Dipartimento di Informatica e Scienze dell'Informazione

Online Data Fusion for building 3D models from acoustical range images

Authors:
Laura Papaleo, Enrico Puppo

Technical Report DISI-TR-05-04

DISI, Università di Genova
Via Dodecaneso, 35, 16146 Genova, Italy http://www.disi.unige.it

Abstract

This Technical Report presents a *Data Analysis approach* which on the fly, starting from multiple acoustic/optical range images, elaborates the acquired unknown object by mosaicing multiple single frame meshes.

In the context of the European ARROV project, we developed a 3D reconstruction pipeline, which provides a 3D model of an underwater scene from a sequence of range data captured by an acoustic camera mounted on board a Remotely Operated Vehicle (ROV). Our approach works on line by building the 3D model while the range images arrive from ROV. The method combines the range images in a sequence by minimizing the workload of the rendering system.

Table of Contents

1. INTRODUCTION 11

2. THE ARROV PROJECT 11

3. MOSAICING FROM MULTIPLE RANGE IMAGES 13

4. THE ARROV RECONSTRUCTION PIPELINE 14

4.1 DATA CAPTURE 16
4.2 DATA PRE-PROCESSING 18
4.3 REGISTRATION 22
4.4 GEOMETRIC FUSION 22

5. RESULTS 28

6. BIBLIOGRAPHY 35

1. Introduction

This report describes the work done in solving the problem of *online mosaicing* starting from multiple range images in the context of the European Project ARROV [1]. First, a brief introduction to the problem and the framework of the project are presented, next we present the details of the developed *Data Analysis Pipeline* and finally results are reported and open issues are outlined.

2. The ARROV Project

The **European ARROV project** [1] has been founded from the European Community within the Fifth Framework Program, specific research, and technological development program "*Competitive and Sustainable Growth*". It started in 2001 with duration of three years and aims at improving the applicability and performance of multifunctional *Remotely Operated Vehicles* (ROV) for the inspection, monitoring and survey of underwater environments and manmade installations. In particular, the project points at solving/improving the following activities:

- *Surveying and monitoring of underwater structures*, such as oil rigs, pipes, ship wrecks, and other submersed structure in general;
- *Inspection of underwater installations*, such as dams, dikes, docks, tanks, canals and tunnels (e.g., in hydro- and thermo-electric power plants), in order to detect causes of danger or damage;
- *Surveillance for detection* of oil and gas leakage from pipes, submersed tanks and wrecks;
- *Control of unmanned industrial operations* (e.g., pipe spool metrology).

At the state of the art, these problems are either solved through intervention of divers, which are consequently exposed to high risks; or addressed with insufficient reliability, because of the poor accuracy and quality of feedback provided by sensors on board an ROV.

The goal of the project has been achieved by the development of a **novel integrated sensorial system**, which exploits both optical and acoustical data to provide the ROV pilot with an **augmented representation of the scene** (*augmented reality*). The system will be also able to synthesize automatically a 3D model of the scene from sensorial data (*model acquisition*). Moreover, a unified man machine interface will be provided to control an ROV as well as the sensors mounted on board.

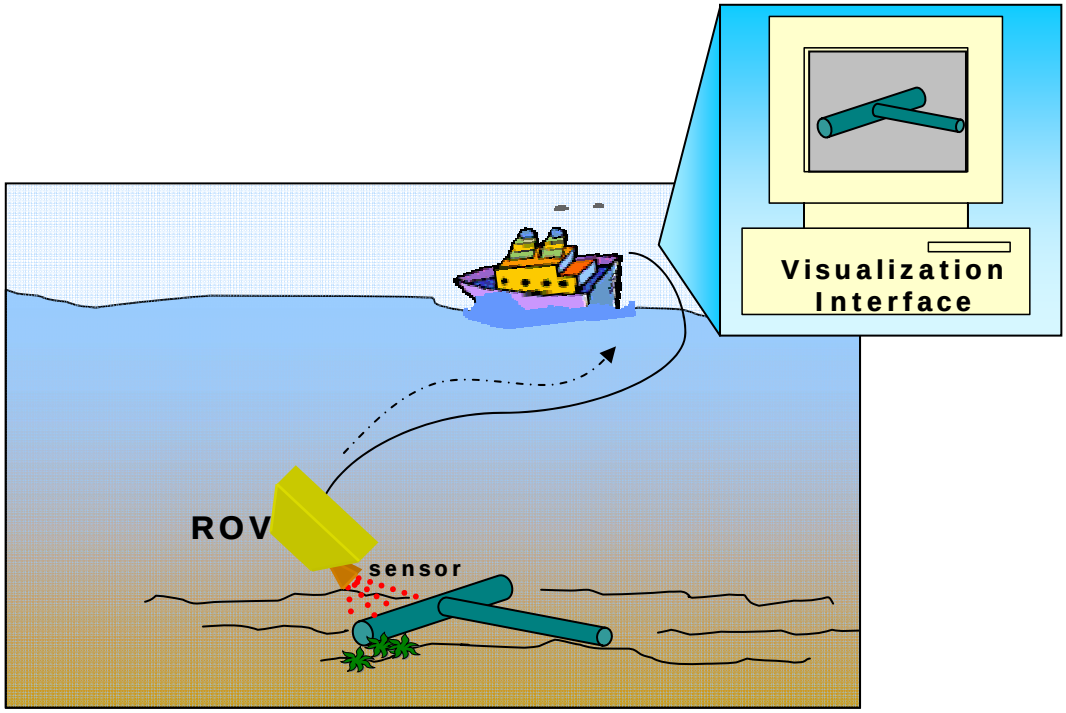

Figure 1 – The ARROV measurements system

Therefore, the main innovation of ARROV lies in the use of a compact set of imaging sensors (optical and acoustic cameras) aimed at the accurate 3D reconstruction of the surrounding environment, and in the effective and efficient visualization of a scene to the vehicle operator through the use of augmented reality (mixed real and synthetic images).

The ARROV project is a frame in which partners of different technical and scientific cultures cooperate together. The consortium includes six partners distributed across three European countries:

- *Our Research Group (The Computer Graphics and Geometric Modelling Group at DISI),* active in the design of multiresolution models and data structures and in the reconstruction of 3D surfaces and object models from sampled data.
- *Department of Computer Science- University of Verona,* with its research focus to Computer Vision problems.
- *OmniTech AS* - the first producer of a commercial 3D underwater acoustic camera - the EchoScope 1600.
- *General Robotics Limited*, a small, high tech company with unique knowledge and experience in software control and simulation systems for the offshore oil and gas industry.
- *Centro Elettrotecnico Sperimentale Italiano (CESI)*, which is a market leader in testing, certification of electromechanical equipment and electrical power system studies.
- *RSN* specializes in survey, positioning, operation of the Sky-Fix Global DGPS Network and ROV services as contractor in the offshore Oil and Gas Industry.

Figure 2 - two different types of commercial ROVs Errore. L'origine riferimento non è stata trovata.

In conclusion, the augmented reality visualization system, developed in the context of the project ARROV, significantly contributes to reduce the cognitive load on teleoperators by providing cues that help prevent them getting lost and disoriented. Moreover and accurate 3D interpretation of data and automatic fault detection significantly contribute to improve surveillance and control of unmanned operations.

The role of our research group in the project was mainly the development of methods for Data Analysis starting from single or multiple range images able to satisfy real time requirements. Our methods have been integrated with the work done by the other academic partner (*Department of Computer Science-University of Verona)* in acquisition and registration in order to produce the complete pipeline. In this report, we will present the entire pipeline going into more details for those parts which see our group as leader (Single Frame Reconstruction and Mosaicing).

3. Mosaicing from Multiple Range Images

Several steps are involved in reconstructing a 3D model and further scene analysis using multiple range images. In a general framework, a full 3D reconstruction of the observed environment requires scanning from *different views* **Errore. L'origine riferimento non è stata trovata.**. The surface measured from each scan is recorded in a *local* coordinate system centered at the scanner. In order to integrate different views, all the surfaces must be registered into a global coordinate system. In general, the challenges in automatic registration are how to efficiently find correspondences from two partially overlapped surfaces and robustly deal with the noise and different surface sampling resolutions [4],[6],[2].

Integrating surfaces from different views is a *fusion* process that weaves a whole surface from a set of overlapping surface patches. The integration process should be robust to surface noise and registration error. Usually, it is impossible to scan the whole object, so holes are left at regions where the laser cannot reach. These holes can be filled by space carving or by an automatic hole filling process after integration. The reconstructed surface may contain a number of parts. In many

cases, a segmentation or decomposition of the surface gives a better interpretation of the surface **Errore. L'origine riferimento non è stata trovata.**.

The framework used in our research (in the context of the project ARROV) is depicted in Figure 3. The integration of multiple range images (*mesh mosaicing*), has to be solved in an *online* and *offline manner*: it is basically the combination of multiple range images in a consistent way in order to obtain a unique range image from which we can derive the mesh representing the part of the object acquired.

In case of *online mosaicing*, at the instant time t+1 we need to implement a fast and efficient algorithm for computing the resulting mesh (R) starting from a mesh A reconstructed until step t and a new mesh B just acquired at instant time t+1. This problem is not so simple for various reasons: above all, the process must work in real time or, at least, fast enough to support interaction. This requirement is in contrast with the complexity of operating. Most phases of the mosaicing process are computationally expensive: registration, fusion, mesh generation and visualization. Consider that, while mesh B has a relatively small and fixed size, mesh A is the mosaic of an arbitrary numerous sequence of previous frames. If the whole mesh A were used in computation and transferred to the visualization engine at each frame, time constraints could not be satisfied.

Therefore, it will be necessary to adopt strategies that can make computation as local as possible.

4. The ARROV Reconstruction Pipeline

As we said before, the final goal of the project ARROV is to provide a 3D scene model of an underwater environment (Figure 1) by the use of a remotely operated vehicle (ROV) with an Echoscope (Figure 4) (acoustic camera) mounted on it. Unfortunately, data provided by an acoustic camera are noisy: speckle noise is typically present due to the coherent nature of the acoustic signals. Resolution is low and depends on the frequency of the acoustic signal (it is about 3cm at 500 KHz): the higher the frequency, the higher the resolution, the narrower the field of view. Consequently, we are forced to operate with a limited field of view and a technique to reconstruct progressively the scene while the sensor is moving is necessary.

In our case: (i) the resolution is never better than some centimeters, unlike classic range data (e.g., from laser range finders); (ii) sensor position is not taken into account for view registration; (iii) the motion of the sensor is quite unstable, and cannot be controlled with precision in any real case, so acquired images from a fixed position may be different due to speckle and sensor floating. As a consequence, some previous solutions based on estimation of surface parameters cannot be taken into account due to the high uncertainty of data.

In order to achieve our goal, we developed, in collaboration with Department of Computer Science- University of Verona a *complete data processing pipeline* (Figure 3), which starts from data acquisition, and produces a geometric model of the observed scene, in the form of a mesh of triangles. This process can come in two versions:

- *on-line*, i.e., processing occurs while new frames come from the sensor as the vehicle carrying it moves; and
- *off-line*, i.e., processing is made after all frames have been collected.

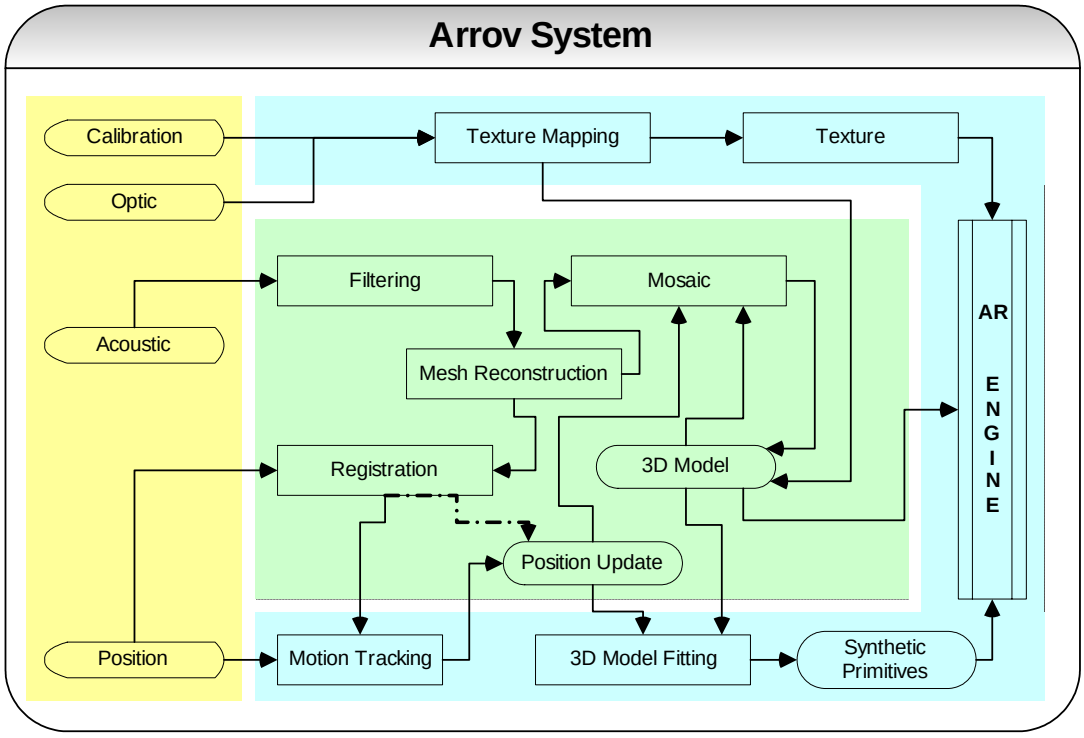

Figure 3 - Arrov data processing scheme

The major differences between the two versions is that the on-line one needs to build the output progressively from a dataset that grows through time, and must do it within strict time constraints, in order to support navigation. The off-line version, on the contrary, can process all data together, taking a long time. We developed fast methods to support the on-line version, as well as refinements to improve the accuracy of the result in the off-line version.

This dissertation will concentrate on the on-line processing pipeline, which includes four stages: *Data Capture*, *Data Pre-processing*, *Registration* and *Geometric Fusion*.

1. **Data capture**, in which the acoustic camera produces a new range image;
2. **Data Pre-Processing**, in which data are pre-processed to eliminate outliers and a range image is processed to obtain a triangle mesh discarding small components;

3. **Registration**, in which all data corresponding to all frames are brought to the same coordinate system. In the on-line version this means just bringing the new frame to the coordinate system of all the previous frames;
4. **Geometric fusion**, in which geometries corresponding to the different frames are merged to form a single mesh representing the whole observed scene. In the on-line version this means to merge the geometry of the new frame into the mesh built on all previous frames.

Our research group actively participates at all the four stages but we were the leader in the context of the Recostruction steps (i.e. Pre-processing and Geometric Fusion). The remaining part of this section describes our reconstruction pipeline, going in details in those steps which have been developed mainly by our research group [3].

4.1 Data Capture

Three-dimensional acoustic data are obtained with a high resolution acoustic camera, the Echoscope 1600 **Errore. L'origine riferimento non è stata trovata.**. The scene is unsonified by a high-frequency acoustic pulse, and a two-dimensional array of transducers gathers the backscattered signals. The whole set of raw signals is then processed in order to form computed signals whose profiles depend on echoes coming from fixed steering directions (called beam signals), while those coming from other directions are attenuated.

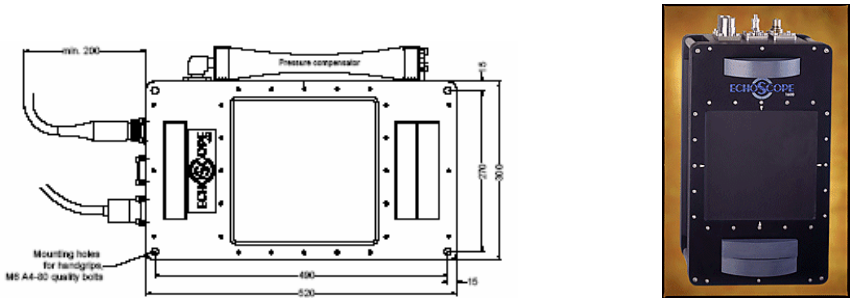

Figure 4 – The Echoscope used for the measurements.

Successively, the distance of a 3D point can be measured by detecting the time instant at which the maximum peak occurs in the beam signal. In particular, acoustic image is formed by the use of the beamforming technique. It is a spatial filter that linearly combines temporal signals spatially sampled by a discrete antenna. In this way, if a scene is insonified by a coherent pulse, the signals, representing the echoes backscattered from possible objects in specific direction, contain attenuated and degraded replicas of the transmitted pulse.

Let us denote by v_k the position of the k-th sensor (transducer), by c the sound velocity, and by $x_k(t)$ the signal received by the k-th sensor. Beamforming can form a *beam* signal, $bs_u(t)$, steered in the direction of the vector u, defined as:

$$bs_u(t) = \sum_{k=0}^{M-1} w_k \cdot x_k(t - \theta_k)$$

where w_k are the weights assigned to each sensor, M is the number of transducers, and $\theta = (v_k \cdot u)/c$ are the delays applied to each signal.

A common method to detect the scattering objects distances is to look for the maximum peak of the beam signal envelope **Errore. L'origine riferimento non è stata trovata.**. Denoting by t^* the time instant at which the maximum peak occurs, the related distance, r^* (i.e., range value) is easily derivable (i.e., $r^* = c \cdot t^*/2$ if the pulse source is placed in the coordinate origin). According to the spherical scanning technology, range values are measured from each steering direction $u(i\ j)$ where i and j are indices related to the elevation (*tilt*) and azimuth (*pan*) angles respectively.

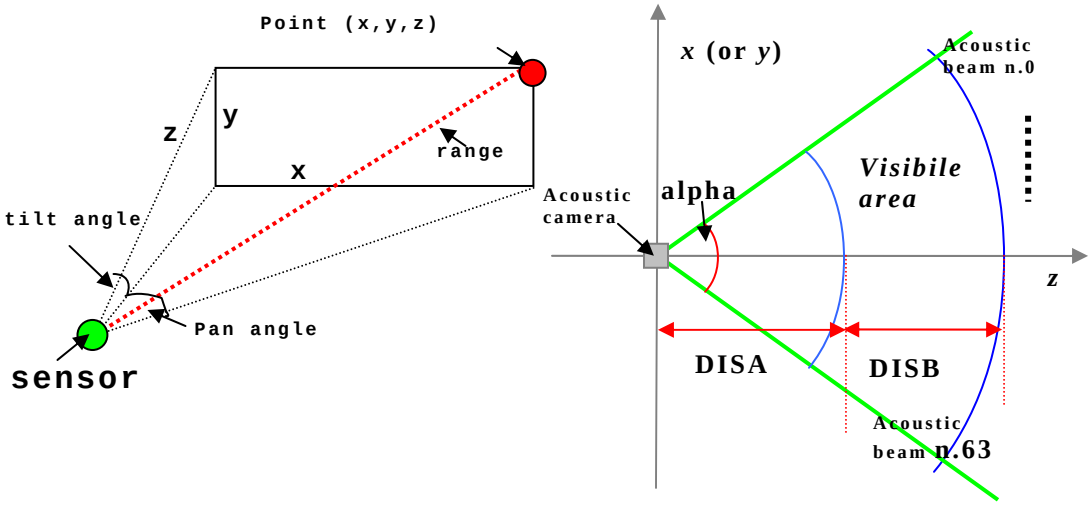

Figure 5- Spherical scanning principle (left) and subdivision of the beams onto the acquiring volume (right). Each beam is associated to a (*i; j*) coordinate of the range image

Figure 5.(left) shows a schema of the scanning principle. Figure 5.(right) shows a projection of the acquiring volume to the *ZX* plane, on which the sectors associated to each beam are marked. The minimum and maximum distances are also depicted.

Going into the details, the Echoscope carries out *64* measures for both tilt and pan by defining a *64×64* range image r_{ij}. The conversion from spherical coordinates to usual Cartesian coordinates is recovered by the use of the following equations **Errore. L'origine riferimento non è stata trovata.**:

$$x = r_{ij} \tan(js_\alpha) \Big/ \sqrt{1 + \tan^2(js_\alpha) + \tan^2(js_\beta)}$$

$$y = r_{ij} \tan(js_\beta) \Big/ \sqrt{1 + \tan^2(js_\alpha) + \tan^2(js_\beta)}$$

$$z = r_{ij} \sqrt{\tan^2(js_\alpha) + \tan^2(js_\beta)}$$

where s_α and s_β are increments of the elevation and the azimuth respectively. Figure 6 shows a range image acquired by the Echoscope and the related points cloud, which will be passed to the Pre-processing step of the pipeline.

There is a tradeoff between range resolution and field of view. Resolution depends on the frequency of the acoustic signal (it is about *5*cm at *500*KHz): roughly speaking, the higher is the frequency, the higher is the resolution, the narrower is the field of view.

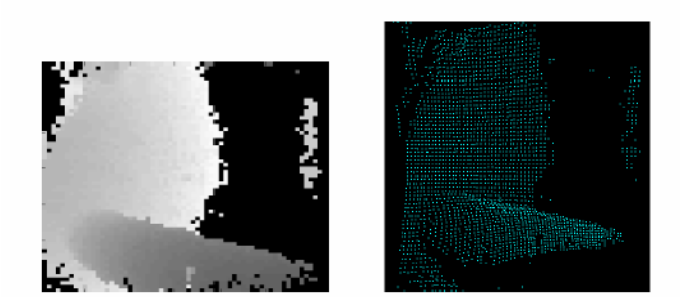

Figure 6 – A range image acquired by the Echoscope 1600 (left) and the relative cloud of points (right). The scene consists of a pipe in underwater

Unfortunately, the acoustic image is affected by false reflections, caused by secondary lobes and by acquisition noise, which is modeled as speckle. Moreover, the intensity of the maximum peak can be used to generate another image, representing the *reliability* of the associate 3D measures, so that, in general, higher is the intensity, safer is the associate distance. A dramatic improvement of the range image quality can be obtained by discarding points whose related intensity is lower than a threshold, depending on the secondary lobes [11], [10].

4.2 Data Pre-processing

The next step of the Recostruction Pipeline is the reconstruction of a triangle mesh from a single range image just acquired by the 3D Echoscope. Our input basically consists of (Table 1):

1. a matrix $I[i][j]$ describing a regular lattice (range image), whose entries (i,j) describe either a point in space (x,y,z), or a vacancy,
2. a connectivity threshold (θ), used to evaluate whether or not two nodes in the lattice should be considered adjacent, and
3. the lateral resolution l of the sensor.

The output of the method is (Table 1):

1. an adjacency matrix A, containing the connection data for each entry in the input frame, and
2. a single frame mesh $M=(V,F)$, where V is a set of 3D vertices and F is a set of faces. The mesh structure contains also additional information such as *face* and *vertex normals*.

Input		**Output**	
INPUT MATRIX	Vertex I[]	ADJACENCY MATRIX	Vertex A[]
CONNECTIVITY THRESHOLD	float t	SINGLE FRAME MESH	Mesh *M
LATERAL RESOLUTION	double l		

Table 1 Input and output of the Single Frame Reconstruction Procedure

The method considers the points corresponding to non-vacant entries in the lattice, and connect them pairwise to form edges: cycles of three edges form triangles of the output mesh. A *potential* edge exists between each pair of points

$v \equiv (x_1, y_1, z_1)$ and $w \equiv (x_2, y_2, z_2)$ if their radial distance is below some threshold (θ), depending from the following formula

$$\left| \sqrt{x_1^2 + y_1^2 + z_1^2} - \sqrt{x_2^2 + y_2^2 + z_2^2} \right| < \theta .$$

We implemented an algorithm, which correctly reconstructs a triangle mesh from a range image as input. It can be divided into four main steps:

1. Find connections for non-vacant entries.
2. Recovery vacant entries
3. Improve connections: find feasible semi-diagonal
4. Generate the mesh

In the following, we will go into details of each step, showing results and pseudo-code of each sub-procedure.

Step 1 : Find connections for non-vacant entries

For each *2×2* block $\begin{bmatrix} v_1 & v_2 \\ v_3 & v_4 \end{bmatrix}$ in the lattice where v_1 is a non-vacant entry (Figure 7.left) the procedure tries to find feasible edges. It first tests independently for horizontal and vertical connectivities (basically v_1, v_2 and v_1, v_3), successively it tests for the descending diagonal (v_1, v_4): if it is found the procedure stops otherwise it checks the other diagonal searching for the connection between v_3 and v_2.

Each time an adjacency is found, the procedure updates the adjacency vector. Starting from the range image in Figure 6.right, the result of the application of the first step is reported in Figure 7.right.

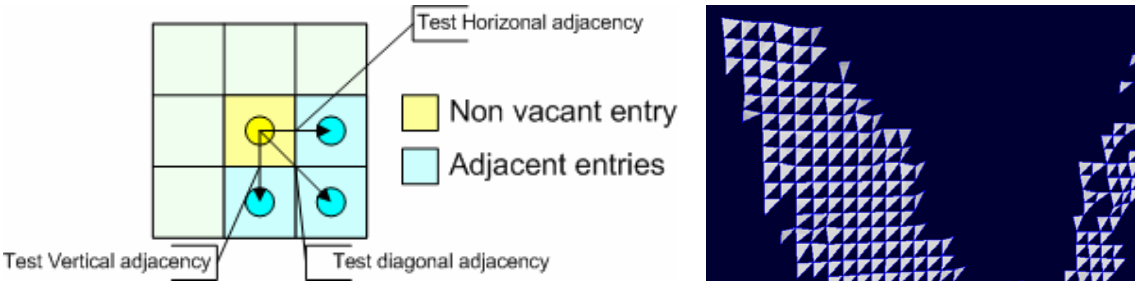

Figure 7 - 2x2 block testing (left) and the result applied on a input range image (right)

Step 2 : Recovery vacant entries

Once all the connections of step 1 from non vacant entries have been discovered, the method considers all the entries in the lattice with no 3D point (*vacant entries*), as in case of a hole in the grid defined by the ARROV sensor.

For each vacant entry v (Figure 8.left) the procedure considers a *3×3* window W_v surrounding v and tests for connectivity between pairs of (non vacant) entries belonging to this W_v:

1. It tests vertical and horizontal adjacencies, checking existence and validity (a connection can exist only if it do not intersect any other already existing connection),
2. It tests the two diagonal adjacencies and validate each potential adjacency only if there are no existing edges crossing it.

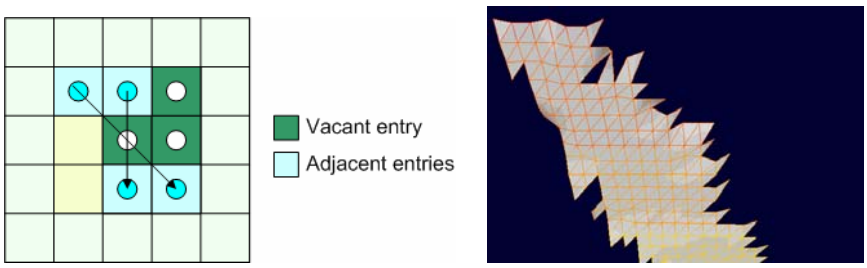

Figure 8 - 3x3 bock testing around a vacant entry (left) and the result of the application of this procedure step applied to an input range image (right)

The results demonstrated that this step is able to fill some holes remained after the application of step_1 and to improve the result of the reconstruction (Figure 8). Unfortunately, too many holes remain in the final mesh and another step is necessary.

Step 3: Improve connections, find feasible semi-diagonals

In order to improve the resulting mesh the method tries to find feasible semi-diagonal connections. For each *3×3* window in the input lattice we check for the possible *eight* semi-diagonal edges among the entries at the boundary of the window (Figure 9.right). At most two of the feasible semi-diagonal edges can coexist and the method searches for them

Step 4: Mesh generation

The last step of our procedure generates the final triangular mesh by forming faces as cycles of three edges (Figure 10).

For each location *v* in the lattice, the method scans the portion of adjacency list (that has been populated during the three steps above spanned) by the first 12 bits of the bit vector encoding it (Figure 9) (the remaining edges in the list will be considered from other vertices).

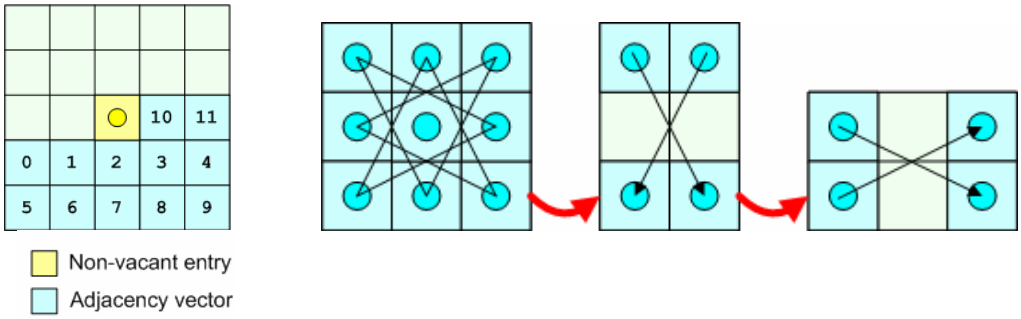

Figure 9 - the adjacency vector of an entry in the lattice (left) - a 3x3 block for testing for feasible semi-diagonals (right)

For each pair of consecutive edges in the list, a triangle of the mesh is generated. A *filter* can be also applied based on the *perimeters* of faces, to obtain a smoother distribution of face dimensions, and to overcome limitations due to the radial nature of the distance (used to test for point connectivity).

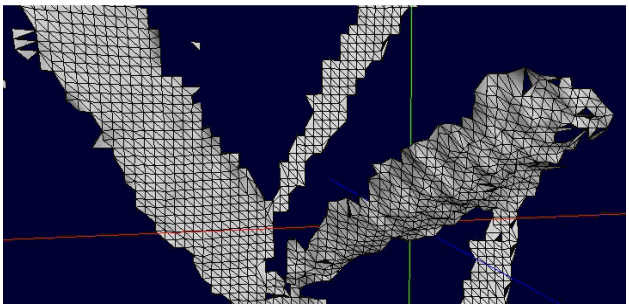

Figure 10 - the result of the application of the SingleFrameReconstruction() procedure.

The procedure cuts also isolated vertices and computes the normal vectors for faces and vertices adding them to the mesh structure. The normal computation is split into two phases.

1. Compute normals for faces by taking the external product of two independent unit vectors lying on the face itself.
2. For each vertex, compute the relative normal by taking the vector sum of the faces' normals sharing the active vertex (the *neighborhood faces* of the vertex).

The connectivity information coming from the mesh is used to discard connected components smaller than a given size. This step, called size *filter*, greatly improves the quality of the acoustic images.

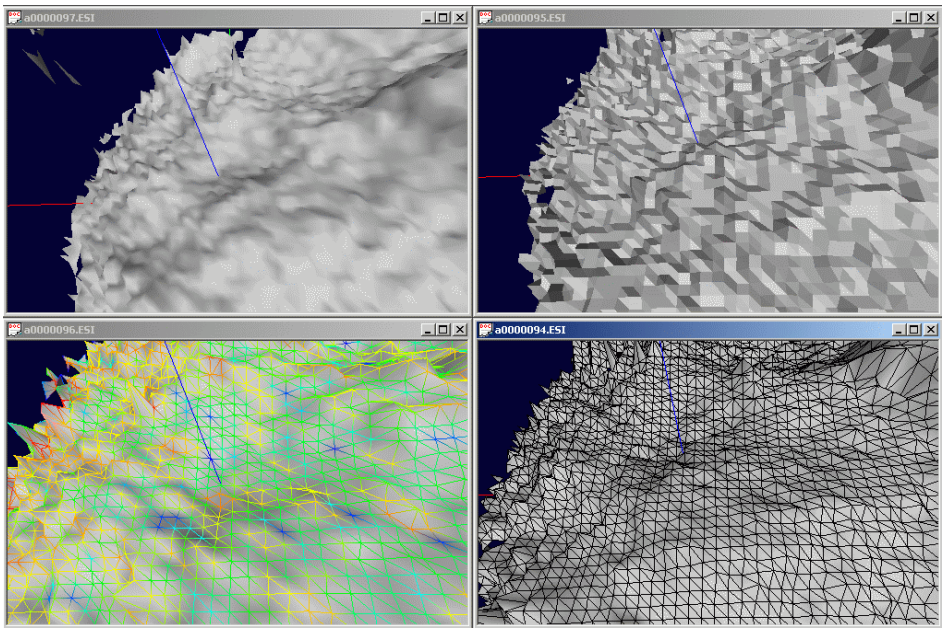

Figure 11 - Tiling of four meshes with different visualization options (from top-left to bottom-right: smooth, simple faces, wireframe with curvature, wireframe and simple faces).

4.3 Registration

After a frame has been reconstructed, we have to deal with the registration on two range images in a unique coordinate system. In the *on-line* version, this means registering the new frame acquired to the coordinate system of all the frames acquired at previous steps.

Registration refers to the geometric alignment of a pair or more of 3D point sets, in the context of the ARROV Project, this was a task of our academic partner, which decided to address the problem of registering a pair of range images using variants and improvements of the classical Iterative Closest Point (ICP) algorithm. ICP can give very accurate results when one set is a subset of the other, but results deteriorate with pairs of partially overlapping range images. In this case, the overlapping surface portions must start very close to each other to ensure convergence, making the initial position a critical parameter [3].

Briefly, in order to be able to work on-line, ICP needs to be modified. In general, the speed enhancement of ICP algorithm can be achieved by: reducing the number of iterations necessary to converge and reducing the time spent at each iteration [2]. In the offline version, during the first phase a table is generated describing which frames overlap and successively, a global ICP is used, distributing registration errors evenly across all views [2].

4.4 Geometric Fusion

In general, the meshes built on the single frames, when represented in the same coordinate system, after registration, will freely overlap and intersect. Jumps, cracks and redundant surfaces will occur at overlaps (Figure 12). Thus, the simple collection of such meshes is not sufficient to give a final mosaic of the sensed objects. Meshes must undergo a process of *geometric fusion*, which produces a single mesh starting at the various registered meshes. If the fusion is to be done in off-line framework, algorithms proposed in [7], [8] serve well the purpose.

Here, we concentrate on the *on-line version* of the problem. In this case, at a given instant of time, a mesh *A* is given, which represents the mosaic obtained from all previous frames, and a new mesh *B* comes, which is built from the current frame as explained in Section 4.2.

The aim is to merge such meshes in a consistent way, in order to obtain a new mesh *R*, which represents the scene spanned by both A and *B*. This must be a *fast* process, since the system must support real time visualization of the mosaic at each frame. In particular, the output of the algorithm must be used to update the graphical system with the new primitives to be rendered. Unfortunately, this is not simply an incremental process in which new graphical primitives are added to the existing ones at each new frame. Indeed, the *fusion* operation may change the portion of *A* that overlaps with *B* (and it must overlap, otherwise registration would fail).

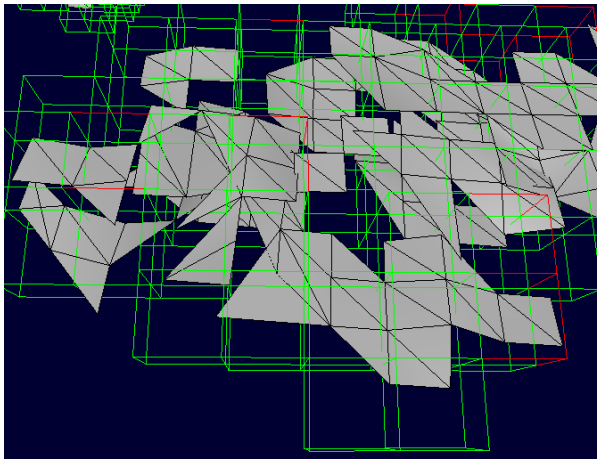

Figure 12 - some faces may overlap after multiple single frame meshes have been added at the model.

This means that some primitives rendered in previous frames will have to be substituted with new ones. Eventually, this means that display lists used by the graphics system will have to be regenerated. This operation can be expensive if applied to the entire mesh, and it could slow down the system.

Because of observations above, we decided to modify the method of the *Marching Intersection Algorithm* (MIA) [7], which exhibits an interesting approach, since it allows a valid trade-off between speed and accuracy and, by the use of quality parameters, it takes into account the reliability of input data.

The original MIA is based on a volumetric approach that locates the geometric intersections between the meshes built on single frames and a virtual 3D reference grid. Intersections are first merged and the output mesh is found by joining intersection points that belong to edges of the same cell, through a scheme defined in a lookup table.

In order to make it applicable to an *on-line setting*, we have modified the algorithm to deal with a pre-computed mesh *A* (as explained before) which *fits* the reference grid (i.e., it has its vertices on grid edges, and each face is completely contained in a grid cell), and a new mesh *B* which intersects the grid properly. Furthermore, in order to support *real time rendering*, we have developed a **lazy update strategy** for display lists, which reduces the load per frame on the graphics system.

So the mosaicing algorithm can be divided into three main phases:

1. Rasterization and Intersection management
2. Fusion, with cleaning and removal operations
3. Mesh generation and Lazy Update

It takes as input:

- a single frame mesh *SFM* with time stamp *t*
- a registration matrix *RM* in homogeneous coordinates, which describes current position and orientation of the sensor in a global coordinate system
- a grid resolution value *GR* (positive real) representing the edge length of square cells in which 3D space is subdivided

- a fusion threshold value FT (positive real) representing the minimum distance between two vertices generated by different frames (The Fusion threshold is usually kept lower than grid resolution)
- an updating threshold value UT (positive real) representing the maximum number of changes in a mesh before a new list must be generated for visualization

The Fusion Procedure produces in output:

- a time stamp t (the same of the input)
- a list of meshes, in which each mesh represents a new portion of mosaic or a modified one, and has a name and the information on the number of triangles in that mesh
- a list of triangles where each triangle is a triple of points (each point represented in a global coordinate system with Cartesian coordinates (x, y,z)).

Phase 1: Rasterization and Intersection Management

This part of the algorithm analyzes every face f_q of one of the registered meshes, searching for intersections with the virtual underlying grid G (the other is the actual mosaic and therefore is already aligned at the grid).

For every face f_q, the minimum enclosing box ($EBox_{fq}$) is computed and the f_q-projection on each plane (called rasterization planes) in the x,y and z directions (XY , XZ, Y Z) is analyzed (Figure 13).

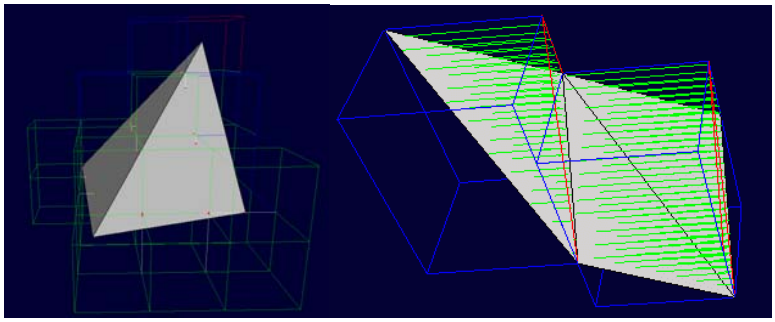

Figure 13 – Rasterization procedure: intersections are marked in red (left) the rasterization lines (X-axis) in green (right)

For doing this, every plane is subdivided in rasterization lines (l_1,l_2,...,l_m), each line being orthogonal to the plane and passing through a node of the grid. These lines may intersect a face at some points. Then, for each line l_h, all the intersections (i_1,i_2,...,i_n) are computed and the collection of all lines (l_1,l_2,...,l_m) gives the set of intersections related to a given rasterization plane p_s (Figure 13, Figure 14). Each intersection i_k is described through a class with the following instance members:

- *ic* as intersection value,
- *name* for the name of the original mesh (or timestamp),

- *sgn* an integer representing the sign of the face normal projection on raster line
- *dir* a number representing the value of the face normal projection on raster line
- *w* a number representing the *weight* of the intersection (computed as a bi-linear interpolation of the weight of surrounding vertices)

Intersections generated by each face f_q are ordered with respect to their value. In the Fusion phase of the algorithm, if two intersections i_1, i_2 are closer than the fusion threshold *FT*, a merging procedure is invoked. When an intersection i_k is computed, it is inserted into a bi-linked list of the corresponding rasterization line l_h: as the number of rasterization lines is very big, we chose to represent a rasterization plane as an associative collection (hash table) of lists, keeping therefore only those structures containing real intersections.

Every time we instantiate a new list, it is also added to a structure that contains all the updated or generated list of the current frame: this structure is a queue (FIFO type) that is scanned during the fusion phase.

Note that, we assume that underlying grid to have no bounds, that is, to be virtually infinite. Therefore we have to generate for each pair of coordinates a unique key as a function $k=f(x,y)$ that describes the rasterization line: unfortunately it is quite difficult to find such a function. For sake of simplicity we chose $k(x,y) = A \cdot x+y$, where A is a constant (usually big). Of course, it is possible for collisions to occur, but for a medium sized mosaic this is quite unlikely.

Actually the hash function is defined in three dimensions as $k(x,y,z)=A \cdot x+B \cdot y+C \cdot z$ and it is used both on the rasterization planes (setting z to zero) and on the whole 3-D mosaic for cell's indexing purposes

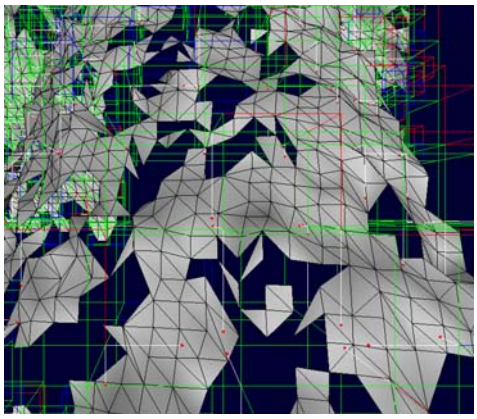

Figure 14 - Cells, edges and intersection (in red)

As we said before, each intersection (object) has a weight *w*, which measures the *reliability* of the data and it is used in the fusion phase. The weight is computed by linearly interpolating the intensity values of the vertices that describe the current face. If d_0, d_1, d_2 are the distances of the intersection point from the vertices, and I_0, I_1, I_2 are their intensities, then the weight w is defined as follow:

$$w = \frac{\left(I_0 \cdot d_1 \cdot d_2 + I_1 \cdot d_2 \cdot d_0 + I_2 \cdot d_1 \cdot d_0\right)}{d_1 \cdot d_2 + d_2 \cdot d_0 + d_1 \cdot d_0}$$

In Figure 15 the result of the rasterization step is presented in the left, while on the right the original mesh of a single frame is depicted.

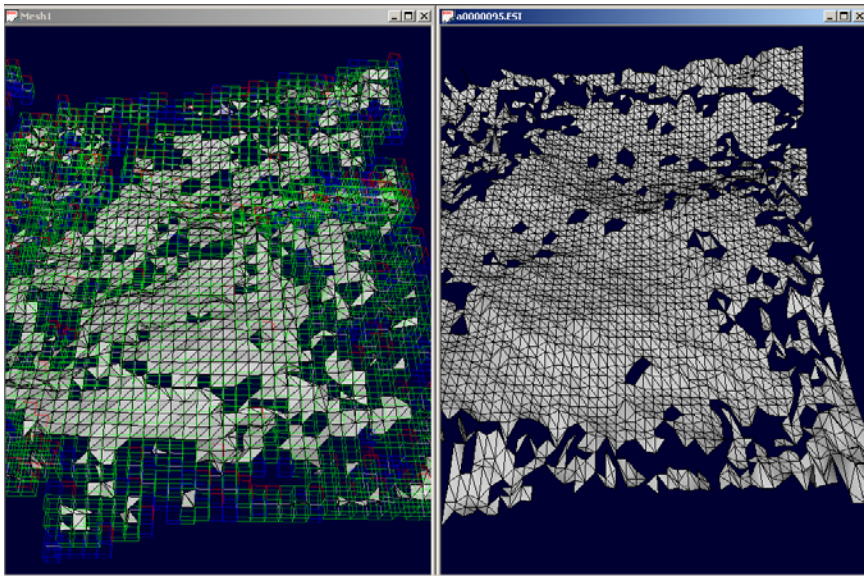

Figure 15 –Surface after rasterization (left) and before (right)

Phase 2: Fusion

After the rasterization phase has been completed on every rasterization plane p_s, the updated intersection lists are scanned and fusion is performed.

As we said before, we follow the same approach as in the original MIA to merge intersections generated by *A* and by *B*, which are close along an edge of the grid.

Note that: (i) while in the original MIA many possible intersections could need to be merged together (because many different frames overlap on the same region), in this case only pairs of intersections will need merge. This suggests how the *on-line approach* is able to distribute the workload through time, by performing only a relatively small number of operations as a new frame comes; (ii) our fusion process uses only those cells intersecting *B* independently from the dimension of *A*.

Merge can be viewed as a warping process that acts on meshes *A* and *B* by moving their vertices along edges of the reference grid in order to align them in regions of overlap.

For every intersection lists (associated with a rasterization line l_h), each couple of consecutive intersections i_1 and i_2 with different name (or timestamp - $t_1 \neq t_2$) and concordant signs are merged together if their distance is shorter than the fusion threshold *FT* taking into account the reliability of the measurement:

$$IC \quad i_1, i_2, i_3;$$
$$if\left(\left(t_{i1} \neq t_{i2}\right) \wedge \left(\mathrm{sgn}_{i1} = \mathrm{sgn}_{i2}\right)\right)\{$$
$$d = \frac{w_{i1} \cdot dir_{i1} + w_{i2} \cdot dir_{i2}}{w_{i1} + w_{i2}};$$
$$D = \frac{\left|ic_{i1} - ic_{i2}\right|}{d};$$
$$if\left(D < FT\right)$$
$$i_3 = Fusion(i_1, i_2);$$
}

The term d in the previous expression takes into account the fact that fusion is performed along the grid lines rather than on minimum distance direction. This operation corresponds to merge two concordant surfaces, which cross the same virtual cell edge.

If the two intersections i_1, i_2 lie on different virtual cells, then the fusion operator performs a shift of the intersections toward the new average location.

During this operation, whenever we move from a virtual cell to another, a couple of artificial intersections are created on all the perpendicular virtual edges the intersection touches. When a new intersection is generated, the algorithm analyzes the other intersections on the same virtual edge in order to verify if a removal operation can be applied.

The removal operation is performed on the currently updated raster lines ($l_1,\ldots,l_q$), that is where a new intersection has been created. A removal action (which removes two intersections from the current raster line) is performed on each pair of intersections (i_1,i_2) which:

- lie on the same edge cell e_c,
- have the same name (or timestamp t_{i1}=t_{i2}), and
- have *different* signs $\mathrm{sgn}_{i1} \neq \mathrm{sgn}_{i2}$.

Phase 3: Mesh generation and lazy update

The mesh generation phase uses a modified versions of the popular marching cubes algorithm for polygonising a scalar field. The basic idea is that 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 mesh construction can be seen as a sequence of single cell meshing.

Given the virtual 3D-grid G, each cell c can be indexed by its edges ($e_{0c},\ldots,e_{11c}$) or its vertices ($v_{0c},\ldots,v_{7c}$). By the use of a proper container structure, such as a hash table or a map, we keep in memory all the necessary structure without allocating too much memory or degrading the performances of the overall system. For cell's indexing we use the same hash function defined above, using the coordinates of the origin vertex, that is the vertex 0 , for generating the key $k(x,y,z)=A\cdot x+B\cdot y+C\cdot z$.

For each virtual cell c, we analyze the intersections along its edges in order to construct (similarly to how we would find out an isosurface) the vertices

(actually, the intersections themselves) and the faces of the relative portion of mesh.

If an edge e_{ic} (i=*0,…,11*) contains an intersection *int* than the classification of its vertices depends on the orientation of that intersection, i.e. sgn_{int}. Successively, faces are generated from the *c*-edges intersections configuration through a lookup table implementing all the possible intersections configurations on edges.

Every time a mesh m_c is created, it is stored as a display list that refers directly to a list of virtual cells and the relative vertices/faces. The current mosaic can be seen as a collection of meshes, each mesh having the same structure described above.

In order to support the graphics system, the algorithm keeps **a geometric structure** consisting of lists of *active cells*, and a corresponding **graphic structure** consisting of different display lists: each list of cells in the geometric structure has a corresponding display list in the graphic structure. Each new frame generates one or more new lists (both geometric and graphic). Such lists contain the new active cells and their corresponding graphical primitives, respectively.

Cells containing portions of surface originated from data with different reliability are distributed among different display lists. This is done because highly reliable data are likely to *survive* longer without modifications.

Old active cells that were modified by processing the new frame *B* are updated in the geometric structure, and each update increases a request value for the relative display list.

In order to speed-up the process of the on-line mosaicing, we decided to implement a *lazy update approach* of the display lists, giving out to the Viewer only the newly generated display lists and those display lists for which the amount of changes exceeds the update threshold *UT*.

So, meshes received by a Viewer at a given frame can be of two kinds:

- **New mesh**: a mesh that was generated in the current frame. In this case, this mesh must be added to the collection held by the Viewer.
- **Modified mesh**: a mesh that was generated in a previous frame and modified in the current frame. In this case, this mesh must substitute the mesh having the same name in the collection held by the Viewer. An empty mesh, i.e., a mesh with zero triangles, means that the mesh with the same name must be deleted from the collection held by the Viewer.

A modified mesh can be discriminated from a new mesh on the basis of existence of his name in the collection of meshes that form the current mosaic.

5. Results

The following section is dedicated to show results obtained for each subsection of the ARROV pipeline. In particular, we report images and function timings for the Registration and Geometric Fusion Steps. At the end of the section, conclusions and future works are listed.

Single Frame Reconstruction

The single frame reconstruction algorithm has been successfully implemented and tested over 3D echoscope data (Figure 11, Figure 18). The results show that a good mesh reconstruction is obtained via a local analysis of the spatial properties of the 3D data, by exploiting the structure of the echoscope sensor.

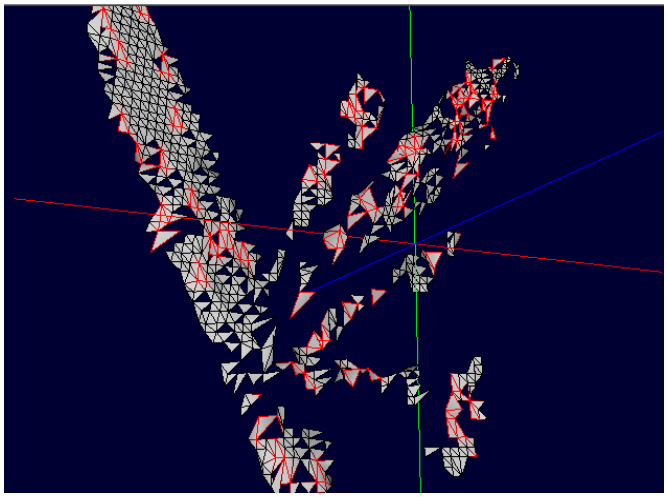

Figure 16 - Faces obtained from step two and three of the procedure are sketched in red.

The reconstruction algorithm depends upon two thresholds, but not critically. The first threshold is over the radial distance between 3D points in the lattice, and is used to evaluate the adjacency relationships among the lattice entries. The second threshold is over the intensity of the 3D points. The intensity values are unevenly distributed over the 3D points, and they represent a measure of the *reliability* of the sensor data itself (Figure 17).

If we decide not to rely on the data in the reconstruction, we can rise up the intensity threshold, therefore excluding points out of the mesh. In this case, the resulting mesh will be much noisier but our approach will show an optimal behavior (Figure 16). Therefore, it is possible to obtain both more reliable and denser meshes.

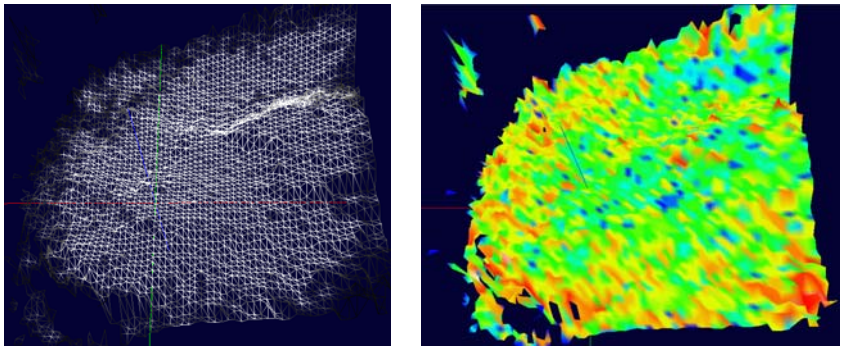

Figure 17 – wireframe visualization. The colour represents the intensities of the points estimated by the Echoscope 1600 (left). Visualization of the local curvature (right)

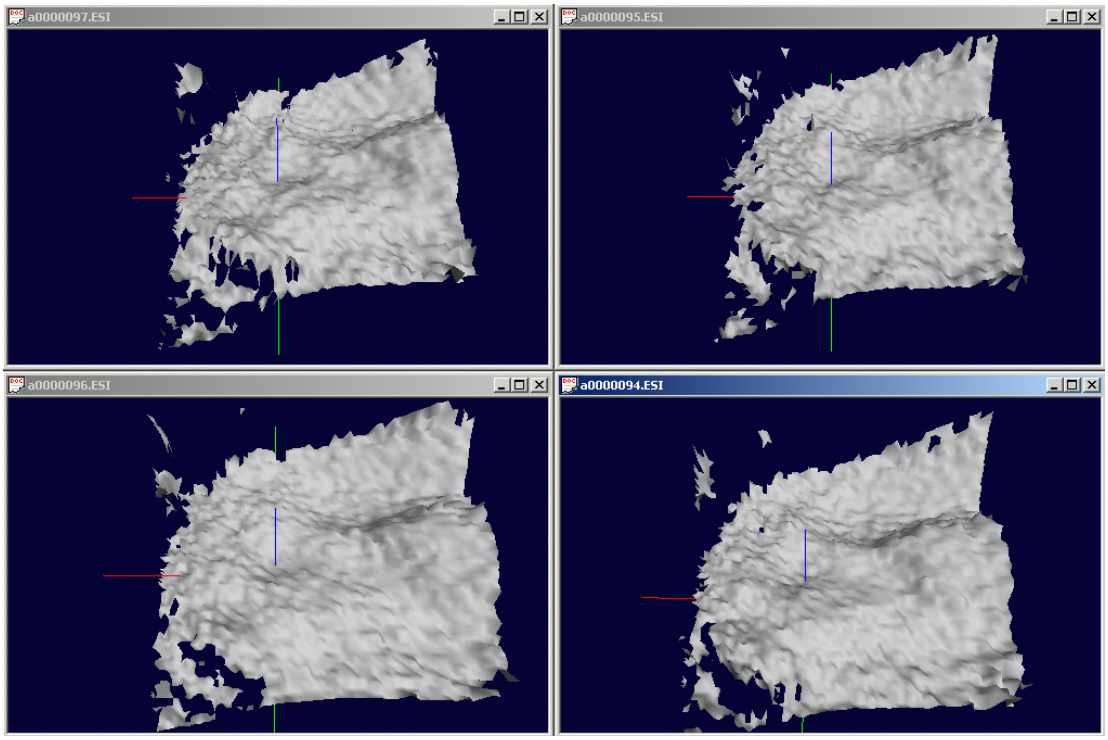

Figure 18 - four different frames reconstructed by the SingleFrameReconstruction() procedure.

Geometric Fusion: Function Timing

The following experimental results have been obtained by elaborating a 10-frame sequence showing an underwater pole structure. Timings have been computed by software profiling, on a P*4 1.5*GHz, with *392*MB Ram, on *10* experiments.

We chose *7* different rasterization steps, from a coarse (*step=50*) to a fine (*step=20*) spatial resolution. Note that spatial resolution depends on the input data range, which is not normalized, and can vary from case to case. Therefore the value of *step=20* means a very fine sampling of the meshes in the current sequence. As we can see the rasterization step affects the total number of the cells, which compose the mosaic, and thus every following computation phase.

Looking at the results, we can remark that the fusion stage does not represent a bottleneck for the mosaic procedure while rasterization does.

We should also stress that rasterization is strongly affected by the size of the original mesh, while the other stages particularly depend on the chosen rasterization step.

Raster Step(int)	Mosaic Cells (msec)	Rasterization (msec)	Cell's Update (msec)	Fusion (msec)	Meshing (msec)
20	14670	587.8	476.6	35.9	1930.5
25	8505	280.9	330.3	29.5	1270.9
30	4343	195.7	247.9	27.5	814.9
35	4010	168.2	217.7	11.2	649.8

40	1521	105.7	134.1	8.6	408
45	1135	62.0	108.6	5.4	285.3
50	942	51.1	91.2	4.5	264.6

Table 2 – rasterization steps, from a coarse (step = 50) to a fine (step = 20) spatial resolution and relative timing form the reconstruction procedures

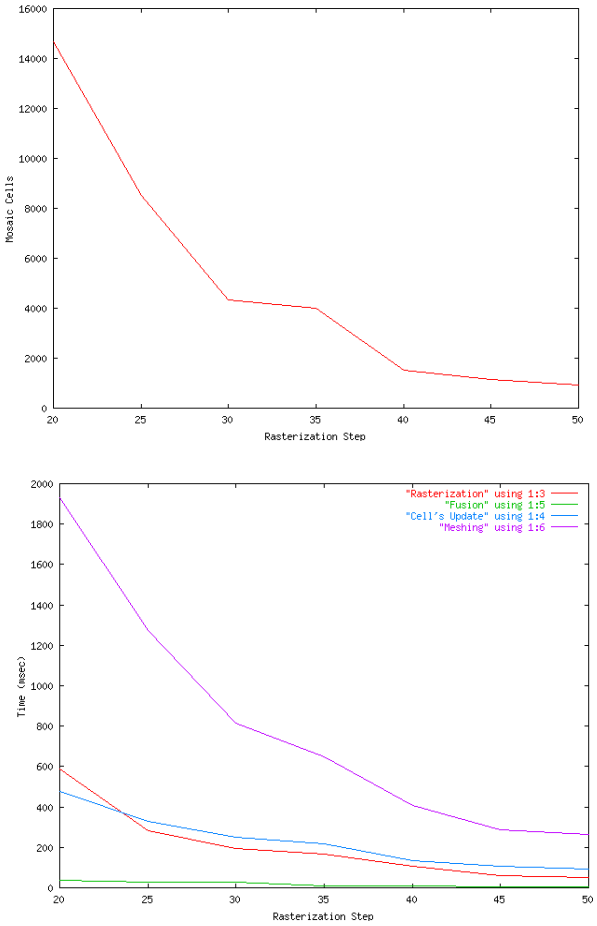

Figure 19 -Graphical timing behavior of the reconstruction phases. (left) Rasterization step (x axis) vs mosaic cells number (y axis).
(right) Time consumption vs rasterization step: red (rasterization), blue (cell update), green (fusion), purple (meshing)

In conclusion, we presented a complete *Data Analysis Framework* developed in the context of the European ARROV project which supports real-time 3D scene reconstruction from a sequence of range data acquired by an acoustic camera (Echoscope 1600) mounted on a ROV.

The Single Frame Reconstruction algorithm depends upon two thresholds, but not critically (radial distance and intensity). The intensity values are unevenly distributed over the 3D points, and they represent a measure of the *reliability* of the sensor data itself (Figure 17).

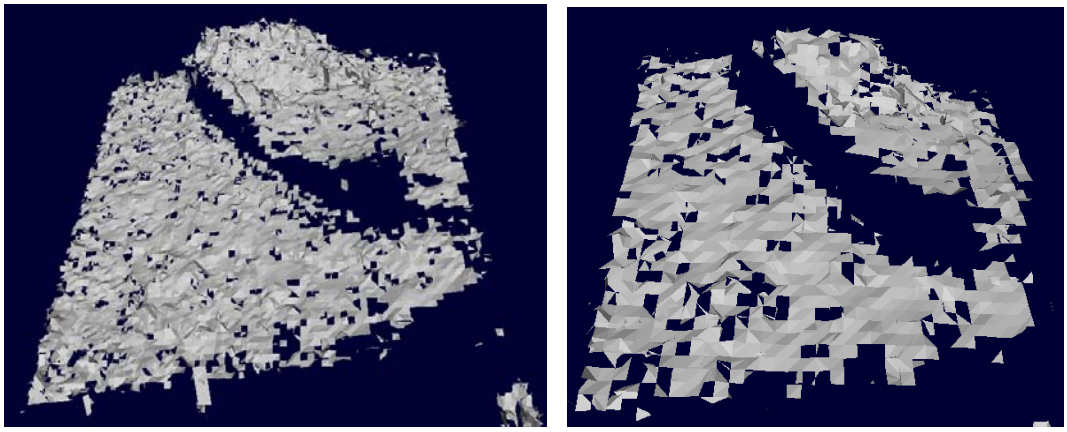

Figure 20 – Mosaic on reconstructed after 4 frames using our Geometric Fusion Procedure. (lfet) gridstep: 20, (right) gridstep:50.

We demonstrated that if we decide not to rely on the data in the reconstruction, we can rise up the intensity threshold but, even if the resulting mesh will be much, approach show goal behavior (Figure 16). Therefore, it is possible to obtain both more reliable and denser meshes.

In the *Geometric Fusion* phase, we developed a reconstruction method on the basis of the Marching Intersection Algorithm (MIA) proposed in [7] adding a lazy update strategy for display lists. Results showed that our method reduces the load per frame on the graphics system and well supports real-time visualization requirements (Table 2 - Figure 23).

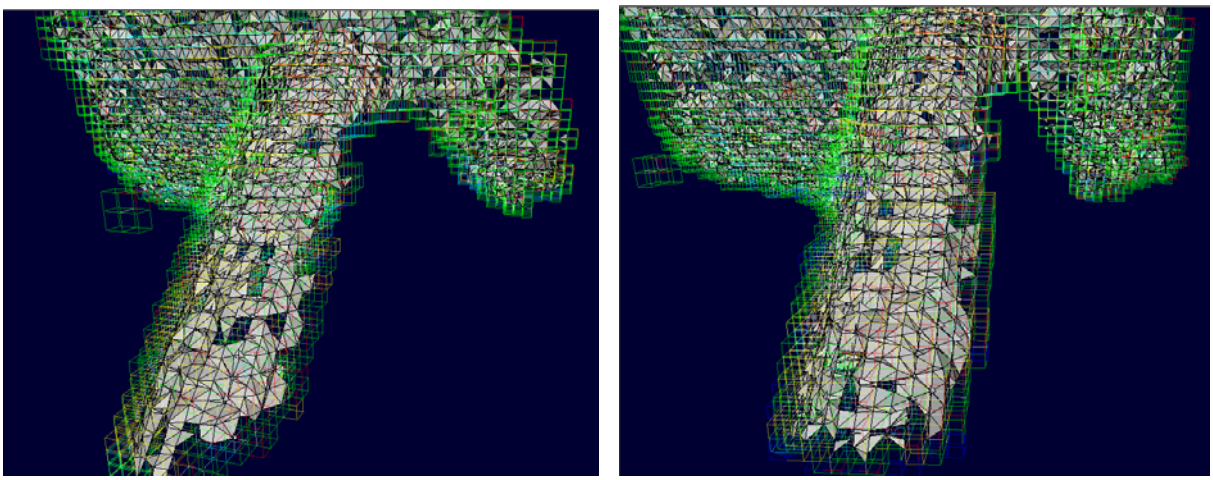

Figure 21 - two mosaic of the same observed scene (left) mosaic of frames from 16 to 22 and (right) mosaic of frame from 16 to 27

The goal achieved by the ARROV project represents a good starting point for the development of fast and accurate methods for Real Time Data Analysis. We will improve accuracy of the results and test the pipeline in different contexts from the underwater inspection.

We believe that our method could well behave also in other situation like fly simulation, inspection, robotics.

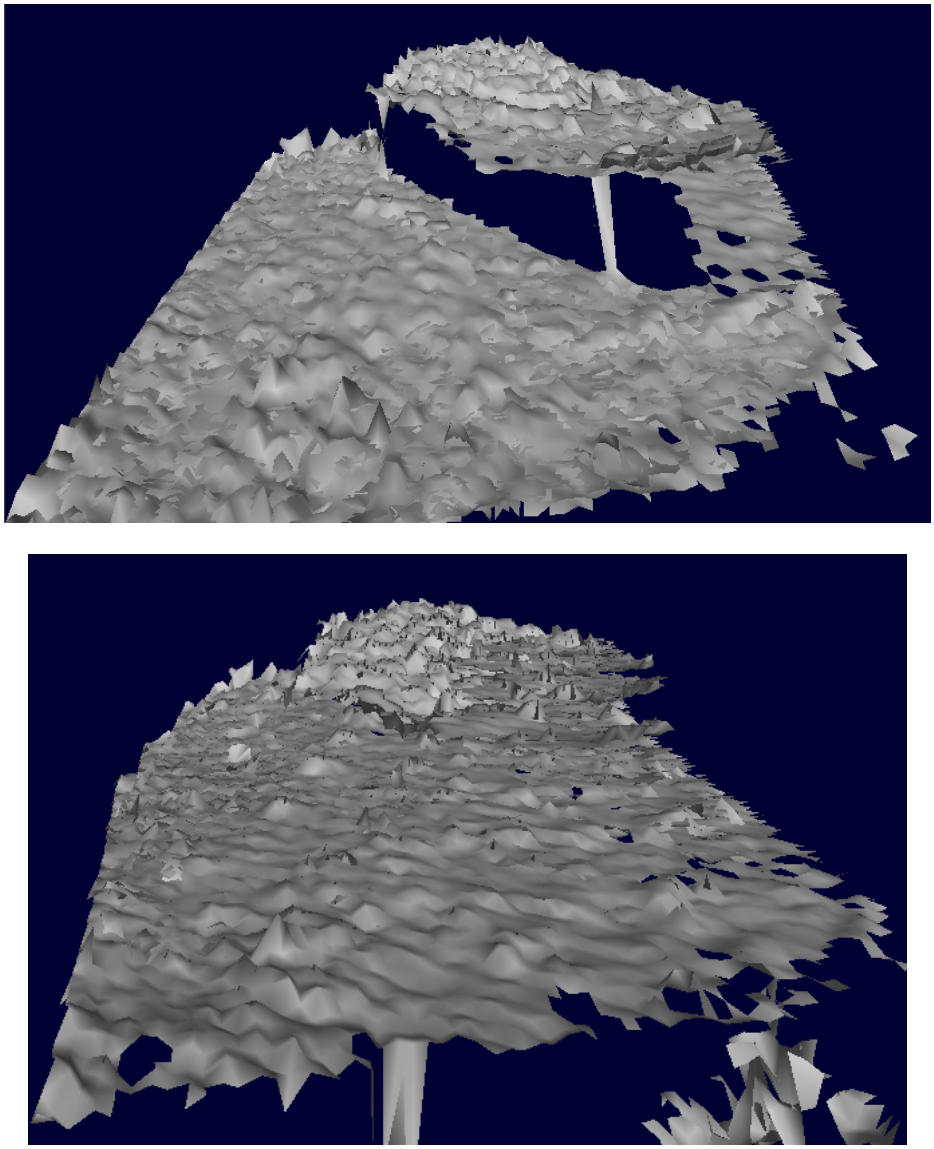

Figure 22 –overlapping of four registered frames (up) accurate registration, (down) fast registration

Finally, the Arrov Pipeline has lead to good results in relation to the types of input coming from the acoustic sensor (Figure 23). In fact, the acquired range images are noisy and at low resolution and present missing data. For these reasons, the Single Frame Reconstruction phase leads to triangular meshes with holes and the problem is present also in the Mosaicing Phase. Part of these holes has been recovered by the use of a method, which searches for connections among entries in a *2x2* and *3x3* window. The real time visualization requirement has been satisfied by the use of a Modified Marching Intersection Algorithm, which maintains a geometric structure consisting of lists of *active cells*, and a corresponding graphic structure consisting of different display lists. In order to speed-up the process, the method *lazily* updates the display lists, giving out to the Viewer only the newly generated display lists and those display lists for which the amount of changes exceeds the *update* threshold.

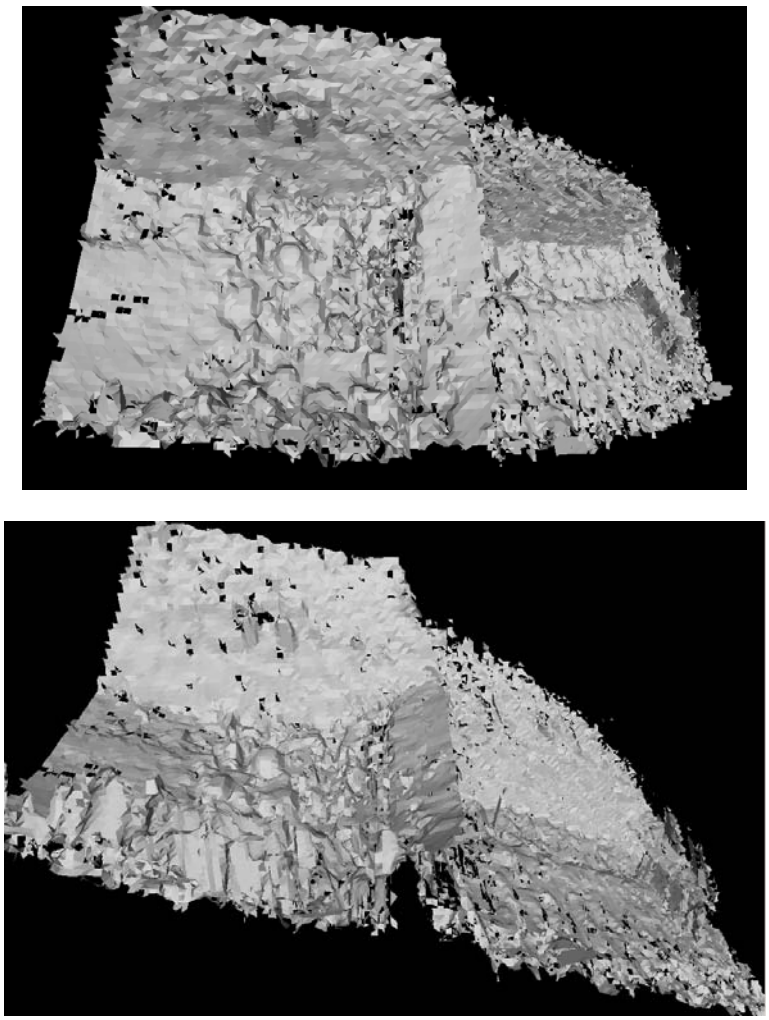

Figure 23 - ARROV mosaic - a result of the reconstruction pipeline on 30 frames.

Some future works can be done in order to optimize the pipeline and improve the results:

- The Marching Intersection Algorithm is sensitive to invalid cells configurations. We will improve the method trying to develop variants, which well-behave in cases of noisy data.
- It is necessary to improve the speed of the pipeline by optimizing the code.
- One important thing is that the pairwise registration of a subsequent frame can lead to error explosion, so sometimes a reset is needed. An improvement of the Registration Techniques is necessary: but this does not fall in the scope of this Dissertation.

The goal achieved by the ARROV project represents a good starting point for the development of fast and accurate methods for Real Time Data Analysis. We will improve accuracy of the results and test the pipeline in different contexts from the underwater inspection (robotics, Aerospatiale research, and so on).

6. Bibliography

[1] ARROV (*Augmented Reality for Remotely Operated Vehicles based on 3D acoustical and optical sensors for underwater inspection and survey*), Project Proposal, 2000.

[2] Castellani U., Fusiello A., Murino V., *Registration of multiple acoustic range views for underwater scene reconstruction*, Computer Vision and Image Understanding, Vol. 87, no. 3, pp. 78-89, 2002.

[3] Castellani U., Fusiello A., Murino V., Papaleo L., Pittore M., Puppo E., *On-line 3D mosaicing from acoustical range data*, 2004, [submitted to VIS04].

[4] Chen Y. and Medioni G., *Description of Complex Objects from Multiple Range Images Using an Inflating Baloon Model,* CVIU Vol. 61, No. 3, May 1995, pp. 325-334.

[5] Chen Y. and Medioni G.. *Object modeling by registration of multiple range images*. Image and Vision Computing, 10(3):145–155, 1992.

[6] Chen Y., Medioni G., *Object Modeling by Registration of Multiple Range Images*. International Journal of Image and Vision Computing, vol.10 n.3, pp.145-155, 1992.

[7] Cignoni P., Montani C., Scopigno R and Rocchini C, *The Marching Intersections algorithm for merging range images*. Technical Report B4-61-00, I.E.I. - C.N.R., Pisa, Italy, June 2000.

[8] Curless B., Levoy M., *A volumetric Method for building complex models from Range Images*. Computer Graphics: Siggraph '96 Proceedings, pp.221-227, 1996.

[9] Fusiello A., Castellani U., Ronchetti L., and Murino V.. *Model acquisition by registration of multiple acoustic range views*. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, Computer Vision - ECCV 2002, number 2351 in Lecture Notes in Computer Science, pages 805-819. Springer, 2002.

[10] Murino V. and Trucco A. *Three-dimensional image generation and processing in underwater acoustic vision*. Proceeding of the IEEE, 88(12):1903–1946, December 2000.

[11] Murino V., Trucco A., and Regazzoni C.. *A probabilistic approach to the coupled reconstruction and restoration of underwater acoustic images*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(1):9–22, January 1998.

[12] Papaleo L., *Surface Reconstruction:Online Mosaicing and Modeling with Uncertainty*, PhD Thesis, University of Genova, 2004.

[13] Papaleo L., Puppo E. , *On-line Mosaicing and Visualization of 3D Meshes from Range Data*, First Conference Eurographics Italian Chapter, Politecnico di Milano, 11-12 July 2002, 2002

[14] Soucy M. and Laurendeau D., *A General Surface Approach to the Integration of a Set of Range Views*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 17, 4, 344-358, 1995.

[15] Trucco E., Fusiello A., and Roberto V.. *Robust motion and correspondence of noisy 3-D point sets withmissing data*. Pattern Recognition Letters, 20(9):889–898, September 1999.

[16] Turk G., Levoy M., *Zippered Polygon Meshes from Range Images*, In proceedings of ACM SIGGRAPH 94, pp.311-318, 1994