The Visual Computer

The *Marching Intersections* algorithm for merging range images

Claudio Rocchini,
Paolo Cignoni,
Fabio Ganovelli,
Claudio Montani,
Paolo Pingi,
Roberto Scopigno

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", Consiglio Nazionale delle Ricerche, Via G. Moruzzi 1, 56124, Pisa, Italy
E-mail: Claudio.Montani@isti.cnr.it

Published online: 4 March 2004
© Springer-Verlag 2004

A new algorithm for the integration of partially overlapping range images into a triangular mesh is presented. The algorithm consists of three main steps: it locates the intersections between the range surfaces and a reference grid chosen by the user, then merges all nearly coincident and redundant intersections according to a proximity criterion, and, finally, reconstructs the merged surface(s) from the filtered intersection set.
Compared with previous methods, which adopt a volumetric approach, our algorithm shows lower computational costs and improves the accuracy of the surfaces produced. It takes into account the quality of the input measurements and is able to patch small holes corresponding to the parts of the 3D scanned object that were not observed by the acquisition device.
The algorithm has been tested on several datasets of range maps; graphical and numeric results are reported.

Key words: 3D scanning – Range map – Range image merging – Volumetric approach to fusion – Marching Intersections

Correspondence to: C. Montani

1 Introduction

The need for 3D digital models of real objects is rapidly increasing in many applications, from reverse engineering to the authoring of 3D virtual worlds. An increasing availability of 3D scanning devices has responded to this need. The selection of *range scanners* currently offered in the market is wide not only in terms of manufacturers but also the different technologies used (active vs. passive devices, casting laser light or structured normal light), scanning resolution and accuracy, and cost. 3D scanning a real object is not as easy as taking a photograph; the construction of a 3D digital replica requires a set of sequential and correlated phases:

3D scanning: The proper input step. Range sensors capture 3D surface measurements with respect to a 2D image plane. For each *view* of the object a 2.5D range image is obtained.[1] Multiple view range images are required to capture the entire surface of the object.

Data registration: The range data are aligned to transform all the measurements into a common Euclidean space.

Data fusion (or merge): All the data are merged into a single 3D surface model. In this phase, the small holes resulting from the fact that parts of the input model are invisible to the acquisition sensor or the small topological anomalies due to acquisition noise must be corrected.

Other processing steps could be directly connected to the creation of the 3D digital model depending on the complexity of the object to be acquired, the requirements of the target application, and the quality of the final model. These further activities are not correlated or sequential, but they can be very important for the acquisition process and/or for improving the final product.

Acquisition planning: The selection of the optimal set of views for the acquisition of the object is important. The availability of a semiautomatic tool for the identification of the next views to capture the whole surface would simplify and speed up the entire process.

Surface properties acquisition: In many applications, the acquisition of the geometry of the object is not sufficient to build a digital model. Color and texture information have to be collected and mapped onto the object surface [3, 11, 29].

[1] A range image contains the depth information of each image point in the viewing direction (direction toward the sensor).

Model simplification and detail preservation: Range scanners return a huge number of samples, and, as a consequence, heavy, noninteractive models are built. The simplification of the geometric complexity of the surfaces is considered a mandatory step and should be able to manage huge triangle meshes [6]. Simplification often leads to a loss of pictorial and geometric (high-frequency) details; appropriate texture and bump mapping techniques have to be developed to overcome these hitches, which can seriously affect the quality of the digital model [5].

All the activities previously summarized require high processing times and/or heavy manual intervention. Nowadays, the software pipeline between the 3D range scanning and the final digital models is doubtless the bottleneck for so-called *3D photography*: solutions for some of the stages above are still missing, some stages are not sufficiently fast, and the overall complexity of the software needed makes the process still excessively complex for potential, non-CG-expert users.

In this work we are concerned with the *data fusion* step. We present a new algorithm for the simultaneous fusion of multiple, partially overlapping range images. Our algorithm first locates the intersections between the range surfaces and a reference grid; the grid resolution is generally chosen by the user, who selects a grid cell size related to the scanning resolution. The grid is not explicitly represented because we do not store or maintain distance values on the grid nodes, as is usual for most volumetric methods. In the second step the method merges all nearly coincident and redundant intersections, according to a proximity criterion that depends on the grid resolution and also taking into account the quality associated with the different parts or observations of the input range maps. Finally, the algorithm reconstructs the merged surface(s) from the filtered intersection set, adopting a recently defined approach called *Marching Intersections (MI)* [28], which is the reciprocal of the classical Marching Cubes (MC) [20] technique: for each (implicit) cell of the grid, the MC configuration is not determined by the configuration of the cell vertices but directly from the disposition of the active intersections on the cell edges. The topology of the surface section is retrieved from the standard MC lookup table and geometry comes for free because we hold, by construction, the intersections of the surface with the cell edges.

Due to the regular reference grid superimposed on the input scene, our algorithm adopts a *volumetric* approach. However, compared with other algorithms of this class, we claim that our proposal produces surfaces that are more accurate with respect to the ideal union of the input range maps, and it is more efficient in terms of processing time and memory usage.

The rest of the paper is organized as follows. We first summarize the related work in Sect. 2. We then describe our proposal in Sect. 3. Experimental results are presented in Sect. 4, and our conclusions are reported in Sect. 5.

2 Related work

Papers dealing with the problem of surface reconstruction from range images very often classify the different algorithms proposed in the literature into two large groups: methods working on unorganized sets of 3D points and methods operating on 2.5D range images. The first algorithms are often considered a generalization of the latter ones.

The methods starting from unorganized or scattered 3D points [21] assume that: (a) given a point p, its n nearest surface neighbors can be located by finding its n nearest 3D neighbors, (b) the density of data points should be quite uniform over the surface, and (c) the points are measured with the same accuracy. According to Soucy and Laurendeau [31], these assumptions are too restrictive for using these methods to integrate sets of multiple range images.

On the other hand, the algorithms working on range images can take advantage of the connectivity relationships between the observations and of the reliability of the measurements according to the characteristics of the scanning device. These methods are aware of the fact that parts of the input scans overlap and the data are possibly affected by calibration or registration errors. Moreover, even though some of the existing algorithms directly operate on range points, the construction of a 3D range surface starting from a 2.5D range image is quite simple. This construction, based on a step discontinuity constrained triangulation algorithm [15], turns out to be easily feasible also in the cases of a nonuniform data distribution. For these reasons we do not include in our short review of related methods algorithms operating on scattered 3D data [21].

Three main approaches have been proposed for the integration of range images:

Mesh integration: Range surfaces (or images) are integrated by removing the redundant parts. These methods are generally indicated as *interpolating* methods because the vertices of the resulting mesh(es) are a proper subset of the vertices of the input range surfaces.

Volumetric fusion: A regular grid is ideally superimposed on the 3D scene; a scalar value is associated with each node of the grid, generally a signed (possibly averaged) distance between the node and the 3D surface(s). Finally, a surface fitting algorithm (e.g., Marching Cubes [20]) is used for the reconstruction of the zero-distance surface(s). In most cases, the vertices of the output mesh(es) only approximate the input range surfaces; for this reason volumetric methods are generally called *approximate* methods.

Deformable models: The 3D object model is obtained from the dynamic deformation of a simple, topologically equivalent 3D surface. The initial object shape is deformed until it reaches the equilibrium state.

Mesh integration algorithms adopt very different approaches, but they are generally characterized by high computational costs and by the need to perform postprocessing smoothing operations due to the integration of data suffering from possible imperfect registration. Examples of mesh integration algorithms are as follows.

Ratishauser et al. [27] integrate triangular meshes by completely retriangulating overlapping areas. Since the triangulation is in 3D, special care must be taken to ensure the mesh does not fold over itself.

Turk and Levoy [34] merge overlapping triangulated meshes using a *zippering* approach. The overlapping meshes are eroded and the boundary correspondences are reconstructed by means of operations in 3D space. A local 2D constrained triangulation is then used to join the mesh boundaries.

Soucy and Laurendeau [31, 32] describe an algorithm based on Venn diagrams to individuate a partition of the surface of an object by finding areas of overlap of the acquired range meshes. For each disjoint part, a virtual viewpoint is defined and a nonredundant integrated triangulation is obtained. Finally, all these local triangulations are connected to yield a global integrated one.

The algorithm by Pito [24] identifies, for each couple of input meshes, all the triangles that sample the same surface patch; the algorithm keeps only one of them – the one acquired with the highest confidence. The redundant triangles are removed, and neighborhood relationships are established between the remaining patches.

Häusler and Karbacher [13] perform *vertex insertion*, *gap bridging*, and *surface growth* operations to merge the input meshes. Merging is followed by *edge swap* operations to obtain more balanced triangulations. The authors do not supply many details on the merge process, but they stress a new postprocessing smoothing filter. This filter, based on the concept of local surface curvature, turns out to be particularly useful in cases of registration error.

More recently, Bernardini et al. [2] presented an algorithm working on range data instead of range surfaces. The data are not completely scattered because for each point the knowledge of a normal to the surface is assumed. Starting with a seed triangle, the method pivots a ball around each edge on the current mesh boundary until a new point is hit by the ball. The edge and the point define a new triangle that is added to the mesh. Interestingly, not all of the input points necessarily become vertices of the integrated mesh, and this reduces the noise of the range data.

As regards the *volumetric fusion* approach (originally introduced by Hoppe et al. [16] for surface reconstruction from unorganized points), the reference work is the paper by Curless and Levoy [8]. We provide more details about this algorithm because we will often refer to it in the subsequent discussion. A volume space is defined encompassing all of the range data. Each node of the grid holds the average of the distances between the node and the range maps *close* to it along the sensor line of sight. Distances are signed, representing the positions of the nodes with respect to the range surface, and weighted. The weight allows one to take into account the reliability of the data based on the characteristics of the acquisition sensor. In the mentioned work, the weight is also used to limit the number of voxels written in the range map-to-volume conversion process: only the voxels *close* to (both sides of) the range maps are initialized. This reduces the number of active voxels in the volume and limits possible interference between *far* surfaces. To reduce memory use, Curless and Levoy represent the volume dataset by means of a run length encoding (RLE) scheme. The possible holes in the resulting isosurface, due to the fact that parts of the model are invisible to the acquisition sensor, are plugged a priori (i.e., directly into the volume) by means of a space carving technique. All the voxels are initialized as *unseen* voxels. The scan conversion of the the range maps writes the *visible* voxels. All the voxels lying between a visible voxel and the sensor are marked as *empty* voxels. The iso-surface separating unseen and empty voxels is the surface plugging the holes.

Roth and Wibowoo [30] use a very similar approach, but they achieve further reductions in computational costs by adopting hash data structures to address occupied voxels and linked lists for an efficient traversal of the volume.

Pulli et al. [25] build an octree starting from a set of registered range maps. The leaves of the octree can be external, internal, or on the boundary of the object. The returned meshes are obtained by generating triangles between the external nodes and those on the boundary. Output surfaces are smoothed by the use of Taubin's algorithm [33].

Wheeler et al. [36] represent the volume by means of an octree; distances are not averaged but just sampled from a subset of plausible input range maps (consensus surfaces). In this way most of the problems due to the presence of noise in the input data are avoided.

To limit the costs of the representation of large volumes, Hilton and Illingworth [14] use a multiresolution geometric fusion algorithm. A hierarchical discrete volumetric structure for implicit surface representation with bounded error is introduced. This approach constructs a low-resolution discrete representation in smooth regions of the surface and a high-resolution one in regions of high surface detail.

Also, the algorithms adopting a volume fusion approach present high computational costs, mainly due to the initialization of the distance volume. Volumetric algorithms are not as precise as mesh integration ones (approximation of the integrated meshes), but they are definitely more robust (for example, in returning watertight surfaces).

Chen and Medioni [7] presented one of the first fusion algorithms based on a *deformable model* approach. An inflating balloon model (initially a sphere) is adaptively inflated starting from a set of registered range images. The process terminates when the sphere reaches the shape of the object to be reconstructed.

In [12], the initial shape is obtained by means of a volumetric approach with a very low resolution of the distance grid. The deformable model is progressively refined by means of adaptive subtriangulations and deformations. These latter are obtained by means of springs that move the vertices toward the real surface and keep the triangulation regularly shaped.

Other methods using similar approaches can be found in [9] and [1].

The algorithms adopting a deformable model approach have the advantage of avoiding difficult integration of misaligned surfaces; on the other hand they have high computational costs, they do not admit changes in the starting topology during the deformation step, and they can produce self-intersections in the case of closely spaced features.

With respect to the taxonomy introduced above, the algorithm we propose in this paper belongs to the volumetric fusion class. However, we do not construct an intermediate implicit function based on the distance between each node of the grid and the nearest point on the object surface, reducing computational times, and we do not even need to explicitly represent the reference grid, reducing in this way the memory needed.

A last observation about related work: the algorithm is mainly based on the processing of Hermite data, i.e., points of intersection of the range surfaces with the 3D reference grid and associated normals. Hermite data have been used recently for feature-sensitive reconstruction of implicit surfaces [18, 19, 35] from volume datasets. While in the referred papers Hermite data are used together with the 3D distance field representing the implicit surfaces, in our solution Hermite data avoid the need to build and maintain a volume grid.

3 A new volumetric algorithm for merging range surfaces

The input data of our merging algorithm are triangular range surfaces with the following characteristics:

- Input range maps were previously registered [4, 26] and are thus defined in a single Euclidean space.
- The range surfaces have been obtained by means of a step discontinuity constrained triangulation algorithm: a continuous surface between adjacent measurements is assumed if their distance is less than a constant threshold [15].

- A direction is associated with each triangle of the input surfaces; this information, which discriminates between the *interior* and the *exterior* of the object, is simple to obtain because during the acquisition of the data each range map is actually a 2.5D height field between the sensor and the object itself. A signed range surface helps to avoid the fusion of intersection pairs corresponding to close but distinct surfaces (the surfaces, for example, that bound a very thin section of the scanned object).
- According to Curless and Levoy [8], a weight representing the quality of the observation is associated with each input measurement. This parameter takes into account both the residual error of the registration and the error inherent in the scanning data. With respect to the Curless and Levoy method, we do not use the weight to prevent the distance function from extending indefinitely in either direction. Our quality parameters effectively take into account the reliability of the data.
- The simple overlapping of all the range maps does not necessarily constitute a watertight model; this is because complex objects could present small surface sections that cannot be reached by the scanning sensor.

As briefly summarized in the introduction, the algorithm locates the geometric intersections between the range images and the virtual 3D reference grid chosen by the user, it merges the corresponding intersections belonging to different range maps, and, finally, it reconstructs the merged surface(s) by means of the *Marching Intersections (MI)* algorithm. These three main steps are detailed in Sects. 3.1–3.3 below. Section 3.4 presents some solutions to cope with possible ambiguities and small holes generated by a missing sampling of the acquisition sensor.

3.1 Range map discretization

The discretization step consists in detecting all the intersections between the range surfaces and the reference grid. To simplify the computations, all the geometric coordinates of the input range data are immersed in the space of the selected grid; the goal of this scaling transformation is to have all the grid lines lying on integer values; this improves the efficiency of the computations and the management of the intersections between the input meshes and the grid edges.

The choice of the grid size should be clearly related to the spatial resolution of the scanning device, which is not always precisely known. A simple heuristic [30] consists in extracting a sufficiently large set of 3D points from the set of range maps, finding the nearest neighbor point to each of these observations, and finally setting the voxel size to three times the average of these minimum interpoint distances. This ensures that each voxel grid element will likely contain at least one data point.

The discretization of the input range images occurs on a per-face basis. For each input triangle, its axis-aligned bounding box is determined, and then three different conversion steps are performed. For each scan conversion step, we compute the intersections between the triangle and the orthogonal set of grid lines. This process leads to the construction of the 2D dynamic data structures XY, XZ, and ZY containing the intersections between all the input faces and the corresponding grid lines.

The entry (i, j) of the 2D pointer data structure XY, for example, refers to the list of intersections between the input range surfaces and the grid line parallel to the Z axis and passing through the point $[i, j, 0]$. A 2D example of the discretization process is shown in Fig. 1.

Each intersection is represented in the dynamic data structures by means of a record in the form

ic	nm	sg	dr	w

where:

ic	is the numeric value (*intercept*) of the intersection. For the XY data structure, for example, the field holds the z component of the intersection;
nm	is a bit set representing the *name* of the range map(s) the observation belongs to. The size of the bit set is equal to the number of input range maps. This implies that multiple range maps may share the same intersection and also allows the efficient management of data fusion or deletion operations implemented by means of bit set operators;
sg	is the *sign* of the observation; if we suppose we are walking along the grid line onto which the intersection has been individuated, a “+” sign means that we are entering the object, a “−” means we are leaving it;

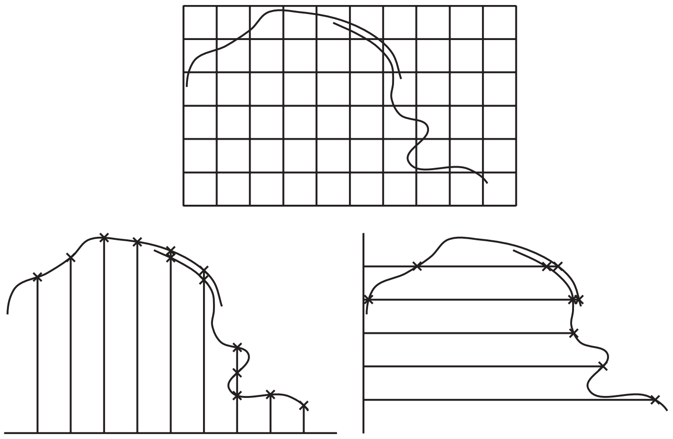


Fig. 1. A 2D example of the discretization process. The two input range curves (*top*) and the two dynamic data structures collecting the vertical (*bottom left*) and horizontal (*bottom right*) intersections of the range curves with the lines of the chosen regular grid

dr is the *direction* field. This field holds the component, along the grid line the intersection belongs to, of the normal to the originating face;[2]

w is the weight associated with the *quality* of the corresponding surface sample. The value is obtained by linear interpolation of the weights of the vertices of the originating triangle.

At the end of the scan conversion process, each list is sorted with respect to the intersection value. Sorting ensures fast detection of nearby intersections and fast search for specific intervals.

It must be remarked that the geometric primitives forming the input range surfaces are read sequentially from secondary memory: there is no need to store all of the range maps in main memory or to maintain topological or connectivity information. This means that, due to the compact representation of the XY, XZ, and YZ pointer data structures and the relative shortness of the intersections lists they subtend, the size of the input surfaces and the number of partially overlapping maps can really be very high. The space complexity in our case is directly dependent on the surface area of the range maps, instead of being proportional to the mesh volume.

[2] In our implementation, *sg* is actually stored as a sign of the *dr* field.

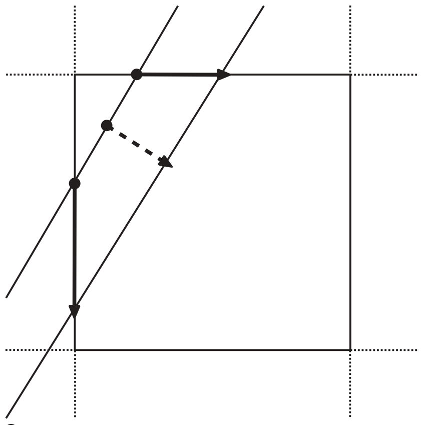
The intersection lists are now ready for the fusion of the nearby intersection pairs and for the reconstruction of the integrated surface(s).

3.2 Merge process

The aim of the second phase of the algorithm is twofold: first, to fuse the multiple range maps discretized in the previous step by merging the corresponding intersections; second, to arrange the intersection data structures in a *Marching Cubes* [20] *compliant* manner, i.e., in such a way that a MC-like surface reconstruction algorithm could be successfully applied. Even though the intersection data structures do not explicitly represent a grid, it is quite easy to reconstruct the *virtual cells* of the grid by simply locating the corresponding edges in the XY, XZ, and YZ data structures. Arranging the data structures in a MC-compliant way means, for example, to ensure the existence of no more than one intersection on each *virtual cell edge*.

The two basic operations that allow us to pursue this goal are the data *fusion* and *removal* actions.

Fusion. The *fusion* operation performs the proper merge step of redundant intersections. Fusion occurs between pairs of consecutive intersections lying on the same grid line. It takes into account the quality of the measurements, the actual distance between the originating range maps, and the



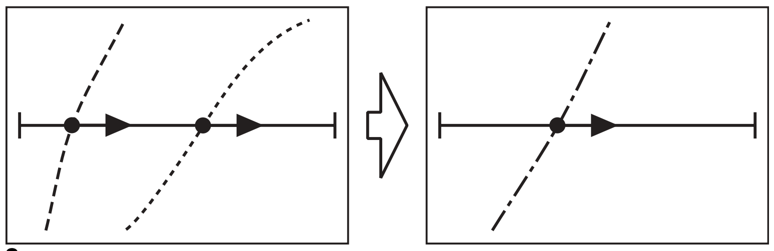


Fig. 2. The *fusion* of intersection pairs is performed along the grid lines (*thick lines*) rather than on the minimum distance directions (*dashed line*)

Fig. 3. An example of *fusion* of two intersections belonging to different range maps

maximum merge distance (*maxdist*) selected by the user.

While the choice of the reference grid should only depend on the resolution of the scanning sensor, the choice of the maximum merge distance has to take into account the residual registration error.

In the merge phase, all the intersection lists are examined: we fuse each couple of consecutive intersections $i1$, $i2$ belonging to different range maps ($nm_{i1} \bigcap nm_{i2} = \emptyset$), with concordant signs ($sg_{i1} = sg_{i2}$), if their distance $\bar{d}(i1, i2)$ is shorter than the maximum merge distance

$$\bar{d}(i1, i2) = \frac{\mid ic_{i1} - ic_{i2} \mid}{\bar{dr}(i1, i2)} \leq maxdist \tag{1}$$

with

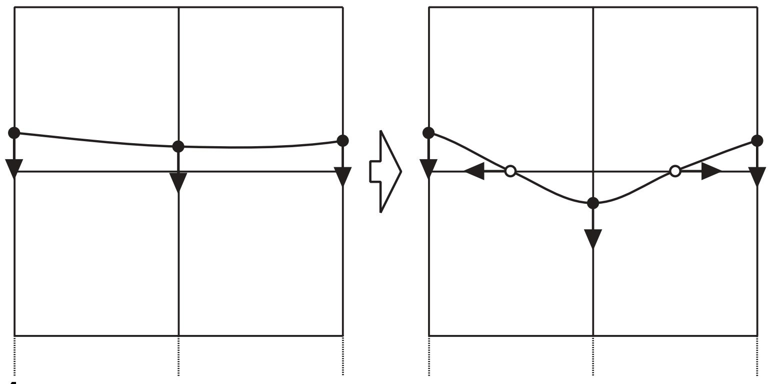
$$\bar{dr}(i1, i2) = \frac{w_{i1} dr_{i1} + w_{i2} dr_{i2}}{w_{i1} + w_{i2}}. \tag{2}$$

The term $\bar{dr}(i1, i2)$ in the previous expression takes into account the fact that fusion is performed along the grid lines rather than on minimum distance directions (Fig. 2).

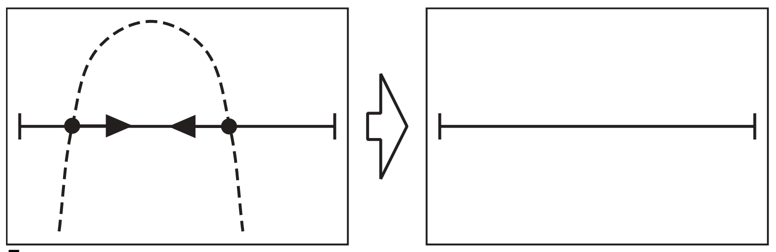
The fusion operation implies the cancellation of $i1$ and $i2$ from the data structure and the creation of the new intersection ir in which the intercept field (ic_{ir}) is given by the weighted average of the corresponding fields, the name is $nm_{ir} = nm_{i1} \bigcup nm_{i2}$, the sign sg_{ir} is concordant with the signs of $i1$ and $i2$, and the direction $dr_{ir} = \bar{dr}(i1, i2)$ as defined in Eq. 2. According to Curless and Levoy [8], also the quality field (w) is incrementally updated. The simple operation described corresponds to merging two concordant surfaces that cross the same virtual cell edge (Fig. 3).

A slightly different behavior is demonstrated by a fusion operation that occurs on intersections lying on the same grid axis but belonging to different virtual cells. In these cases, the fusion operator performs a *shift* of the intersections toward the new average location. During this shift, whenever we pass from a virtual cell to another, a couple of new artificial intersections is inserted on all the perpendicular edges the intersection touches. An example is shown in Fig. 4. Each new intersection is initialized with intercept value undefined (we only know the cell it belongs to), name equal to the name of the originating intersection, sign compatible with the sign of the originating intersection, and direction of movement and quality value set to null.

The insertion of new intersections ensures the consistency of the data structures by simulating a discretization step (i.e., detection of intersections with the grid lines) for those surfaces that slightly change their position due to the merge process.



4



5

Fig. 4. Moving an intersection. If an intersection moves from a (virtual) cell to another, then a couple of artificial intersections is inserted on all the perpendicular (virtual) edges the intersection touches during its movement. From a graphical point of view the movement of the central intersection (*left*) leads the curve to touch the perpendicular edges (*right*; see newly inserted *white dots*)

Fig. 5. Example of the *removal* of two consecutive and discordant intersections belonging to the same range map

Whenever a new intersection is generated, the algorithm analyzes the other intersections on the same virtual edge to verify if a removal operation (described below) can be applied. Most of the new intersections are removed in this way. The locations of the artificial intersections that survived is determined by a simple interpolation method.

Removal operation. The *removal* operation is performed on the currently updated grid edge, i.e., each edge where a new intersection has been created. Moreover, all the grid edges undergo this operation before the merge process starts; this ensures that high-frequency details are removed and aims at transforming the intersection data structures in a MC-compliant set.

During the initial removing phase all the intersection lists are analyzed. A *removal* action is performed on each pair of intersections $i1$ and $i2$ that lie on the same cell edge (that is, if $\lfloor ic_{i1} \rfloor = \lfloor ic_{i2} \rfloor$), belong to the same range map, and have discordant signs. As shown in Fig. 5, this operation corresponds to the removal of the high-frequency details in a range map. In this case, the range map bit index that $i1$ and $i2$ have in common ($nm_{ic} = nm_{i1} \bigcap nm_{i2}$) is subtracted by the examined intersections ($nm_{i1} = nm_{i1} - nm_{ic}$ and $nm_{i2} = nm_{i2} - nm_{ic}$). The intersections whose name field is empty are definitely removed from the structure; otherwise, they maintain their position in the lists, and the intercept, sign, and direction fields are unchanged. Depending on the resulting name field value, two, one, or no intersections are physically deleted. A complete step of the (2D) merge process, in which fusion and removal operations are performed, is shown in Fig. 6.

Unfortunately, the fusion and removal operators are not sufficient to ensure that no more than one intersection lies on each virtual edge, as shown in the

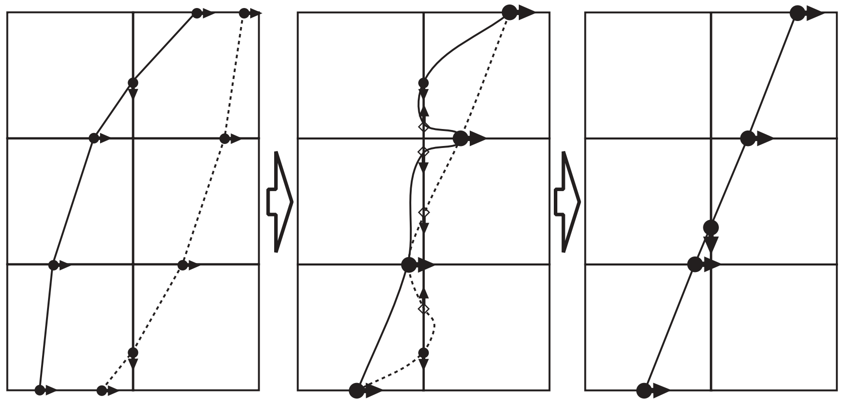


Fig. 6. Complete merge step. Two horizontal intersection pairs lying on the same virtual cell are merged (*uppermost* and *lowermost pairs*). The other two pairs belong to different virtual cells, and therefore their merge generates new intersections (denoted by *diamonds*); these form new pairs that are either merged or removed because of discordant directions

following table in which the relationships between two generic intersections $i1$ and $i2$ belonging to the same edge are discussed.

	$sg_{i1} = sg_{i2}$	$sg_{i1} \neq sg_{i2}$
$nm_{i1} \bigcap nm_{i2} = \emptyset$	fusion	anomaly
$nm_{i1} \bigcap nm_{i2} \neq \emptyset$	anomaly	removal

The cases in which an *anomaly* is detected come from (possibly small) malformations of the original range maps. In these situations, the algorithm maintains more intersections on a virtual edge and postpones their handling to the surface reconstruction step. Explicitly storing the index of the corresponding range map(s) in the *name* bit set field allows for the easy resolution, during the reconstruction step, of the possible ambiguities by analyzing the corresponding range maps. Details are given in the following section.

3.3 Surface reconstruction via Marching Intersections

At the end of the merge process our data structures contain the signed intersections between the integrated surface and the lines of the reference grid. Therefore, we can start the surface reconstruction step.

The basic idea of the *Marching Intersections (MI)* [28] reconstruction method is very simple: the reconstruction of a 3D surface is completely defined if all the *signed* intersections of the surface with the lines of a regular grid are known. The main step of MI is shown in Fig. 7: the reconstruction of the surface parcel contained in a *virtual* cell of the reference grid. As described in Sect. 3.1, the initial scale transformation performed on the range maps allows the nodes of the grid to lie on integer coordinates (i, j, k, ...); given a virtual grid cell it is therefore easy to find the intersections lying on its edges.

The MC-like classification of the vertices of a virtual cell can be obtained easily from the analysis of the intersections existing on its edges. If an edge contains an intersection, then the classification of its vertices depends on the orientation of the intersection, i.e., the value of the sign field sg (Fig. 7). For each classified (i.e., interior/exterior) cell vertex v, the classification of the adjacent vertex v' on an incident edges e is concordant with v if no intersections exist on e or discordant if a single intersection exists with direction compatible with the class of vertex v.

If the intersection configuration along the current cell edges are MC compatible, then we return the corresponding 8-bit binary code. This code allows us to access the standard MC lookup table and to recon-

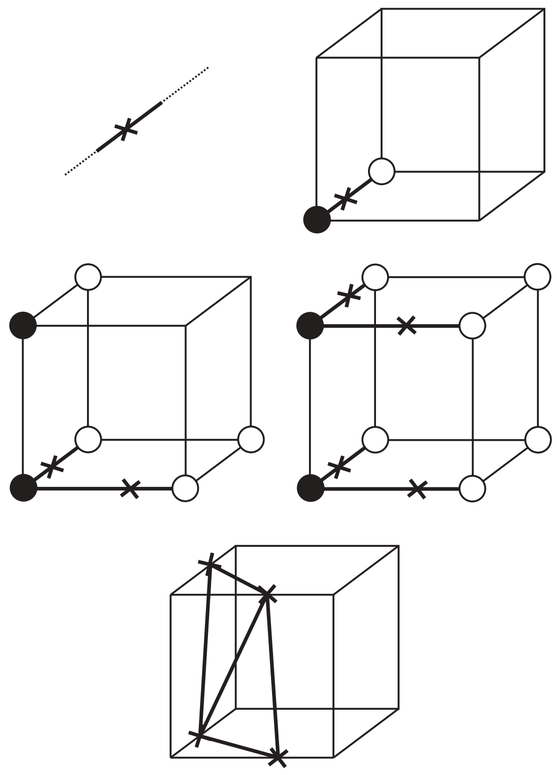


Fig. 7. Reconstruction of a virtual cell and of the corresponding triangular patch performed by the MI method (from *left* to *right*, *top* to *bottom*), starting from a signed intersection on an edge

struct the encoded triangular patch. In our algorithm, we use a MC lookup table that solves the problem of ambiguity in the reconstruction of the triangular patch internal to a grid cell [22].

Different traversal strategies have been proposed for visiting the cell grid in surface fitting. The classic MC approach is in general implemented by adopting an iterative slice-based visit of all the cells of the volume. Another solution visits the cells following a propagation approach [17], tracking the surface from an initial seed cell. This solution has advantages (e.g., it allows one to produce output encoded in triangle strips), but it implies the handling of a huge stack for the addresses of the active cells to be visited (with impact on space and time efficiency). Moreover, a propagation approach is more effective if the output surface is guaranteed to consist of just one component.

The algorithm we designed performs an iterative visit of the intersection lists, which is somehow similar to the slice-based MC visit. We visit, in sequence, the three data structures XY, XZ, and YZ; for each structure, we analyze the entries from left to right and from bottom to top. We do not have to maintain a trace of the processed cells because we adopt the following visiting strategy:

- Visit the XY structure; for each intersection, we process the corresponding cell (searching for possible intersections in the XZ and YZ structures).
- Visit the XZ structure; for each intersection, we process the corresponding cell (searching for possible intersections in the XY and YZ structures) only if it does not contain intersections in the XY structure (because in that case it has been processed in the previous step).
- Visit the YZ structure; for each intersection, we process the corresponding cell only if it does not contain intersections in the other XY and XZ structures (because in that case it has been processed in the previous steps).

All the intersections used in the reconstruction phase (i.e., the intersections being now vertices of some triangle in the integrated mesh) are marked. Moreover, the surface fitting process can fail on some cells (see next subsection); MI stores the addresses of these cells in an auxiliary structure for further processing.

3.4 Solving anomalies and closing holes

The use of the MI algorithm permits one to reconstruct most of the integrated surfaces. However, a noncorrect configuration can be obtained on some cells due to multiple or nonexisting intersections on some cell edges. A 2D example of an irregular cell is shown in Fig. 8, where the scan conversion of the boundary of a range map produces an anomalous intersection that survives the merge process. Moreover, missing intersections can be produced in the case of small holes contained in the surface (frequently produced by the range scanner as a result of parts of the model being unseen by the acquisition sensor). In our experiments the percentage of cells that, after intersection merging, present a correct pattern of intersections and that can be directly reconstructed is between 86% and 94% depending on data quality, number of range maps, and residual registration error.

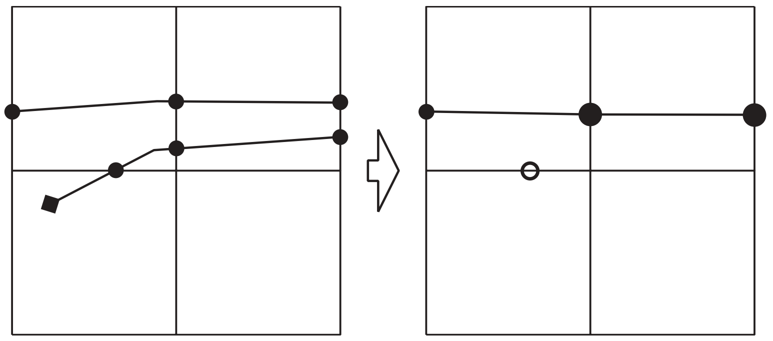


Fig. 8. The boundary of a range surface can generate anomalous intersections (e.g., the *circled intersection*, which remains isolated after some intersection merging actions)

All virtual cells that contain intersections and have not been safely reconstructed by MI in the previous step are processed as follows:

- If all the intersections on the cell edges are marked (i.e., they have been already used for the reconstruction of the surface in the adjacent cells), then the cell is considered to cover part of a hole on the object surface; all the edges of the triangles generated in neighbor cells that lie on faces of the current cell are inserted into an auxiliary edge list that encodes the boundaries of the holes.
- If the cell holds some unmarked intersections and we have multiple possible MC configurations, we test their compatibility with respect to the configurations of the adjacent cells. If just one configuration is acceptable, then this is the solution; in the case of more than one acceptable configuration, the analysis is postponed in order to verify whether the ambiguity of neighboring cells can be resolved. If a complete analysis of the ambiguous cells does not produce any correct configuration, then these cells are treated as parts of holes.

The last phase of the algorithm regards closing holes. The edges inserted into the appropriate list are analyzed: starting from a seed edge, the boundary of a hole is reconstructed by simply connecting edges with compatible extremes. The hole is then triangulated by means of a simple 3D extension of a 2D triangulation algorithm [23]. The only hypothesis is that the boundary of the hole is simple and not self-intersecting. This hypothesis is not too strong if the data to be processed are multiple and accurately selected range maps.

4 Experimental results

The algorithm proposed has been tested on several datasets of range maps. It is one of the algorithms we currently use in our projects for the automatic acquisition of 3D artistic objects. In this context, we have chosen some examples from the *Stanford 3D Scanning Repository*[3] and the data of a gypsum copy of the Ippolita Maria Sforza's bust carved by Francesco Laurana. Some of these examples have already been used for the demonstration of other integration algorithms, and this allows us to compare our method with other existing algorithms.

The results obtained are shown in Table 1 for the datasets *Stanford Bunny*, *Stanford Dragon*, *Stanford Happy Buddha*, and *Ippolita M. Sforza*. For each dataset we have extracted the integrated surface using two different resolutions of the virtual reference grid. The tests have been carried out on a Windows 2000 PC with an Intel Pentium IV 2.53-GHz processor and 1 GB RAM. The times reported are in *minutes:seconds* and include I/O operations.

Figure 9 shows the obtained results from a graphical point of view.

Based on these results, great emphasis has to be given to the time efficiency of our proposal. The algorithm by Curless and Levoy [8] takes 56 and

[3] http://www-graphics.stanford.edu/data/3Dscanrep/

Table 1. The results obtained by the integration of the three datasets (Bunny, Dragon, and Happy Buddha) from the Stanford 3D Scanning Repository and from the reconstruction of the Ippolita Maria Sforza's bust by Francesco Laurana. Times include I/O operations. For each dataset, two different grid resolutions have been selected

Dataset	# Scans	# Input Triangles	Grid size	# Output Triangles	Time (mm:ss.d)	Memory Usage (Mb)
Bunny	10	690 534	$109 \times 107 \times 84$	79 150	00:01.3	3
			$333 \times 338 \times 266$	797 497	00:23.1	30.5
Dragon	71	3 681 894	$271 \times 226 \times 162$	184 423	00:12.9	20.5
			$612 \times 511 \times 367$	1 054 846	01:45.0	151
Happy	58	5 985 189	$224 \times 329 \times 135$	312 200	00:21.3	19
Buddha			$564 \times 831 \times 340$	2 331 421	04:07.0	140
Ippolita	29	10 211 802	$246 \times 264 \times 153$	313 313	00:07.4	36
M. Sforza			$530 \times 569 \times 331$	1 472 153	01:01.0	170

47 min on a 250-MHz MIPS R4400 processor for the integration of the Dragon (grid size $712 \times 501 \times 322$) and Happy Buddha ($407 \times 957 \times 407$) datasets, respectively. If we include a scaling factor to make these times comparable to the times obtained on a more modern architecture, our solution still maintains higher efficiency because the computation of intercepts on the grid lines is faster than the computation of distances along a viewing line. Moreover, our times include the triangulation of small holes. The computing times of the method of Curless and Levoy rise considerably when the *space carving* modality is selected to perform hole filling; the times reported for the same two datasets are in fact 257 and 197 min. Note that the space carving technique used by Curless and Levoy is more powerful than our solution because it permits one to correctly fill also very large holes that do not have a simple boundary. Unfortunately, its computational cost is very high, and so less expensive techniques [10] have been developed recently.

Our method also turned out to be comparable to the performance of the *Ball Pivoting* technique proposed by Bernardini et al. [2]. In this case the comparison cannot be done on the basis of a common grid size, and so we tried different virtual cell sizes to obtain a comparable number of triangles in the output meshes. For the Bunny dataset, for example, *Ball Pivoting* produces 710K triangles in 132 s (I/O + CPU times) on a 450-MHz Pentium II Xeon PC. Our method generates a mesh composed of 800K triangular faces in 23 s.

Table 1 shows that the computational complexity of the method is strictly related to the number of (virtual) cells in the output MI data structure rather than to the number of input range maps or input triangles. The low memory consumption of our algorithm is also shown in Table 1. It depends on the adopted reference grid resolution. In particular, space complexity can easily be limited to being linear in the number of the vertices of the integrated mesh (i.e., the number of intersections at the end of the merge process). For comparison purposes, *Ball Pivoting* requires 43 MB on the Bunny dataset, 114 MB on the Dragon dataset, and 163 MB on the Buddha dataset. Results on space consumption are not reported for the volumetric method of Curless and Levoy [8]; they only say that the run length encoding representation adopted allows for the reduction of memory size to 5%–10% of the rough volume representation. Memory requirements could be further reduced by integrating the range maps in an iterative way rather than performing a simultaneous fusion. Another memory saving approach could be to divide the logical volume into slabs (to be operated separately) and then to simply integrate the resulting meshes.

We stated in Sect. 3.4 that: (a) most of the integrated surface is generated during the first application of the MI algorithm, (b) a small number of triangles is generated with the combinatorial analysis of the cells with ambiguous configurations, and (c) some triangles are produced by the triangulation of the boundaries of the holes (true unseen parts of the model or unsolved ambiguous situations). Empirical results relative to the Bunny mesh are presented in Table 2.

5 Conclusions

We have presented a new and efficient algorithm for the integration of range images. Based on a volume approach, the main characteristics of the meth-

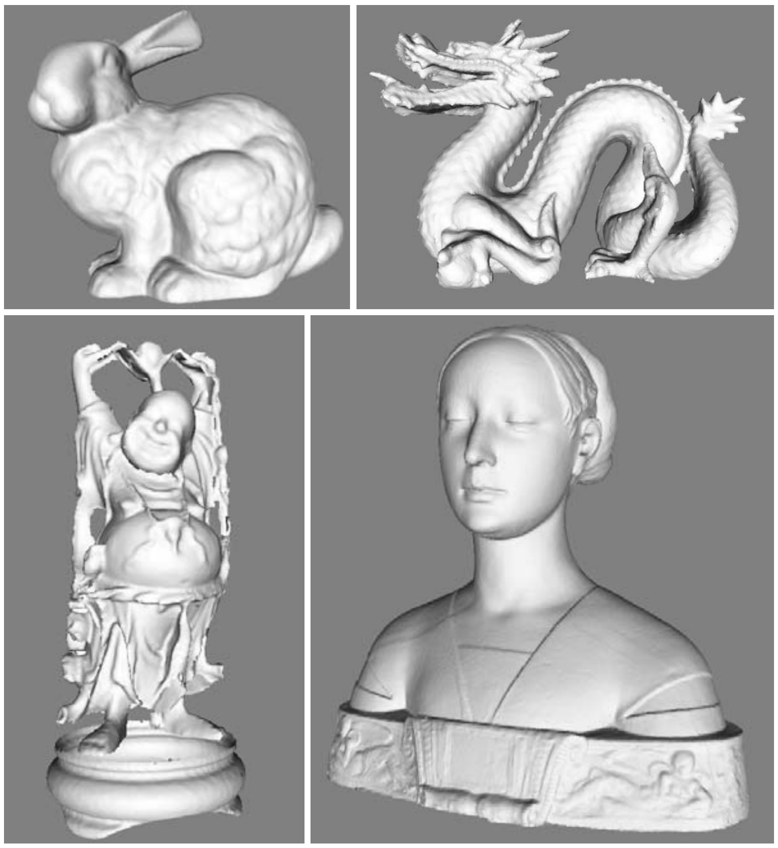


Fig. 9. Some views of the reconstructed models. From *left* to *right*, *top* to *bottom*: Stanford Bunny (a grid size of 142 × 134 × 104 was used), Stanford Dragon (612 × 511 × 367), Stanford Happy Buddha (106 × 264 × 176), and Ippolita M. Sforza (530 × 569 × 331)

Table 2. Percentage of the reconstructed mesh faces produced in the three phases by the MI algorithm. **a** Faces produced by direct reconstruction. **b** Faces produced after combinatorial analysis of adjacent cells. **c** After hole filling

Dataset	Resolution	% case *a*	% case *b*	% case *c*
Bunny	109 × 107 × 84	98.62	0.72	0.66
	333 × 338 × 266	98.28	1.12	0.60

ods are: (a) the location of intersections between the range images and the reference grid rather than the computation of distances in a voxel space and (b) an efficient solution for surface reconstruction.

A main point of this solution is the improved accuracy with respect to previous approaches. In fact, our solution first of all computes the intersection between the input meshes and a set of grid lines; second, it merges corresponding intersection pairs (which means, in some cases, evaluating the geometrical coordinates of the corresponding point, taking into account the quality of the merged intersections); finally, the output surface is produced by reconstructing the mesh topology from a per-cell analysis of the intersection configurations. With respect to other voxel-based methods, we do not resample mesh geometry (taking into account voxel-based distances); this means that instead of performing a double conversion step (from boundary to voxel space, and again from voxel space to boundary), the geometry returned by MI is obtained by the precise computation of intersections between input meshes and the reconstruction grid; interpolation is limited to the mutually overlapping sections of the input range maps.

Our algorithm has proved to be an efficient solution, both in time and space complexity. Analogously to other voxel-based solutions, the size of the output mesh depends on the size of the 3D grid used in resampling and reconstruction. This is an advantage common to all voxel-based solutions because the user can produce representations at a different accuracy and size, according to the selected grid resolution. Moreover, the output mesh is in general more compact than the one produced by integration-based methods (e.g., the *Ball Pivoting* technique [2]), even when a high resolution grid is used, because the mutually overlapping sections are fused. The resulting size of the fused mesh is in general much lower than the total size of the input range maps.

Another problem common to many integration-based methods is the possible C^1 discontinuity that may be produced in the output mesh on the border of adjacency between different range maps. Abrupt staircase discontinuity (due to small registration errors or inaccuracies) may be easily produced by all methods based on the retriangulation of the set of surface samples. Blending corresponding overlapping sections of the input range maps mitigates this possible data inaccuracy.

Another important aspect is the capability of a given fusion method to manage attribute data that can be defined on the input range maps. An example is color data produced by many 3D scanners as an attribute of each geometrical sample. Managing color with a voxel-based fusion approach is not straightforward: for each voxel we generally have a distance value from the surface that may depend on many surface samples. One possibility is to resample the color attribute after the output mesh has been produced, but this means that for each output vertex we should maintain a trace of the corresponding original range map(s). Conversely, simplicity of color management is an important characteristic of MI: for each intersection, computing the corresponding color is straightforward, and color can be managed in the same manner in which we manage geometry (i.e., use weighted composition of both geometry and color for each pair of merged intersections).

On the other hand, the intensive use of the algorithm since its first implementation (2001) has highlighted the limits of the proposal as summarized below.

The MI algorithm is more accurate and faster than the volumetric approaches based on distance fields, but it is certainly less robust. Its lower robustness mainly depends on the presence of noise in the input data rather than on the choice of the maximum merge distance or the grid size. While the latter parameter can be easily fixed from the sampling density and the residual registration error, noisy data – generally border faces in the range maps with the wrong orientation – generate *non-MC-compliant* cell configurations and therefore many holes. Finally, robustness can also be reduced by a nonproper implementation of the range map discretization process. An inaccurate rasterization could generate wrong placement of some intersections in the virtual cells.

The ability of the algorithm in closing holes is limited to simple, non-self-intersecting boundaries. The triangular mesh closing the hole is intended to be planar.

Small parts of the reconstructed surfaces can be dependent on their *position* in the grid: if two close intersections with opposite signs belong to the same virtual cell, then they are considered high-frequency detail and removed from the structure. If the same intersections belong to different cells, then they are not removed.

Acknowledgements. We acknowledge the financial support of the EU IST-2001-32641 "ViHAP3D" and RBAU01MZJ5 MIUR Firb "Macro-Geo" Projects.

References

1. Algorri M, Schmitt F (1995) Deformable models for reconstructing unstructured 3D data. Lecture notes in computer science, vol 905. Springer, Berlin Heidelberg New York, pp 420–428
2. Bernardini F, Mittleman J, Rushmeier H, Silva C, Taubin G (1999) The Ball-pivoting algorithm for surface reconstruction. IEEE Trans Vis Comput Graph 5(4):349–359
3. Callieri M, Cignoni P, Scopigno R (2002) Reconstructing textured meshes from multiple range RGB maps. In: Proceedings of the 7th international fall workshop on vision, modeling, and visualization 2002, Erlangen, Germany, 20–22 November 2002
4. Callieri M, Cignoni P, Ganovelli F, Montani C, Pingi P, Scopigno R (2003) VCLab's tools for 3D range data processing. In: Arnold D, Chalmers A, Niccolucci F (eds) Proceedings of VAST 2003 and EG symposium on graphics and cultural heritage, Brighton, UK, 5–7 November 2003
5. Cignoni P, Montani C, Rocchini C, Scopigno R, Tarini M (1999) Preserving attribute values on simplified meshes by re-sampling detail textures. Vis Comput 15(10):519–539
6. Cignoni P, Rocchini C, Montani C, Scopigno R (2003) External memory management and simplification of huge meshes. IEEE Trans Vis Comput Graph 9(4):525–537

7. Chen Y, Medioni G (1995) Description of complex objects from multiple range images using an inflating balloon model. Comput Vision Image Understand 61(3):325–334
8. Curless B, Levoy M (1996) A volumetric method for building complex models from range images. In: Proceedings of SIGGRAPH'96, New Orleans, 4–9 August 1996. Comput Graph 30:303–312. Addison-Wesley, Reading, MA
9. Da Silva R, Shin-Ting W (1998) Reconstructing a 3D model from range images using radial flow model. In: Proceedings of the 11th SIBGRAPI conference, Rio de Janiero, 20–23 October 1998, pp 123–131
10. Davis J, Marschner SR, Garr M, Levoy M (2002) Filling holes in complex surfaces using volumetric diffusion. In: Proceedings of the 1st international symposium on 3D data processing, visualization, and transmission (3DPVT '02), Padua, Italy, 19–21 June 2002
11. Goesele M, Granier X, Heidrich W, Seidel H-P (2003) Accurate light source acquisition and rendering. ACM Trans Graph 22(3):621–630
12. Grosskopf S, Neugebauer J (1998) Fitting geometrical deformable models to registered range images. In: Kock R, van Gool L (eds) Proceedings of the SMILE '98 conference, Freiburg, Germany, 6–7 June 1998. Lecture notes in computer science 1506. Springer, Berlin Heidelberg New York, pp 266–274
13. Häusler G, Karbacher S (1997) Reconstruction of smoothed polyhedral surfaces from multiple range images. In: Girod B, Niemann H, Seidel H-P (eds) Proceedings of 3D Image Analysis and Synthesis '97, Infix, Sankt Augustin, Germany, pp 191–198
14. Hilton A, Illingworth J (1997) Multi-resolution geometric fusion. In: Proceedings of the international conference on recent advances in 3D digital imaging and modeling, Ottawa, Ontario, Canada, 12–15 May 1997. IEEE Press, New York, pp 181–188
15. Hilton A, Stoddart AJ, Illingworth J, Windeatt T (1998) Implicit surface-based geometric fusion. Comput Vision Image Understand 69(3):273–291
16. Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W (1992) Surface reconstruction from unorganized points. In: Proceedings of SIGGRAPH '92, Chicago, 26–31 July 1992. Comput Graph 26(2):71–78
17. Howie CT, Blake EH (1994) The mesh propagation algorithm for isosurface construction. Comput Graph Forum 13(3):65–74
18. Ju T, Losasso F, Schaefer S, Warren J (2002) Dual contouring of hermite data. In: Proceedings of the 29th conference on computer graphics and interactive techniques (SIGGRAPH '02), San Antonio, TX, 21–25 July 2002. ACM Trans Graph 21(3):339–346
19. Kobbelt LP, Botsch M, Schwanecke U, Seidel H-P (2001) Feature-sensitive surface extraction from volume data. In: Proceedings of SIGGRAPH 2001, Los Angeles, 12–17 August 2001. Comput Graph 35(3):57–66. ACM Press, New York
20. Lorensen WE, Cline H (1987) Marching Cubes: a high resolution 3D surface construction algorithm. In: Proceedings of SIGGRAPH '87, Anaheim, CA, July 1987. Comput Graph 21(4):163–170
21. Mencl R, Müller H (1998) Interpolation and approximation of surfaces from three-dimensional scattered data points. In: Proceedings of Eurographics'98 STAR – State of the Art Reports, pp 51–68
22. Montani C, Scateni R, Scopigno R (1994) A modified look-up table for implicit disambiguation of Marching Cubes. Vis Comput 10(6):353–355
23. Narkhede A, Manocha D (1995) Fast polygon triangulation based on Seidel's algorithm. Graphics Gems V. Academic Press Professional, Boston, pp 394–397
24. Pito R (1996) Mesh integration based on co-measurements. In: Proceedings of the international conference on image processing, Lausanne, Switzerland, 16–19 September 1996, pp 397–400
25. Pulli K, Duchamp T, Hoppe H, McDonald J, Shapiro L, Stuetzle W (1997) Robust meshes from multiple range maps. In: Proceedings of the international conference on recent advances in 3D digital imaging and modeling, Ottawa, Ontario, Canada, 12–15 May 1997. IEEE Press, New York, pp 205–211
26. Pulli K (1999) Multiview registration for large datasets. In: Proceedings of the 2nd international conference on 3D digital imaging and modeling, Ottawa, Ontario, Canada, 4–8 October 1999. IEEE Press, New York, pp 160–168
27. Ratishauser M, Stricker M, Trobina M (1994) Merging range images of arbitrarily shaped objects. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Seattle, 20–24 June 1994, pp 573–580
28. Rocchini C, Cignoni P, Ganovelli F, Montani C, Pingi P, Scopigno R (2001) Marching Intersections: an efficient resampling algorithm for surface management. In: Proceedings of the international conference on shape modeling and applications (SMI 2001), Genoa, Italy, 7–11 May 2001
29. Rocchini C, Cignoni P, Montani C, Scopigno R (2002) Acquiring, stitching and blending appearance attributes on 3D models. Vis Comput 18(3):186–204
30. Roth G, Wibowoo E (1997) An efficient volumetric method for building closed triangular meshes from 3-D image and point data. In: Proceedings of Graphics Interface '97, Kelowna, BC, Canada, 21–23 May 1997, pp 173–180
31. Soucy M, Laurendeau D (1995a) A general surface approach to the integration of a set of range views. IEEE Trans Patt Anal Mach Intell 17(4):344–358
32. Soucy M, Laurendeau D (1995b) A dynamic integration algorithm to model surfaces from multiple range views. Mach Vision Appl 8(1):53–62
33. Taubin G (1995) A signal processing approach to fair surface design. In: Proceedings of SIGGRAPH '95, Los Angeles, 6–11 August 1995, pp 351–358
34. Turk G, Levoy M (1994) Zippered polygon meshes from range images. In: Proceedings of SIGGRAPH '94, Orlando, FL, 24–29 July 1994. pp 311–318. ACM Press, New York
35. Varadhan G, Krishnan S, Kim YJ, Manocha D (2003) Feature-sensitive subdivision and isosurface reconstruction. In: Proceedings of IEEE Visualization 2003, Seattle, 19–24 October 2003, pp 99–106
36. Wheeler MD, Sato Y, Ikeuchi K (1998) Consensus surfaces for modeling 3D objects from multiple range images. In: Proceedings of the IEEE international conference on computer vision, Bombay, India, January 1998

Photographs of the authors and their biographies are given on the next page.

CLAUDIO ROCCHINI collaborates with the Visual Computing Lab of the Istituto di Scienza e Tecnologie dell'Informazione (ISTI) of the National Research Council (CNR) in Pisa, Italy; he is a government official at the Istituto Geografico Militare (IGMI) in Florence. His research interests include surface modeling, simplification, multiresolution and automatic acquisition of 3D models. Rocchini received in 1997 an advanced degree (Laurea) in Computer Science from the University of Pisa.

PAOLO CIGNONI is research scientist at the Istituto di Scienza e Tecnologie dell'Informazione (ISTI) of the National Research Council (CNR) in Pisa, Italy. His research interests include computational geometry and its interaction with computer graphics, scientific visualization, volume rendering, simplification and multiresolution. Cignoni received in 1992 an advanced degree (Laurea) and in 1998 a PhD in Computer Science from the University of Pisa.

FABIO GANOVELLI is research scientist at the Istituto di Scienza e Tecnologie dell'Informazione (ISTI) of the National Research Council (CNR) in Pisa, Italy. His research interests include deformable object modelling, collision detection, multiresolution and isosurfaces extraction. He received in 1995 an advanced degree in Computer Science (Laurea) and in 2001 a PhD in Computer Science from the University of Pisa.

CLAUDIO MONTANI is a research director with the Istituto di Scienza e Tecnologie dell'Informazione (ISTI) of the National Research Council (CNR) in Pisa, Italy. His research interests include data structures and algoritms for volume visualization and rendering of regular or scattered datasets. Montani received an advanced degree (Laurea) in Computer Science from the University of Pisa in 1977. He is member of IEEE and AICA.

PAOLO PINGI is a PhD student at the Istituto di Scienza e Tecnologie dell'Informazione (ISTI) of the National Research Council (CNR) in Pisa. His research interests include 3D scanning, sensor tracking and applications of Computer Graphics. Pingi received an advanced degree (Laurea) in Computer Science in 1999 from the University of Pisa.

ROBERTO SCOPIGNO is senior research scientist at the Istituto di Scienza e Tecnologie dell'Informazione (ISTI) of the National Research Council (CNR) in Pisa, Italy. He is currently engaged in research projects concerned with scientific visualization, volume rendering, web-based graphics, multiresolution data modeling and rendering, 3D scanning and applications of 3D computer graphics to Cultural Heritage. Scopigno received an advanced degree (Laurea) in Computer Science from the University of Pisa in 1984. He is member of IEEE, Eurographics and ACM Siggraph. He is elected member of the Executive Committee of the Eurographics association and, since 2003, Vice Chair of the association.