Efficient On-line Mosaicing from 3D Acoustical Images

U. Castellani, A. Fusiello, V. Murino *
Department of Computer Science
University of Verona

L. Papaleo, E. Puppo †
Department of Computer and Information Sciences
University of Genova

Stefania Repetto‡
Biophysical and Electronic Engineering Department
University of Genova

M. Pittore §
e-Magine IT srl
Genova

Abstract

This paper presents a system for the 3D reconstruction of an underwater environment on the basis of multiple range views from an acoustical camera. The challenge is to provide the reconstruction on-line, as the range views are obtained from the sensor. The final target of the work is to improve the understanding of a human operator driving an underwater Remotely Operated Vehicle (ROV). The acoustic camera provides a sequence of 3D images in real time (about 5fps in the current version). Data must be registered and fused to generate a unique 3D mosaic in the form of a triangle mesh, which is rendered through a graphical interface. Available technologies for registration and meshing have been modified to match time constraints. We report experiments on real data.

1 Introduction

A Remotely Operated Vehicle (ROV) is a vehicle attached through an umbilical cable to either a ship or a docking station, which is teleoperated by a remote pilot, and is used to perform complex tasks underwater in a variety of domains, including offshore oil industry, underwater construction work, research, environmental studies, sea bottom surveys, survey of shipwrecks, dredging, fisheries etc. A ROV must be equipped with imaging devices and other sensors, in order to provide the necessary feedback to the pilot. Optical cameras and sonar are standard devices which often provide only poor and hardly useful information to the pilot. In fact, most activities are carried out in turbid water.

In the context of European Project ARROV (GROWTH Programme - V Framework) we have investigated the use of an acoustic camera - or 3D multibeam sonar - which generates 3D data from a volume spanned by a single acoustic pulse. This device can offer great advantages over traditional ones, since it is able to provide 3D data in real time. Our final goal is to provide a 3D scene model to the human operator(s) of a ROV, in order to facilitate navigation and understanding of the surrounding environment. The main challenge here is to use 3D data from the acoustic camera in order to build a 3D model on-line, while range images are captured.

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 3 cm at 500 KHz): the higher the frequency, the higher the resolution, the narrower 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.

*e-mail: castella—fusiello—murino@disi.unige.it
†e-mail: papaleo—puppo@disi.unige.it
‡e-mail: repetto@dibe.unige.it
§e-mail: pittore@e-magine-it.it

In order to achieve our goal, we have developed a complete data processing pipeline, which starts from data acquisition, and produces and visualizes a geometric model of the observed scene, in the form of a mesh of triangles. The mesh is called a *mosaic* because it is built by adding new portions of scene frame by frame. The pipeline includes the following stages, each of which is described in a separate section of the paper:

1. *Data capture*, in which the acoustic camera produces a new range image (Sec. 3);
2. *Single frame reconstruction*, in which and a single range image is processed to obtain a triangle mesh (Sec. 4);
3. *Registration*, in which the new frame is brought to the same coordinate system of the 3D mosaic built on all previous frames (Sec. 5);
4. *Geometric fusion*, in which the new frame is merged into the 3D mosaic to update its geometry (Sec. 6);
5. *Visualization*, in which the updates of the mosaic are delivered to the graphic engine (Sec. 7).

In Section 8 we show some results, and in Section 9 we make some concluding remarks.

2 Related work

In this section we report on related work, by focusing on the two most crucial stages of the pipeline, i.e., registration and geometric fusion.

The registration of two points sets is usually performed by the Iterative Closest Point (ICP) procedure [Besl and McKay 1992; Chen and Medioni 1992]. Many variants to ICP have been proposed to cope with partially overlapping views and false matches in general, including the use of thresholds to limit the maximum distance between points [Zhang 1994], disallowing matching on the surface boundaries [Turk and Levoy 1994], and the use of robust regression [Masuda and Yokoya 1995; Trucco et al. 1999].

Albeit early approaches also focused on efficiency, recently this issue has become more and more relevant, as real-time registration has started to become more feasible. In [S. Rusinkiewicz 2001] a survey on the main ICP variations is presented focusing both on the accuracy of results and speed. In order to reduce the time spent finding corresponding points in the ICP procedure, we implemented a technique inspired to the reverse calibration [Blais and Levine 1995], that exploits the spatial organization of range data.

Many methods have been proposed in the last few years to address the problem of shape reconstruction from 3D data. Our input comes in the form of a sequence of range images, each of which is very noisy, and we need to process such images on-line by integrating each of them in the mosaic. So we are interested in methods that:

0-7803-8669-8/04/$20.00 ©2004 IEEE.

accept sets of range images as input; produce an approximating (not interpolant) mesh; can take into account data accuracy; can process sequences of images on-line; offer a good trade-off between speed and accuracy.

The methods proposed in [Curless and Levoy 1996; Rocchini et al. 2001] fulfill all but the last two requirements. They are both based on a discretization of the 3D space. The resolution of the discretization can be used as a trade-off between time performance and accuracy of the result. Both methods rely on the preliminary construction of mesh from each frame, although they use it differently. The method in [Curless and Levoy 1996] uses the registered meshes to evaluate a signed distance field from the surface to be reconstructed. The surface is then extracted as the zero level set of such a field, through the Marching Cubes algorithm [Lorensen and Cline 1987]. On the contrary, the method in [Rocchini et al. 2001] intersects each mesh with the edges of the cells in the discretized space, performs a fusion of intersections that lie close in space, and connects such intersections through a variant of the same Marching Cubes algorithm.

No method has been specifically designed to handle data on-line. A crucial point is that processing a new image should have only local effect on the mosaic and its complexity should depend only on the size of the new mesh. We developed a variant of the method in [Rocchini et al. 2001] – which seemed more suitable to this purpose – which fulfills all such requirements.

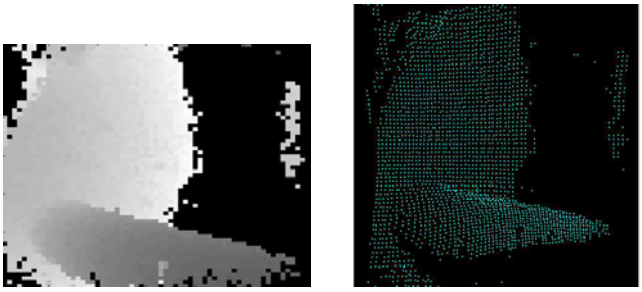
3 Data Capture

Three-dimensional acoustic data are obtained with a high resolution acoustic camera, the *Echoscope* 1600 [Hansen and Andersen 1996]. The scene is insonified 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. Successively, the distance of a 3D point is measured by detecting the time instant at which the maximum peak occurs in the beam signal [Urik 1983].

According to the spherical scanning technology, range values are measured from each steering direction $\mathbf{u}(i,j)$ where i and j are indices related to the elevation *(tilt)* and azimuth *(pan)* angles respectively. The Echoscope carries out 64 measures for both tilt and pan by defining a 64×64 range image $r_{i,j}$. Coordinates are eventually converted from spherical to Cartesian.

Figure 1 shows a range image and the related points cloud. 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 500KHz): roughly speaking, the higher the frequency, the higher the resolution, the narrower the field of view.

Unfortunately, the acoustic image is affected by false reflections, caused by secondary lobes, and by acquisition noise, which is modelled as speckle. The intensity of the maximum peak is used to generate another image, representing the reliability of the associate 3D measures. In general, the higher the intensity, the more reliable the associated distance. A dramatic improvement of the range image quality is obtained by discarding points whose related intensity is lower than a threshold, depending on the secondary lobes [Murino et al. 1998; Murino and Trucco 2000]. The Echoscope 1600 can provide pre-processed images at the rate of about five frames per second.



(a) (b)

Figure 1: A range image (a) and the corresponding cloud of points (b). The scene consists of a pipe underwater

4 Single frame reconstruction

The problem of reconstructing a triangle mesh from a single image is often considered trivial and solved in a straightforward procedure that connects each group of four adjacent pixels to form a pair triangles. Unfortunately, acoustic images are very noisy and data are missing at many pixels. Therefore, we developed a more sophisticated algorithm that tests for possible connections of each pixel and all other pixels in a 5×5 window around it.

Our input consists of an input matrix $I[i][j]$, where each entry (i,j) contains either the coordinates (x,y,z) of a point, or a vacancy. We use a single parameter, a *connectivity threshold* (θ), used to evaluate whether or not two points should be considered adjacent.

The method considers the points corresponding to non-vacant entries and connect them pairwise to form edges: cycles of three edges form triangles of the output mesh. A potential edge exists between a 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 smaller than threshold (θ):

$$\left| \sqrt{x_1^2+y_1^2+z_1^2} - \sqrt{x_2^2+y_2^2+z_2^2} \right| < \theta. \qquad (1)$$

The first phase of the algorithm considers each 2×2 block

$$\mathbf{Bl} = \begin{bmatrix} \mathbf{v_1} & \mathbf{v_2} \\ \mathbf{v_3} & \mathbf{v_4} \end{bmatrix}$$

where v_1 is not vacant tries to find feasible edges: the method tests independently for horizontal and vertical connectivity (v_1, v_2 and v_1,v_3); then it tests for connectivity along diagonal(v_1,v_4); if there is no connectivity along this diagonal, it tests for connectivity along the other diagonal v_3,v_2. The second phase considers each vacant entry v in the lattice, and tests for connectivity between pairs of entries in the 3×3 window surrounding v. The approach first tests vertical and horizontal connectivity, then it tests the two diagonal adjacencies;

In a third phase, 3×3 windows are scanned again, checking for the possible eight semi-diagonal edges connecting entries at the boundary of the window. Again, each potential adjacency is validated only if there are no existing edges crossing it. At most two of the feasible semi-diagonal edges can coexist.

Finally, the isolated vertices are cut off, and the normal vectors for faces and vertices are computed and added to the mesh structure. The connectivity information coming from the mesh is used to filter out connected components smaller than a given size. This latter step, called *size filter*, greatly improves the quality of the acoustic images by filtering speckle noise.

5 Registration

Registration refers to the geometric alignment of a pair or more of 3D point sets. We addressed this problem using the classical Iterative Closest Point (ICP) algorithm [Besl and McKay 1992], a general purpose method for the registration of rigid 3D shapes.

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. However, modifications to the original ICP are now widely used to achieve accurate registration even with fairly general starting positions [Zhang 1994; Turk and Levoy 1994; Trucco et al. 1999]. We implemented a variation similar to the one proposed by Zhang [Zhang 1994], using a modified cost function based on robust statistics to limit the maximum distance between closest points [Castellani et al. 2002].

However, 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: i) reducing the number of iterations necessary to converge and ii) reducing the time spent in each iteration. Finding closest points is the responsible for the bulk of the time spent in each iteration. Corresponding points, however, needs not to be necessarily the closest points. For example, [Chen and Medioni 1992] uses normal shooting and [Blais and Levine 1995] suggests to use the so-called reverse calibration technique, that projects the source point onto the destination mesh, from the point of view of the destination mesh's range camera. We focused on the latter approach and developed an acceleration method based on it.

The technique is based on the fact that the sensor outputs both an unorganized cloud of 3D point V and range image r_V [Besl 1988]. Given a 3D point $v \in V$ (data set), let $r_W(i,j)$ be the projection of v onto the range image of the model set W. The 3D point $w \in W$ associated to $r_W(i,j)$ is the *tentative* corresponding point of v. The connectivity information given by the range image is used to search in the neighborhood N_w of w defined as:

$$N_w = \{w' \in W \mid w' = B(r_W(i+k, j+h)) \; ; \\ h, \ell = -d, \ldots d\}$$

where d is the dimension of a window centered on $r_W(i,j)$ and $B(\cdot)$ is the operator that re-projects a range point onto the Euclidean 3D space. The closest point to v in N_w is taken as the corresponding point of v.

It is worth noting that the range image is not defined everywhere, because after the initial filtering step points have been discarded. If the projection of v falls onto an empty area, this point remains without correspondence.

The projection of a 3D point (x,y,z) onto the range image is specified by the following equation:

$$i = \frac{\alpha - I_{OFF}}{s_\alpha}; \qquad j = \frac{\beta - J_{OFF}}{s_\beta} \tag{2}$$

where

$$\alpha = \arctan\frac{y}{z}; \qquad \beta = \arctan\frac{x}{z} \tag{3}$$

and s_α, s_β, I_{OFF} and J_{OFF} are sensor's parameter that determine its field of view and resolution (see Section 3).

The cost of the projection (i.e., computing i and j) is largely due to the computation of the arctan. In order to avoid it, we store in a table the values of $\tan\alpha_i$ and $\tan\beta_j$, where α_i and β_j are the angles on the model image (i.e., according to the spherical scanning principle), which we know *a priori*. When a 3D point (x,y,z) from data set is processed, the values $\frac{x}{z}$ and $\frac{y}{z}$ are compared with the values stored in the table.

We verified that the alignment based on the reverse projection could fail when the two views are not close enough. To cope with this problem and to increase the robustness of the registration, we run few iteration of the classical ICP algorithm (without reverse projection), which allow to obtain a good pre-alignment from which the reverse-calibration alignment converge fast and correctly to the optimal solution.

6 Geometric Fusion

The meshes built on single frames, when represented in the same coordinate system, after registration, would freely overlap and intersect. Jumps, cracks and redundant surfaces would occur at overlaps. 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.

Since our mosaic is built on-line, at a given instant of time a mesh `M` is given, which represents the mosaic obtained from all previous frames, and a new mesh `SFM` comes, which is built from the current frame, as explained in Section 4. Our aim is to merge such meshes in a consistent way, in order to obtain a new mesh `M'` which represents the scene spanned by both `M` and `SFM`. This must be a fast process, since real time visualization of the mosaic should be supported at each frame. In particular, the workload at each frame should be proportional to the size of mesh `SFM`, which is bounded from above by the fixed size of input images.

Our method is a modification of the *Marching Intersection Algorithm (MIA)* [Rocchini et al. 2001], which offers a valid trade-off between speed and accuracy, and 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 grid is found by joining together intersection points that lie close to each other along an edge of the grid.

In order to make it applicable to an on-line setting, we have modified the MIA to deal with a pre-computed mesh `M` 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 `SFM` which intersects the grid properly. Basically, 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

6.1 Input

At each iteration, our method takes as input: a *single frame mesh* `SFM`; and a *registration matrix* `RM` in homogeneous coordinates, which describes current position and orientation of mesh `SFM` in a global coordinate system. Moreover, the algorithm makes use of two global parameters: a *resolution* value `GR` used to define a *virtual* grid `G` of cubic cells that discretize 3D space; and a *fusion threshold value* `FT`, representing the minimum distance along a grid edge between two vertices generated by different frames. The Fusion threshold is usually smaller than grid resolution.

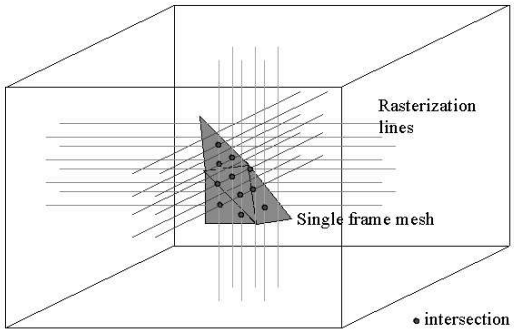


Figure 2: A portion of mesh and the rasterization lines intersecting it.

6.2 Data structures

The current mosaic `M` is initialized by the first frame and it is kept as an internal status of the algorithm. We actually do not store the mesh directly. We rather store the collection of all and only those cells of grid `G` that intersect `M` and, for each cell, the configuration of intersections of its edges with `M`. Such cells are maintained in a map and each cell can be addressed via the coordinates of its center.

We also store three (virtually infinite) 2D arrays, called the *rasterization planes*, which correspond to discretizations at resolution `GR` of the coordinate planes XY, XZ and YZ. Each entry in each array corresponds to a straight line orthogonal to the plane represented by the array, called a *rasterization line* (see Figure 2). This entry actually contains a pointer to a *list of intersections* between meshes `M` and `SFM` and the rasterization line, which are computed during the rasterization phase, and modified during the fusion phase. Each rasterization plane is stored as an associative collection (hash table) of doubly-linked lists, keeping only real intersections.

6.3 Rasterization and Intersection Management

Every face f_q of mesh `SFM` is projected independently on the three rasterization planes XY, XZ and YZ. Each projection is rasterized at resolution `GR` so that each point obtained from rasterization will correspond to an intersection between f_q and one of the rasterization lines, as described before. Thus a simple scan of the raster points is sufficient to update the content of intersection lists attached to the rasterization planes (see Figure 3).

Each intersection is characterized by the intersection value `ic` (i.e., its coordinate along the rasterization line) and a *weight* `w` which measures the *reliability* of the datum and is used in the fusion phase. This 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 follows:

$$\mathtt{w} = \frac{(I_0 \cdot d_1 \cdot d_2 + I_1 \cdot d_2 \cdot d_0 + I_2 \cdot d_1 \cdot d_0)}{(d_1 \cdot d_2 + d_2 \cdot d_0 + d_1 \cdot d_0)} \quad (4)$$

Intersections on a given rasterization line are inserted into the corresponding list, sorted with respect to their values `ic` (see Figure 4). Note that each list of intersections is composed of just a few elements, usually one or two intersections with `M` and one intersection with `SFM`.

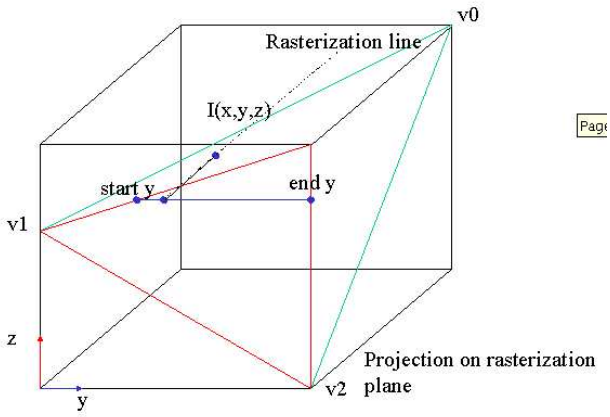


Figure 3: Computation of an intersection point between a triangle (green) and a rasterization line (dotted black). The red triangle is the projection of the green triangle on the rasterization plane (YZ). Each raster point along the blue raster line corresponds to an intersection point. The third coordinate (x in this case) is obtained by linear interpolation.

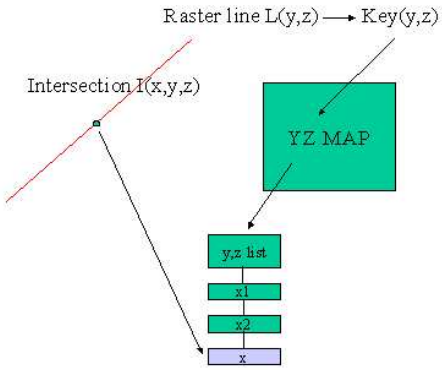


Figure 4: Each intersection point is inserted into the list corresponding to its rasterization line.

6.4 Fusion

Fusion can be viewed as a warping process that acts on meshes `M` and `SFM` by moving their vertices along edges of the reference grid in order to align them in regions of overlap (see Figure 5).

In order to avoid processing all existing intersection lists, we keep track of all and only lists that were updated with intersections on `SFM` during the rasterization phase. Only such lists are scanned to perform fusion. For every intersection list, each pair of consecutive intersections i_1 and i_2 is considered for fusion. Intersections i_1 and i_2 are merged if and only if: (i) one of them belongs to `M` and the other belongs to `SFM`; (ii) the surface normals at both intersections (which are computed during rasterization) have concordant signs; and (iii) the distance between i_1 and i_2 is shorter than the fusion threshold `FT`. Fusion is computed as a weighted average that takes into account the reliability of the measurements w_{i_1}, w_{i_2}.

Figure 6 shows an example of two overlapping surfaces, and their points of intersection with the grid, which are candidate for fusion. Note that, unlikely the case of MIA, in our case only pairs of inter-

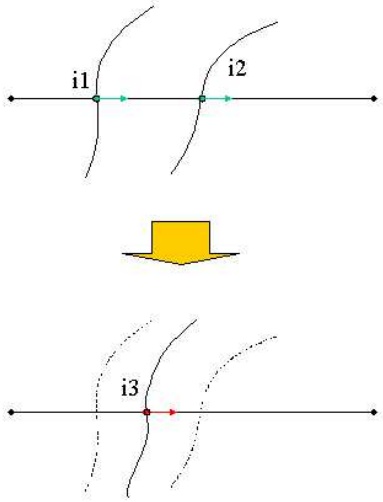


Figure 5: Effect of fusion on two overlapping surfaces (example is 2D for simplicity).

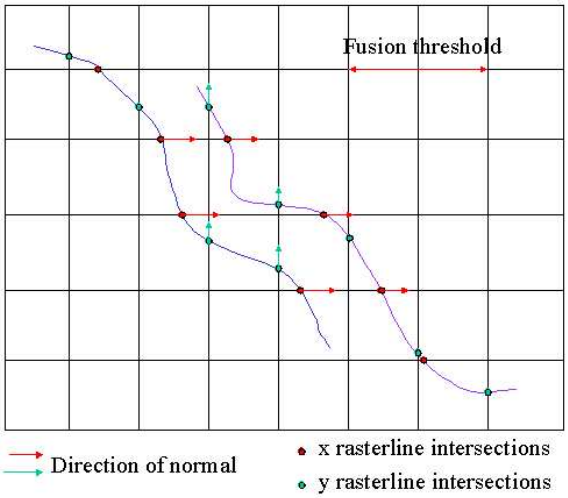


Figure 6: Two overlapping surfaces (example is 2D for simplicity). Pairs of intersection points marked with surface normals will be merged.

sections (one from M and one from SFM) will need to be merged. This suggests how the on-line approach is able to distribute the workload through time, performing a small number of operations as a new frame comes.

The fusion 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. Whenever we move an intersection from a virtual cell to another, additional intersections must be created or some existing intersections must be removed. For details on these latter operations see the original MIA [Rocchini et al. 2001].

6.5 Mesh generation

The mesh generation phase uses a modified versions of the popular Marching Cubes (MC) algorithm [Lorensen and Cline 1987]. The basic idea is that the reconstruction of a 3D surface is completely defined if all the intersections of the surface with the lines of a regular grid are known. Faces of the output mesh can be computed independently on each cell of grid G that contains intersections on its edges, via a MC lookup table. We call such cells the *active cells*, and store them in a map, as described previously.

Each time, during fusion, we either find a new intersection with SFM, or we shift an existing intersection with M because it is fused with another intersection with SFM, we update the existing pool of active cells of grid G accordingly. A cell may change its status for three reasons:

(a) it contains a new intersection;

(b) an existing intersection along one of its edges is moved to a different location along the same edge;

(c) it lost one intersection (which was shifted to a adjacent cell because of fusion).

We keep track of cells that changed their status and we generate only the portion of M corresponding to those cells. Note that the rest of M was generated during previous cycles (when processing previous frames) and remains unchanged.

7 Visualization

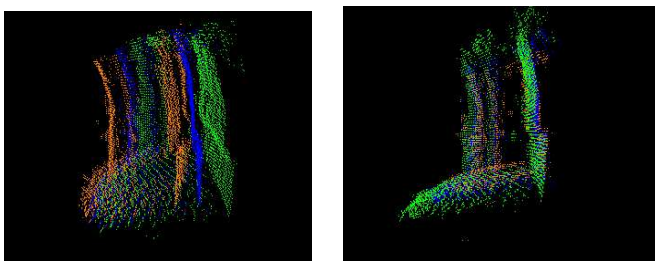
Our data processing pipeline must work in a distribuited system, where the computer performing visualization (called the *viewer*) is different from that performing data processing. Therefore, meshes must go through the bottleneck of a network with limited bandwidth before being delivered to the viewer. In this situation, it would not be practical to resend the whole mosaic at each frame. Therefore, we segment our mesh M into different portions and transmit such portions in batches ba following a *lazy update* approach.

All cells of type (a) found when processing a new frame contribute to form a new portion of mesh, that will correspond to a display list at the viewer. Note that cells of type (a) are not sufficient to update the graphics at a given frame. In fact, also portions of mesh containing existing active cells of type either (b) or (c) should be updated. On the other hand, many lists at the viewer could be possibly updated only because of few changes for each of them.

In order to avoid uncontrolled overload of the transmission channel, we update lists only when changes are beyond a given *update threshold* UT. Active cells of type (b) and (c) are updated in the local geometric structure, and each update increases a *request value* for its related display list. The list is regenerated and transmitted only when the request value exceeds the update threshold.

So, display lists received by the viewer at a given time can be of two kinds:

- **New mesh:** a mesh that was generated in the current frame, from cells of type (a). In this case, this list must be added to those held by viewer.
- **Modified mesh:** a mesh that was generated in a previous frame and modified in the current frame, for more than UT cells with respect to that stored at the viewer. In this case, this list must substitute that having the same name in the collection held by the viewer.



(a) (b)

Figure 7: In (a) three range images before the registration procedure is applied. In (b) the result of registration.

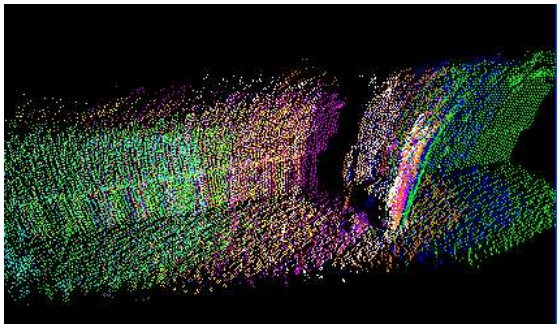


Figure 8: Input data from the whole system after registration.

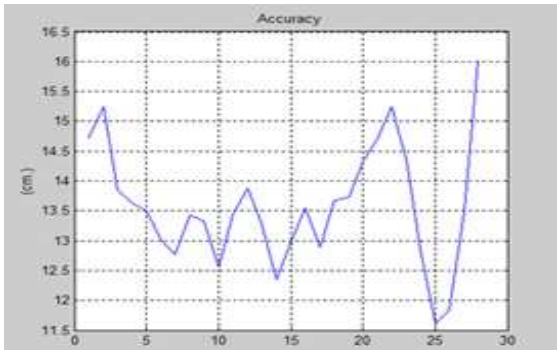
This mechanism proved to be sufficient to support real time visualization in a distribuited environment, without showing relevant artifacts because of updates deferred from the lazy update mechanism.

8 Experiments

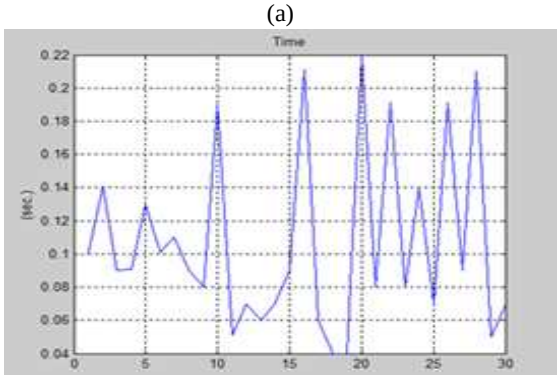
In this section some experiments are presented. We tested the registration algorithm on a sequence of real images in which the scene is composed of an underwater wall and some pillars. The aperture of the sensor was about 90×90 degrees. The sequence is composed by 30 frames and each frame has been registered with respect to the previous one. Figure 7 shows the registration of three images, while Figure 7.a shows the images before the registration and Figure 7.b shows their alignment by allowing the enlargement of the field of view.

Figure 8 shows the points from all the range images represented in the same reference system after registration. The transformations that bring each view on to the mosaic are computed just combining the sequential pairwise matrices. Figure 9 show shows the performance of the registration in terms of accuracy and speed of each frames pair. The mean values of accuracy is 13.58 cm; the mean value of time necessary for registering a pair of images is 0.1069 sec. Therefore, speed of registration supports more than 9 frames per second and it is sufficient to observe a rough mosaic of the scene in real time by rendering just the points. The accuracy of registration is reasonable since it is little higher than the image resolution.

For what concern the integration with the geometric fusion phase, we presented some initial experimental results. They have been obtained by elaborating the same sequence of the experiments above. Figure 10 shows the final result of mosaicing the 30 frames. Timings have been computed by software profiling, on a *P*4 1.5*GHz*,



(a)



(b)

Figure 9: Evaluation of the pairwise registration: accuracy (a) and speed (b). Each value *i* refers to the registration between frame *i* and $i-1$. The accuracy is given by the residual of the last ICP iteration

with 392Mb Ram, on 10 experiments. We chose 5 different rasterization steps, from a coarse ($\texttt{step} = 50$) to a fine ($\texttt{step} = 15$) 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 $\texttt{step} = 20$ 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 will compose the mosaic, and for this reason, every successive computation phases.

Looking at the results, we can remark that the fusion phase of the mosaicing pipeline does not represent a bottleneck for the entire procedure, while rasterization does. We should also stress that rasterization is strongly affected by the size of the original mesh, while the other phases (particulary meshing) depend on the chosen rasterization step.

Raster Step*(int)*	Mosaic Cells *(msec.)*	Rasterization *(msec.)*	Fusion *(msec.)*	Cell Update *(msec.)*	Meshing *(msec.)*
15	23414	209	228	981	1000
20	11838	125	135	554	561
30	4384	63	67	250	251
40	2224	43	45	163	163
50	1233	32	33	99	99

Table 1: 7 rasterization steps, from a coarse ($\texttt{step} = 50$) to a fine ($\texttt{step} = 15$) spatial resolution and relative timing form the reconstruction procedures

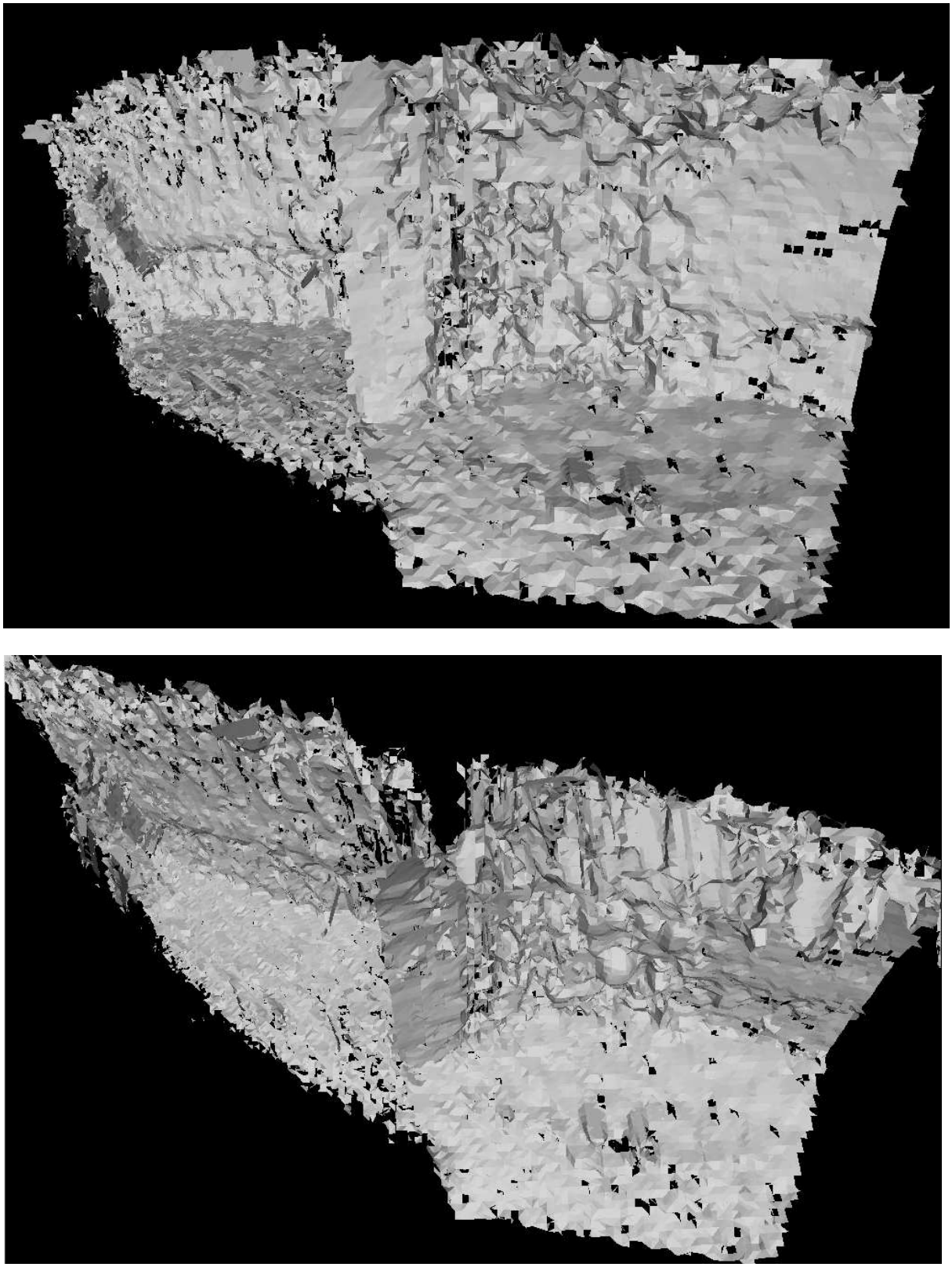


Figure 10: On-line mosaic from 30 frames. Total number of faces is about 100.000

9 Conclusions

This paper presented a technique for on-line 3D scene reconstruction from a sequence of range data acquired by an acoustic camera. The final goal was to provide a 3D scene model to the human operator of an underwater remotely operated vehicle (ROV), in order to facilitate navigation and understanding of the surrounding environment.

As we said in Section 1, registration and geometric fusion are the critical phases for the on-line version of our pipeline. In the registration phase, we modified the ICP algorithm reducing the number of points to be registered by subsampling. Moreover, we reduced the number of iterations and the time spent in each iteration by implementing a corresponding point search procedure inspired to the reverse calibration [Blais and Levine 1995]. For the geometric fusion phase, we developed an on-line reconstruction method on the basis of the Marching Intersection Algorithm (MIA) proposed in [Rocchini et al. 2001]. At each new frame, only the grid cells intersecting it need to be analyzed, so that the computational load is dependent on the size of the new image, but independent on the size of the whole mosaic.

In order to prevent overload of the graphics engine, we also developed a lazy update strategy for display lists, which updates the graphical structures in batches and avoids delivering large meshes all togther and too frequently.

Our processing pipeline already meets the rate at which images are captured by the acoustic sensor, thus supporting interactive use. We are currenting integrating a motion tracking technique based on Kalman filter, which exploits feedback from motion sensors on board the ROV in order to improve and speed up registration. We are also optimizing the code of the fusion phase. We are confident that with such improvments, our system will became more accurate and at least twice as fast.

Acknowledgments

This work was partially supported by the European Commission under the project no. GRD1-2000-25409 ARROV (Augmented Reality for Remotely Operated Vehicles based on 3D acoustical and optical sensors for underwater inspection and survey) and by the Italian Space Agency (ASI) under project AUREA (AUgmented REAlity for the teleoperation of free-flying robots"). The implementation of the ICP is partially due to Linmi Tao.

References

BESL, P., AND MCKAY, N. 1992. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 2 (February), 239–256.

BESL, P. J. 1988. Active, optical imaging sensors. *Machine Vision and Applications*, 127–152.

BLAIS, G., AND LEVINE, M. D. 1995. Registering multiview range data to create 3-D computer objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence 17*, 8, 820–824.

CASTELLANI, U., FUSIELLO, AND MURINO, V. 2002. Registration of multiple acoustic range views for underwater scene reconstruction. *Computer Vision and Image Understanding 87*, 3 (July), 78–89.

CHEN, Y., AND MEDIONI, G. 1992. Object modeling by registration of multiple range images. *Image and Vision Computing 10*, 3, 145–155.

CURLESS, B., AND LEVOY, M. 1996. A volumetric method for building complex models from range images. *Computer Graphics 30*, Annual Conference Series, 303–312.

HANSEN, R. K., AND ANDERSEN, P. A. 1996. A 3-D underwater acoustic camera - properties and applications. In *Acoustical Imaging*, P.Tortoli and L.Masotti, Eds. Plenum Press, 607–611.

LORENSEN, W., AND CLINE, H. 1987. Marching cube: A high resolution 3d surface construction algorithm. *Computer Graphics 21*, 4, 163–170.

MASUDA, T., AND YOKOYA, N. 1995. A robust method for registration and segmentation of multiple range images. *Computer Vision and Image Understanding 61*, 3 (May), 295–307.

MURINO, V., AND TRUCCO, A. 2000. Three-dimensional image generation and processing in underwater acoustic vision. *Proceeding of the IEEE 88*, 12 (December), 1903–1946.

MURINO, V., TRUCCO, A., AND REGAZZONI, C. 1998. A probabilistic approach to the coupled reconstruction and restoration of underwater acoustic images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 1 (January), 9–22.

ROCCHINI, C., CIGNONI, P., GANOVELLI, F., MONTANI, C., PINGI, P., AND SCOPIGNO, R. 2001. Marching Intersections: an efficient resampling algorithm for surface management. In *Int. Conf. on Shape Modeling and Applications*, IEEE Comp. Society, Genova, Italy, 296–305.

S. RUSINKIEWICZ, M. L. 2001. Efficient variants of the icp algorithm. In *IEEE Int. Conf. on 3-D Imaging and Modeling, 3DIM '01*.

TRUCCO, E., FUSIELLO, A., AND ROBERTO, V. 1999. Robust motion and correspondence of noisy 3-D point sets with missing data. *Pattern Recognition Letters 20*, 9 (September), 889–898.

TURK, G., AND LEVOY, M. 1994. Zippered polygon meshes from range images. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, ACM Press, A. Glassner, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, 311–318. ISBN 0-89791-667-0.

URIK, R. J. 1983. *Principles of Underwater Sound*. McGraw-Hill.

ZHANG, Z. 1994. Iterative point matching of free-form curves and surfaces. *International Journal of Computer Vision 13*, 2, 119–152.