Incremental integration of multiresolution range images

A D Sappa*[a] and **M A Garcia[b]**
[a]Computer Vision Center, Edifici O Campus UAB, 08193 Bellaterra-Barcelona, Spain

[b]Dept. of Informatics Engineering, Autonomous University of Madrid, Cra. Colmenar Viejo Km. 15, 28049 Madrid, Spain

Abstract: An efficient incremental algorithm for integrating overlapping, registered range images acquired from different points of view is presented. Every new range image, which is represented by a 3D triangular mesh, is merged with a second triangular mesh which represents the current reconstructed model. The merging process consists of three stages. The first stage finds the area of overlap between the new 3D mesh and the reconstructed model. Triangles that belong to the area of overlap of the coarsest mesh are removed. In the second stage, the boundary created after removing the previous triangles, along with its adjacent boundary from the unaltered (fine) mesh, are projected to a reference plane and triangulated with a 2D constrained Delaunay algorithm. Finally, the last stage projects the previous 2D mesh back to the 3D space, leading to a 3D mesh that sews the new range image to the current reconstructed model. The algorithm is suitable for integrating range images acquired with different spatial resolutions and containing one or more separate objects.

Keywords: incremental multiview integration, range images, multiple objects

1 INTRODUCTION

Range images are gaining popularity in computer vision and robotics owing to the variety of applications that can benefit from them, including world modelling, reverse engineering and object segmentation or recognition. Many of these applications require the automatic reconstruction of 3D objects present in a scene. In order to solve this problem, it is necessary to acquire several range images from different points of view and integrate them into a single 3D model.

Assuming that frame transformations among the different views can be reliably computed by applying well-known techniques such as the iterative closest point (ICP) algorithm[1] or scanning mechanisms,[2] two basic processes must be performed to solve the 3D object reconstruction problem. First, an exploration process is necessary to determine the positions where the range sensor must be placed in order that all the surfaces of the objects present in the scene can be observed. This issue is known as the next-best-view problem (e.g. Hilton[3] and Bottino and Laurentini[4]). Secondly, a multiview integration process must combine the set of range images acquired from the different viewpoints with the aim of generating a consistent 3D model of the object or objects present in the explored scene. Alternatively, the integration process can be performed simultaneously with the registration as it is presented by Tubic *et al.*[5]

If a single range sensor is available and it is mounted, for instance, on a robot that changes the sensor's position over time, the integration process has some important constraints that must be accounted for. The first one is that range images will be acquired sequentially. This implies that the

The MS was accepted for publication on 14 October 2006.
* *Corresponding author. A D Sappa, Computer Vision Center, Edifici O Campus UAB, 08193 Bellaterra-Barcelona, Spain; e-mail: angel.sappa@cvc.uab.es*

integration process must be incremental, being able to update a current reconstructed model as new images are obtained. The second constraint is that the various range images being integrated may have different spatial resolutions, depending on the distance from the sensor to the objects of interest and the angle with which they are observed. Finally, if the reconstruction must be done close to real time, the integration process must be efficient enough not to introduce a significant time penalty.

The different techniques proposed in the literature for range image integration can be broadly classified as unstructured or structured methods.

Unstructured integration methods generate a polygonal surface from an arbitrary set of unorganized points obtained from the available views. Following this approach, Boissonnat[6] presented an algorithm for integrating a set of 3D points corresponding to different views of a single object by applying a 3D Delaunay triangulation algorithm. Edelsbrunner and Mücke[7] proposed alpha shapes as a generalization of the convex hull of a point set. Hoppe[8] presented a complex algorithm that takes as input an unorganized set of points and produces as output a non-redundant surface model. To do so, various assumptions about the data points are made (uniform data point density, same accuracy for all the data, and feasibility of estimating the k nearest surface neighbours of any point p), which are quite restrictive given the above-mentioned constraints of the problem. Another unstructured method for reconstructing surface models by integrating a cloud of 3D points was presented by Bajaj *et al.*[9] This proposal requires a dense uniform sampling of the object to be reconstructed and generates the integrated model by means of regular triangulations and weighted alpha shapes.

Following a different approach, Liao and Medioni[10] extended the principle of 2D snakes to 3D deformable models in order to approximate a 3D surface from a cloud of points. This technique starts by deforming an initial closed surface (e.g. closed cylinder, sphere) trying to bring it as closely as possible to the 3D input data points. The deformation process is carried out by minimizing an energy functional that considers an attraction force field around each original data point—each point pulls the initial surface toward itself. This approach cannot handle multiple objects or objects with deep cavities.

The integration of unstructured data points was tackled at the same time as the registration of multiple range images in Masuda.[11] In that work, integration and registration are alternately iterated until the input shapes are properly registered to the integrated shapes. Finally, a mesh model is generated from the resulting integration by a volume-based algorithm different from the classical marching cubes.

A common problem with unstructured integration methods is that, since their input is a point cloud, when a new range image must be integrated into the reconstructed model, all the previously acquired points must be reconsidered. Therefore, these methods are not adequate for an incremental update of the model, since the computational complexity of that process would grow exponentially. Moreover, the majority of methods require that the points that belong to the same surface be uniformly distributed, complicating or even preventing the integration of range images of different resolutions. Finally, those methods assume that the cloud of points belongs to a single object. However, if a complex scene is to be reconstructed, the acquired range images will contain surfaces belonging to separate objects.

In contrast, structured integration methods assume that the data points that belong to the range images to be merged have been structured in some way prior to the integration process, such as in a triangular mesh. If the range images are originally represented by bidimensional arrays of 3D points, a triangular mesh can be trivially created by linking adjacent points along rows, columns and diagonals. More involved techniques aimed at producing triangular meshes that adapt to the shapes present in the given range images can also be applied.

By taking advantage of the complementary structural information, the performance of the integration process can dramatically improve with respect to unstructured methods. Moreover, such information allows useful features related to the surfaces that are being represented to be computed. For instance, it is possible to determine efficiently regions of overlap between the triangular meshes to be integrated, as well as to identify redundant 3D points contained in these regions. Those points can be discarded or they may help to improve the accuracy with which the overlapped regions are represented in the integrated model. Another important advantage of structured methods is that the range images to be integrated can be locally merged by only considering their

overlapped regions. This may have an important impact on the performance of the whole process.

Following this second approach, Soucy and Laurendeau[12] presented a structured integration technique that decomposes the input range images into subsets of canonic views. A canonic view is an area in which several input range images partially or fully overlap. A specific reference plane is then defined for each canonic view. All the points of the input range images that belong to every canonic view are first projected onto the reference plane associated with that view, and then triangulated with a 2D Delaunay algorithm. The resulting meshes are finally combined to form the complete integrated model. This algorithm is static in the sense that all the range images are necessary in order to be able to compute the overlapping subsets. This problem is overcome by Soucy and Laurendeau[13] by allowing the incremental integration of new range images. However, all the points contained in the area of overlap between the range images involved in the integration process are retriangulated, with a subsequent time penalty in the case of large overlapped areas.

Another well-known structured integration technique was presented by Turk and Levoy.[14] In this case, the integration process is directly carried out in the 3D space. This process begins by converting two meshes that may have a considerable overlap into a pair of meshes that barely overlap along portions of their boundaries. This is done by simultaneously eroding the overlapped boundaries of each mesh. Next, the meshes are 'zippered' together by means of a constrained 3D triangulation process. Finally, each vertex of the zippered mesh is moved to a consensus position obtained by taking an average of nearby positions from each of the original range images. This zippering technique works satisfactorily for relatively smooth surfaces, but it has been shown to fail in regions of high curvature. Furthermore, it was designed to reconstruct single objects from dense range images of similar resolution.

A differently structured method for merging a new range image represented by a triangular mesh with a second mesh that represents the reconstructed object was proposed by Pito.[15] First, triangles that approximate the same surface patch on both meshes are identified. From those triangles, only the ones acquired with the largest confidence are kept. The removal of low confidence triangles produces gaps between adjacent patches that are subsequently retriangulated, making sure that each new triangle does not cause the mesh to self-intersect. This technique is also carried out in the 3D space. In an extension to this work presented by Ju *et al.*,[16] range images are decomposed into subset patches that are then evaluated according to a confidence value. Redundant patches are removed, while winning patches are merged to complete a single mesh.

A regular 3D mesh reconstruction approach that integrates different views by mapping them on a 2D plane is presented by Khan *et al.*[17] The authors present a prototype of a system to generate 3D mesh models with a 3D scanner and a turntable. This approach exploits not only the data point structure, but also the way in which they were scanned (i.e. the turntable). Therefore, a plane wrapping a cylindrical representation is used to depict the different views.

In the present paper, a new structured method for efficiently integrating range images represented by 3D triangular meshes is presented. This technique can integrate range images of different resolutions and can reconstruct single objects as well as separate multiple objects. The proposed algorithm consists of three stages. Given a 3D triangular mesh representing a new range image, the first stage finds out the area of overlap between the new 3D mesh and the reconstructed model, which is also represented by a triangular mesh. The mesh whose overlapped triangles have the largest average perimeter will be referred to as the coarse mesh. The triangles belonging to the area of overlap of the coarse mesh are then removed.

In the second stage, the boundary created after removing the previous triangles, along with its adjacent boundary from the unaltered mesh, are projected onto a reference plane and triangulated with a 2D constrained Delaunay algorithm.[18] The last stage projects the previous 2D triangular mesh back to the 3D space. This 3D mesh allows the new range image to be sewn to the current reconstructed model.

Although the overall scheme of the proposed technique resembles the zippering technique, it has two major differences that make the proposed technique more adequate for the efficient integration of multiresolution range images describing single or multiple objects. The first difference is in the detection of the region of overlap between the new mesh and the current reconstructed model. In Turk and Levoy,[14] this area is constituted by those

triangles from a mesh whose vertices are closer than a certain experimental threshold to the triangles of the other mesh. This heuristic is exclusively based on point-to-triangle distances, disregarding the orientation of the surfaces. This may lead to interior vertices from one mesh being erroneously considered to overlap with boundary triangles of the other mesh just because their distance is lower than the given threshold. This problem can be worsened if both meshes have different resolutions (Turk and Levoy[14] assumed that the triangles of both meshes have comparable sizes). Instead, the proposed technique is based on a more robust process that determines the inclusion of vertices of a mesh into volume elements (polytopes) associated with the triangles of the other mesh. Those polytopes are defined by taking the surface orientation and curvature into account.

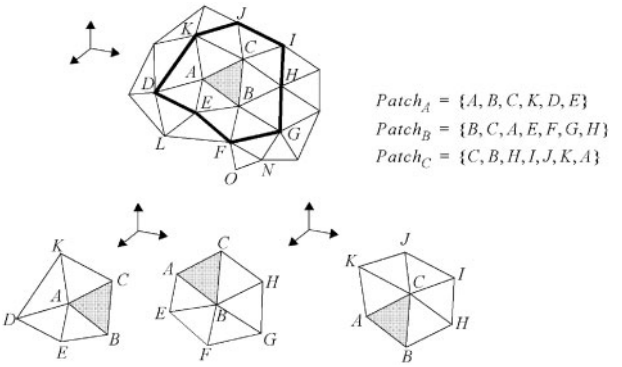
The second basic difference from Turk and Levoy[14] is in the way the two meshes are merged. In Turk and Levoy,[14] the sewing is performed by applying an ad hoc constrained triangulation in the 3D space. New points to be triangulated must be found by determining all the intersections between the lines of one of the boundaries and planes orthogonal to the edges of the other boundary. In addition to the high computational cost of this operation, a large number of new points can result from the process, all of them being included in the final mesh. Instead, the proposed technique obtains a sewing triangular mesh in a more straightforward and robust way, by applying a well-known constrained Delaunay triangulation in the 2D space, without addition of new points to the result.

This paper is organized as follows. Section 2 introduces concepts and notation that will be used throughout the rest of the paper. Section 3 describes the three stages of the proposed integration algorithm. Section 4 presents experimental results with real and synthetic range images. Finally, conclusions and future lines are given in section 5.

2 PRELIMINARY CONCEPTS

2.1 Range images and triangular meshes

Generally, a range image is a rectangular sampling of the surfaces present in a scene. Its usual representation is a 2D array **R**, where each array element $\mathbf{R}(r, c)$, $r \in [0, R)$ and $c \in [0, C)$, is a scalar that represents a surface point of coordinates: $(x, y, z) = \{f_x(r), f_y(c), f_z[\mathbf{R}(r, c)]\}$ referred to a local coordinate system associated with the range sensor. The definition of f_x, f_y and f_z depends on the properties of the actual range sensor being used. In general, $\mathbf{R}(r, c)$ can be considered to be the distance between a surface point and a given reference plane, which is orthogonal to the axis of the sensor and placed opposite to it at a specified distance. Invalid points in the image will be considered to have the background value β. The viewing direction **D** is aligned with the Z axis of the local reference frame, and it is pointing towards the scene. Thus, vector **D** can be defined as $\mathbf{D} = (0, 0, -1)$.



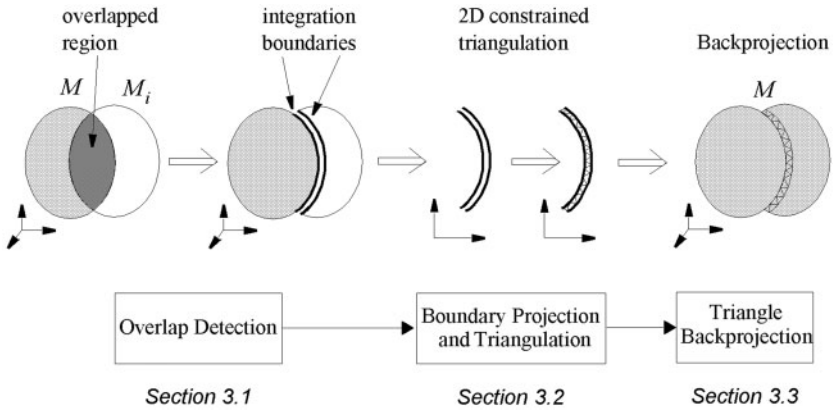
1 3D triangular mesh and polygonal patches associated with triangle ABC: $\text{Patch}_A = \{A, B, C, D, E\}$; $\text{Patch}_B = \{B, C, A, E, F, G, H\}$; $\text{Patch}_C = \{C, B, H, I, J, K, A\}$

A 3D triangular mesh is a piecewise linear surface consisting of triangular faces connected along their edges. Formally, a 3D triangular mesh M is a set $\{V, T\}$, where $V = \{v_1, \ldots, v_m\}$, $v_i \in \Re^3$, is a set of vertex positions that define the shape of the mesh in $\Re^3$, and T is a description of the mesh topology. Each vertex is defined by three coordinates (x, y, z). The topology of the triangular mesh is defined by a set of polygonal patches. A polygonal patch is associated with every vertex and keeps the identifier of that vertex as well as the identifiers of the vertices adjacent to it. The adjacent vertices are kept in counter-clockwise order by convention. Figure 1 shows an example of a 3D triangular mesh and the three polygonal patches that contain a certain triangle ABC.

A vertex that belongs to the boundary of a triangular mesh M, such as vertices J and F in Fig. 1, will be referred to as an exterior vertex of M.

2.2 Binary space partition trees

In order to locate objects contained in a certain region of space efficiently, binary space partition



2 Illustration of three stages of integration algorithm

(BSP) trees can be used.[19] A BSP tree is a binary tree intended to store the objects contained in a certain parallelepiped in 3D space (the domain parallelepiped) aligned with the axes of a reference frame—this frame is the local frame associated with the reconstructed 3D model—in such a way that it is efficient to recover the objects contained in another (usually smaller) parallelepiped (the query parallelepiped).

Each node of the BSP (binary) tree is associated with a parallelepiped, with the root of the tree being associated with the domain parallelepiped. The parallelepiped associated with the node at any level of the tree is split into two equally sized parallelepipeds that are assigned to the two children of the node. The cutting plane at each level may be parallel to the XY, XZ or YZ planes. Among the possible cutting planes, the one that produces the most regular parallelepipeds is chosen at each level. The number of levels (and thus the resolution of the partition) is fixed and defined a priori. The parallelepipeds associated with the leafs of the tree are the ones that contain the stored objects, and will be referred to as BSP cells.

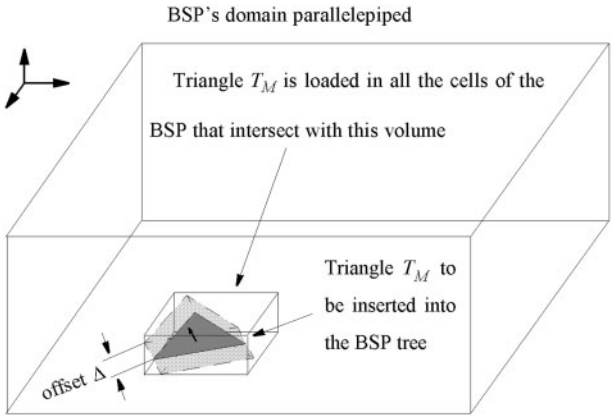
Every time a new 3D object (bounding box) is loaded into a BSP tree, it is added to the BSP cells that cover the volume of space that the object occupies. These cells are found by traversing the tree from its root. If the number of objects stored in any cell reaches a specified threshold and the maximum number of levels at that part of the tree has not yet been reached, that BSP cell is split into two new cells, and the objects contained in it are redistributed between the new ones.

Once a set of objects has been loaded into the BSP tree, a query function will return the objects contained in the BSP cells that intersect the given query parallelepiped. Two parallelepipeds intersect if they share a region of 3D space.

3 INCREMENTAL INTEGRATION OF RANGE IMAGES

Given a triangular mesh M_i that approximates a new range image, the objective consists of merging it with the current reconstructed model M, also represented by a triangular mesh, generating thus a non-redundant surface model that becomes the new current reconstructed model M.

The algorithm consists of three stages (see Fig. 2). The first stage identifies the overlapped regions between the two triangular meshes to be merged (the new range image and the current reconstructed model) and, by taking these regions into account, it defines the integration boundaries. In the second stage, the integration boundaries are projected onto an integration plane, which is a plane orthogonal to a projection direction δ, computed by considering the orientation of the triangles that belong to the overlapped regions found in the previous stage. The points that constitute the projected integration boundaries are triangulated over that 2D space through a constrained Delaunay algorithm, preserving their connectivity (polylines defined by the projected integration boundaries). Finally, the obtained triangulation is backprojected to the original 3D space. The final result is a non-redundant triangular mesh that will be integrated with further range images. The three stages of the algorithm are described in more detail below.



3 Right prism and axis-aligned bounding box associated with a certain triangle T_{M}

3.1 Overlap detection

Given two 3D triangular meshes, M_{i} (new mesh) and M (current reconstructed model), referred to the same global coordinate frame, this first stage identifies the triangles of M_{i} that overlap with triangles of M, and then removes the triangles that belong to the coarsest mesh, since they are considered to provide redundant information.

A triangle T_{i} of M_{i} is considered to overlap current model M if any of the vertices of T_{i} overlaps with any of its nearby triangles in M. In order to find out efficiently the triangles of M located in the neighbourhood of a given vertex of T_{i}, a BSP tree is used. At the beginning of the integration process, the triangles of M are loaded into the BSP tree as follows. A right prism (a prism whose top and bottom polygonal faces lie on top of each other, such that the vertical polygons connecting their sides are rectangles) is associated with each triangle T_{M} of M. The right prism of a triangle is computed by displacing that triangle an offset Δ upwards and downwards along the direction of its normal vector.

In order to define the regions of the domain parallelepiped (see section 2) where each triangle is inserted (the domain parallelepiped encloses M and M_{i}, being aligned with the axes of the reference frame), axis-aligned boxes bounding the previously obtained prisms are computed. Each triangle T_{M} is stored in all the cells of the BSP tree that intersect the bounding box of the right prism of T_{M}. Figure 3 illustrates the right prism and bounding box corresponding to a certain triangle T_{M}.

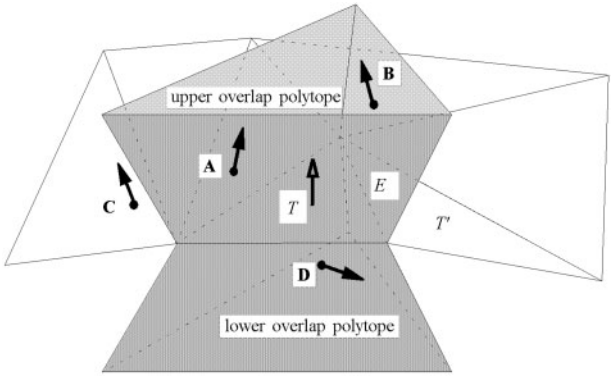
Once all the triangles of M have been loaded into the BSP tree, given the 3D coordinates of a vertex of M_{i} as the query parallelepiped (see section 2), the BSP tree will return the triangles of M close to that vertex.

There are two special cases in which the overlap between a vertex of M_{i} and a triangle of M is directly discarded in considering that they both belong to separate surfaces: (a) when the distance between the vertex of M_{i} and the plane that contains the triangle of M is above a certain threshold (half the average length of the edges of the triangle has been considered), and (b) when the normal vector associated with the vertex of M_{i} has an angle of more than 90 degrees with respect to the normal vector of the triangle of M. The normal vector associated with a vertex is obtained by averaging the normal vectors of the triangles that belong to the polygonal patch associated with that vertex (polygonal patches are defined in section 2).

If the previous conditions are not satisfied, the vertex of M_{i} will be considered to overlap with the triangle of M if the former belongs to either the upper or lower overlap polytopes of that triangle. The upper overlap polytope of a triangle is defined as the intersection between the positive half-space supported by the triangle's plane and the positive half spaces supported by three upper overlap planes associated with the edges of the triangle. Given an edge E of a triangle T, the upper overlap plane associated with E is the plane that contains E and is orthogonal to the plane of the triangle T' adjacent to T along E, in the case where such a neighbour exists, or orthogonal to the plane that contains T if either T' does not exist or triangles T' and T form a concave surface. Conversely, the lower overlap polytope is the symmetry of the upper overlap polytope with respect to the plane that contains triangle T.

Figure 4 illustrates the concept of overlap polytopes and overlap detection considering a certain triangle T and a set of vertices. In this example, vertices **A** and **B** are considered to overlap with triangle T, while vertices **C** and **D** do not overlap. Vertex **D** does not pass the normal orientation test, whereas vertex **C** is outside both the upper and lower overlap polytopes of T.

The overlap detection process starts by insertion of all the exterior vertices of M_{i} in a FIFO queue. Afterwards, an iterative process removes the vertex V at the head of the queue and tests it for overlap against its nearby triangles in M. These triangles are found from the BSP tree, given the coordinates of V. If vertex V happens to overlap with any of those



4 Determination of overlap between vertices of M_i and triangle T belonging to M; overlap polytopes of T are darker; points **A** and **B** overlap with triangle T while points **C** and **D** do not

triangles from M, the vertices of M_i that belong to V's polygonal patch and have not yet been tested for overlap are enqueued. This iterative process stops when the queue is empty. In this way, it is not necessary to test all the vertices of M_i. The triangles of M_i that contain any overlapped vertex are referred to as the overlapped triangles of M_i. The triangles of M that overlap with vertices of M_i are referred to as the overlapped triangles of M.

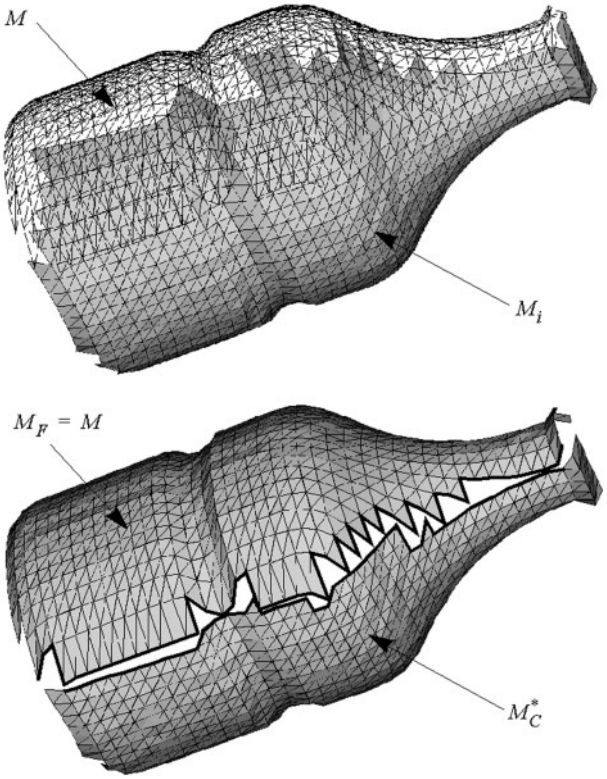
The mesh whose overlapped triangles have the largest average perimeter is referred to as the coarse mesh, M_C, while the other mesh is referred to as the fine mesh, M_F. Before removing the overlapped triangles of M_C, the integration boundaries of M_F are determined. Those boundaries are constituted by those edges that link the exterior vertices of M_F that overlap with triangles of M_C.

Finally, the overlapped triangles of M_C are removed, giving rise to a new mesh M_C^*. After this process, some interior vertices of M_C may have become exterior vertices of M_C^*. The edges that link these new sets of exterior vertices are considered to be the integration boundaries of M_C^*.

Figure 5 shows an example of two overlapped meshes and the result after eliminating the overlapped triangles. Integration boundaries are indicated by thickened polylines.

3.2 Boundary projection and triangulation

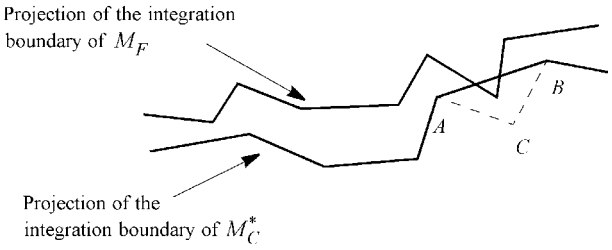
At this stage, the integration boundaries found above are projected over a reference plane orthogonal to a projection direction δ. Vector δ is computed as the average direction of the normal vectors associated with those triangles of M_F that have some of their



5 *Top*: Overlapped triangular meshes representing new range image (M_i) to be integrated with current model M; both meshes have same resolution, although boundaries of both meshes have coarser triangles due to oblique orientation of surface with respect to sensor. *Bottom*: Triangular meshes after eliminating overlapped triangles of M_i

edges belonging to the integration boundaries. Notice that, since 3D points defining triangular mesh M_i belong to a range image obtained from viewing direction **D** (the last scanned range image that will be integrated with the current model M) in the worst case, when all the boundaries of M_i overlap with M, the projection direction δ will almost be the same as the viewing direction **D**. In other words, independently of the complexity of the surface geometry of the scanned object there is always a valid projection direction δ.

It is possible that some edges of the projected polylines that define the integration boundaries of M_C^* intersect some edges of the projected polyline corresponding to the integration boundaries of M_F. In this case, every edge belonging to the projected boundaries of M_C^* that intersects the projected boundaries of M_F is replaced by the other two edges of its corresponding triangle. Figure 6 shows an example where two non-overlapped vertices define an



6 Integration boundaries to be triangulated by means of constrained 2D Delaunay algorithm

edge that intersects the projected boundary of M_{F}. In this case, edge (A, B) is replaced by edges (A, C) and (B, C) (i.e. the other edges that define triangle ABC). This process is applied iteratively until all the intersections are removed. In this way, it is guaranteed that no new vertices are inserted and that the original topology is preserved.

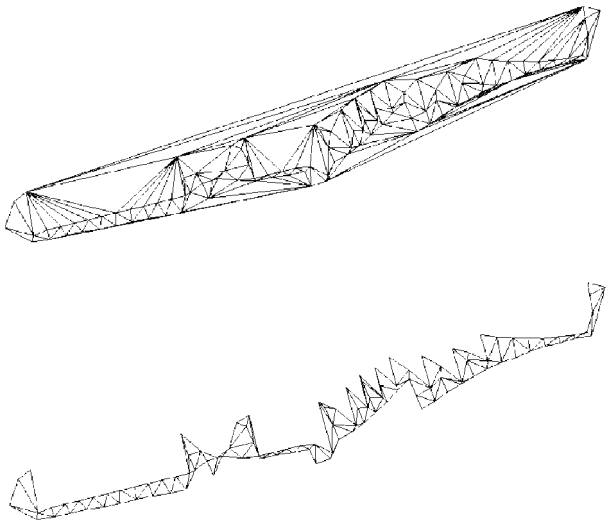
After all the intersections between the projected boundary edges have been removed, the points that define those boundaries are triangulated by means of a constrained 2D Delaunay algorithm.[18] The constraints of this triangulation are the edges corresponding to the projected integration boundaries. The result of the triangulation is a 2D triangular mesh whose convex hull encloses the projected boundaries (Fig. 7(*top*)).

Finally, the exterior edges of the 2D triangulation that do not correspond to projected integration boundaries are removed, except for those cases in which an edge joins a vertex of the integration boundary of M_{F} with a vertex of the integration boundary of M_{C}^*. Figure 7 illustrates the previous triangulation and removal stages.

After a triangular mesh whose exterior edges are projected integration boundaries has been obtained, the next stage maps each 2D triangle to the original 3D space.

3.3 Triangle backprojection

In this final stage, the 2D triangular mesh obtained is mapped back to the original 3D space, in this way merging meshes M_{F} and M_{C}^*. This stage simply maintains the topology generated in the 2D projection. In order to do this, first it is necessary to remove in the original 3D space those triangles of M_{C}^* that have been suppressed during the previous boundary projection stage owing to intersection between projected boundaries (see Fig. 6). Afterwards, the 2D triangular mesh is backprojected to the 3D space. Figure 8 shows the result of integrating the two



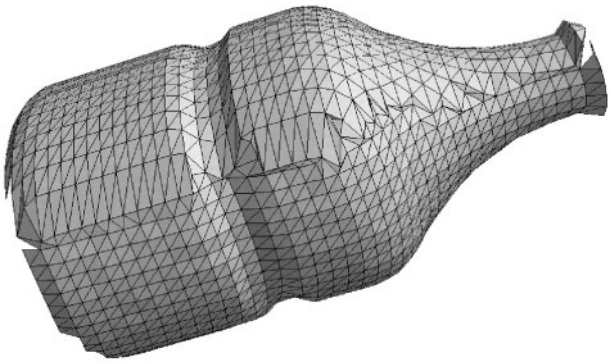
7 *Top*: 2D triangular mesh generated through constrained triangulation of projected boundaries. *Bottom*: Triangular mesh obtained after removing all boundary edges that are not integration boundaries

triangular meshes that have been used as example throughout these sections.

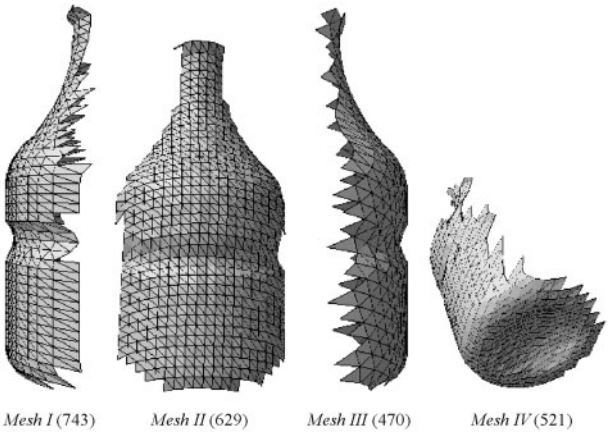
The above-mentioned stages–boundary projection, triangulation and backprojection–do not introduce additional error, as they only perform geometrical manipulation of the meshes to be integrated. Although new topologies are created, the spatial positions of the vertices are not modified. This is another difference in comparison with most of the previous approaches, where new vertices are introduced during the integration process.

4 EXPERIMENTAL RESULTS

This section presents the experimental results of the proposed integration algorithm considering both real



8 Final result obtained after integrating M and M_{i}: triangular mesh will be integrated with further range images

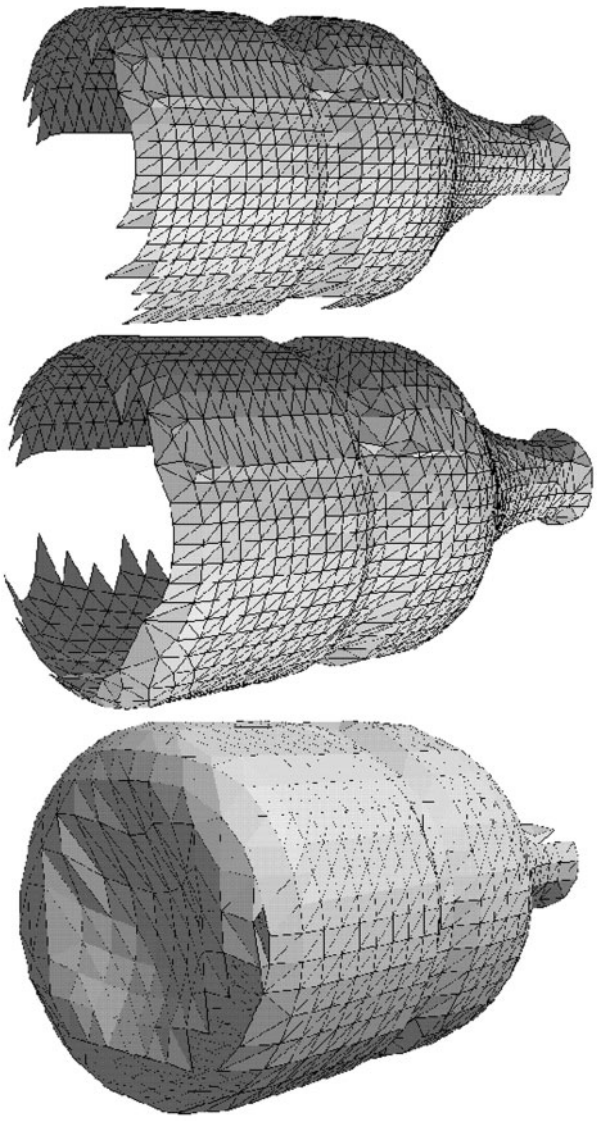


9 Original triangular meshes integrated in Fig. 10; number of vertices included in every mesh is shown in brackets; triangular meshes I, II and III are shown as observed from same point of view

and synthetic multiresolution range images. All CPU times were measured on a SGI Indigo II with a 175 MHz R10000 processor.

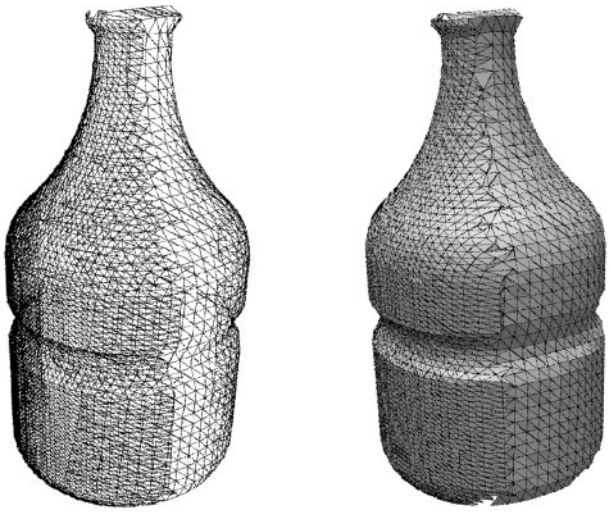
Figure 10 shows the incremental integration of four synthetic range images approximated by the triangular meshes shown in Fig. 9. Figure 10(*top*) shows the triangular mesh obtained after integration of the first two range images, with 743 and 629 vertices each. The CPU time to compute the three stages of the whole integration process was 0.62 s. The resulting triangular mesh contains 1218 vertices; this triangular mesh was integrated with a third range image that was approximated with a mesh containing 470 vertices. The result of the integration is shown in Fig. 10(*middle*). The CPU time to compute the integration process was 0.53 s. The integrated triangular mesh contains 1505 vertices. Finally, Fig. 10(*bottom*) shows the result when a fourth range image, approximated by a triangular mesh with 521 vertices, is added. This last integration took 0.69 s.

Range images of the same object approximated with triangular meshes of different resolutions have also been generated and integrated. Figure 11 shows the result of integrating two synthetic range images represented by triangular meshes of different resolutions (2976 and 629 vertices, respectively). Figure 11(*left*) shows a wireframe model in which the overlap between the original triangular meshes to be integrated can be appreciated. Figure 11(*right*) shows the triangular mesh (3581 vertices) obtained by integrating the two given meshes. This final triangular mesh was obtained in 2.2 s.



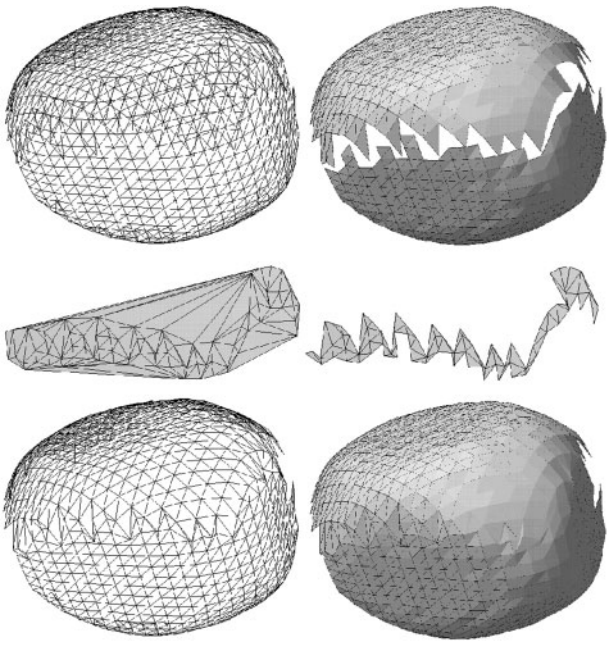
10 *Top:* Triangular mesh obtained after integrating two views. *Middle:* Addition of new view to previous result. *Bottom:* Fourth view is integrated with previous resulting mesh. Latter corresponds to integration of three preceding meshes

Figure 12 shows different stages of the integration of two synthetic range images approximated with triangular meshes of 398 and 368 vertices, respectively. The triangular mesh resulting from that integration contains 684 vertices. The CPU time to compute the three stages of the integration process was 0.7 s. The triangular mesh obtained was integrated with a new range image, the latter approximated with a triangular mesh of 486 vertices. The result of the integration is shown in Fig. 13(*left*). The triangular mesh obtained after the integration process contains 1063 vertices and was obtained in 0.79 s.

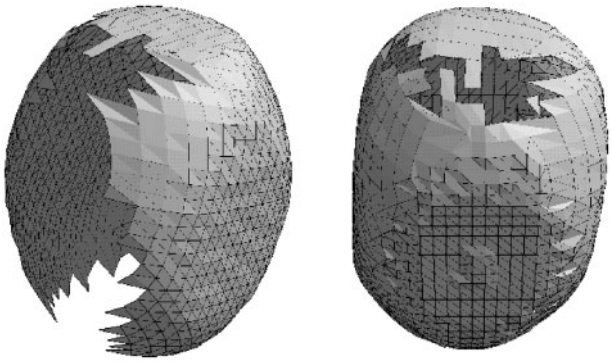


11 *Left*: Overlapped triangular meshes to be integrated (triangular meshes with different resolutions). *Right*: Triangular mesh obtained after integration process

Finally, a new triangular mesh with 496 vertices was integrated with the previous one (containing 1063 vertices). The CPU time of the integration process was 1.1 s (Fig. 13(*right*)).

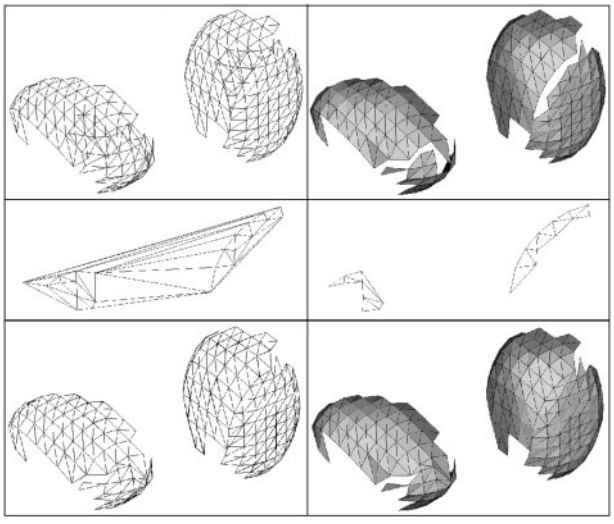


12 *Top left*: Overlap of original triangular meshes to be integrated. *Top right*: Generation of integration boundaries by eliminating overlapped triangles of M_i. *Middle left*: 2D triangulation of projected boundaries. *Middle right*: Triangular mesh obtained after removing all boundary edges that are not integration boundaries. *Bottom left*: Resulting triangular mesh obtained after integrating M and M_i. *Bottom right*: Rendering of resulting mesh

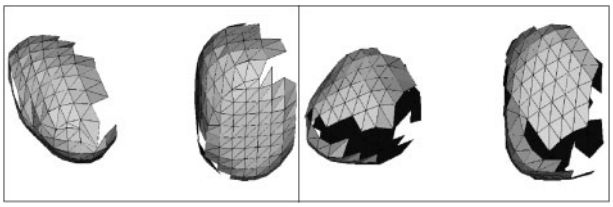


13 *Left*: Triangular mesh obtained after incremental integration of triangular meshes that approximate three range images. *Right*: Triangular mesh obtained after integration of new range image with previous result

Figure 14 shows an example of incremental integration of two synthetic range images obtained from a scene that contains two separate objects. The integrated mesh contains 212 vertices and was obtained in 0.07 s (Fig. 14(*bottom*)). The triangular mesh obtained was integrated with other two range images from the same scene. Figure 15(*left*) shows the triangular mesh obtained after integrating a new mesh of 131 vertices and the mesh shown in



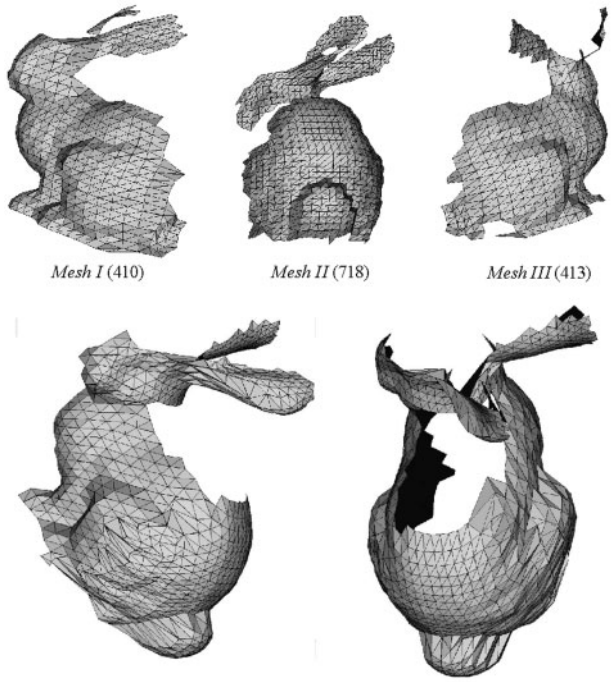
14 *Top left*: Original meshes to be integrated. *Top right*: Integration boundaries obtained after eliminating overlapped triangles. *Middle left*: 2D triangular mesh of integration boundaries. *Middle right*: Triangular meshes obtained after removing boundary edges that are not integration boundaries. *Bottom left*: Resulting triangular mesh obtained after integration process (in wireframe). *Bottom right*: Rendering of resulting mesh



15 *Left*: Triangular mesh obtained after integration of new range image with resulting triangular mesh presented in Fig. 14(*bottom*). *Right*: Triangular mesh obtained after integrating fourth image

Fig. 14(*bottom*). The CPU time to compute the three stages of the integration algorithm was 0.11 s. The triangular mesh of 242 vertices obtained was integrated with a fourth range image, the latter being approximated with a triangular mesh of 110 vertices. The result, shown in Fig. 15(*right*), is a triangular mesh of 340 vertices, obtained in 0.08 s. Notice that the proposed technique also produces valid results when the scene contains more than a single object. As indicated in Fig. 14(*middle*), although the constrained 2D Delaunay algorithm generates a single 2D triangulation (i.e. the convex hull), isolated planar meshes are obtained after removing exterior edges that do not correspond to projected integration boundaries. These isolated meshes (Fig. 14(*middle left*)) integrate the views corresponding to the different objects.

Finally, Fig. 16 shows the results after integrating different real range images of a single object downloaded from the Stanford 3D scanning repository. These images have been approximated by triangular meshes of different resolutions (410, 718 and 413 vertices, respectively). Figure 16(*bottom left*) shows



16 *Top*: Three different real range images approximated by triangular meshes of different resolutions: number of vertices shown in brackets. *Bottom left*: Triangular mesh obtained after integration of first two views. *Bottom right*: Resulting mesh after integration of third view

the triangular mesh of 746 vertices obtained from the integration of the first two views. This mesh was computed in 0.24 s. Figure 16(*bottom right*) shows the result when the third view is integrated. This final mesh contains 897 vertices. The integration process took 0.46 s. All the above experimental results are summarized in Table 1.

Table 1 Summary of experimental results for examples containing single object

Input triangular mesh	No. of vertices	No. of vertices in the final mesh	Total time (s)
Mesh I, Fig. 9	743	1,218	0.62
Mesh II, Fig. 9	629		
Mesh (I, II), Fig. 9	1,218	1,505	0.53
Mesh III, Fig. 9	470		
Mesh (I, II, III), Fig. 9	1,505	1,817	0.69
Mesh IV, Fig. 9	521		
High resolution, Fig. 11	2,976	3,581	2.2
Low resolution, Fig. 11	629		
Mesh I, Fig. 12	398	648	0.7
Mesh II, Fig. 12	368		
Mesh (I, II), Fig. 12	648	1,063	0.79
Mesh III, Fig. 13	486		
Mesh (I, II, III), Fig. 13	1,063	1,415	1.1
Mesh IV, Fig. 13	496		
Mesh I, Fig. 16	410	746	0.24
Mesh II, Fig. 16	718		
Mesh (I, II), Fig. 16	746	897	0.46
Mesh III, Fig. 16	413		

Zipper, a public version of the algorithm presented by Turk and Levoy,[14] was also tested on the same examples shown throughout this section. Zipper produced valid integrated meshes with CPU times of the same order of magnitude as the proposed technique for all the examples containing a single object. However, no triangular mesh was generated for the example containing two separate objects (Fig. 15). Zipper was designed to integrate range images of similar resolution obtained from a single object.

5 CONCLUSIONS

An efficient incremental algorithm for integrating overlapped registered range images acquired from different points of view was presented. Every new range image, which is represented by a 3D triangular mesh, is merged with a second triangular mesh that represents the current reconstructed model. The merging process consists of three stages.

The first stage finds out the area of overlap between the new triangular mesh and the reconstructed model. A BSP tree loaded with all the triangles of the reconstructed model is used to locate efficiently which of those triangles are close to the vertices of the new mesh. Overlap polytopes defined by those triangles are used to detect the vertices of the new mesh that overlap with the reconstructed model. The overlapped triangles corresponding to the coarsest mesh are removed.

In the second stage, the boundary created after removing the previous triangles, along with its adjacent boundary in the unaltered mesh are projected onto a reference plane and triangulated with a 2D constrained Delaunay algorithm. The reference plane is orthogonal to a direction found by averaging the normal vectors associated with the triangles that have some of their edges belonging to the integration boundaries found before. Finally, the last stage projects the previous 2D mesh back to the 3D space, leading to a 3D triangular mesh that stitches the new range image up to the current reconstructed model.

The proposed technique allows the dynamic integration of new range images as they are acquired. Therefore, it is suitable for integrating images obtained by a range sensor that is being moved over a complex scene. Range images of different resolutions can be handled owing to the use of overlap polytopes in the overlap detection stage and conventional 2D constrained Delaunay triangulations for generating the triangular patches that produce the final merging. The proposed technique is more efficient than previous ones thanks to the use of BSP trees and 2D constrained Delaunay triangulations. Previous techniques apply more time-consuming 3D triangulations or triangulate all the points that belong to the area of overlap between the new range image and the currently reconstructed model.

Future work will consist of determining efficient strategies for smoothing the union between integrated regions through spatial filters that take into account the positions of the original points as well as the shape of the integrated surfaces. Another interesting research line consists of extending the proposed algorithm in order to be able to integrate textured images associated with the given range images.

ACKNOWLEDGEMENTS

This work was partially supported by the Spanish Ministry of Education and Science under projects DPI2004-07993-C03-03 and TRA2004-06702/AUT. The first author was supported by the Ramón y Cajal Program.

REFERENCES

1 Besl, P. J. and McKay, N. D. A method for registration of 3-D shapes, *IEEE Trans. Pattern Anal. Machine Intell.*, 1992, **14**(2), 239–256.
2 Zagorchev, L. and Goshtasby, A. A mechanism for range image integration without image registration, Proc. IEEE Int. Conf. on *3-D Digital Imaging and Modeling*, Ottawa, Canada, June 2005, IEEE, Piscataway.
3 Hilton, A. Scene modeling from sparse 3D data. *Image Vision Comput.*, 2005, **23**, 900–920.
4 Bottino, A. and Laurentini, A. What's NEXT? An iteractive next best view approach. *Pattern Recognition,* 2006, **39**, 126–132.
5 Tubic, D., Hebert, P. and Laurendeau, D. A volumetric approach for interactive 3D modeling. *Computer Vision Image Understand.*, 2003, **92**, 56–77.
6 Boissonnat, J. Geometric structures for three-dimensional shape representation. *ACM Trans. Graphics*, 1984, **3**(4), 266–286.
7 Edelsbrunner, H. and Mücke, E., Three-dimensional alpha shapes. *ACM Trans. Graphics*, 1994, **13**(1), 43–72.
8 Hoppe, H., DeRose, T., Duchamp, T., McDonald, J and Stuetzle, W. Surface reconstruction from

unorganized points, Proc. *Computer Graphics (SIGGRAPH)*, July 1992, ACM Press, New York, pp. 71–78.

9 Bajaj, C., Bernardini, F. and Xu, G. Automatic reconstruction of surfaces and scalar fields from 3D scans, Proc. *Computer Graphics (SIGGRAPH)*, 1995, ACM Press, New York, pp. 109–118.

10 Liao, C. and Medioni, G. Surface approximation of a cloud of 3D points. *Graph. Models Image Process.*, 1995, **57**(1), 67–74.

11 Masuda, T. Registration and integration of multiple range images by matching signed distance fields for object shape modeling. *Computer Vision Image Understanding*, 2002, **87**, 51–65.

12 Soucy, M. and Laurendeau, D. Multi-resolution surface modelling from multiple range images, Proc. IEEE Int. Conf. on *Computer Vision and Pattern Recognition*, 1992, IEEE, Piscataway, pp. 348–353.

13 Soucy, M. and Laurendeau, D. A dynamic integration algorithm to model surfaces from multiple range views. *Mach. Vision Appl.*, 1995, **8**, 53–62.

14 Turk, G. and Levoy, M. Zippered polygon meshes from range images, Proc. *Computer Graphics (SIGGRAPH)*, 1994, ACM Press, New York, pp. 311–318.

15 Pito, R. Mesh integration based on co-measurements, Proc. IEEE Int. Conf. on *Image Processing*, Lausanne, Switzerland, 1996, IEEE, Piscataway.

16 Ju, X. *et al.* Integration of range images in a multi-view stereo system, Proc. 17th Int. Conf. on *Pattern Recognition*, Cambridge, UK, August 2004, IEEE, Piscataway.

17 Khan, I., Okuda, M. and Takahashi, S. Regular 3D mesh reconstruction based on cylindrical mapping, Proc. IEEE Int. Conf. on *Multimedia and Expo*, Taipei, Taiwan, 2004, IEEE, Piscataway.

18 Shewchuk, J. R. Triangle: engineering a 2D quality mesh generator and Delaunay triangulator, Proc. First ACM Workshop on *Applied Computational Geometry*, Philadelphia, May 1996, ACM Press, New York, pp. 124–133.

19 De Berg, M., van Kreveld, M., Overmars, M. and Schwarzkopf, O. *Computational Geometry: Algorithms and Applications*, 2000 (Springer, Berlin).