Computer Vision and Image Understanding **79,** 92–122 (2000)
doi:10.1006/cviu.2000.0846, available online at http://www.idealibrary.com on IDEAL®

Feature Tracking in Video and Sonar Subsea Sequences with Applications

E. Trucco, Y. R. Petillot, I. Tena Ruiz, K. Plakas, and D. M. Lane

Ocean Systems Laboratory, Department of Computing and Electrical Engineering, Heriot–Watt University, Riccarton, Edinburgh EH14 4AS, United Kingdom

Received March 26, 1999; accepted February 4, 2000

This paper deals with automatic target tracking in video and sonar subsea sequences, an essential capability for automating tasks currently performed by remotely operated vehicles under pilot control. We describe two trackers, one for video sequences, the other for sector scan sonar sequences. No assumptions are made about the images, scene, or motion observed. To illustrate applications, we report results of our systems for 3-D structure reconstruction and panoramic mosaic building from video sequences and describe in some detail our path planning and obstacle avoidance system using sonar sequences. © 2000 Academic Press

Key Words: subsea vision; video tracking; sonar tracking.

1. INTRODUCTION

1.1. The Need for Reliable Tracking Systems

The recent renaissance of interest in autonomous underwater vehicles (AUVs) for subsea applications makes the development of reliable navigation and control technologies crucial. Image sequences, both sonar and video, are rich sources of data for many tasks; for instance, images can supply visual servoing loops with positional and motion measurements or provide information about the seafloor profile in surveying mission or about the structure and state of humanmade installations in maintenance missions. The introduction of autonomous, reliable visual servoing modules would also take away the burden of constant manual control from pilots of remotely operated vehicles (ROVs), allowing them to concentrate on the target task.

1.2. Video and Sonar Tracking as an Image Processing Problem

Tracking is an essential requirement for most subsea tasks based on image sequences, including following pipes, keeping station in front of targets in predefined orientations, and building panoramic mosaics of benthic objects, industrial installations, or the seafloor. The software implementing any such task relies ultimately on tracking the motion of some image elements, or targets.

In this paper, *target tracking means estimating the motion of one or more image regions through the frames of a sequence.* Notice that the problem is cast from the viewpoint of computer vision for both video and sonar sequences, traditionally two separate domains in subsea applications. Of course, the exact definition of "target" depends, among other things, on the image type (e.g., video or sonar) and the task for which tracking is performed (e.g., servoing, image stabilization).

1.3. Tracking as a Computer Vision Technique

Feature tracking is a *pedigree* problem in computer vision [39, 44, 45, 54] and its literature is vast. Not so for sonar tracking (as an image processing problem), mainly due to limitations of sensor technology which have been overcome only recently. These are discussed in the next section.

Tracking has been approached in two basic image processing frameworks. One is to track *image features*, that is, image regions with special properties making them clearly identifiable and efficiently detectable. Classic examples of features in computer vision are corners [43], lines [39], and deformable contours [5]. Such features are detected automatically in each frame and tracked through a sequence for as long as possible. The resulting motion field is *sparse*: all motion information computed refers only to the feature regions. Finding the same feature in subsequent frames is a problem similar to stereo matching [10, 12, 22, 31, 54]. Sequence analysis can exploit also temporal continuity; i.e., the features' position and appearance change slowly through most of a sequence. Kalman filtering has proven a popular technique for this purpose [30, 48, 40].

The alternative framework is *optical flow methods*, a class of differential techniques estimating image motion at each pixel [2, 6, 34, 42, 49, 50]. The advantage is that a *dense* motion field is produced and, in principle, can be used for segmentation, tracking, and 3-D motion analysis. In practice, the main disadvantage with underwater sequences seems to be the differential nature of the methods, which requires a high frame rate and negligible changes between consecutive frames. This is clearly not the case with sonar sequences, and only very good water conditions seem to allow the computation of usable optical flows from video sequences. Moreover, optical flow methods can be computationally demanding. For these reasons, we have turned to feature-based techniques, which allow the use of stereo matching techniques to cope with finite and large disparities, in combination with Kalman filtering to take advantage of temporal information.

The size of the technical literature on image-based video tracking is not matched by that of image-based sonar tracking, mostly because of sensing limitations. Until recently, most subsea obstacle avoidance systems have used low-resolution or low-frame-rate sonar, yielding inaccurate estimates of obstacle position and motion. These systems were suitable for *reactive obstacle avoidance* (or *reflex behavior*), but not for tracking or path planning in real, changing environments.

Much work was done on segmenting side-scan sonar images, but not so much on forward-looking sonar images, and then again on still images [33]. Separating static from moving image regions using FFT techniques is one of the few examples of forward-looking sonar sequence analysis [7, 27]. But the noisy nature of forward-looking sonar images makes it very difficult to compute meaningful segmentations from a single return.

The advent of multibeam sonars, yielding high resolution and good frame rates, has made it feasible to exploit temporal features in sonar image segmentation and, in general, has opened the door to a whole new range of applications [7, 21, 26, 51, 52] including computing detailed descriptions of time-changing environments [9, 36].

1.4. A Brief Overview of Subsea Sonars

A brief description of the available types of sonar sensors and why forward-looking is the best option for path planning seems now in order.

Sonar (sound navigation and ranging) is an acoustic sensor used for sensing the environment. Sonar sensors can be divided into two main categories, *passive* and *active*. The passive sonar is a very sophisticated apparatus that senses acoustic energy in the environment. Passive sonars are mainly used in military applications, *stealth* being a major constraint for this type of operation. Active sonars are used in a broader range of applications such as charting or surveying for the offshore oil industry, and we shall concentrate on them. The active sonar emits pulses of acoustic energy, which will be reflected back to the sonar as they collide with surfaces within the sonar's range.

In water, acoustic sensors have proved by far the more popular choice. Electromagnetic waves attenuate far more rapidly and their working range is greatly reduced. Although cameras and lasers are extensively used for close-range inspection, their working range is typically well below 5 m and their performance is seriously affected by variable factors like water turbidity and marine snow.

For obstacle avoidance and path planning, one of the main applications for our work, the vehicle must be protected from the risk of collisions, and obstacles detected at safe range. The use of short-range sensors like cameras is therefore limited, whereas sonars offer ranges up to 200 m and are clearly the best option.

Why is the *forward-looking multibeam sonar* currently the best sensor for sonar-based tracking and allied applications? To answer, we must review briefly the most common types of active sonars available.

1.4.1. Echo Sounder Sonar Transponder

This is the most common and widely available commercial sonar. It emits a pulse at a given frequency; the pulse is reflected back from a surface to the receiver unit of the sonar. The distance between surface and sonar can be estimated by measuring the time interval between emission and reception of the pulse. Normally, these sonars are used in boats or UVs to estimate the vehicle's altitude from the seabed.

1.4.2. Side-Scan Sonar

This acoustic imaging device is towed by a vessel, typically to provide wide-area images of the seabed for surveying purposes. The images are generated in sequence; acoustic energy is transmitted to the side of the sonar, and the sound reflected back is used to create the image. For a detailed description the reader is referred to [3]. Side-scan sonars have been used as tools for mapping in many commercial applications. Slant-range and geometric corrections of images can be used to work out the position of a vehicle, but this requires too many constraints, including the assumption of a flat seabed.

1.4.3. Bathymetric Sonar

Also known as the *multibeam echo sounder*, this sonar is similar in operation to the echo sounder sonar transponder, except that it uses an array of hydrophones (as opposed to a single one), which allows for multiple beams. The result is a profile (cross-section) view of the seabed. This type of sonar is commonly used to build maps.

1.4.4. Forward-Looking Mechanically Scanned Sonar

This active sonar is used for a number of diverse applications, such as obstacle avoidance, midwater mine detection, and surveillance. The sonar consists of a single hydrophone which is mechanically scanned along the horizontal axis, sweeping a so-called *sector*. The returns are then used to create an image. Most systems provide the user with the option of choosing the size of the sector to scan and with some degree of control on the resolution. In most cases higher resolution results in a slower refresh rate. Typical ranges are up to 200 m, but this can be altered by the user. Again a longer range will also result in a slower refresh rate. Forward-looking sonars have been used for many years by ROV pilots for remotely controlled navigation, obstacle avoidance, and localization around known structures.

The major advantage of this type of sonar is its capability of detecting objects or seabed features, such as protruding rocks, at large distances. These can be observed in subsequent scans and tracked, providing the UV is moving at a slow speed. Another advantage is its price, much less than that of its more advanced cousins, described below.

1.4.5. Forward-Looking Multibeam Sonar

This type of sonar uses a fixed array of hydrophones, scanned electronically, which allows much faster updates of sectors (e.g., the Seabat 6012 can update a sector up to 30 times a second). In all other accounts, this sonar is similar to a mechanically scanned sonar. These sonars are more expensive than mechanical systems; nevertheless their popularity in the underwater community has been growing. Automatic methods for obstacle avoidance [21], motion estimation [9], and image recognition [11] using forward-looking multibeam sonar images have already appeared.

1.4.6. 3-D Acoustic Cameras

This type of sensor has been made available recently in the shape of the Echo-Scope 1600 [19]. This sensor allows 3-D data visualization as it uses a 2-D array of hydrophones. The cost of the sensor (the latest figure of £180,000 is almost three times the cost of a multibeam, forward-looking sonar), its weight (40 kg in air), and its size make it still prohibitive for several applications, especially those involving AUVs.

1.4.7 . Which Sonar for Tracking and Path Planning?

In conclusion, forward-looking sonars are the only sensors satisfying two main requirements for tracking and path planning: they can scan an area *in front of* the vehicle and at a *sufficiently high frame rate* to yield large overlaps between consecutive frames. This makes it possible to track targets using image processing techniques. Instead, side-scan and bathymetric sonars build maps of the seafloor *under* the vehicle as they are dragged along; in addition, their images need corrections requiring position information from independent sensors.

1.5. About This Paper: Contributions and Structure

The main contributions of this work are the development of

- a robust, real-time video tracker offering very good reliability with subsea imagery in a veriety of environments;
- a sonar tracker locating and tracking obstacles reliably in real, multibeam sonar sequences, thus taking full advantage of dense spatial and temporal information, fast enough to support real-time ROV or AUV operations, and successfully integrated in a fully working path planning and obstacle avoidance system.

This paper is organized in three parts. The first (Section 2) describes the theory behind our real-time video tracker and reports some experimental results. The second (Section 3) does the same for our sonar tracker. The third (Section 4) sketches three applications developed in our laboratory: reconstructing 3-D structures and building panoramic mosaics from video sequences, and path planning and obstacle avoidance using sonar sequences, the latter with some more details on its importance in AUV operations. A summary of our work and a brief account of our future plans close the paper.

2. ROBUST VIDEO TRACKING

2.1. Introduction

Robust tracking means detecting automatically unreliable matches, or *outliers*, over an image sequence (see [32] for a survey of robust methods in computer vision). Recent examples of robust video trackers include [47], which identifies tracking outliers while estimating the fundamental matrix, and [46], which adopts a RANSAC approach to eliminating outliers for estimating the trifocal tensor. Such approaches increase the computational cost of tracking significantly.

This section presents a robust tracker based on an efficient outlier rejection scheme suitable for subsea video sequences. The basis is the Shi–Tomasi–Kanade tracker [29, 41, 43], a feature tracker based on SSD matching and assuming affine frame-to-frame warping. The system tracks small image regions and classifies them as *good* (reliable) or *bad* (unreliable) according to the residual of the match between the associated image regions in the first and current frames. Our work extends the Shi–Tomasi–Kanade tracker in several ways: First, we have introduced an *automatic* scheme for rejecting bad features, thus making the process fully automatic. We employ a simple, efficient, model-free outlier rejection rule, called X84, and prove that its assumptions are satisfied in the feature tracking scenario. Second, we have developed a system running for indefinite lengths of time. Third, our tracker runs in real time on nondedicated hardware.[1]

2.2. How the Video Tracker Works

Consider an image sequence $I(\mathbf{x}, t)$, with $\mathbf{x} = [u, v]^\top$ the coordinates of an image point. If the frame rate is sufficiently high, we can assume that intensity values within small regions remain practically unchanged after displacement,

$$I(\mathbf{x}, t) = I(\delta(\mathbf{x}), t + \tau), \tag{1}$$

[1] Code for a basic version of our robust tracker is available at ftp://taras.dimi.uniud.it/pub/sources/rtrack.tar.gz.

where $\delta(\cdot)$ is the *motion field*, specifying the *warping* that is applied to image points. If we choose an *affine* motion field

$$\delta(\mathbf{x}) = D\mathbf{x} + \mathbf{d},$$

where

$$D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix}$$

is a deformation matrix and $\mathbf{d}$ is the translation of the image windows center, then a point $\mathbf{x}$ in the first image I moves to a point $A\mathbf{x} + \mathbf{d}$ in the second image J,

$$J(A\mathbf{x} + \mathbf{D}) = I(\mathbf{x}),$$

where $A = \mathbf{1} + \mathbf{d}$ and $\mathbf{1}$ is the 2×2 identity matrix.

We estimate the motion parameters, D and $\mathbf{d}$, by minimizing the residual

$$\epsilon = \sum_{\mathcal{W}} [I(A\mathbf{x} + \mathbf{d}, t + \tau) - I(\mathbf{x}, t)]^2. \tag{2}$$

By plugging the first-order Taylor expansion of $I(A\mathbf{x} + \mathbf{d}, t + \tau)$ into (2), and imposing that the derivatives with respect to D and $\mathbf{d}$ are zero, we obtain the linear system

$$T\mathbf{z} = \mathbf{a}, \tag{3}$$

in which $\mathbf{z} = [d_{11}\ \ d_{12}\ \ d_{21}\ \ d_{22}\ \ d_1\ \ d_2]^\mathsf{T}$ contains the unknown motion parameters, and

$$\mathbf{a} = -\tau \sum_{\mathcal{W}} I_t [uI_u\ \ uI_v\ \ vI_u\ \ vI_v\ \ I_u\ \ I_v]^\mathsf{T},$$

$$T = \sum_{\mathcal{W}} \begin{bmatrix} U & V \\ V^\mathsf{T} & G \end{bmatrix},$$

with

$$U = \begin{bmatrix} u^2 I_u^2 & u^2 I_u I_v & uv I_u^2 & uv I_u I_v \\ u^2 I_u I_v & u^2 I_v^2 & uv I_u I_v & uv I_v^2 \\ uv I_u^2 & uv I_u I_v & v^2 I_u^2 & v^2 I_u I_v \\ uv I_u I_v & uv I_v^2 & v^2 I_u I_v & v^2 I_v^2 \end{bmatrix},$$

$$V^\mathsf{T} = \begin{bmatrix} u I_u^2 & u I_u I_v & v^2 I_u^2 & v I_u I_v \\ u I_u I_v & u I_v^2 & v I_u I_v & v I_v^2 \end{bmatrix}.$$

Of course, the above equation is satisfied only approximately because of the approximations introduced (among other reasons). A better estimate of $\mathbf{z}$ can be determined from Eq. (3), using a Newton–Raphson iterative scheme.

The quality of such estimates depends critically on several factors, including the size of the feature window, the amount of texture in the image, and the amount of camera motion. One generally favors small windows as they give better localization and are less likely to straddle depth discontinuities; however, they make estimates of the deformation matrix less reliable. Under the assumption of a high frame rate, image motion and feature deformation between frames are minimal over limited time intervals. For these reasons, a *pure translational* model, that is,

$$\delta(\mathbf{x}) = \mathbf{x} + \mathbf{d},$$

is adequate as long as the tracker runs continuously and reinitializes the feature sets as soon as enough features are lost (Section 2.2.3).

The tracker's task now is to compute **d** for a number of selected points in each pair of successive frames in the image sequence.

In this translational case, the residual takes the form

$$\epsilon = \sum_{\mathcal{W}} [I(\mathbf{x} + \mathbf{d}, t + \tau) - I(\mathbf{x}, t)]^2. \tag{4}$$

By plugging the first-order Taylor expansion of $I(\mathbf{x} + \mathbf{d}, t + \tau)$ into (4) and imposing that the derivatives with respect to **d** are zero, we obtain the linear system

$$G\mathbf{d} = \mathbf{e}, \tag{5}$$

where

$$G = \sum_{\mathcal{W}} \begin{bmatrix} I_u^2 & I_u I_v \\ I_u I_v & I_v^2 \end{bmatrix}, \quad \mathbf{e} = -\tau \sum_{\mathcal{W}} I_t [I_u \;\; I_v]^{\mathsf{T}},$$

with $[I_u \;\; I_v] = \nabla I = [\partial I/\partial u \;\; \partial I/\partial v]$ and $I_t = \partial I/\partial t$. Given a pair of successive frames, the solution of (5), that is, $\hat{\mathbf{d}} = G^{-1}\mathbf{e}$, is used to predict a new (registered) frame. The procedure is iterated according to a Newton–Raphson scheme until the displacement estimates converge.

Our tracker incorporates a *normalized* SSD matcher for residual computation. This limits the effects of intensity changes between frames, extremely frequent in underwater sequences, by subtracting the average gray level (μ_J, μ_I) and dividing by the standard deviation (σ_J, σ_I) in each of the two regions considered,

$$\epsilon = \sum_{\mathcal{W}} \left[\frac{J(A\mathbf{x} + \mathbf{d}) - \mu_J}{\sigma_J} - \frac{I(\mathbf{x}) - \mu_I}{\sigma_I} \right]^2, \tag{6}$$

where $J(\cdot) = I(\cdot, t + 1)$, $I(\cdot) = I(\cdot, t)$.

A more elaborate normalization is described in [8]; [17] reports a modification of the Shi–Tomasi–Kanade tracker based on explicit photometric models.

2.2.1. *What Is a Feature?*

The question that now arises is *what constitutes a feature?* Researchers have traditionally proposed to track corners, or windows with high spatial frequency content, or regions with a sufficiently high mix of second-order derivatives, to overcome difficulties such as the *aperture problem*. Here a less intuitive, but more principled, kind of feature is used; *a* *good feature is one that can be tracked well, making the extraction criterion optimal by construction.*

In this framework, a feature can be tracked reliably if a numerically stable solution to Eq. (5) can be found, which requires that G be well conditioned and that its entries be well above the noise level. In practice, since the larger eigenvalue is bounded by the maximum allowable pixel value, the requirement is that the smaller eigenvalue be sufficiently large. Calling λ_1 and λ_2 the eigenvalues of G, we accept the corresponding feature if $\min(\lambda_1, \lambda_2) > \lambda$, where λ was a user-defined threshold in the work of Shi *et al.* [41]. This constitutes an automatic procedure for identifying reliable regions to track.

2.2.2. Feature Monitoring and Robust Tracking

To summarize the tracker's algorithm, features are detected in an initial frame. In all subsequent frames (within a given time interval, or until enough features are dropped from the initial set), and for each feature, the tracker solves a linear, overconstrained system to determine the current position of the feature. The next problem is how to find features failing to comply with the motion model (feature monitoring).

To monitor the quality of the features, the tracker checks the residuals between the first (after reinitialization of the feature set) and the current frame: high residuals indicate features which violate the motion model and must be rejected. Such bad features are caused by occlusions, perspective distortions, and strong intensity changes (e.g., specular reflections). However, after enough time has elapsed without reinitialization of the feature set, most features eventually undergo significant rotation, scaling or shearing and are discarded. A tracker running continuously must reinitialize before so many features are lost that the task supported (e.g., visual servoing, structure reconstruction) cannot be performed any longer (Section 2.2.3).

We have developed a method for selecting a robust rejection threshold *automatically* within sequences of limited length (i.e., where only a small subset of good features could be erroneously discarded). We begin by establishing which distribution is to be expected for the residuals when good features, i.e., almost identical regions, are compared. We assume that the intensity $I(\delta(\mathbf{x}), t)$ of each pixel in the current-frame region is equal to the intensity of the corresponding pixel in the first frame $I(\mathbf{x}, 0)$ plus some Gaussian noise $n \simeq \eta(0, 1)$. Hence

$$I(\delta(\mathbf{x}), t) - I(\mathbf{x}, 0) \simeq \eta(0, 1).$$

Since the square of a Gaussian random variable has a chi-square distribution, we obtain

$$[I(\delta(\mathbf{x}), t) - I(\mathbf{x}, 0)]^2 \simeq \chi^2(1).$$

The sum of n chi-square random variables with one degree of freedom is distributed as a chi-square with n degrees of freedom (as it is easy to see by considering the moment-generating functions). Therefore, the residual computed according to (4) over an $N \times N$ window $\mathcal{W}$ is distributed as a chi-square with N^2 degrees of freedom:

$$\epsilon = \sum_{\mathcal{W}} [I(\delta(\mathbf{x}), t) - I(\mathbf{x}, 0)]^2 \simeq \chi^2(N^2). \tag{7}$$

As the number of degrees of freedom increases, the chi-square distribution approximates a Gaussian, which is in fact used to approximate the chi-square whenever $N > 30$. Therefore, since the window $\mathcal{W}$ associated to each feature is at least 7×7, we can safely assume a Gaussian distribution of the residual for the good features:

$$\epsilon \simeq \eta(N^2, 2N^2).$$

When the two regions over which we compute the residual are bad features (i.e., they are not warped by an affine transformation), the residual is not a sample from the normal distribution of good features, but an outlier. Hence, *the detection of bad features reduces to a problem of outlier detection*, which is equivalent to estimating the mean and the variance of the corrupted Gaussian distribution. To do this, we employ a simple but effective model-free rejection rule, X84 [18], which achieves robustness by employing median and median deviation instead of the usual mean and standard deviation. This rule prescribes to reject values which are more than k median absolute deviations (MADs) away from the median:

$$\mathrm{MAD} = \operatorname*{med}_i \left\{ \left| \epsilon_i - \operatorname*{med}_j \epsilon_j \right| \right\}. \tag{8}$$

In our case, ϵ_i are the tracking residuals. A value of $k = 5.2$ is adequate in practice, as it corresponds to about 3.5 standard deviations, and the range $[\mu - 3.5\sigma, \mu + 3.5\sigma]$ contains more than the 99.9% of a Gaussian distribution [18]. The rejection rule X84 has a breakdown point of 50%: any majority of the data can overrule any minority.

2.2.3. Implementation Aspects

The theory above has been implemented in a tracker running continuously and capable of supporting real-time applications requiring tracked data several times per second. In our implementation, a set of features is extracted and tracked until a sufficient number are visible. Feature extraction is performed only in the first frame. As outlined in the previous section, the tracker then looks for the corresponding feature within a window in the second image centered at the feature position in the first image. The size of the tracking window for $t = 2$ is the same as that for the extraction phase. Subsequently, the window size is adjusted in accord with the history of feature displacements in the image sequence.

Robustness monitoring (i.e., discarding unreliable features) is performed periodically, whenever it is deemed necessary by the overlaying applications, or when reinitialization is performed. When the number of tracked features drops below a threshold, feature extraction is triggered and the feature set reinitialized. The old features are carried over to the new batch and newly extracted features added. The period of the robustness checks, the threshold triggering feature reinitialization, and the reinitialization strategy itself depend on the particular application incorporating the tracker. For instance, the feature set could be reinitialized when the tracked features are less than the minimum number of features needed for reconstructing 3-D structure, or the reextraction of features could be focused in new areas appearing in the image due to image motion. It is important to note that one tries to limit the number of times that a robustness check is performed to those absolutely necessary for the overlaying applications, as the outlier rejection operation is at least an order of magnitude more expensive than the tracking operation.

Our C++ implementation runs at 6 to 10 frames per second on a Sun Ultra-10 under Solaris, with no special or dedicated hardware. We have measured a speedup by approximately a factor of 2 on a high-performance Alpha card. The full feature extraction phase takes approximately 0.1 s to extract about 35 features from a 250×370 image on the Ultra-10. Notice that full feature extraction is necessary *only* to reinitialize the feature set: subsequent tracking is a local and inexpensive operation.

2.2.4. Experimental Results

We report briefly two examples illustrating video tracking.

The Hyball sequence (Fig. 1) was acquired by a Hydrovision Hyball ROV swimming in our laboratory tank along a metal structure. The sequence consists of 127 gray-level frames, each of 250×370 pixels. The figure shows all the features tracked (left column) and those

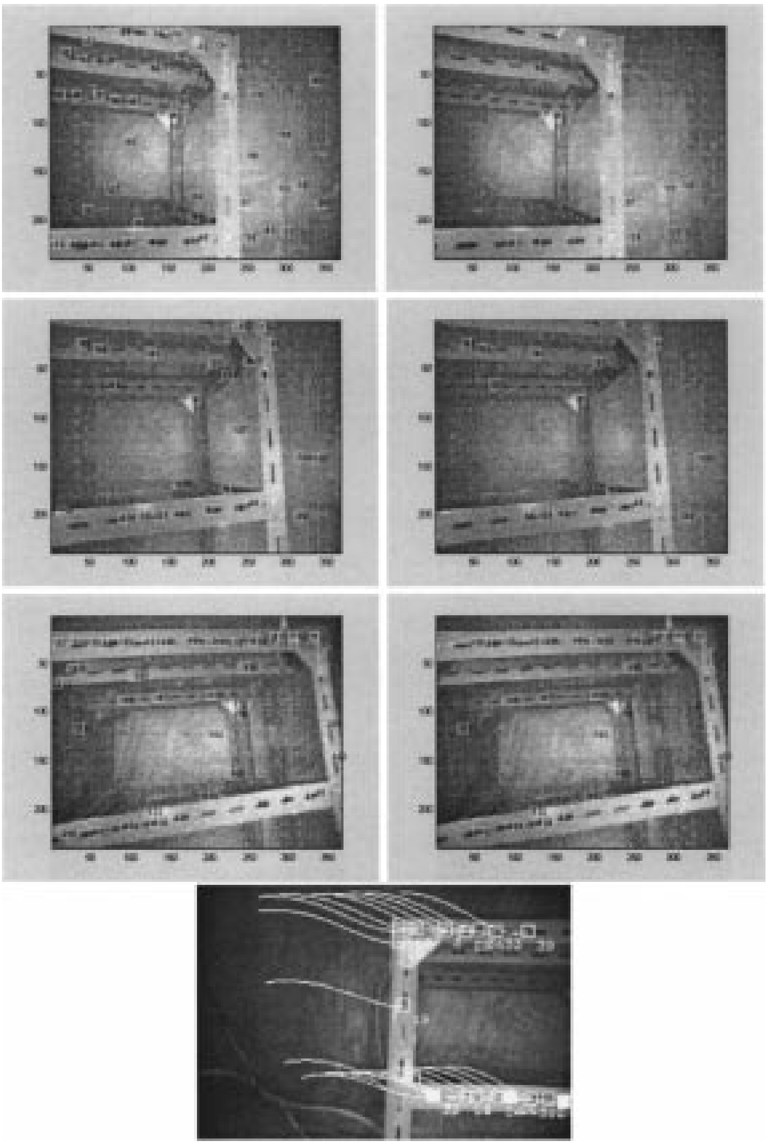
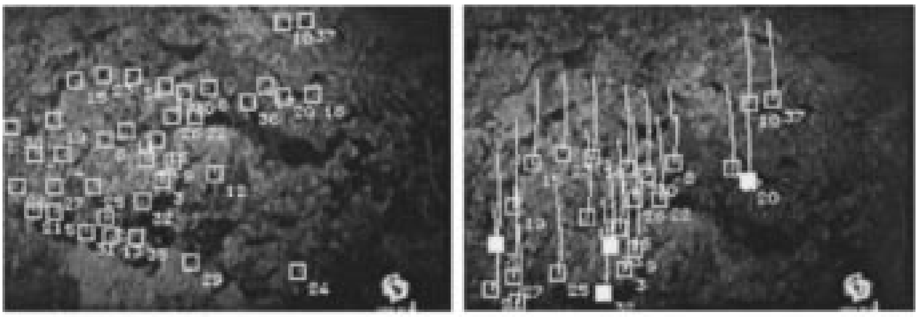


FIG. 1. Robust feature tracking in the Hyball sequence. The rows show frames 17, 30, and 45, with all the features extracted (left column) and those preserved after robust monitoring (right column). The bottom figure shows the tracks of the features present in the last batch.



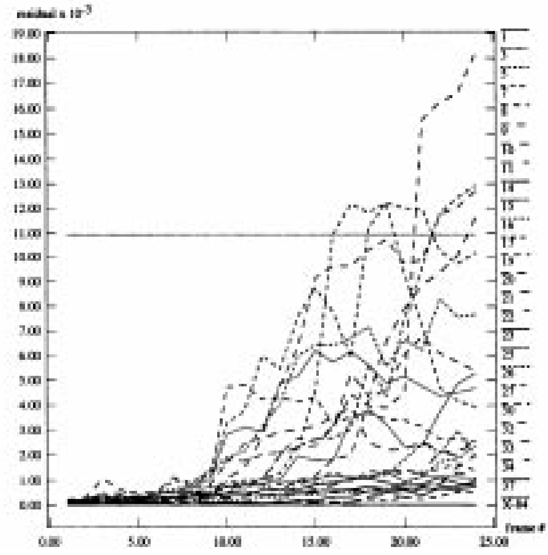


FIG. 2. First and last frame of the Smallrock sequence (white windows in the last frame indicate features rejected as unreliable) and plot of residual magnitude against frame number.

preserved by the robust monitoring (right column), as well as the tracks of the features present in the last frame.

The Smallrock sequence (Fig. 2) consist of 46 frames, each of 240×160 pixels. It was acquired by an ROV operated by the Institute of Marine Biology, Crete, in shallow Mediterranean waters near Psira Island. The figure shows the first and last frames of the sequence, the latter with tracks superimposed. The white squares indicate features dropped as unreliable. Figure 2 also plots the residuals of all features against the frame number. The horizontal line indicates the threshold set automatically by X84. Features associated to residuals higher than the threshold are rejected as unreliable.[2]

In both examples, the tracker adopts the translational model. This has proven adequate, in all our experiments, to support applications running continuously (see discussion in Section 2.2).

To evaluate performance quantitatively in the context of an application, we used the features tracked to compute the fundamental matrix [49] between the first and the last frame of the two sequences. The fundamental matrix was estimated using Zhang's nonlinear

[2] 2MPEG sequences of these and other tests can be found on the Internet at our laboratory's Web site, http://www.cee.hw.ac.uk/Oceans.

TABLE 1
1 RMS Distances

	Hyball	Smallrock
All	0.23	0.58
X84	0.18	0.53

Note. RMS distance of points from epipolar lines.

method [54]. To assess the goodness of the estimated epipolar geometry, we computed the RMS distance of the tracked points from the corresponding epipolar lines. If the epipolar geometry is estimated accurately, indicating accurate tracking, all points should lie on epipolar lines. Table 1 reports, for comparison, results with and without the robust X84 rejection. With both sequences, the robust tracker brings a decrease in the RMS distance, suggesting that features corrupting the estimation have been dropped (this is the prime purpose of robust tracking). This behavior was confirmed in a number of similar tests.

3. SONAR TRACKING

3.1. Introduction

We adopt a feature-based scheme for tracking, so that the first element becomes *segmentation*, that is, identifying the features to be tracked in each frame, or more precisely, *identifying regions of high backscatter* which we will refer to as *objects*.

Segmenting sonar images reliably can be very time consuming. For this reason, we restrict segmentation to *areas of interest* in the image, namely:

- where a basic, fast segmentation algorithm indicates the presence of a new object;
- where a static object was detected in past frames;
- where a moving object is expected to be when the next frame is acquired.

Our segmentatin algorithm was designed for real-time processing (in our case, 3 frames per second) and for accuracies suitable for path planning purposes. The algorithm is organized as follows.

- *First-layer segmentation:* noise smoothing, detection of new objects, location of areas of interest (where new objects are detected and where existing objects, static or tracked, are expected to be found).
- *Second-layer segmentation:* segmentation within the areas of interest.

3.2. First-Layer Segmentation

A common segmentation procedure for sonar images consists of median filtering followed by thresholding [7]. Filtering for noise smoothing is an absolute necessity as backscatter is common, especially in the case of multibeam sonar images. However, it is generally time consuming. We have tested several filtering techniques (mean, median, Gaussian) and found that a good compromise between quality and speed was reached by the following scheme:

- Attenuate backscatter noise using a 7×7 Gaussian filter, which performs almost as well as a median filter but at a reduced computational cost [15].
- Threshold the image with an adaptive thresholding scheme based on the image histogram, which is independent of the actual signal level. The idea is to estimate the probability

density function of the noise from the image histogram, assuming that the latter is a good estimate of the former. Notice that the calculation of the histogram is done on the *original image*, not its filtered version. The estimation of the probability density function is made on the area of the image where no objects have been detected (by the first-layer segmentation) and where no previous objects are likely to be found.

The thresholds are derived from the histogram and only depend on a fixed false alarm rate. To determine a threshold, we fix a target alarm rate and enter this into the cumulative histogram (y axis), which gives us the desired threshold value (x axis). The thresholds are also independent of the energy of the returns. This technique has proved effective even for varying transducer gains and intensity returns. If an object is part of the image from which the histogram has been computed and was not detected by the first-layer algorithm, it contributes most probably to the higher part of the histogram and will be selected even with a high false alarm rate, while most of the noise will be rejected. The noise that is not filtered out corresponds to backscatter returns at the same levels as the objects; this kind of noise cannot be removed without removing part of the interesting objects. We have not run any quantitative analysis of the sensitivity of our tracker to threshold variations, but our thresholding scheme has successfully supported tracking in real sequences (including drop ins/outs) longer than 300 frames at 9 frames per second.

Special images composed mainly of obstacles (with high returns) or containing much backscatter noise from the seabed can be detected easily from their high variance. The process can then be adapted to these special cases.

Our Matlab 5.2 implementation of this technique runs in real time (as defined above) on a Sparc Ultra-10 under Solaris and represents a working compromise between real adaptive filtering (where the threshold value is derived *locally* with respect to the surrounding pixels) and fixed thresholding.

3.3. Second-Layer Segmentation

The second layer takes advantage of the tracking module described in Section 3.4, which tracks the positions of objects from frame to frame and predicts their dynamic characteristics and next locations. This second layer is an object-based algorithm (as opposed to the first layer, which is pixel-based); the necessary object identification is implemented via a labeling algorithm.

The process can be decomposed into two distinct parts, namely:

1. selection of the areas of interest in the image;
2. segmentation within these regions.

3.3.1. Selection of the Areas of Interest

A Kalman filter is associated with each object detected in the scene. Let $\{O_i\}$ denote the set of objects present in the scene at a given instant. This set can be decomposed into two subsets, $\{NewO_i\}$ and $\{TrackO_i\}$, representing respectively the objects just appeared in the image and those tracked from previous frames. For each object in $\{NewO_i\}$, an area of interest is set which matches exactly the labeled object as resulting from the first-layer segmentation. For each object in $\{TrackO_i\}$, an area of interest is set which matches the previously tracked object; the area is positioned using the prediction of the Kalman filter.

The size of the area of interest depends on the uncertainty on position and area computed by the Kalman filter associated with the object.

3.3.2. *Segmentation*

The segmentation algorithm is again based on the histogram of the original image (previously computed) to set the thresholds, but is restricted to the areas of interest. These are filtered using a 7×7 Gaussian filter, after which a double threshold is applied. First, the regions to be considered are selected as those above the higher threshold. Then, the areas connected (by 8-connectivity) to the regions selected by the higher threshold by a continuous chain of pixels whose values are above the lower threshold are selected.

The first merit of this algorithm is that it discards noisy middle-value peaks, which would be kept by a simple thresholding technique. This algorithm also keeps relatively low-intensity pixels connected to high returns, which correspond to less reflective parts of objects. An example of segmentation is shown in Fig. 3. The results shown include objects which have been tracked for a few frames.

Our Matlab 5.2 implementation of the second-layer segmentation runs in 0.1 s on an Ultra-5 workstation under Solaris, mainly due to the confinement of expensive processing to small areas of interest.

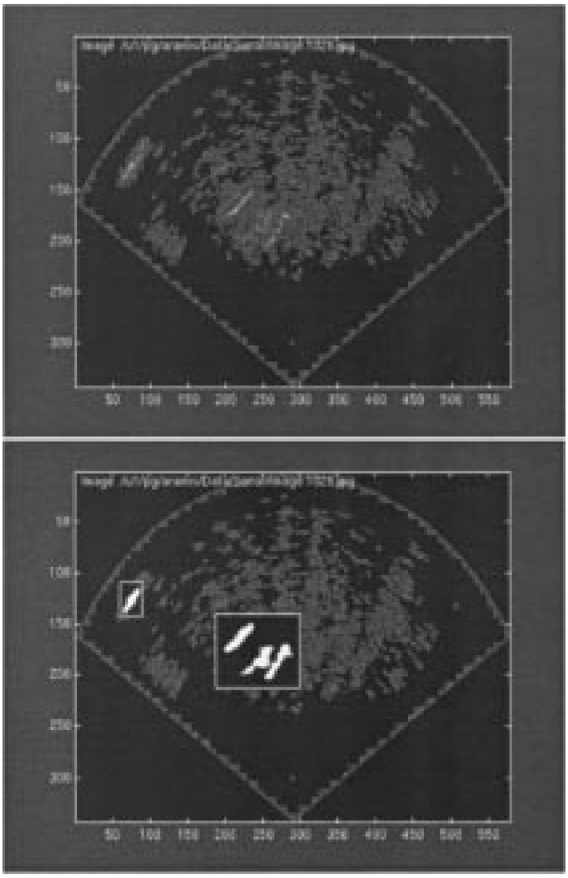


FIG. 3. Example of second-layer segmentation of multibeam, high-resolution sonar images. The input image is shown on top. Data courtesy of the Department of Ocean Engineering, Florida Atlantic University.

3.3.3. Object Labeling

The segmented object regions are labeled using a standard labeling algorithm, and the following features are computed for each object:

- image position;
- object area in pixels;
- object perimeter in pixels;
- second moments.

3.4. How the Sonar Tracker Works

3.4.1. Object-Based Kalman Tracking

Tracking in forward-looking sonar images has been tackled using pixel-based techniques such as optical flow associated with tracking trees or multiple-hypothesis tracking [9, 26]. These techniques perform well when the images are not too noisy, but they often require knowledge of the vehicle movement. To obviate this, we have based our tracker on a combination of segmentation and feature extraction which is object-based rather than pixel-based.

The features extracted are the basis of the tracking algorithm. The tracker has two main functions:

- to reduce the computational cost of the segmentation;
- to extract the dynamic characteristics of the objects to be used for path planning.

Kalman filters are the core elements of our tracking scheme. As mentioned earlier, the noisy nature of sonar images limits strongly the range of tracking techniques that can be used successfully. For instance, the performance of pixel-based (and computationally expensive) techniques such as correlation or optical flow can be very limited. Kalman filters, instead,

- work fast when the vector state is small;
- allow motion estimation, a useful source of information for path planning;
- attach uncertainty estimates to the components of the vector state, again a most useful source of information for path planning;
- allow inclusion of a model of the vehicle dynamics (through extended Kalman filters);
- provide a framework for fusing data from different sensors.

As multibeam sonar frame rates are between 4 and 30 frames per second, simple, standard Kalman filters are suitable for tracking purposes. In our filter, the state vector, X, is composed of the image position in x and y coordinates, the area a of the object, and their first and second derivatives; that is,

$$[x \quad \dot{x} \quad \ddot{x} \quad y \quad \dot{y} \quad \ddot{y} \quad a \quad \dot{a}].$$

The area (a) and its variation are additional features useful for characterizing a target, even without a model of their behavior. Both a and $\dot{a}$ change, in typical sequences, in such a way to allow the use of a linear model. These changes are smooth and therefore suitable for the linear model.

Notice that a single filter integrating all the objects (and thus accounting for their correlation) proves very costly, as the complexity of the Kalman filter is $O(n^3)$, n being the dimensionality of the state vector.

3.4.2. Data Association

Data association, a difficult problem extensively studied [1], is the component of a multi-target tracking system matching the objects detected in the current frame with those tracked in previous frames. Considering the constraints imposed by AUV and ROV operational scenarios, we adopted a *nearest-neighbor algorithm* as a working compromise between speed and performance. The algorithm uses position and area to perform the association. The standard nearest-neighbor method has been modified to cope with temporary merging of two objects, as well as splitting one object into two distinct objects, a frequent occurrence caused by the highly variable intensity of sonar returns.

The data association algorithm has the following structure:

1. Calculate the distances between all the observations and the predicted positions of the tracked objects.
2. For each tracked object, select the observations that
   - either fall within the validation gate of the Kalman filter for position in x and y (the validation gate is defined as the uncertainty on a state of the Kalman filter, given by the covariance matrix of the filter) or
   - intersect the tracked object.
3. For all the selected observations, verify that areas are compatible.
4. Prune the selected observations as follows:
   - If a single observation falls within the validation gate of the tracked object and areas are compatible, keep it and discard the others.
   - If two or more observations fall in the validation gate of the tracked object and areas are compatible, choose the closest one in terms of position.
   - If no observations falling within the validation gate have compatible area, there might be a split or a merge. Try to aggregate observations and check resulting areas. In case of failure, choose the closest observation.
   - There is no observation at all within the validation gate of the tracked object. An error in position estimation might have occurred. Select the observations that are intersecting with the tracked object, if they are not the best choice for another tracked object, and repeat the merge and split test described above.

Once data association has been applied, the tracks can be updated. Three cases are possible:

- A new observation matches the predicted position. The Kalman filter recursion is then applied, a new state vector is derived, and new internal values are computed.
- No new observation matches the prediction. The obstacle prediction is then updated using the filter's internal values. If no match is found between the observations and a given tracked object over a predefined number of frames, the object is discarded as a false alarm.
- An observation is not associated with any tracked object. A new object is created and its corresponding filter initialized.

3.4.3. Experimental Results

This scheme has been successfully tested with real sonar data provided by Florida Atlantic University (FAU). The sonar sequences used were acquired by a forward-looking sonar developed by FAU and fitted onto the front of one of their AUVs (the Ocean Explorer,

FIG. 4. The AUV “Ocean Explorer.” Courtesy of the Department of Ocean Engineering, Florida Atlantic University.

shown in Fig. 4). The sonar has 120 beams of width 1° horizontally and 30° vertically. The range of the sonar was set to 40 m. Figure 5 shows the estimated trajectory of the tracked objects with respect to the vehicle on a sequence of 300 frames at a 9 frame/s frame rate. Images were subsampled by a factor of 2 in both x and y to speed up segmentation, yielding 550×400 images. The latest estimated speed vector is also displayed in the figure. The figure shows four examples of the tracking at different stages.

As a benchmark, we compared our results against the position and heading given by the AUV’s intertial navigation system, also made available with the sequences. As inertial sensors cannot take currents into account, leading to considerable positional uncertainties, we have used only the heading values for our comparison. We notice that differences between measurements can also be due to the different refresh rates (about 1 s for the inertial sensor, 9 frames per second for our tracker). Table 2 shows our heading estimates, computed using the best trajectory of the objects tracked, and those given by the inertial sensors for different frames. For objects too close to the sonar head (less than 10 m), near-field effects corrupt the results. Therefore, they were discarded.

TABLE 2
Heading Estimates Comparison

Frame numbers	Tracker differential heading	Inertial differential heading	Inertial differential heading (interp.)
1048–1148	73.5°	75°	73.2°
1098–1148	36.17°	35.08°	35.08°

Notes. Comparison of the heading estimates using the tracker and the readings of the inertial sensors on the vehicle. The right column corresponds to the interpolated readings of the inertial sensors to take into account the lower refresh rate of these sensors.

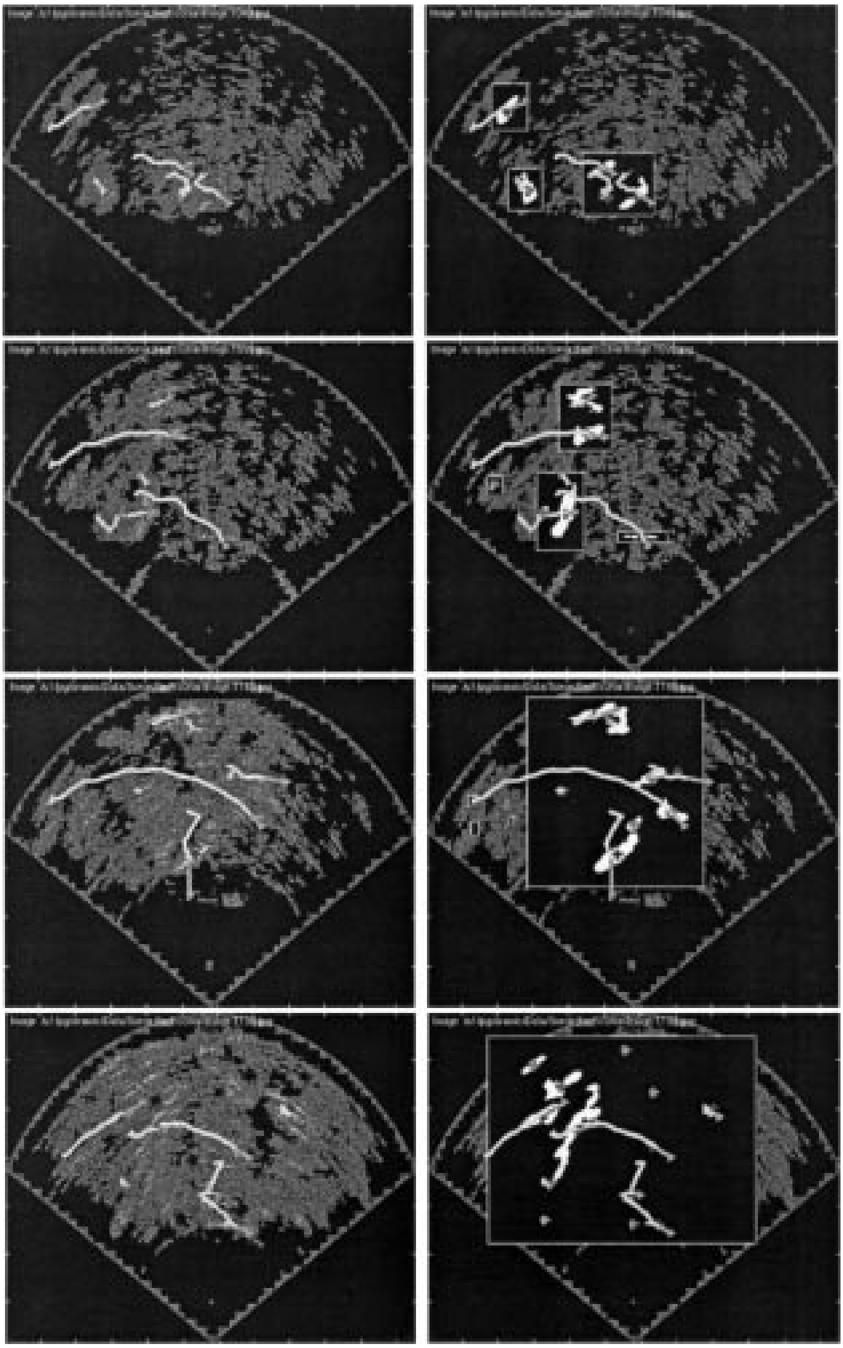


FIG. 5. Example of tracking results on a sequence of 300 multibeam sonar images. Tracked trajectories are shown as white lines, estimated velocity vectors are shown as gray arrows, and object centers are marked with crosses.

These results indicate a good match between our estimate, based on image processing only, and the inertial sensor measures of the vehicle. This suggests that the tracker could be used not only within the path planning system described in Section 4.3, but also as a *secondary navigation sensor*. This would offer two advantages over classical inertial sensors: high frame rate and accounting for currents while tracking objects on the seabed. This application is planned for future work.

4. APPLICATIONS

This section gives a brief account of three applications of our video and sonar trackers.

4.1. Panoramic Mosaics from Video Sequences

We have developed software for building *panoramic mosaics* [25, 35] of scenes scanned by a video camera fitted on a vehicle. No information on the camera motion is necessary; we assume that the scene is mostly planar, although experience indicates that good results (for panoramic displaying, not measuring) are achieved also when the scene is not entirely planar. Reliable frame-to-frame correspondences are computed by the continuous video tracker (translational motion model) and fed into a module estimating the best plane homography aligning two consecutive frames [20, 22]. The homography is then used to warp the current frame onto the reference one: that is, the current image is warped to simulate an image plane coplanar with that of the reference frame. The warped image can be aligned to the reference image without further geometric transformation. Image blending, guaranteeing seamless frame-to-frame borders, is implemented by locally averaging the values of all the pixels mapped to the same pixel in the final mosaic.

An example is given in Fig. 6, which shows 6 frames from a benthic, 150-frame sequence acquired by VICTOR (Fig. 10). Notice the compensation for unknown motion (rotation and translation of the image plane) and zoom effects, visible from the deformation in shape and size of the image frame. The quality of this result is comparable with that of similar techniques applied underwater [16, 34] and in air [25, 35].

Notice that, although the mosaicking algorithm assumes that the scene observed is planar, very good results are achieved with many nonplanar targets; the one in Fig. 6 is an example. Ultimately, quality depends on the combination of scene depth, scene–camera distance, and focal length (assuming good visibility); our experience with video sequences acquired by real ROVs on scientific and industrial missions suggests that our technique yields good-quality mosaics in a large number of practical situations.

4.2. 3-D Structure from Uncalibrated Motion

A video sequence acquired by a single camera in unknown motion contains complete information about the 3-D structure, but not the absolute size, of the scene observed [10, 49]. The absolute size can be computed if the distance in space between two visible points is known. This means that video data acquired by ROV cameras can be used to build 3-D models of unknown targets (e.g., hydrothermal vents, rocks, and wrecks) or to verify the shape and size of humanmade structures (e.g., pipelines, valves, manifolds, and similar installations).

An example of 3-D shape reconstruction from uncalibrated motion is shown in Fig. 7 [38]. The figure shows four frames from a sequence of a calibration pattern acquired in our laboratory tank by a camera moved by hand. No calibration at all was performed.

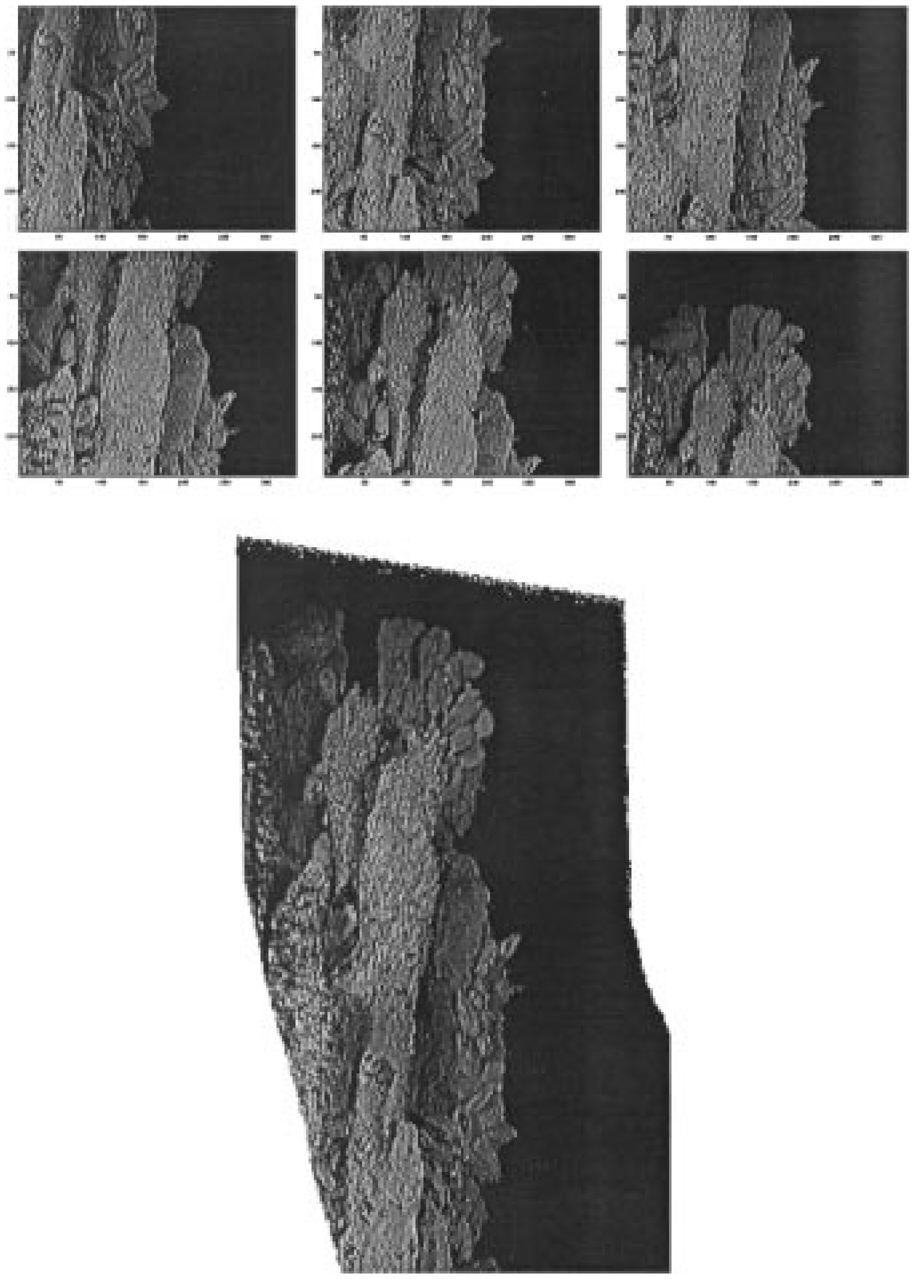


FIG. 6. Six frames from a benthic, 150-frame sequence acquired in the Pacific Ocean, and panoramic mosaic of the whole sequence. Original sequence courtesy of IFREMER Brest.

The video tracker was used to compute reliable point correspondences throughout the sequence. The motions of the image points tracked were then used to estimate the shape (relative position) of the corresponding points in space. The reason for the small number of 3-D points reconstructed is that the feature set was never reinitialized. The accuracy of this reconstruction, computed as the average mean error in the relative position of known points, is approximately 2%.

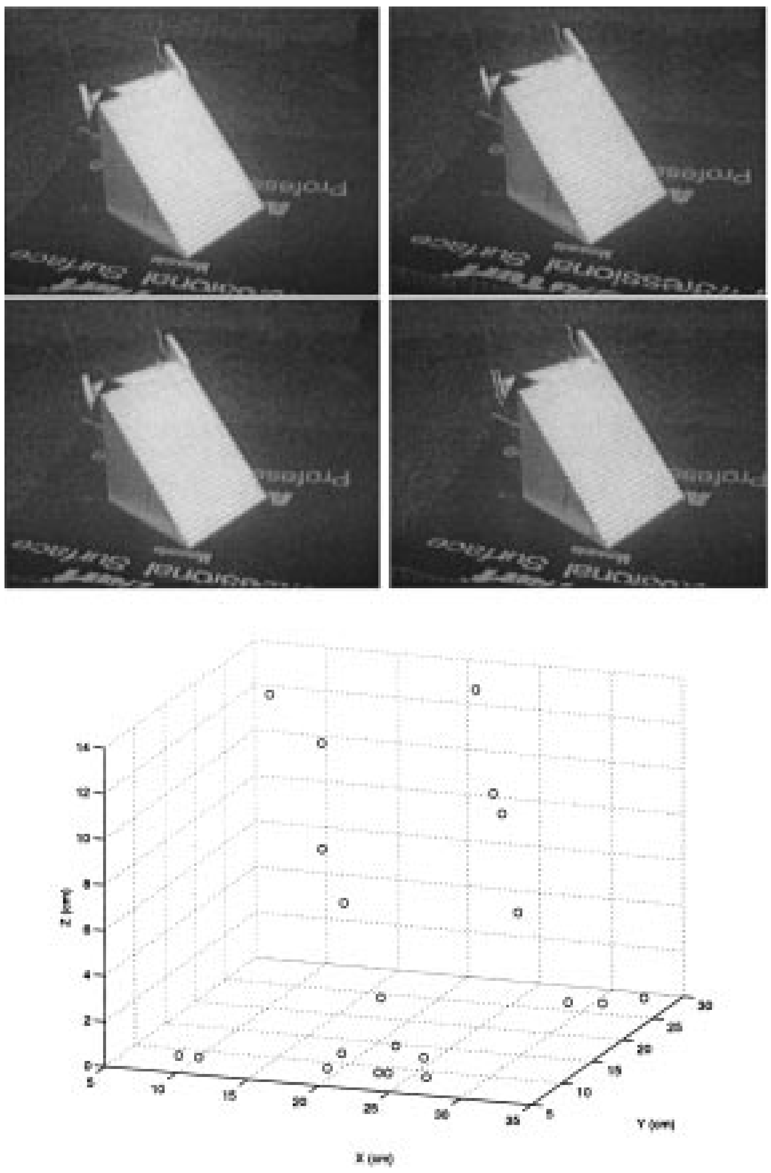


FIG. 7. Four frames from a sequence acquired in our laboratory tank by a hand-held camera (time increases from top to bottom and left to right) and reconstructed 3-D structure of the scene points tracked throughout the sequence.

4.3. Dynamic Obstacle Avoidance and Path Planning from Sonar Sequences

Given the importance of this application, this section explains in some detail our obstacle avoidance and path planning system incorporating the tracker.

4.3.1. Introduction and System Overview

The goal of this research is to develop an obstacle avoidance system for the ARAMIS (advanced ROV package for automatic mobile investigation of sediments) tool-skid. The ARAMIS project (MAST-CT97-0083) provides a geological/scientific tool-skid which will be mounted on two different ROVs, VICTOR from IFREMER, France, and ROMEO from CNR-IAN, Italy, both shown in (Fig. 10) operating at a close distance from the seabed (2 m) at depths ranging from 50 to 2000 m. The cruising speed for both ROVs is around 1 knot and the movements of the ROVs are measured by several on-board sensors feeding the obstacle avoidance system with position, speed, and orientation of the vehicle in world coordinates. The main missions of the ROVs are geological and biological surveys of the seabed and water column, including benthic and pelagic missions which could last up to 72 h. The need for an automated piloting system, or at least an aided piloting system, is clear.

Our ultimate aim is to detect and avoid obstacles using sonar data; therefore, segmenting the image into regions containing the obstacles is a crucial task. It is also useful to know how the obstacles are moving with respect to the vehicle, as this information can be exploited by the obstacle avoidance and path planning algorithms. This information is computed by the tracker. Once the static and dynamic characteristics of the obstacles have been estimated, we create a model of the workspace surrounding the ROV using *constructive solid geometry* (CSG). The choice of CSG to represent obstacles is based on the fact that classical surfaces such as spheres, cylinders, and half-spaces are CSG primitives that can be very easily combined, and we adopt a *convex representation* for the obstacles. In addition, working with convex obstacles facilitates path planning and fosters convergence. The workspace model takes into account the currently visible obstacles as well as those that have gone out of the field of view but the positions of which are still deemed critical to the definition of a safe path.

The system we have designed (Fig. 8) is modular in nature. Modularity is seen as a key feature for handling different needs within the same framework. Short descriptions of each module in Fig. 8 follow.

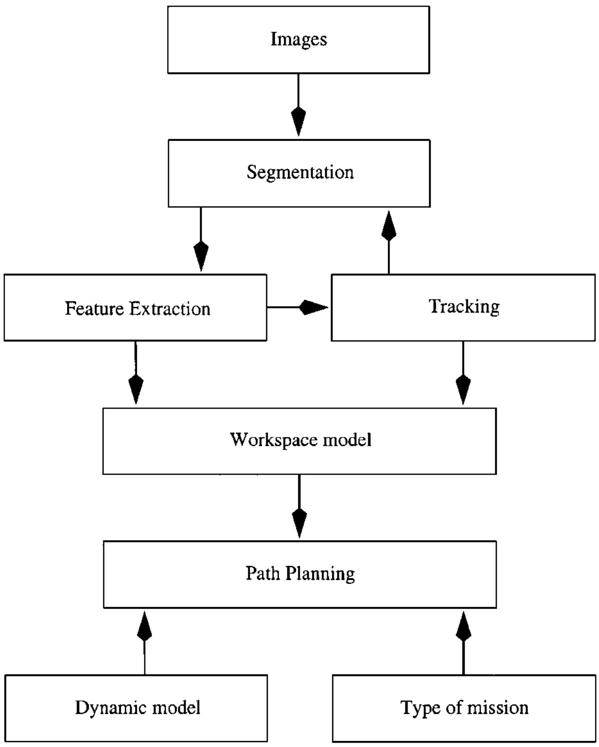


FIG. 8. Architecture of the sonar-based, real-time path planning system.

Segmentation. The purpose of this module is to identify regions of the image corresponding to obstacles. Considering the very nature of multibeam sonar images, we discarded the certainty grid approach [28] often used in motion planning systems based on air ultrasonic sensors and focused on an object-oriented description of the workspace.

Feature extraction. Once the image has been segmented, potential obstacles are located and their features (position, moments, area) are computed. These features will be used later to discard false alarms and to track obstacle motion with respect to the vehicle.

Tracking. This module is described in detail in Section 3. It drives the segmentation, reduces its computational cost, and enables the creation of a map in world coordinates of the obstacles surrounding the ROV (the workspace model).

ROV dynamic modeling. We have a dynamic and kinematic model of Angus 002, a ROV developed by our laboratory [4]. This model takes into account any type of sea current. The model is implemented in a software package interfaced to the rest of our software within a Matlab 5 environment.

Workspace representation. We build a workspace model from the obstacles and features extracted from each image, called an *intrafame workspace model*. Combining the intraframe models over time, we build and maintain a dynamic workspace, updated from frame to frame. This dynamic model is the main input to the path planning algorithm.

Path planning. This module is detailed in the following sections. In essence, we use a nonlinear programming technique based on a CSG representation of the obstacles. Each obstacle, static or moving, is represented as a constraint on the search space (i.e., the path cannot cross the obstacle). The algorithm minimizes the Euclidean distance to the goal subject to such constraints. This approach advances some of our previous work [51, 52].

4.3.2. *Workspace Representation*

The choice of a workspace representation is intimately linked with the path planning technique. Most path planning algorithms assume a convex representation of the obstacles to ensure that the goal is reached [28]. When dealing with a changing environment sensed on the fly, it is advisable to use *reactive path planning*, which does not need a complete description of the workspace between the current position and the goal. The reasons for this are:

- only partial information is available, due to the limitations of the sensor;
- new obstacles can appear in the workspace at any time;
- the accuracy of the obstacle representation changes with the distance from the vehicle.

Global path planning needs a complete description of the workspace, as it defines the *complete* path from the starting point to the goal. This is generally done through *visibility graphs* (see [28] for a review on the subject). *Local path planning*, instead, defines only a *partial* path to the goal and needs only a possibly incomplete representation of the workspace [23, 24, 53]. Global path planning is therefore not advisable in our case, as the workspace is sensed while moving and therefore continuously changing and only partially known.

On the basis of the considerations above, we use a local path planning technique which advances some of our previous work [51, 52]. Only the immediately next step of the path is calculated before new sensor data is acquired. The central idea of the method is to represent the free space as a set of inequality constraints, using CSG as detailed below, within a nonlinear programming problem. The goal is modeled as the unique global minimum of the objective function. The initial configuration of the vehicle is the starting point of the nonlinear search.

Workspace representation using CSG. Planning and obstacle avoidance take place in the *configuration space* of the vehicle, which integrates both its kinematics and its link geometry. In this space, the vehicle is represented as a point. Each obstacle in the workspace is modeled as a constraint. Let S be the 2-D or 3-D surface of a Euclidean space E representing the obstacle, and let us denote the set of its interior points by I, the set of its boundary points by B, and the set of its exterior points by T; that is,

$$\begin{gathered} I \cup B \cup T = E \\ I \cap B = B \cap T = I \cap T = \emptyset. \end{gathered} \tag{9}$$

The nonnegative function g on E is called a *defining function of the obstacle S*, in the CSG sense, if

$$\begin{aligned} &\forall p \in I, \quad 0 < g(p) < 1, \\ &\forall p \in B, \quad g(p) = 1, \\ &\forall p \in T, \quad g(p) > 1. \end{aligned} \tag{10}$$

For example, the defining function of an ellipse when $E = \mathbb{R}^2$ is

$$\forall p \in \mathbb{R}^2, \quad g(p) = (x/a)^2 + (y/b)^2, \tag{11}$$

where a and b are the half-axes of the ellipse and p is the point of coordinates (x, y) in the plan.

An attractive point of CSG lies in the fact that complex objects can easily be constructed from simple canonical objects using union and intersection operations. For instance,

$$\forall p \in E, \quad g^I(p) = \max(g_1(p), g_2(p), \ldots, g_n(p)) \tag{12}$$

defines the *intersection of n objects* with defining functions $g_1, g_2, \ldots, g_n$ respectively, while

$$\forall p \in E, \quad g^U(p) = \min(g_1(p), g_2(p), \ldots, g_n(p)) \tag{13}$$

defines the *union* of the same objects.

However, these functions are difficult to compute in practice and are replaced by approximations. In particular, given a positive real number, m, and n objects represented as above,

$$g^I = \left(g_1^m + g_2^m + \cdots + g_n^m\right)^{1/m} \tag{14}$$

and

$$g^U = \left(g_1^{-m} + g_2^{-m} + \cdots + g_n^{-m}\right)^{-1/m} \tag{15}$$

approximate the objects' intersection and union, respectively. The number m can be used to control the accuracy of the smooth approximation and to obtain convex unions and intersections. Here, for the sake of simplicity but without loss of generality, we represent obstacles as *ellipses*. More general representation are of course possible, for instance, polygonal ones (see for instance [52]), but ellipses strike a good compromise between effectiveness of planning and ease of computation. The system fits an ellipse to the contour of each detected obstacle using our implementation of a linear (and therefore efficient) ellipse-specific fitting algorithm [37]. More complex object representations using more than one CSG primitive are forecast in the future and this is why we present the general CSG framework here.

4.3.3. *Path Planning with Static Objects*

In this case we assume all the objects are static objects in the world reference frame. All obstacles O_i, $i \in [1, n]$ of the workspace are defined as ellipses with defining functions g_i, defined by Eq. (11) in a 2-D Euclidean space. The free space with respect to obstacle O_i is defined as

$$\{p \in E \mid 1 - g_i(p) < 0\}, \tag{16}$$

and the complete free space of the vehicle can be represented as

$$\{p \in E \mid \forall i \in [1, n], 1 - g_i(p) < 0\}. \tag{17}$$

The objective function representing the practical problem to be solved, f, is the minimum distance from the start to the goal point in the configuration space

$$\forall p \in E, \quad f(p) = (p - p_g)^{\mathsf{T}}(p - p_g), \tag{18}$$

where p_g designs the goal point and $^{\mathsf{T}}$ indicates transpose.

Our path planning problem is now cast as a classical nonlinear optimization problem, that is, *minimizing f under constraints $g_1, \ldots, g_n$*. A first, obvious advantage is that well-proven numerical techniques are available to compute a solution. A second advantage is that this approach generates very smooth paths, compatible with feasible vehicle motion. A third advantage is that the effect of each constraint is identified clearly; alternative optimization techniques, e.g., potential fields [53], hide the constraints in the objective function, which often leads to an increased influence of local minima. Finally, modeling obstacles through CSG offers considerable flexibility for the representation of the workspace.

4.3.4. *Path Planning with Moving Objects*

Dealing with moving objects is a very desirable property for an obstacle avoidance system. A static workspace representation can be used, but this requires very frequent updating to take proper account of obstacle movements. Moreover, objects are never found at their real positions during planning (that is, between consecutive updates of the workspace), and significantly suboptimal paths can result when the robot is pushed away from its trajectory by a moving obstacle.

Our solution is to *incorporate time explicitly* into the representation [13, 14], thus moving to a three-dimensional workspace. It is straightforward to include the dynamic properties of the obstacles (estimated by the tracker) in the CSG representation. This is done by making the parameter p, describing the position of the robot in the configuration space, and the defining function g dependent on time. As an example, the defining function of an ellipse with $E = \mathbb{R}^3$ is made time-dependent and follows

$$\forall p \in \mathbb{R}^3, \quad g(p) = (x(t)/a(t))^2 + (y(t)/b(t))^2, \tag{19}$$

where a and b are the half-axes of the ellipse and p is the point of coordinates (x, y, t) in $\mathbb{R}^3$. The workspace should represent the objects in a world reference frame, and only objects which are moving in the current frame (*and not with respect to the vehicle*) should be considered moving objects.

We must reformulate the path planning problem. All the equations derived for the static case are still valid, but p is now a function of position, p (in configuration space), and also of time, t. This has two main consequences. First, *new constraints must be added to the nonlinear optimization*, to model the fact that t is a positive and monotonically increasing variable. Second, *the goal must now be defined in time as well*. As it is impossible to know *a priori* when the vehicle reaches the goal, we estimate the goal time as the time taken by the vehicle to travel a straight path (no obstacles) between starting and goal position at maximum speed. Then, at each iteration of the path planning algorithm, a new time to go is computed and the time value of the goal representation is updated.

This technique ensures a feasible solution at each iteration; moreover, the time-varying representation of the workspace is perfectly suited to integrate tracking information, thus yielding a reliable, dynamic path planning algorithm.

4.3.5. Results

For reasons of space, we show only one example of path generation by the system integrating segmentation, feature extraction, tracking, and dynamic path planning, with real sequences of sonar data. The sequence was taken by a forward-looking sonar developed by FAU and fitted on the front of the Ocean Explorer (Fig. 4). The sonar has 120 beams of width 1° horizontally and 30° vertically. As no model of the Ocean Explorer was available for integration in the simulation, we have simulated the movement of a “blind” ROV, driven by the data received from the sonar. Path planning was performed in the ROV reference frame, not in the world reference frame, to simulate a relative motion between objects and vehicle. Of course, other conditions being equal, the relative motion (and therefore the images) generated by a still sonar looking at a moving environment is the same as the motion generated by a moving sonar in a still environment, which is the situation expected in a real mission.

The goal was intentionally set so that the generated path would cross that of moving obstacles. The left column of Fig. 9 shows four frames from the original sequence; the right column shows the same frames after segmentation. The obstacles identified are highlighted. The boxes are rectangular regions containing obstacles. The path generated for the vehicle and the trajectory of tracked, moving obstacles are shown in both columns.[3]

[3] An animated MPEG version of these and other results can be found on our Web site, http://www.cee.hw.ac.uk/~aramis/resources/.

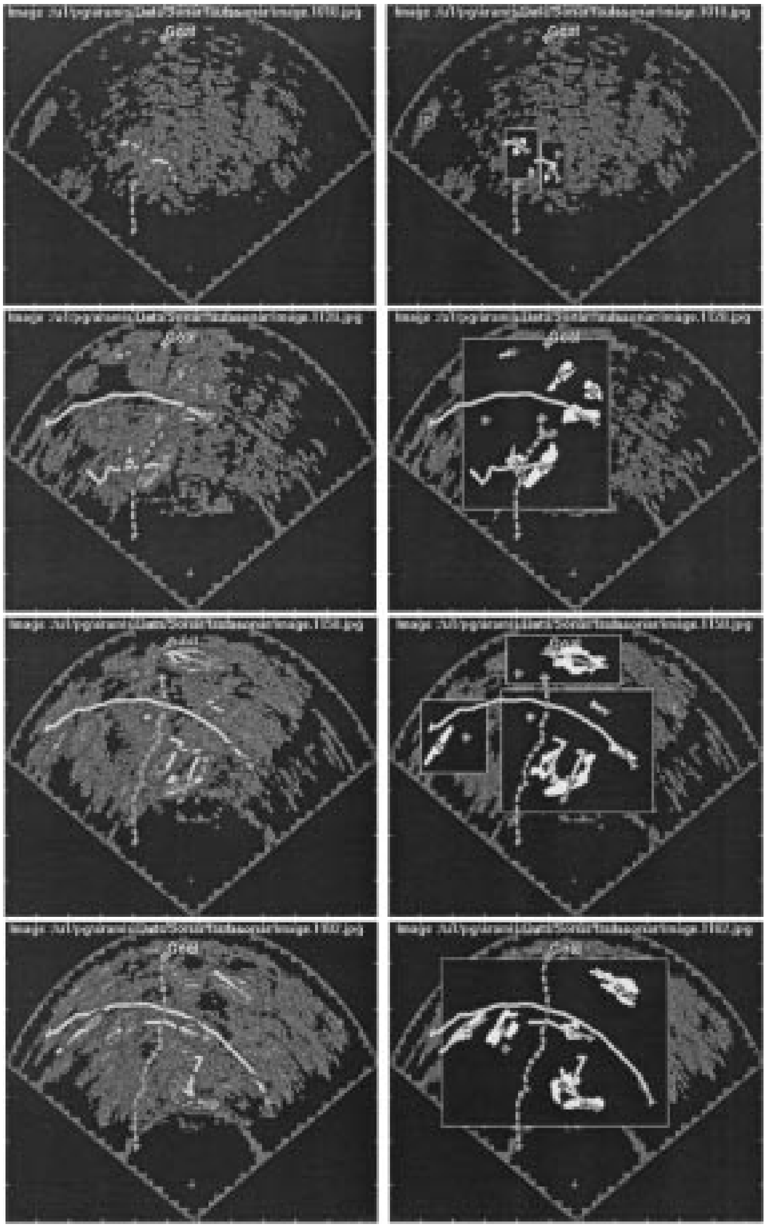


FIG. 9. Left column: Example of dynamic path generation with a sequence of real images containing moving obstacles. Right column: The same with superimposed segmentation output. In each image, the goal is indicated by the top middle cross, and the starting point is bottom left.

In order to reduce processing time, the original images (1200 × 700 pixels) were subsampled by a factor of 2 in both dimensions. The whole process (from segmentation to path generation), implemented in Matlab 5.2 on a Sun Ultra-10 under Solaris, runs at 3 s per frame. This suggests that an optimized C++ version, part of which could run on separate, high-speed hardware, could easily cope with an input frame rate of a few images per second, certainly sufficient for a real-time system fitted on an AUV at cruising velocity standard for data collection missions (a few knots).

5. CONCLUSIONS

The main contributions of our work on video and sonar tracking for subsea sequences are summarized below.

The real-time video tracker:

- has a much lower computational cost than trackers based on robust regression and random sampling techniques like RANSACK or LMedSq [32, 46];
- shows very good reliability in our experiments with sequences acquired during real ROV missions in a variety of environments;
- runs at frame rates suitable for real-time applications;
- is used as a standard components in ROV/AUV software modules developed in our laboratory.

The sonar tracker:

- locates and tracks obstacles reliably in real, multibeam sonar sequences, taking full advantage of dense spatial and temporal information;

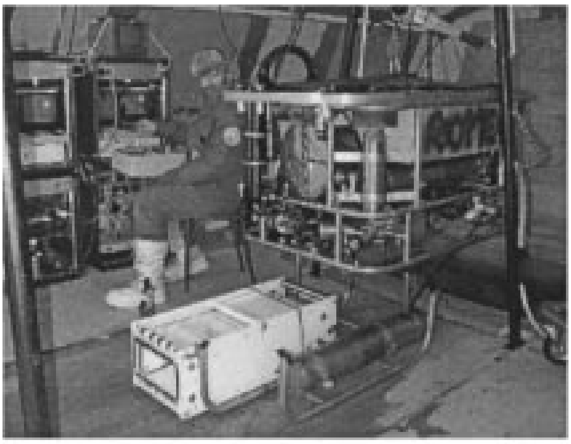
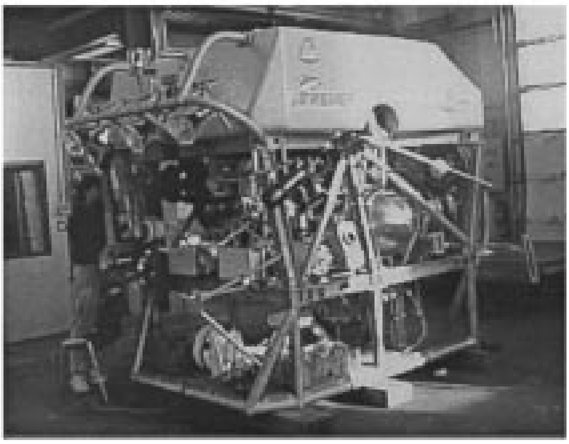



FIG. 10. 10. The two ROVs allocated to the ARAMIS project. Top: ROMEO, developed by the Institute of Naval Automation (IAN), Genoa (Italy, here seen during its mission in Antarctica in 1998. Bottom: VICTOR, developed by IFREMER Toulon (France). Images courtesy of IAN and IFREMER.

- is fast enough to support real-time ROV or AUV operations;
- has been integrated in a fully working path planning and obstacle avoidance system.

Current and future work involving video tracking include incorporating the video tracker in:

- the visual servoing control system of RAUVER, a 2-m, twin-hulled, class 5 robot submersible developed in our laboratory, which can be operated as a ROV or an AUV[4];
- a real-time video mosaicking system under development.

Current work and future work involving sonar tracking include:

- evaluating our tracker as an alternative navigation sensor to inertial sensors;
- testing the obstacle avoidance and path planning system on RAUVER, and subsequently on ROMEO and VICTOR (see Fig. 10);
- addressing the problem of unwanted returns, typically from the sea bottom when looking at obstacles in the water column, due to the aperture of the receiving beams (typically in the range [0.5°, 2°] horizontally and [15°, 30°] vertically).

ACKNOWLEDGMENTS

We are indebted to Professor John Cushieri of Florida Atlantic University, Department of Ocean Engineering, for the Ocean Explorer sonar sequences; to Chris Smith of the Institute of Marine Biology, Crete, for video data of their ROV missions; to Fabrice Lecormu of IFREMER (Brest) for the VICTOR sequences. At OSL, the mosaicking software was written by Anthony Doull; RAUVER was designed and assembled by Kelvin Hamilton and Leonard McLean. Our thanks are given to Andrea Fusiello, Jean-Francois Lots, Katia Lebart, and Alistair Cormack for their many and varied contributions. This work was supported by MAST Contract MAS3-CT97-0083 "ARAMIS" and Marie Curie Fellowship MAS-CT97-5038.

REFERENCES

1. Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*, in Mathematics in Science and Engineering, Vol. 179, Academic Press, San Diego, 1988.
2. J. L. Barron, D. J. Fleet, and S. Beauchemin, Performance of optical flow techniques, *Internat. J. Comput. Vision* **12**(1), 1994, 43–77.
3. J. M. Bell, *A Model for the Simulation of Sidescan Sonar*, Ph.D. thesis, Heriot–Watt University, Edinburgh, Scotland, September 1995.
4. P. Bellec, Simulation of the 6 Degree of Freedom Motion of a Remotely Controlled Unmanned Submersible, Master's thesis, Heriot–Watt University, Edinburgh, EH14 4AS, Riccarton, 1980.
5. A. Blake, R. Curwen, and A. Zisserman, A framework for spatio-temporal control in the tracking of visual contours, *Int. J. Comput. Vision* **11**(2), 1993, 127–145.
6. M. Campani and A. Verri, Motion analysis from first order properties of optical flow, *CVGIP: Image Understand.* **56**, 1992, 90–107.
7. M. J. Chantler and J. P. Stoner, Automatic interpretation of sonar image sequences using temporal feature measurements, *IEEE J. Ocean. Engrg.* **22**(1), 1997, 47–56.
8. I. J. Cox, S. Roy, and S. L. Hingorani, Dynamic histogram warping of image pairs for constant image brightness, in *Proceedings of the IEEE International Conference on Image Processing, 1995*, pp. 366–369.
9. J. Cushieri and S. Negahdaripour, Use of forward scan sonar images for positioning and navigation by an AUV, in *Proceedings of OCEANS'98, Nice, France, September 1998* (IEEE/OES, Ed.), Vol. 2, pp. 752–756.

[4] For more information on RAUVER, see http://www.cee.hw.ac.uk/∼kelvin/rauver_index.htm.

10. O. D. Faugeras, What can be seen in three dimensions with an uncalibrated stereo rig? in *Proceedings of the 2nd European Conference on Computer Vision, S. Margherita, Italy, 1992*, pp. 563–578.

11. G. L. Foresti, V. Murino, C. S. Regazzoni, and A. Trucco, A voting-based approach for fast object recognition in underwater acoustic images, *IEEE J. Ocean. Engrg.* **22**(1), 1997, 57–65.

12. A. Fusiello, V. Roberto, and E. Trucco, Efficient stereo with multiple windowing, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Puerto Rico, June 1997*, pp. 858–863, IEEE Comput. Soc., Los Alamitos, CA.

13. J. Gil de Lamadrid and J. Zimmerman, Avoidance of obstacles with unknown trajectories: Locally optimal paths and path complexity, I, *Robotica* **11**, 1993, 299–308.

14. J. Gil de Lamadrid and J. Zimmerman, Avoidance of obstacles with unknown trajectories: Locally optimal paths and path complexity, II, *Robotica* **11**, 1993, 403–412.

15. R. Gonzalez and R. Woods, *Digital Image Processing*, Addison–Wesley, Reading, MA, 1992.

16. N. Gracias and J. Santos-Victor, Automatic mosaic creation of the ocean floor, in *Proceedings of the IEEE Oceans '98 Conference, Nice, France, June 1998*, pp. 257–262, IEEE Comput. Soc., Los Alamitos, CA.

17. G. D. Hager and P. N. Belhumeur, Real-time tracking of image regions with changes in geometry and illumination, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, California, 1996*, pp. 403–410.

18. F. R. Hampel, P. J. Rousseeuw, E. M. Ronchetti, and W. A. Stahel, *Robust Statistics: The Approach Based on Influence Functions*, in Wiley Series in Probability and Mathematical Statistics, Wiley, New York, 1986.

19. R. K. Hansen and P. A. Andersen, The application of real 3D acoustical imaging, in *OCEANS'98 IEEE Proceedings, Nice, France, September 1998*, pp. 738–741.

20. R. I. Hartley, Theory and practice of projective rectification, *Internat. J. Comput. Vision* **35**(2), 1999, 1–15.

21. L. Henriksen, Real-time underwater object detection based on electrically scanned high-resolution sonar, in *1994 Symposium on Autonomous Underwater Vehicle Technology, AUV'94, Cambridge, Massachusetts, July 1994*, IEEE Oceanic Engineering Soc.

22. F. Isgro and E. Trucco, Projective rectification without epipolar geometry, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Fort Collins, Colorado, June 1999.*

23. O. Khatib, *Commande dynamique dans l'espace opérationnel des robots manipulateurs en présence d'obstacle*, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, 1980. [in French]

24. O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, *Int. J. Robot. Res.* **5**, 1986, 90–98.

25. A. Krishnan and N. Ahuja, Panoramic image acquisition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, California, 1996*.

26. D. M. Lane, M. J. Chantler, and D. Dai, Robust tracking of multiple objects in sector-scan sonar image sequences using optical flow motion estimation, *IEEE J. Ocean. Engrg.* **23**(1), 1998, 31–46.

27. D. M. Lane and J. P. Stoner, Automatic interpretation of sonar imagery using qualitative feature matching, *IEEE J. Ocean. Engrg.* **19**, 1994, 391–405.

28. J. C. Latombe, *Robot motion planning*, Kluwer Academic, Boston, 1991.

29. B. D. Lucas and T. Kanade, An iterative image registration technique with an application to stereo vision, in *Proceedings of International Joint Conference on Artifical Intelligence, 1981.*

30. L. Matthies, T. Kanade, and R. Szelisky, Kalman filter based algorithms for estimating depth from image sequences, *Internat. J. Comput. Vision* **3**, 1989, 209–236.

31. P. S. Maybeck, *Stochastic Models, Estimation, and Control, Volume 1.* Academic Press, New York, 1979.

32. P. Meer, D. Mintz, D. Y. Kim, and A. Rosenfeld, Robust regression methods in computer vision: A review, *Int. J. Comput. Vision* **6**, 1991, 59–70.

33. M. Mignotte, C. Collet, P. Perez, and P. Bouthemy, Unsupervised Markovian segmentation of sonar images, in *ICASSP, IEEE International Conference on Acoustics, Speech, Signal Processing—Proceedings, 1997*, Vol. 4, pp. 2781–2784.

34. S. Negahdaripour, X. Xu, and A. Khamene, Applications of direct 3-d motion estimation for underwater vision systems, in *Proceedings of the IEEE Oceans '98, Nice, France, September 1998*, pp. 51–55, IEEE Computer Soc., Los Alamitos, CA.

35. S. Peleg and J. Herman, Panoramic mosaics by manifold projection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1997*, pp. 338–343.

36. Y. Petillot, I. Tena Ruiz, D. M. Lane, Y. Wang, E. Trucco, and N. Pican, Underwater vehicle path planning using a multi-beam forward looking sonar, in *Proceedings of OCEANS'98, Nice, September 1998*, Vol. 2, pp. 1194–1199, IEEE/OES, New York.

37. M. Pilu, A. W. Fitzgibbon, and R. B. Fisher, Ellipse-specific least-squares fitting, in *Proceedings of the IEEE Conference on Image Processing, Lausanne, Switzerland, 1996*.

38. K. Plakas, E. Trucco, and A. Fusiello, Uncalibrated vision for 3-D underwater applications, in *Proceedings of OCEANS'98, Nice, September 1998*, Vol. 1, pp. 272–276, IEEE/OES, New York.

39. L. Robert, C. Zeller, O. Faugeras, and M. Hébert, Applications of non-metric vision to some visually-guided robotics tasks, in *Visual Navigation: From Biological Systems to Unmanned Ground Vehicles* (Y. Aloimonos, Ed.), Chap. 5, pp. 89–134, Erlbaum, Hillsdale, NJ, 1997.

40. L. S. Shapiro, H. Wang, and J. M. Brady, A matching and tracking strategy for independently moving objects, in *Proceedings of the British Machine Vision Conference*, pp. 306–315, BMVA, Edinburgh, 1992.

41. J. Shi and C. Tomasi, Good features to track, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June 1994*, pp. 593–600.

42. F. Spindler and P. Bouthemy, Real-time estimation of dominant motion in underwater video images for dynamic positioning, in *Proceedings of the IEEE Int. Conf. on Robotics and Automation, Louven, Belgium, 1998*, pp. 1063–1068.

43. C. Tomasi and T. Kanade, *Detection and Tracking of Point Features*, Technical Report CMU-CS-91-132, Carnegie Mellon University, Pittsburg, Pennsylvania, April 1991.

44. C. Tomasi and T. Kanade, Shape and motion from image streams under orthography—A factorization method, *Internat. J. Comput. Vision* **9**(2), 1992, 137–154.

45. T. Tommasini, A. Fusiello, V. Roberto, and E. Trucco, Robust feature tracking in underwater video sequences, in *Proceedings of OCEANS'98, Nice, France, September 1998*, Vol. 1, pp. 46–50, IEEE/OES, New York.

46. P. H. S. Torr and A. Zisserman, Robust parameterization and computation of the trifocal tensor, in *British Machine Vision Conference* (R. Fisher and E. Trucco, Eds.), pp. 655–664, BMVA, Edinburgh, 1996.

47. P. H. S. Torr, A. Zisserman, and S. Maybank, Robust detection of degeneracy, in *Proceedings of the IEEE International Conference on Computer Vision* (E. Grimson, Ed.), pp. 1037–1044, Springer-Verlag, Berlin/New York, 1995.

48. E. Trucco, V. Roberto, S. Tinonin, and M. Corbatto, SSD disparity estimation for dynamic stereo, in *Proceedings of the British Machine Vision Conference* (R. B. Fisher and E. Trucco, Eds.), pp. 342–352, BMVA, Edinburgh, 1996.

49. E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, New York, 1998.

50. A. Verri and E. Trucco, Finding the epipole from uncalibrated optical flow, *Image Vision Comput. J.* **17**, 1999, 605–609.

51. Y. Wang and D. M. Lane, Subsea vehicle path planning using nonlinear programming and constructive solid geometry, in *IEE Proceedings on Control Theory Applications, 1997*, Vol. 144, pp. 143–152.

52. Y. Wang and D. M. Lane, Path planning for underwater vehicles using constrained optimisation, in *Oceanology International Conference 1998, Brighton, March 1998*, pp. 175–186.

53. Y. Zhang and K. P. Valavanis, A 3-D potential panel method for robot motion planning, *Robotica* **15**, 1997, 421–434.

54. Z. Zhang, Determining the epipolar geometry and its uncertainty: A review, *Int. J. Comput. Vision* **27**(2), 1998, 161–195.