A Complete System for On-line 3D Modelling from Acoustic Images

U. Castellani*, A. Fusiello*, V. Murino*, L. Papaleo+, E. Puppo+, M. Pittore§

*Dipartimento di Informatica - Università di Verona, Italy

+Dipartimento di Scienze dell'Informazione - Università di Genova, Italy

§E-magine-it, Genova

March 21, 2005

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. The acoustic camera provides a sequence of 3D images in real time. 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 and extended to match time constraints. Some experiments on real data are reported.

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 (*Augmented Reality for Remotely Operated Vehicles based on 3D acoustical and optical sensors for underwater inspection and survey* - 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 (about 5fps in the version available to us). 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 5 cm at 500 kHz): the higher the frequency, the higher the resolution, the narrower the field of view. Consequently we are forced to

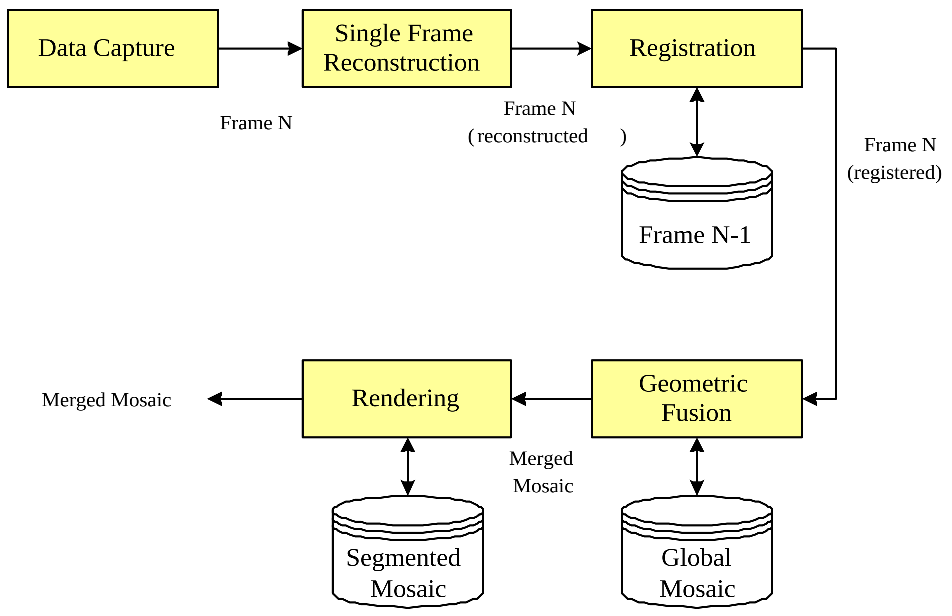


Figure 1: Overall architecture of the system.

operate with a limited field of view and a technique to reconstruct progressively the scene while the sensor is moving is necessary. This technique is called *3D mosaicing*, because it consists in juxtaposing new portions of scene frame by frame.

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 overall architecture of the system is outlined in Figure 1. The data processing pipeline includes the following stages, each of which is described in a separate section of the paper:

1. Data capture (Sec. 3), in which the acoustic camera captures a new range image;
2. Single frame reconstruction (Sec. 4), in which a single range image is processed to obtain a triangle mesh;
3. Registration (Sec. 5), in which the mesh produced by the previous step is brought

into alignment with the (partial) mosaic. This is accomplished by aligning it with the previous frame, which needs to be stored;

4. Geometric fusion (Sec. 6), in which the new mesh is merged into the 3D mosaic that is being built, thereby updating its geometry;

5. Rendering (Sec. 7), in which only the parts of the mosaic that have been updated are delivered to the graphic engine (*lazy update*).

Experimental results are reported in Section 8. Section 9 concludes the paper with some remarks and directions for future work.

2 Related work

Very few systems are addressed to the reconstruction of underwater environments and, for the best of our knowledge, none of them is able to operate in real-time [33]. Kamgar-Parsi [17] proposed an *acoustic lens* technique for 3D data acquisition from which 3D models are recovered by using standard volumetric approach. Negahdaripour [25, 26] focused on some computer visions techniques such as *shape from stereo and video, optical flow estimation, 2D mosaicing* by using optical camera(s). Regarding non-underwater applications, several works have been proposed in the literature [16, 1]. In particular, some examples of real time system are proposed in [30, 19]. Rusinkiewicz [30] introduced a complete model acquisition system that permits the user to see continuously the update of the model as the object is scanned. The real time constraint is satisfied by the use of a simple and fast structured-light range scanner and by speeding up the performance of the registration phase. A similar approach is proposed by Koninckx et. al. [19]. The real time acquisition is guaranteed by adapting the code of the projected stripes. The reconstruction is carried out in *one-shot* by allowing the modelling of deformable surfaces. In a previous project called VENICE (*Virtual Environment Interface by Sensory Integration for Inspection and Manipulation Control in Multifunctional Underwater Vehicles* - EU, IV Framework) a 3D synthetic model of the observing scenario was available and the research was mainly focused on the alignment between synthetic and real environment [12, 22]. In the context of ARROV project, the scene is unknown and the 3D reconstruction of the observation is obtained by adopting an automatic *modelling* approach [2, 37].

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 a resulting *mosaic*. The most crucial stages of the modelling pipeline are registration and geometric fusion, especially for the time constraint.

The registration of two points sets is usually performed by the Iterative Closest Point (ICP) procedure [3, 10]. 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 [38], preventing matching on the surface boundaries [35], and the use of robust regression [21, 34].

The finding of closest points is responsible for the bulk of computational complexity of ICP algorithm. Early approaches focused on data structures tailored to answer efficiently to closest-point queries (e.g., k-D trees [38]). The problem of speeding-up the ICP algorithm is still open and new techniques [13, 18, 32, 28] have been recently proposed.

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 [31] 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 based on the reverse calibration [6], that exploits the spatial organization of range data.

As for the geometric fusion, we are mainly interested in methods that accept sets of range images as input and produce an approximating (not interpolant) mesh; moreover, they should be able to take into account the accuracy of the data, to process sequences of images on-line and, finally, to offer a good trade-off between speed and accuracy.

The methods proposed in [11, 29] fulfill all but the last two requirements. They are both based on a subdivision of the 3D space via a regular volumetric grid. The resolution of subdivision determines the resolution of the output mesh and it can be used as a trade-off between speed of processing 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 [11] 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 well known Marching Cubes algorithm [20]. On the contrary, the method in [29] 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 therefore started from the approach in [11] and we modified it to meet requirements of the on-line setting.

3 Data Capture

Three-dimensional acoustic data are obtained with a high resolution acoustic camera, the *Echoscope* 1600 [15]. 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 can be measured by detecting the time instant at which the maximum peak occurs in the beam signal [36]. 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.

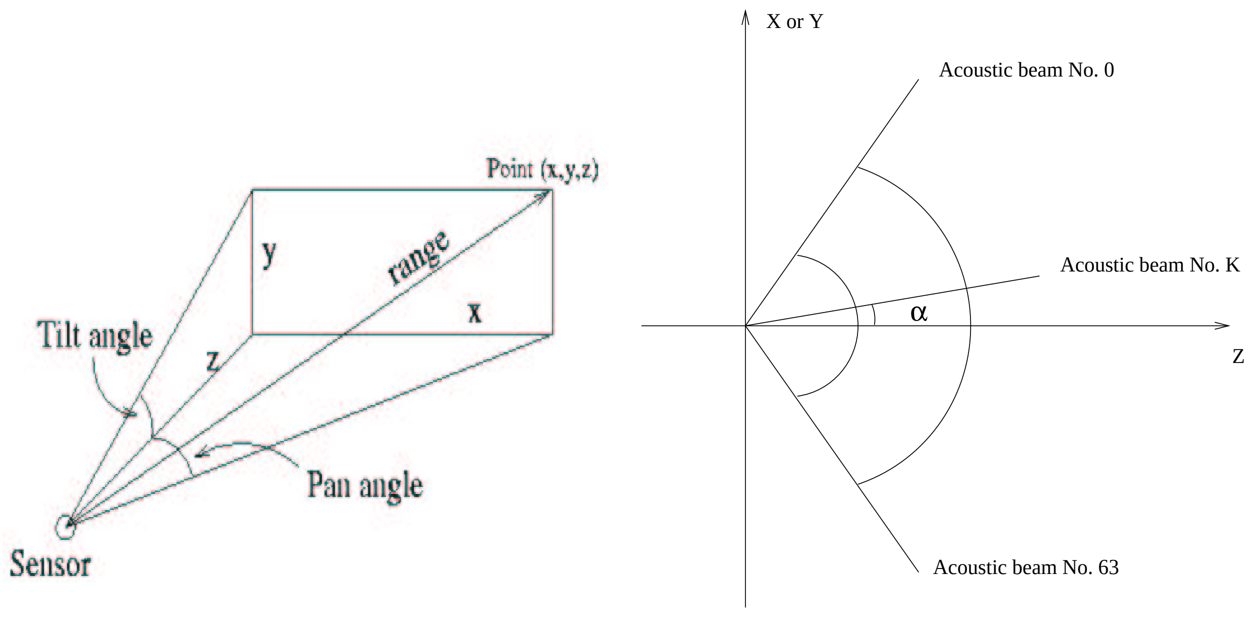
Let us denote by $\mathbf{v_k}$ the position of the k-th sensor (transducer), let c be the sound velocity, and let $x_k(t)$ be the signal received by the k-th sensor. Beamforming can form a *beam signal*, $bs_{\mathbf{u}}(t)$, steered in the direction of the vector $\mathbf{u}$, defined as:

$$bs_{\mathbf{u}}(t) = \sum_{k=0}^{M-1} \omega_k \cdot x_k(t - \theta_k) \tag{1}$$

where ω_k are the weights assigned to each sensor, M is the number of transducers, and

$\theta = (\mathbf{v}_k \cdot \mathbf{u})/c$ 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 [36]. 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 $\mathbf{u}(i, j)$ where i and j are indices related to the elevation *(tilt)* and azimuth *(pan)* angles respectively. Figure 2.(a) shows a schema of the scanning principle. Figure 2.(b) 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.



(a) (b)

Figure 2: Spherical scanning principle (a) and subdivision of the beams onto the acquiring volume (b). Each beam is associated to a (i, j) coordinate of the range image

Going into details, the Echoscope carries out 64 measures for both tilt and pan by defining a 64×64 range image $r_{i,j}$. Spherical coordinates are converted to usual Cartesian coordinates, referring to a coordinate system centered at the camera, by the use of the following equations [5]:

$$x = r_{i,j} \tan(js_\alpha)/\sqrt{1 + \tan^2(is_\alpha) + \tan^2(js_\beta)} \qquad (2)$$

$$y = r_{i,j} \tan(is_\beta)/\sqrt{1 + \tan^2(is_\alpha) + \tan^2(js_\beta)} \qquad (3)$$

$$z = r_{i,j} \sqrt{\tan^2(is_\alpha) + \tan^2(js_\beta)} \qquad (4)$$

where s_α and s_β are elevation and azimuth increments respectively.

The result is a cloud of 3D points in xyz coordinates, each of which refers to an entry of a 64×64 matrix. Figure 3 shows a range image and the related points cloud. The scene consists of a joint composed by a vertical pipe and an horizontal pipe which is oriented toward the viewer. Both images are provided by the Echoscope. 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 is the frequency, the higher is the resolution, the narrower is the field of view.

Unfortunately, due to secondary lobes and acquisition noise, the acoustic image is affected by false reflections which is modelled as speckle. Moreover, the intensity of the maximum peak can be used to generate another image, representing the reliability of the associated 3D measures, so that, in general, the higher is the intensity, the safer the associated measure. 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 [24, 23].

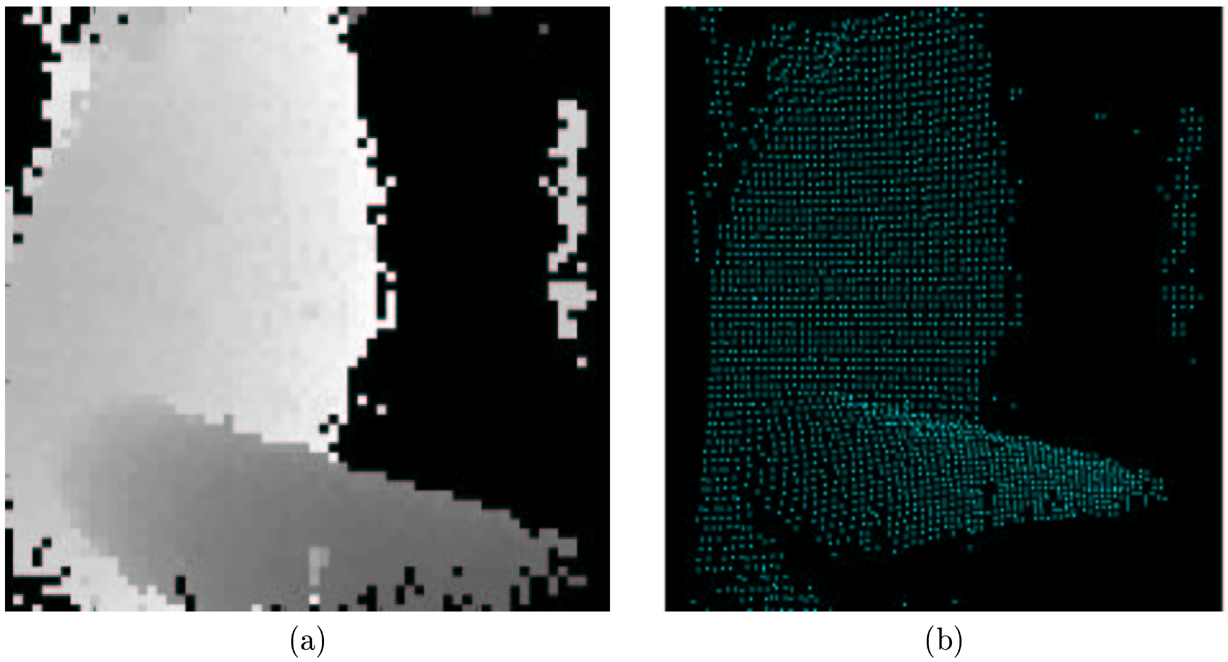


Figure 3: Range images (a) and cloud of points (b). The scene consists of a join of pipes in underwater

4 Single-Frame reconstruction

The next phase of our data processing pipeline consists of taking the point cloud, in the form of a matrix of 3D points, and building a triangle mesh by properly connecting such points.

The problem of reconstructing a triangle mesh from a single range image is usually considered trivial and solved in a straightforward way by connecting each group of four adjacent points in the image to form a pair of triangles. Therefore, only points having adjacent entries in the input image may be connected.

On the other hand, acoustic range images are very noisy and data are missing at many pixels. In order to reduce the number of undesired holes, we extended the classical approach by developing an algorithm that, besides testing for usual adjacencies, also tests for possible adjacencies in a 3×3 window that surrounds each vacancy in the input.

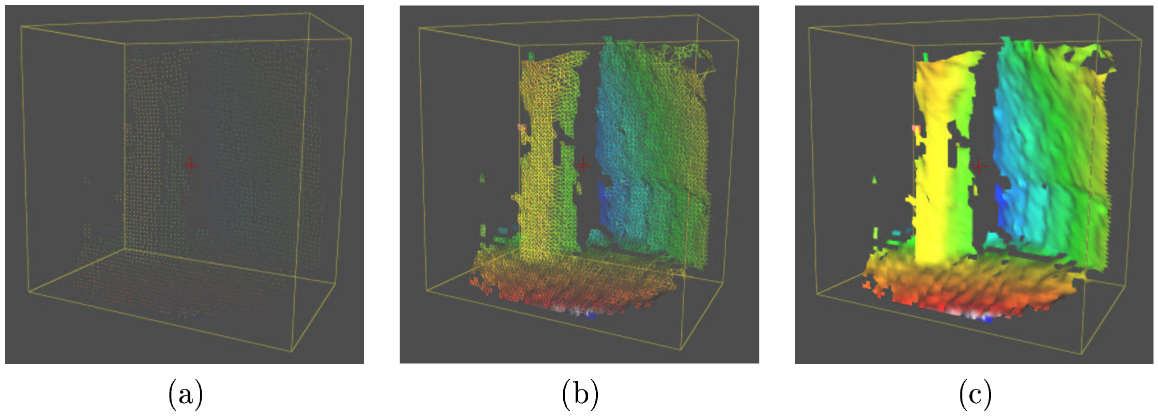
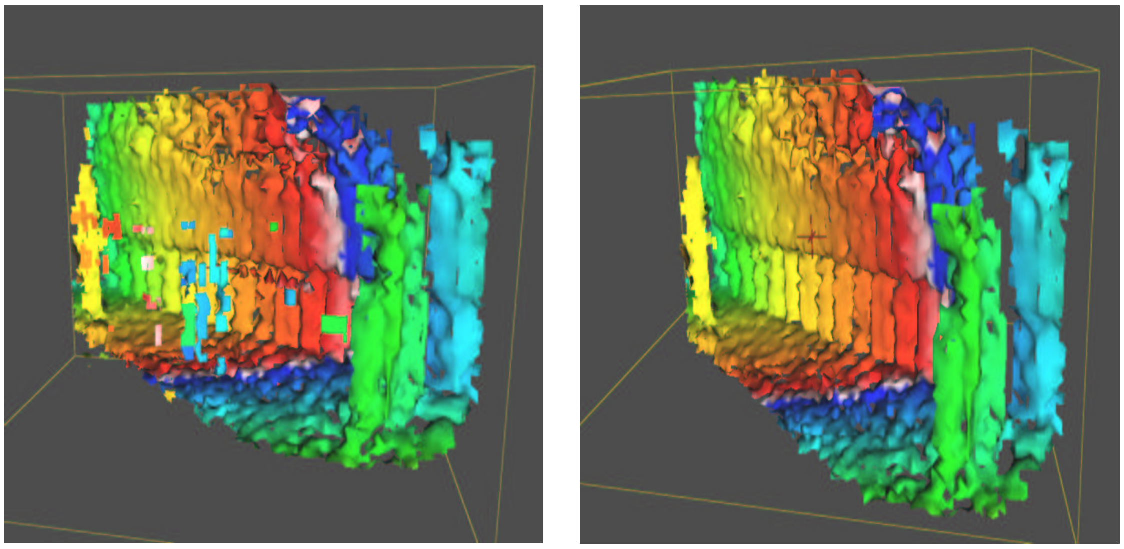


Figure 4: Single frame reconstruction: the cloud of points received from sensors (a) and the single frame mesh resulting from it in wire-frame (b) and shaded (c). Bounding box and false colors are used to enhance perception of depth.

In order to avoid connecting pairs of points that have adjacent entries but belong to different surfaces (i.e., they lie across a crack in the range image), a potential adjacency generates an edge only if the radial distance of points is smaller than a given threshold. Triangles are obtained by considering cycles of three edges. The output of this phase is a mesh of triangles, with surface normals estimated at all vertices of the mesh and pointing towards the observer (sensor), which will be called a *single frame mesh* in the following (see Figure 4).

A topological data structure is built on-the-fly, which allows us traversing the mesh by triangle adjacencies in order to identify connected components in the mesh. Connected components having a size smaller than a given threshold are filtered out, since they are likely to be generated by spurious data. This method proved to be very effective in filtering speckle noise (see Figure 5).

Further details on single frame reconstruction and on filtering of connected components can be found in [9, 27].



(a) (b)

Figure 5: Speckle noise from sensor generates many fragments (clearly visible in different colors) located in front of the main wall and roughly aligned with the two pillars in (a); the effects of speckle noise are removed in by filtering small connected components of the mesh (b).

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 [3], 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 [31]. We adopted an approach similar to the one proposed by Zhang [38], using a modified cost function based on robust statistics to limit the maximum distance between closest points

[7]. As pointed out by Zhang [38], the distribution of the residuals for two fully overlapping sets approximates a Gaussian, when the registration is good. The non-overlapped points skew the distribution of the residuals, hence the threshold on the distance must be set using a robust statistics. In our algorithm the threshold is automatically defined by introducing the so called X84 rule [14]: the points whose residual differ more than 5.2 MAD (Median Absolute Deviations) from the median are discarded. The value 5.2 corresponds to about 3.5 standard deviations, which encloses more than 99.9% of a Gaussian distribution.

However, in order to satisfy the time constraints, 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, need not to be necessarily the closest points. For example, [10, 28] use normal shooting and [6, 31] suggest 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 proposed technique is based on the fact that the sensor outputs both an unorganized cloud of 3D point V and range image r_V [4]. 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:

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

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 Cartesian 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:

$$i = \frac{\alpha - I_{OFF}}{s_\alpha}; \qquad j = \frac{\beta - J_{OFF}}{s_\beta} \tag{5}$$

where s_α and s_β are described in section 3 , I_{OFF} and J_{OFF} are offsets and finally α and β are given by:

$$\alpha = arctg\frac{y}{z}; \qquad \beta = arctg\frac{x}{z} \tag{6}$$

The parameters s_α, s_β, I_{OFF} and J_{OFF} are fixed by the acquisition sensor and they determine the aperture of the acquisition (i.e., field of view and resolution)(Figure 2.b).

In summary, the algorithm for speed up the finding of corresponding points is given by the following steps:

For each 3D data-point $v_i \in V$

- find i and j by using eq. 5 and 6

- project v_i on to the model range image $r_W(i,j)$
- find the *tentative* corresponding point w_i
- find the neighborhood N_{w_i} of w_i
- find the *definitive* corresponding point w_i' (if it exists)

We verified that the alignment based on the reverse projection could fail when the two views are not close enough. In order to cope with this problem, we run few iterations of the classical ICP algorithm (without reverse projection) which provides a good pre-alignment. Moreover, as we mentioned before, the $X84$ rule is applied to the corresponding points in order to discard the outliers. It is worth noting that both pre-alignment and outlier rejection may slow down the processing by trading-off between speed and accuracy. More details on the proposed registration algorithm are reported in [8, 9]. Figure 6 shows a sideways view of the registration of three images. The scene consists of an underwater wall (on the right part), the seabed (on the bottom part) and one pillar (on the central part). Figure 6.a shows the images before the registration and Figure 6.b shows their alignment by evidencing the preserving of the shapes profiles.

6 Geometric Fusion

Different single frame meshes registered in a common coordinate system contribute to form the 3D mosaic. Such mosaic must be built and visualized on-line, while frames are acquired from the sensor. This requirement poses some serious challenges:

- The method used to build the mosaic must be fast. For instance, the complexity

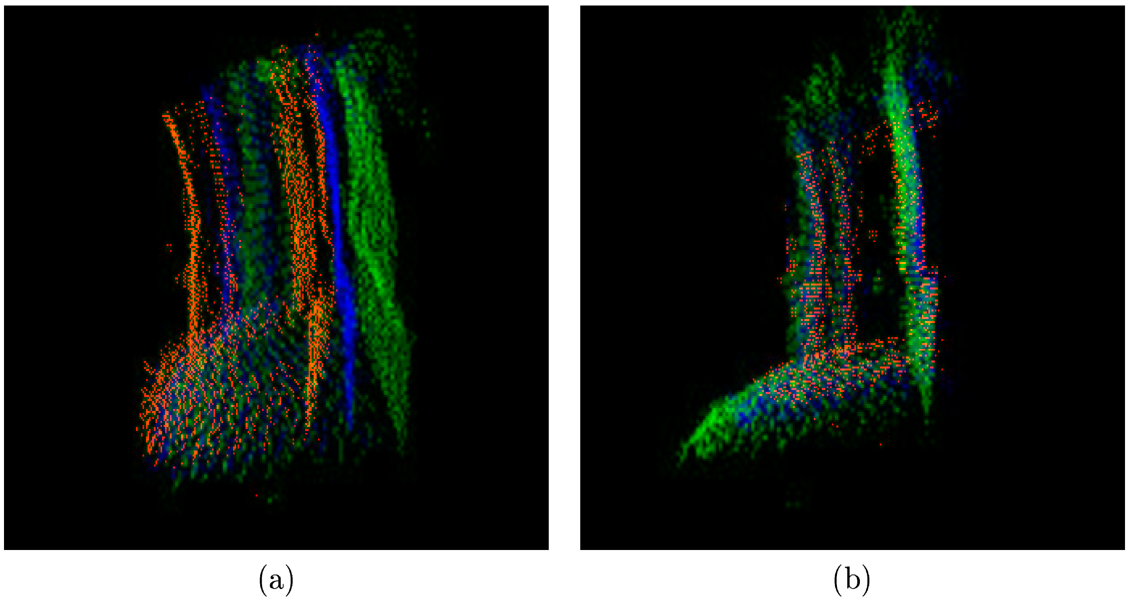


Figure 6: In (a) three images before the registration procedure is applied. In (b) the result of registration.

of updating the mosaic with one new frame should be proportional to the size of a single frame, not to the size of the whole mosaic.

- Frame-to-frame updates must be delivered to the rendering system without the need of regenerating the whole mosaic mesh.
- It should be possible to trade-off between speed, accuracy of the result and size of the output mesh.

We therefore started from the method proposed by Curless and Levoy in [11] and we modified it to meet our requirements. We first give a brief description of the base method, and next we describe our modifications.

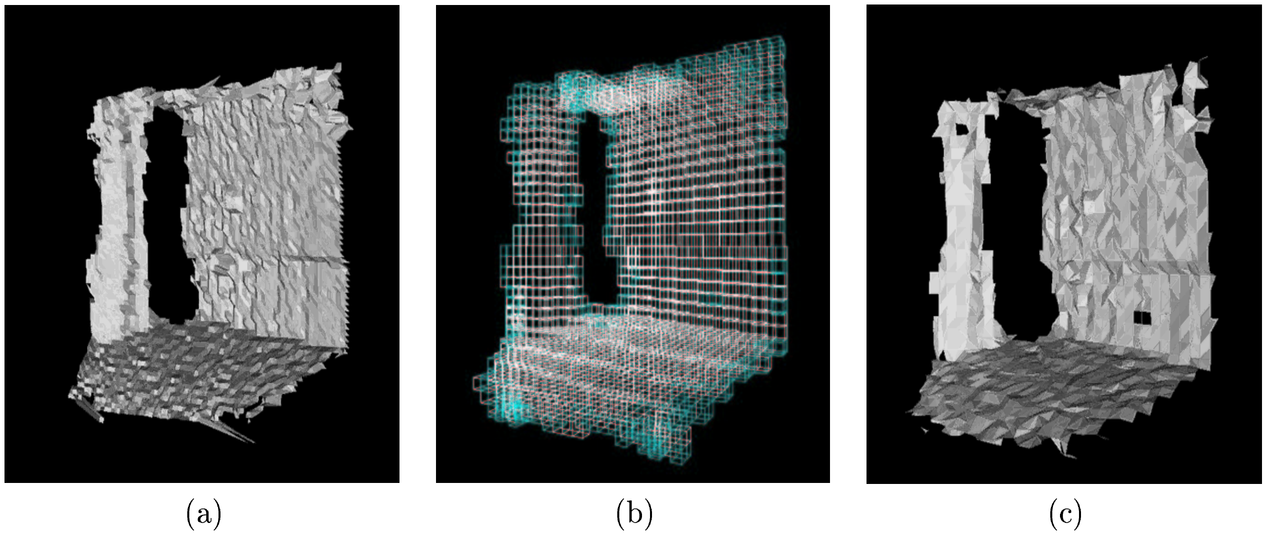


Figure 7: A single frame mesh (a); the corresponding set of active cells (b); the mesh at lower resolution extracted via MC from the active cells (c)

6.1 The volumetric approach

The mesh reconstruction method of Curless and Levoy is based on a virtual grid of *cubic cells* that decomposes the 3D space containing the object to be reconstructed. *Nodes* of the grid coincide with vertices of the cubic cells. A signed distance function is computed at all nodes of the grid that lie close to the surface reconstructed from each single frame (registered in a common coordinate system), which measures the distance of points in space from the surface itself. Contributions from different frames to the same node are combined to obtain a global estimate of the distance function. Finally, all active cubic cells are traversed, and a fragment of mesh is obtained inside each cube as an isosurface of value zero of the distance field, via the well known Marching Cubes (MC) algorithm [20]. See Figure 7 for an illustration of the method on a single frame.

6.2 On-line mosaicing

Following the same volumetric approach, we amortize computation of the distance field, and update of active cells through the sequence of frames that come on-line. The virtual grid is initially empty and it is created while frames are processed. Note that the grid is virtually infinite, while only active cells are really created and maintained in memory. The size of cells is a parameter that allows us to trade-off speed and accuracy and also to perform data reduction if large objects must be reconstructed without saturating physical and graphical memory.

The architecture of this phase is depicted in the upper part of Figure 8. We maintain a dual data structure via two *maps* (map is a mechanism provided by standard template libraries, which is similar to hash tables). One map, called the *Node Map*, contains grid nodes, while the other map, called the *Cell Map* contains grid cells. Each node is addressed by a key consisting of its integer coordinates in the virtual grid, while each cell is addressed by the key of its corner having the smallest coordinates.

At each new frame, we update the distance field only at those nodes that are corners of cubic cells traversed by the single frame mesh, and we save such updates in the node map. In the same time, we generate a list of (keys of) all cells that are incident at updated nodes. Such cells have changed their configuration and must be updated as a consequence. We scan such list and, for each cell, we either retrieve it from the cell map, or, if it is not present, we create it. In both cases, we update the MC configuration of the cell.

We maintain also a *Segmented Mosaic* that is a collection of (keys of) all cells that contribute to the mosaic as a segmented list, where: there are as many segments as the processed frames; and each segment contains all cells that were created when processing

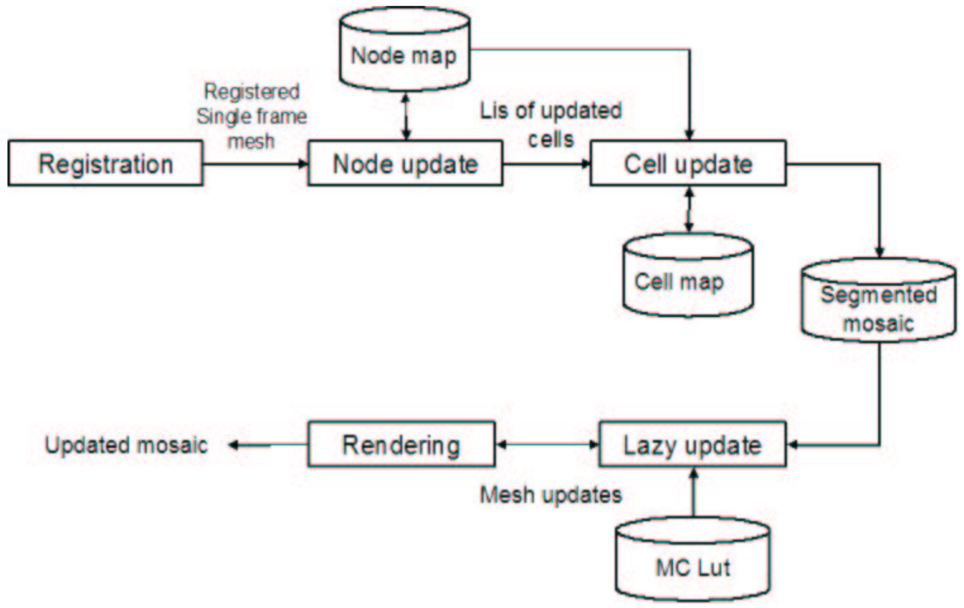


Figure 8: Architecture of the mosaic (upper) and rendering (lower) phases.

the corresponding frame. At each frame, a new segment is generated, while the fact that cells of other segments have been updated is recorded in a segment hit counter. This will be used during the phase of rendering, which is depicted in the lower part of Figure 8 and will be described in the next section. More details of the various phases of the on-line mosaicing algorithm are given in the following.

We have two alternative methods for updating the distance field at nodes:

- A faster method considers only the vertices of the single frame mesh, and, for each vertex, updates the distance field at the corners of the cell containing that vertex. The key to this cell is obtained by arithmetic computation and the cell is either created or retrieved from the map of cells. Its corner nodes are in turn either created or retrieved from the map of nodes. We compute the signed distance between each node and the tangent plane through the vertex. Contribution of different vertices (possibly from different frames) to the same node are combined as a weighted average, where weight of a given contribution is directly proportional to the reliability of measure, and inversely proportional to the square distance computed. Let v be the current vertex and w_v its weight (weight comes from the sensor and represents the reliability of measure v), let d_v be the signed distance of a given node n from v, let d_n the current distance field at n, and w_n be the corresponding weight. We compute $W = w_v/(d_v^2 + 1)$ and we update n as follows:

$$d_n = (d_n w_n + d_v W)/(w_n + W)$$

$$w_n = (w_n^2 + W^2)/(w_n + W).$$

In this way, the closer a vertex v is to a node n, and the more reliable the measure of v is, the more important the contribution of v to the distance field at n will be.

This method works well as long as the resolution of the single frame mesh is higher than that of the volumetric grid. For a high resolution grid, however, the result tends to be fragmented because many cells traversed by the surface are never classified, since they do not contain any vertex.

- A slightly slower but more accurate method is based on random sampling of points inside faces of the single frame mesh. We sample a number n_f of points inside each triangular face f of the single frame mesh, where n_f is proportional to the area of f and to the resolution of the grid. If l is the edge length of the volumetric grid, and Δ_f is the area of f, we set $n_f = k\Delta_f/l^2$, where k is a coefficient corresponding to the average number of points sampled for each cell traversed by f. The actual number of points sampled for each such cell is proportional to the area of the intersection between f and the cell itself. Too small values of k tend to generate holes where a cell has only small intersection with a face (because no points are sampled inside that cell). On the on the other hand, very large values of k slow down processing. We experimentally found that values of k beyond 10 do not give visible benefits to the result on the data sets we have processed, so we set $k = 10$. For each point p sampled from a face f, we compute the baricentric coordinates of p in f and we use them to estimate the surface normal at p by linearly interpolating the normals at the vertices of f. Then we proceed as in the previous case to classify the corners of cell containing p.

All grid nodes that are updated while processing a given frame may contribute to update the mosaic. Therefore, we save a list UC of (keys to) all cells that have such nodes as corners. The pseudo-codes of the algorithms corresponding to the two methods described above are shown in Figure 9.

After all nodes have been updated, we scan list UC and re-compute the MC configuration of each cell in it. For each cell, we classify its eight nodes according to the sign of distance field at them. This gives us a code of 8 bits that uniquely identifies the connectivity of the fragment of mesh traversing the cell. Then, we compute the value of intersections between the edges of the cell and the isosurface of value zero for the distance field. These intersections occur at all edges having endpoints where the distance field has different signs, and their locations along edges are found by weighted linear interpolation, which uses both the values of the distance field at the vertices of the cell and their weights. The collection of such intersections gives the position of vertices of the fragment of mesh traversing the cell. The code of 8 bits plus the (at most 12) intersections constitute the

```
ClassifyNodesPerVertex (Mesh $M$) // $M$ is the registered single frame mesh
begin
     // update distance field at nodes of each cell containing a vertex of $M$
     for every vertex $v$ of $M$
          compute the key of cell $c$ containing $v$;
          for every node $n$ of $c$
               compute signed distance $d_v$; // see text
               compute weight $W$; // see text
               if $n$ does not exist in the Node Map then
                    create and initialize $n$ in the Node Map;
               update $d_n$ and $w_n$ in the Node Map; // see text
          // the cell containing $v$ plus its 26 neighbors need to be updated
          for every cell $c$ incident at some updated node
               put the key of $c$ into a list $UC$ of updated cells
end;
ClassifyNodesPerCell (Mesh $M$) // $M$ is the registered single frame mesh
begin
     // update distance field at nodes of each cell intersecting $M$
     for every face $f$ of $M$
          compute $n_f$; // see text
          for $i = 1$ to $n_f$
               sample a point $p$ in $f$;
               compute normal at $p$; // see text
               compute signed distance $d_p$; // see text
               compute weight $W$; // see text
               if $n$ does not exist in the Node Map then
                    create and initialize $n$ in the Node Map;
               update $d_n$ and $w_n$ in the Node Map; // see text
               // the cell containing $p$ plus its 26 neighbors need to be updated
               for every cell $c$ incident at some updated node
                    put the key of $c$ into a list $UC$ of updated cells
end;
```

Figure 9: The two alternative algorithms for updating the distance field at nodes of the grid.

MC configuration of a cell [20].

The pseudo-code of procedure for updating cells is given in Figure 10.

Note that the *Cell Map* only collects all active cells, with their MC configuration, while the triangle mesh is generated via MC only when necessary, in order to send it to the rendering system. This will be explained in the next section.

```
UpdateCells (List $UC$) // $UC$ is the list of keys to cells that must be updated
begin
    // update MC configurations at all cells in the list
    for every key $k$ of $UC$
        retrieve cell $c$ corresponding to $k$ from Cell Map;
        if $c$ does not exist then
            create $c$;
        retrieve vertices of $c$ from Node Map;
        recompute MC configuration of $c$;
        insert/update $c$ into Cell Map;
        if $c$ was just created then
            add $k$ to the segment corresponding to current frame in Segmented Mosaic
        else
            increment the hit counter of the segment that created $c$ in Segmented Mosaic;
end;
LazyUpdate (int $lt$) // $lt$ is a threshold to decide when a segment of mosaic must be regenerated
begin
    // scan the segmented mosaic and segments of mesh
    for every segment $s$ of the Segmented Mosaic
        if $s$ is the last segment or $s.HitCounter > lt$ then
            for every key $k$ in $s$
                retrieve cell $c$ corresponding to $k$ from Cell Map;
                regenerate mesh $M_c$ internal to $c$;
                send $M_c$ to rendering engine;
end;
```

Figure 10: The algorithms for updating cells of the mosaic and for lazy update of meshes at rendering engine.

7 Rendering

In order to meet real time requirements, our hardware architecture is based on a network of personal computers. A typical configuration includes one PC to drive the acoustic sensor and collect data, one PC to perform data processing, and one PC to support the user interface for the pilot. The latter computer (called the *viewer*) must render the mosaic, which was built from data processing on a different computer. More complex configurations (e.g., those concerning activities by one or more surveyors) may require rendering to several viewers, under different user interfaces.

So, the hardware architecture must be necessarily distributed, and the computer that makes data processing is usually not the same that renders the results. As a consequence, 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 into different portionsand transmit such portions in batches by following a *lazy update* approach.

Active cells after processing a given frame can be divided in three groups:

- Group N of new cells, i.e., cells that did not exist before.
- Group U of updated cells, i.e., cells that existed before but were modified by contributions of the current frame.
- Group S of stable cells, i.e., cells that existed before and were not modified by current frame.

The output mesh should be generated by adding faces from cells of group N, substituting faces from cells of group U and leaving faces corresponding to group S untouched. The *Segmented Mosaic* data structure described in the previous section is used to this purpose. The output mesh in the rendering system is also segmented into

different display lists, corresponding to the list of cells in the *Segmented Mosaic.*

Cells of group N form a new list, corresponding to current frame. This list is scanned, a display list of

triangles is generated by using MC lookup tables [20] and it is delivered to the rendering engine. Cells of group U instead belong to lists corresponding to previous frames. We maintain a hit counter for each such list in the *Segmented Mosaic*, and we increment such counter every time a cell in a given list is updated. When such a counter exceeds a given threshold lt, we regenerate the corresponding display list and we substitute it to the old one in the rendering system. If $lt = 0$, all lists containing cells of U are regenerated at each frame; otherwise, some updates are deferred to later frames. The pseudo-code of the procedure for lazy update of meshes is given in Figure 10.

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

8 Experiments

In this section we present some experiments. We tested the proposed algorithms on two sequences of real images collected during trials of the project. The aperture of the sensor was about 90×90 degrees. Each frame has been registered with respect to the previous one. The transformations that bring each view on to the mosaic are computed just combining the sequential pairwise registration matrices.

In experiment 1 (60 frames) a portion of a quayside in La Spezia (Italy) has been reconstructed. The scene is quite complex and it is composed of a big vertical pillar with several small columns on its sides. Figure 11 shows the pilot interface while the mosaic is growing. The ROV (which is depicted on the figure) is rotating from right to left. New interesting parts of the quayside appear while the ROV is moving. In early frames the scene is little defined, while, at the end, the whole structure is better recognizable.

Figure 12 shows two views of the reconstructed scenarios. The distance between the two farthest columns is approximately 12m, that is a very wide field of view for underwater environments. Each of the pillars is well defined and, furthermore, the pilot can benefit from the advantages of the 3D representation. For example the top view highlights the

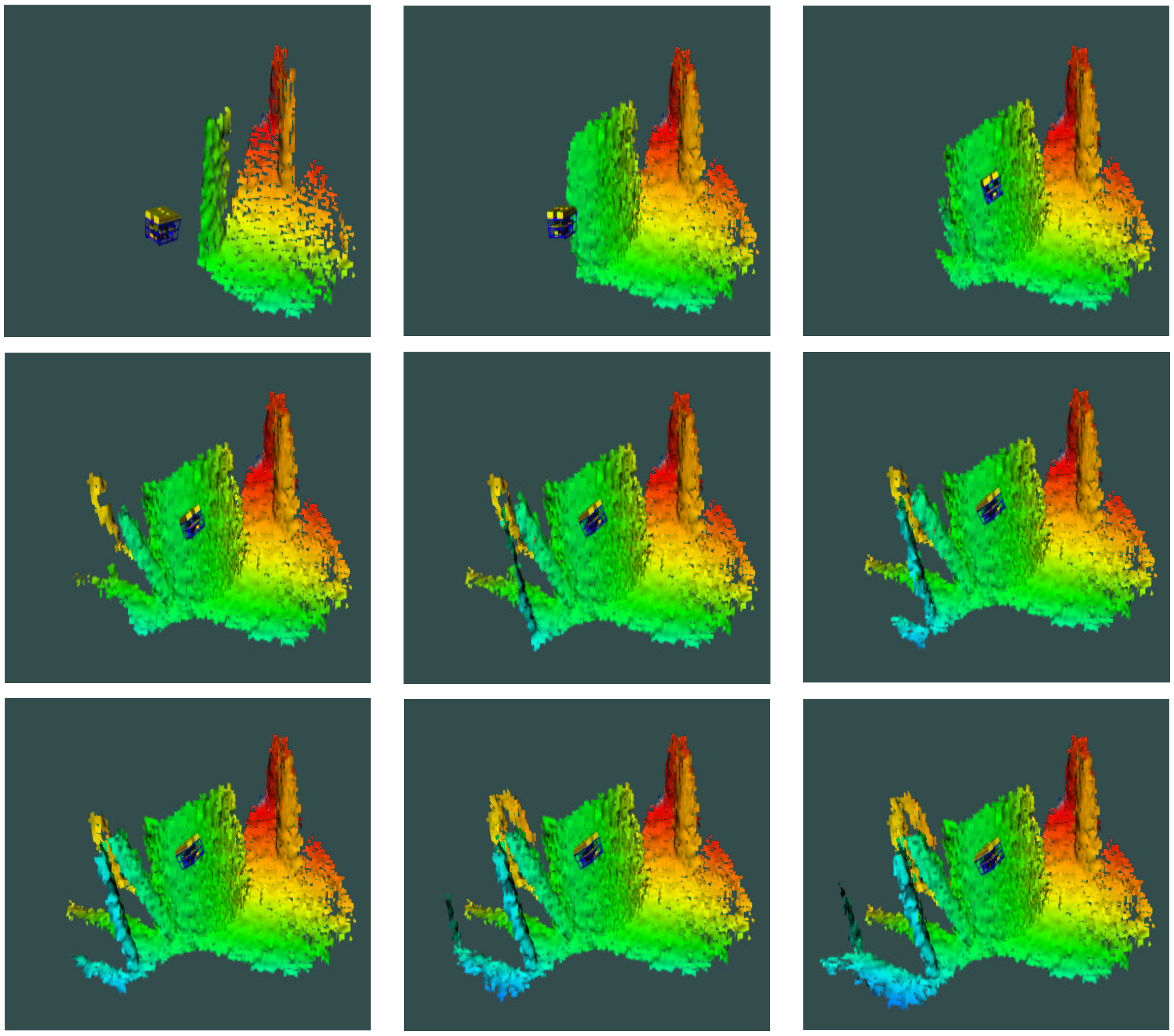
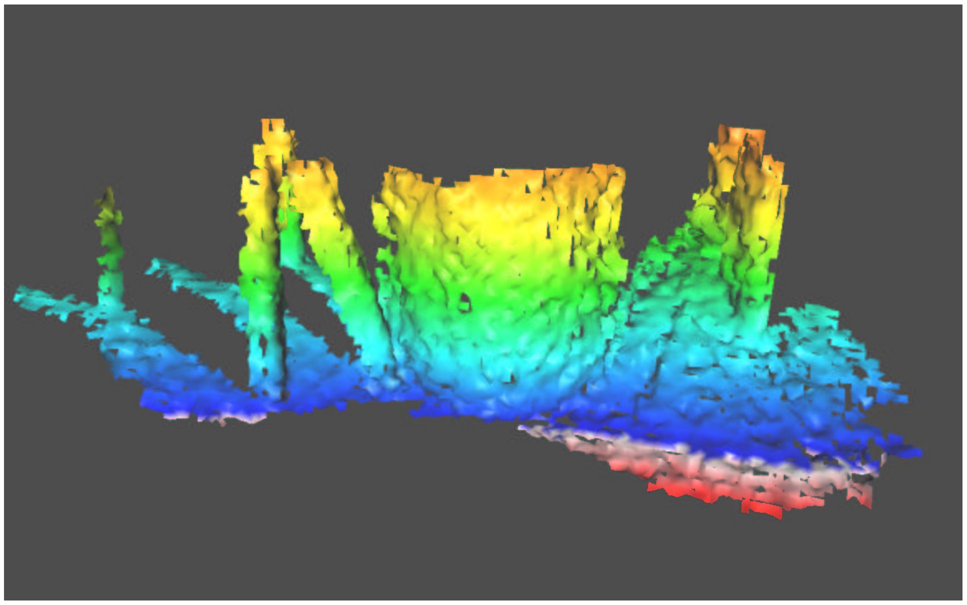


Figure 11: Selected frames of experiment 1 while mosaic is growing

curvature of the pillars and their distance from each other (12.b).

In experiment 2 a longer sequence has been analyzed (460 frames). A whole quayside in Bergen (Norway) has been reconstructed. The scene is composed of the seabed and 10 pillars. Figure 13 shows the final mosaic.

Even if the images are quite noisy the scenario is well recognizable. Figure 14 shows a zooming on a pair of pillars by highlighting interesting details. The seabed is a rough and


(a)

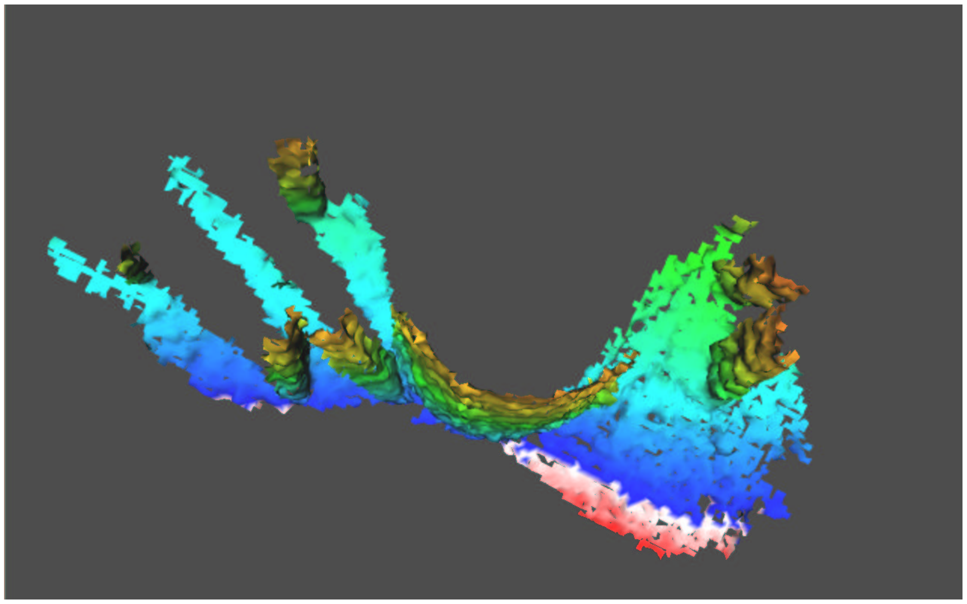
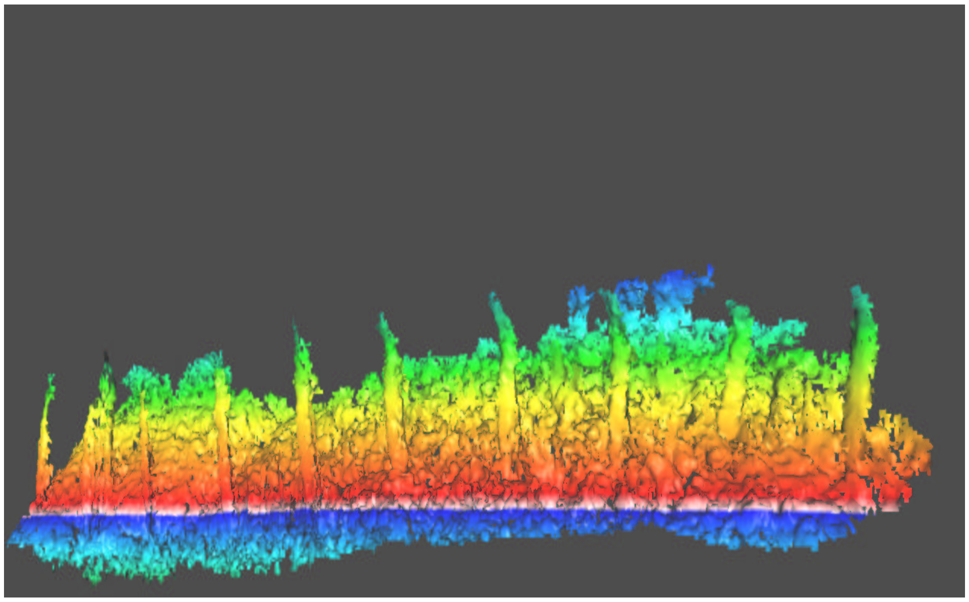

(b)

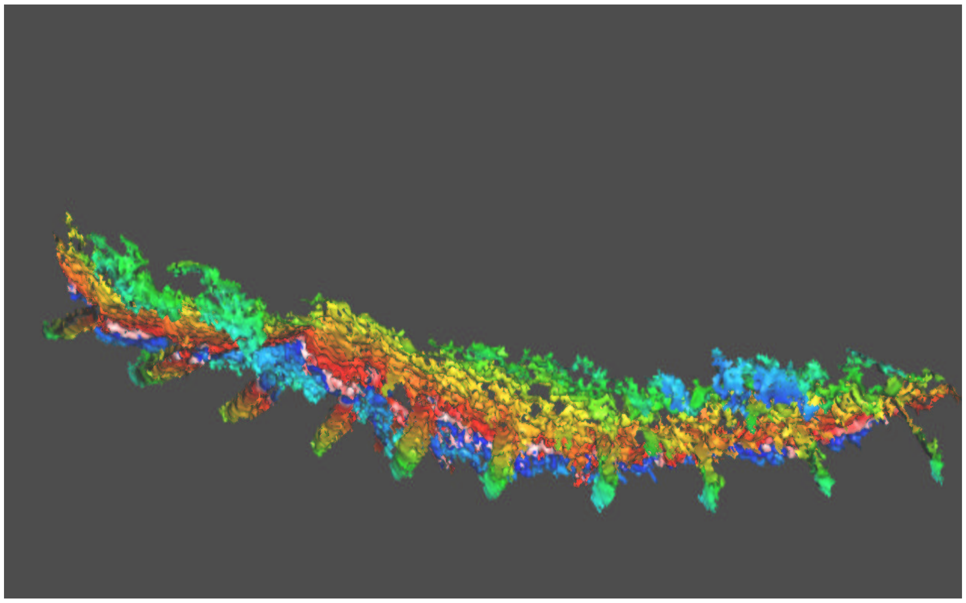
Figure 12: Final reconstruction of experiment 1. Front view (a) and top view (b)

steep slope and some rocks are visible (see for example near the base of pillars). Furthermore all the pillars are preserved. The distance between two pillars is about 7 m, thus the reconstructed area is approximately 70 m wide.

It is worth noting that pillars in the reconstruction of Figure 13.b are not lying along



(a)



(b)

Figure 13: Final reconstruction of experiment 2. Front view (a) and top view (b)

a straight line, as they should. The drift that makes reconstruction bend slightly is due to accumulation of registration errors, which is fairly typical of systems based on pairwise registration. This effect could be reduced by integrating in the registration phase information on the ROV position obtained from motion sensors devices, when available.

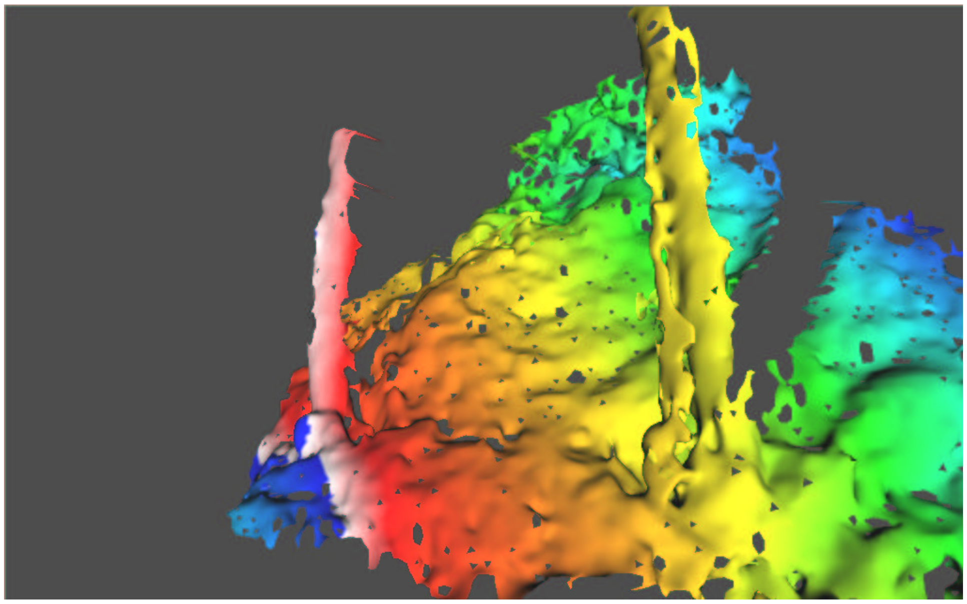


Figure 14: Zooming on a pair of pillars and the seabed

Table 1 shows the registration performance in terms of accuracy and speed of each sequence. The speed is the average time per pairs of views. In order to highlight the improvement of the proposed approach, a comparison with the classic ICP algorithm has been carried out. The speed obtained with our method, based on the reverse calibration paradigm, is approximately 20 time faster.

	Classic ICP		Proposed ICP	
Sequence	Time *(sec.)*	Accuracy *(cm.)*	Time *(sec.)*	Acuracy *(cm.)*
Experiment 1	2.195737	10.611615	0.117299	10.820742
Experiment 2	1.695990	16.679747	0.130076	16.830383

Table 1: Performance of registration. The accuracy is given by the residual of the last ICP iteration

The accuracy is given by computing the mean distance between the corresponding points of the last ICP iteration. It is worth noting that accuracy is comparable with the accuracy obtained by using the classic ICP algorithm. Moreover, accuracy is reasonable since error is just a little higher than the image resolution (i.e., the registration error is mainly affected by the noise of the acquired data).

For what concern the integration with the geometric fusion and rendering phases, we have analyzed four different rasterization steps for both sequences, by using grids with edge lengths of 10, 20, 30, and 40 cm, respectively (Table 2). As expected, most of the processing time is spent in the geometric fusion phase, while the rendering phase is very fast. In particular, from step 10 to step 20 and from step 20 to step 30 the speed is quite increased, while from step 30 to step 40 the improvement is rather small. On the other hand, the mosaic resolution becomes more rough as the grid step becomes larger. Figure 15 shows some wire-frame representations of the same mosaic detail obtained with different rasterization steps. The value of `step` = 20 provides a resolution fine enough for the current sequence and it performs sufficiently fast on the hardware we used for our experiments. The mosaics shown in Figure 11, 12, 13 and 14 have been generated by setting a raster step 20.

	Experiment 1		Experiment 2	
Raster Step	Geometric fusion *(sec.)*	Rendering *(sec.)*	Geometric fusion *(sec.)*	Rendering *(sec.)*
10	0.21102	0.04768	0.19779	0.06033
20	0.12494	0.00903	0.10210	0.01036
30	0.07076	0.00470	0.05952	0.00502
40	0.06702	0.00418	0.04203	0.00385

Table 2: Spatial resolution and relative timing from the geometric fusion and rendering procedures

Note that spatial resolution of data depends on the input data range, as well as on the working conditions of the sensor that admits using alternatively three different frequencies of the acoustic signal, which correspond to three different fields and ranges of view. So the best choice of the rasterization step is indeed a trade-off between speed of processing and quality of the result and may vary from case to case.

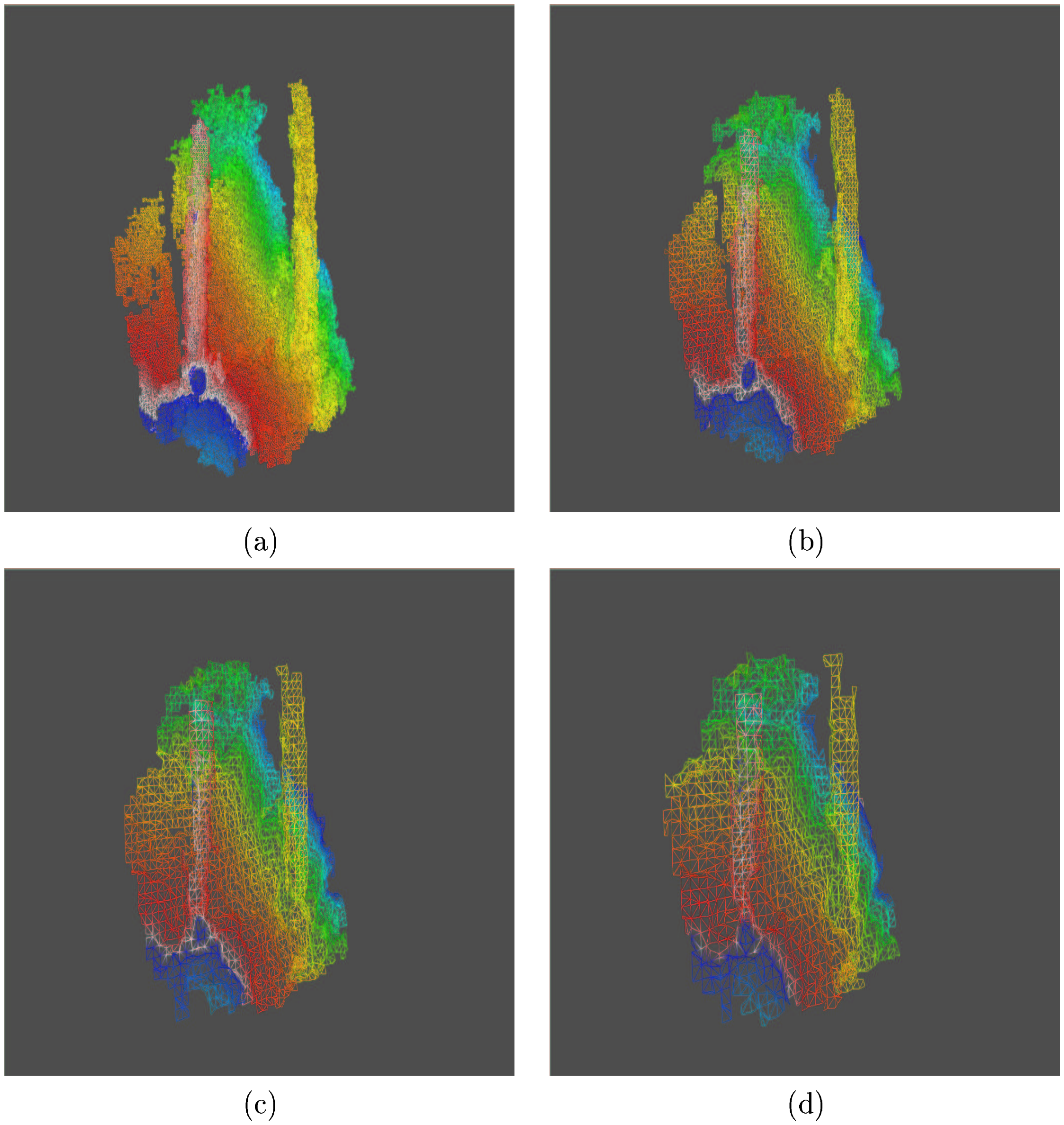


Figure 15: Wireframe representations with different raster steps: 10 (a), 20 (b), 30 (c) and 40 (d)

Finally, in table 3 the timing of the main phases are summarized (the time is the average per pairs of views). The mosaicing column refers to the sum of the geometric fusion and the rendering phases for the raster step 20. Timings have been computed by software profiling, on a $P4$ 2.8Ghz, with 2Gb Ram. Overall, the speed of the whole data processing pipeline supports about 4 frames per second that is compatible with the speed of the

Experiments	Single Frame Reconstraction *(sec.)*	Registration *(sec.)*	Mosaicing *(sec.)*	Total *(sec.)*
exp 1	0.02662	0.11729	0.13397	0.27788
exp 2	0.01742	0.13007	0.11246	0.25995

Table 3: Summary of the total timing for the analyzed sequences. The mosaicing column refers to the sum of the geometric fusion and the rendering phases for the raster step 20.

acquisition device (i.e., the acoustic camera). Note that most of the involved algorithms could actually run in parallel in the context of a pipeline architecture. Therefore, a dramatic improvement of speed could be obtained by splitting the processing on different machines.

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 main contribution consists in proposing an automatic 3D modeling approach for a very hostile environment such as the underwater scenarios.

The challenging goal is to provide a 3D scene model on-line 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 proposed a variation of the reverse calibration approach in order to improve the speed of the corresponding point search procedure and to increase the robustness in handling noisy data. For the geometric fusion phase, we developed an on-line reconstruction method on the basis of the volumetric approach proposed by Curless and Levoy in [11], which has been modified by speeding up the computation of the distance field and the definition of the updated cells in order to manipulate effectively a growing 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 together and too frequently.

Our processing pipeline meets the rate at which images are captured by the acoustic sensor. Furthermore, the user can interact with the graphical interface on-line, while the mosaic is growing, thus improving the perception of the scene during the operation.

We are currently integrating a motion tracking technique based on Kalman filter, which exploits feedback from motion sensors on board the ROV in order to improve convergence of registration and reduce the accumulation of registration errors. Moreover, we are planning to use more the system in live experiments in order to get also some user feedback for future improvements. We are also working on how to combine the display of sensed data together with a priori knowledge of the environment, such as underwater charts and maps.

Acknowledgments

This work was 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 Ministry of Reseach and Education under project LIMA3D (Low Cost 3D imaging and modelling automatic system). The implementation of the ICP is partially due to Linmi Tao.

References

[1] J. Batlle, E. Mouaddib, and J. Salvi. Recent pogress in coded structured light as a technique to solve the correspondence problem: Survey. *Pattern Recognition*, 31(7), 1998.

[2] F. Bernardini and H. E. Rushmeier. 3d model acquisition pipeline. *Computer Graphics forum*, 21(2):149–172, 2002.

[3] P. Besl and N. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[4] Paul J. Besl. Active, optical imaging sensors. *Machine Vision and Applications*, pages 127–152, 1988.

[5] Paul J. Besl. Range imaging sensors. Technical report, General Motors Research Publications, March 1988.

[6] G. Blais and M. D. Levine. Registering multiview range data to create 3-D computer objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):820–824, 1995.

[7] U. Castellani, A. Fusiello, and V. Murino. Registration of multiple acoustic range views for underwater scene reconstruction. *Computer Vision and Image Understanding*, 87(3):78–89, July 2002.

[8] U. Castellani, A. Fusiello, and V. Murino. 3d registration by fast icp. Technical report, Dipartimento di Informatica, Verona University, December 2004. http://arrov.disi.unige.it/publications.html

[9] U. Castellani, A. Fusiello, V. Murino, L. Papaleo, M. Pittore, and E. Puppo. Object reconstruction and 3d mosaicing. Technical report, Project ARROV, July 2004. http://arrov.disi.unige.it/publications.html

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

[11] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *Computer Graphics*, 30(Annual Conference Series):303–312, 1996.

[12] A. Fusiello and V. Murino. Augmented reality by optical and acoustic sensory integration. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):625–636, 2003.

[13] M. Greenspan and M. Yurick. Approximate kd tree search for efficient icp. In *In Proceedings of 3-D Digital Imaging and Modeling (3DIM 2003)*, pages 442– 448, 2003.

[14] F.R. Hampel, P.J. Rousseeuw, E.M. Ronchetti, and W.A. Stahel. *Robust Statistics: the Approach Based on Influence Functions.* Wiley Series in probability and mathematical statistics. John Wiley & Sons, 1986.

[15] R. K. Hansen and P. A. Andersen. A 3-D underwater acoustic camera - properties and applications. In P.Tortoli and L.Masotti, editors, *Acoustical Imaging*, pages 607–611. Plenum Press, 1996.

[16] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision.* Cambridge University Press, 2000.

[17] B. Kamgar-Parsi, L.J. Rosenblum, and E.O. Belcher. Underwater imaging with a moving acoustic lens. *IEEE Transactions on Image Processing*, 7(1):91 – 99, 1998.

[18] C. A. Kapoutsis, C.P. Vavoulidis, and I. Pitas. Morphological iterative closest point algorithm. *IEEE Transaction on Image Processing*, 8(11), 1999.

[19] T.P. Koninckx, A. Griesser, and L. Van Gool. Real-time range scanning of deformable surfaces by adaptively coded structured light. In *In Proceedings of 3-D Digital Imaging and Modeling (3DIM 2003)*, pages 293–300, 2003.

[20] W. Lorensen and H. Cline. Marching cube: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–170, 1987.

[21] T. Masuda and N. Yokoya. A robust method for registration and segmentation of multiple range images. *Computer Vision and Image Understanding*, 61(3):295–307, May 1995.

[22] V. Murino and A. Fusiello. Augmented reality by integrating multiple sensory modalities for underwater scene understanding. In *In Confluence of Computer Vision and Computer Graphics*, pages 331–349, 2000.

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

[24] V. Murino, A. Trucco, and C.S. Regazzoni. 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.

[25] S. Negahdaripour and H. Madjidi. Stereovision imaging on submersible platforms for 3-d mapping of benthic habitats and sea-floor structures. *IEEE Transactions on Oceanic Engineering*, 28(4):625 – 650, 2003.

[26] S. Negahdaripour, C. H. Yu, and A. Shokrollahi. Recovering shape and motion from undersea images. *IEEE Transactions on Oceanic Engineering*, 15(4):189–198, 1990.

[27] L. Papaleo and E. Puppo. On-line data fusion for building 3d models from acoustical range images. Technical report, Dipartimento di Informatica e Scienze dell'Informazione, University of Genova, December 2004. http://arrov.disi.unige.it/publications.html

[28] S.Y. Park and M. Subbarao. A fast point-to-tangent plane technique for multi-view registration. In *In Proceedings of 3-D Digital Imaging and Modeling (3DIM 2003)*, pages 276–283, 2003.

[29] C. Rocchini, P. Cignoni, F. Ganovelli, C. Montani, P. Pingi, and R. Scopigno. Marching Intersections: an efficient resampling algorithm for surface management. In *Int. Conf. on Shape Modeling and Applications*, pages 296–305, Genova, Italy, 2001. IEEE Comp. Society.

[30] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3d model acquisition. In *In Proceedings of Siggraph (SIGGRAPH 2002)*, pages 438–446, 2002.

[31] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *IEEE Int. Conf. on 3-D Imaging and Modeling, 3DIM '01*, Quebec City (Canada), 2001.

[32] R. Sagawa, T. Masuda, and K. Ikeuchi. Effective nearest neighbor search for aligning and merging range images. In *In Proceedings of 3-D Digital Imaging and Modeling (3DIM 2003)*, pages 79– 86, 2003.

[33] H. Singh, Xiaoou Tang, E. Trucco, and D. Lane. Guest editorial special issue on underwater image and video processing. *IEEE Transactions on Oceanic Engineering*, 28(4):569 – 776, 2003.

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

[35] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 311–318. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[36] R. J. Urik. *Principles of Underwater Sound.* McGraw-Hill, 1983.

[37] Hongbin Zha, Hideo Saito, Andrea Fusiello, and Vittorio Murino. Special issue on 3-d image analysis and modeling. *IEEE Transactions on Systems, Man and Cybernetics*, 33(4):550–553, 2003.

[38] Z. Zhang. Iterative point matching of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, 1994.