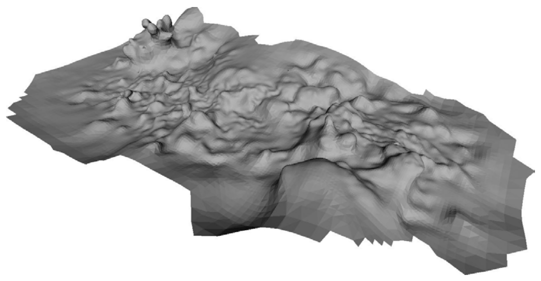


(b) Dense Input 3D Points

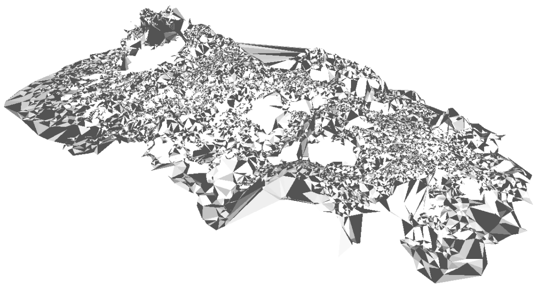
Fig. 2. Sparse and dense 3D point clouds used in section V.



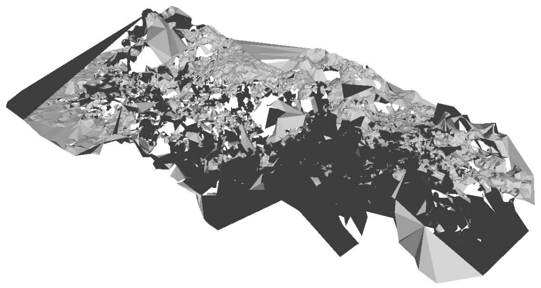
(a) Hoppe



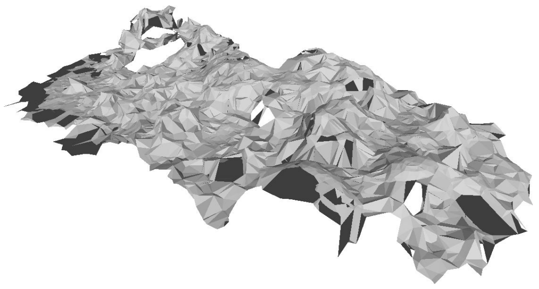
(b) Poisson



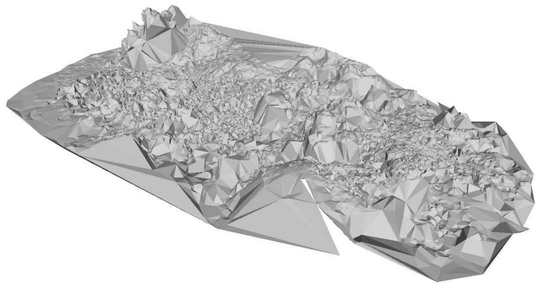
(c) Cocone



(d) Greedy

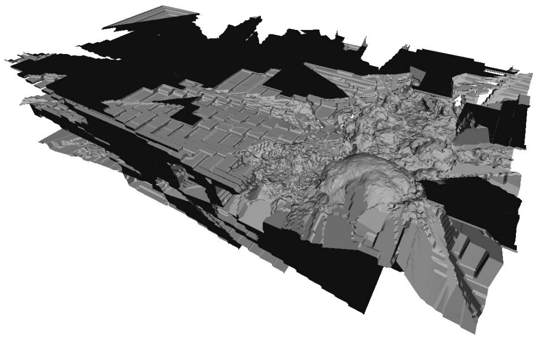


(e) Ball Pivoting

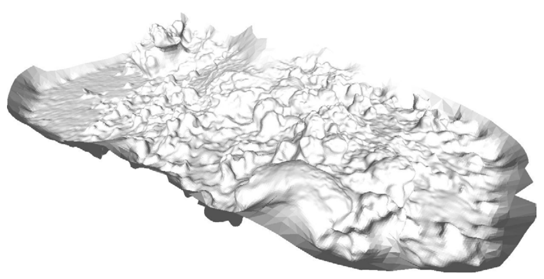


(f) Graph Cuts

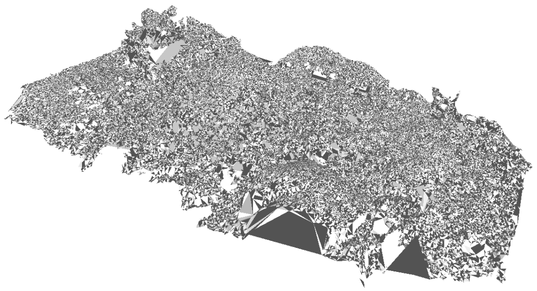
Fig. 3. Experimental results for the Sparse dataset. Black parts represent badly oriented triangles. This images show that the majority of methods has problems when dealing with undersampled regions. Only Poisson and Graph Cuts method are recovering a fully consistent surface.



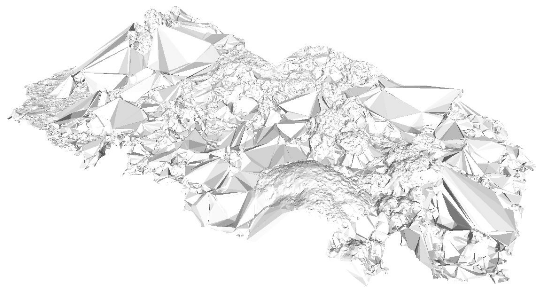
(a) Hoppe



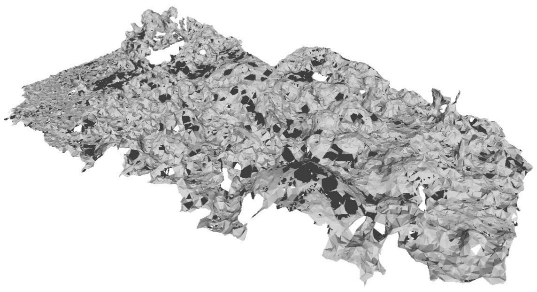
(b) Poisson



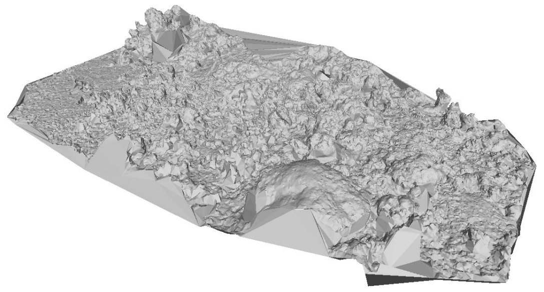
(c) Cocone



(d) Power Crust



(e) Ball Pivoting



(f) Graph Cuts

Fig. 4. Experimental results for the Dense dataset. Black parts represent badly oriented triangles. Despite the recovered surface has larger coverage than with the sparse input, the methods are having problems with abrupt discontinuities.

[26] ——, "Patch-based multi-view stereo software," http://grail.cs.washington.edu/software/pmvs.