HARBOUR IMAGES SEQUENCE ANALYSIS FOR CONTROL AND MONITORING

P. Gamba[1] *A. Mecocci*[2]

[1]Dipartimento di Elettronica, Università di Pavia, Via Abbiategrasso, 209, I-27100 Pavia, Italy
Tel. +39-382-505923 Fax. +39-382-422583 e-mail: gamba@ipvmw2.unipv.it
[2]Facoltà di Ingegneria, Università di Siena, Via Roma, 77, I-54100 Siena, Italy

ABSTRACT

A system devoted to ship traffic control in a harbour environment is proposed, where optic flow approach and monocular image sequences are used. In each frame the scene is segmented, moving and still objects are found, and the movement of each ship is completely tracked. Partial and/or total occlusions are correctly handled by means of suitable grouping algorithms. Quantitative motion estimation is obtained by an iterative procedure that extract precise 3D information from the monocular sequence. The implemented version of the system is able to monitor and control a harbour environment with a substantially low computational effort.

1. INTRODUCTION

More and more interest can be observed in literature for complex environments controlling and monitoring. The reason of this interest is the growing traffic (of airplanes, cars, ships, people, ...), so that a considerable effort has to be made to control each moving target, guide it along the right path, and avoid collisions.

To monitor environments characterized by a great number of structures, moving and/or still, it is possible to exploit the integration of different sensors located around the area to control [1]; if the targets to be tracked are ships, usually these sensors are radars. This is useful in open sea, but not in harbours: harbour environments are filled with buildings and still structures, whose clutter makes difficult the use of radars for traffic control. Instead, to compute the motion parameters of moving ships by means of other type of sensors, like commercial videocameras, is a possibility that, as far as we know, has not yet been examined. The most practical setup for harbour controlling is a single monocular sensor, exploiting its capabilities to analyze the environment, and retrieve motion and depth parameters of the targets.

Many approaches to monocular sequence analysis has been developed, ranging from active vision algorithms [2] to edge grouping and tracking methods [3], from active contours procedures [4] to model-based tracking algorithms [5]. Moreover, in a harbour we must observe that moving targets can be partially or even totally occluded by still structres. This other problem reveals the need for algorithms able not only to recognize and reconstruct the shape of an object, but also to follow it among different moving structures. In literature this problem is addressed in [6], where the concept of "history" of the frames has been introduced: the knowledge of the behaviour in the past frames of a structure could give an initial guessed position if we consider a long image sequence and a slow-time varying situation. Fortunately, this is the case for a harbour environment, where ships generally move slowly.

However, despite of the number of theoretical approaches developed, as far as we know little effort has been made to test all these algorithms in a complex environment, where noise level is very high and the robustness of the procedures can be completely evaluated. Therefore, we tried to set up an automatic system for control, evaluating a number of different solution and testing on a practical site their efficiency and reliability. We implemented the Optic Flow approach and a tracking algorithm able to consider also occlusions in a very flexible and user-friendly environment: with the proposed system architecture we are able to follow many targets.

2. OVERALL SYSTEM DESCRIPTION

To develop a complete system for harbour monitoring and controlling, it is necessary to set up a complex procedure: a schematic block diagram of the whole software architecture is presented in fig. 1. There are three main logical subparts: the first is devoted to the definition of the regions of interest; the second to the pre-processing of the image sequences; the third refers to the detection of changing regions and to their characterization and tracking.

2.1. Definition of the regions of interest

The input interface aims to a first refinement of the objectives of the control system: since in a harbour there are a lot of different targets, with very different behaviours, the operator is invited to suggest his regions of interest in a still frame. In the proposed environment, in fact, it is possible to have ships (moving or still), persons, cars and trucks (also moving or still), large infrastructures (still) and smaller ones, like quay-cranes (that can be considered as objects *partially* moving). The operator identifies some regions of the scene, defining their nature:

1. regions *still*, where no interesting moving object will ever be found;
2. regions *movable*, where non interesting moving objects may be or where *partially* moving structures are located;
3. regions *moving*, suitable for proper processing and search for targets of interest

This definition allows to greatly reduce, by means of the knowledge of the operator, the regions of the frames where successive processing operations are to be performed. User preferences, defined only once at system setup, are stored

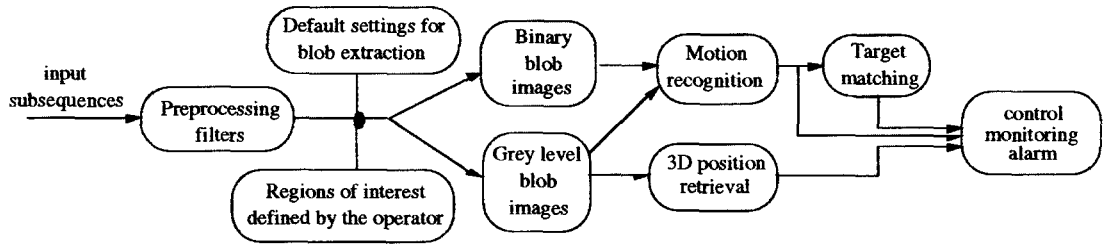

Figure 1. The harbour control system block diagram.

in a file and retrieved at every successive startup. Nevertheless, each time the operator assume that a re-definition of the regions of interest is necessary, he can run the procedure again and make the control system more suitable to the new situation.

2.2. Pre-processing techniques

Pre-processing techniques are compulsory in an external, complex, environment where the lighthnening conditions may vary in an extremely wide range to grant a uniform input to the successive segmentation and tracking steps. The approach followed has been to implement this logical subpart as a learning procedure: samples of the scene in different moments during the day and with different weather conditions are stored in order to provide the parameters useful for a suitable pre-processing of the images. Moreover, a dynamic library with efficient normalization, correction and image registration routines has been implemented, allowing to extend and readapt the chosen techniques to the environment.

2.3. Scene Segmentation

Next step in the procedure is to recognize in the scene the different moving objects in order to evaluate their movements. Even if this particular scene segmentation process can be accomplished in a number of way, a fast solution in tracking problems is to find moving parts (generally called *blobs*) by computing the difference between subsequent frames of a sequence [11]. We followed this simple procedure applying it only to the regions of interest, adding an adaptive thresholding operation, where the grey-level distribution of the foreground and background are considered as Gaussian, and their mean value and variance $m_{1,2}, \sigma_{1,2}$) are computed minimizing the difference between the measured and the proposed behaviours (fig. 2).The binary images found, however, define moving regions as well as non-interesting areas, since our environment is very noisy and the images are taken by commercial CCD cameras. To eliminate the noise effects, a simple morphological filtering technique is used.

Then, in order to simplify the successive processing and reduce the cpu-time needed, a set of windows is created in the binary images around one or more blobs among those detected. A list of *moving windows* is therefore associated to each frame, and the successive image processing steps are applied only to the corresponding parts of the grey-level original images, resulting in a faster algorithm, but also in the possibility of introducing a parallel procedure. For each *moving window*, a chain-code algorithm efficiently stores in memory the blob perimeter. However, even if the previous processing locate regions where changes has arised, not all these areas correspond actually to moving objects in the scene. We need to recognize, within the changing windows, the regions where real motion exists. As observed in the introduction, we can compute the optic flow in the interesting zones of the sequence frames. Windows associated to blobs whose optic flow is equal to zero or motion vectors are incoherent are discarded as meaningless and probably due to noise. To grant real-time capabilities, the spatio-temporal approach [7] has been applied. The method is based on the following equation (the so called *motion-constraint* equation)

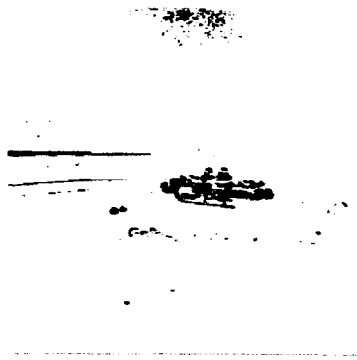

Figure 2. The parts of an harbour scene recognized as moving objects.

$$\frac{\partial g}{\partial x}u_x + \frac{\partial g}{\partial y}u_y + \frac{\partial g}{\partial t} = 0 \qquad (1)$$

stating that the function $g(x, y, t)$ (grey level of the point with cohordinates x, y at time t) satisfies for some ∂x, ∂y and ∂t

$$g(x + \partial x, y + \partial y, t + \partial t) = g(x, y, t) \qquad (2)$$

provided that no change in the light is occurred. In equation (1), in fact, the values of ∂x, ∂y and ∂t are related through u_x and u_y, respectively the x- and y-components of the optic flow. Variables in equation (1) are underconstrained, since it defines a line in the (u_x, u_y) space, and needs an additional constraint to be solved. Morever, noise due to light variations and quantization problems may produce computational errors. A way to solve the problem is the so called *constraint line clustering* algorithm, proposed by Schunck [9], that starts from the polar form of the same equation

$$d = \rho \cos(\alpha - \beta) \qquad (3)$$

where $\rho(x, y)$ and $\beta(x, y)$ are respectively the motion velocity and direction, and d, α are the distance and complementary angle of the motion constraint line. The polar form is a better choice when *gradg* is not linear (e.g. where sharp boundaries are present, like in our case).

Our approach applies a pyramidal procedure to the algorithm, that allows to exploit the high spatial correlation

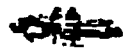

Figure 3. The tracked ship in the previous scene.

among neighbour pixels, and compute more efficiently the values of the optic flow. Pyramidal procedures are common in classification algorithms, where have proved to be useful, and even in our situation, this technique helps and fastens the procedure if there are relatively large groups of pixels in the images with the same or similar optic flow value, e.g. dealing with large ships (or close range sequences) in a harbour. The optic flow values are computed mainly at the low resolution level, while higher resolutions are useful for boundaries and fine structures. However, we need to observe that the approach outlined in [9] is efficient and error-free only for high frame-acquisition rate, that is not our case. The pyramidal implementation corresponds to a spatial averaging of the velocity filed, thus giving an easier estimate of the cluster center position in noisy data. The averaging procedure, that works also as a noise-filtering step, is tolerable since the moving targets (ships) are rigid body objects, whose optical flow values over the image plane are indeed the same for any pixel of their projection. This is the main reason for the pyramidal implementation.

As the final step in segmentation, a merging algorithm is implemented, in order to logically connect different blobs characterized by similar optic flow values, and therefore by a probably very similar movement in the scene. Adjacent regions, characterized by a similar behaviour of the optic flow, are joined in a single region having as direction of motion the one determined by the optic flow common to all the subregions individuated. The results are visible in fig. 3. All the previous processing steps are applied to subsequences involving three consecutive frames. The reason is that moving targets in a harbour are usually so slow that we can use three frames without loosing too much information. On the other side, averaging results on more frames makes our procedure less sensible to noise.

2.4. Blob tracking and deformation analysis

The tracking process is aimed first to find a correspondence among these *motion windows*, and the different moving objects contained. To this aim, we evaluate the *global similarity* of each window pair: if $\mathbf{a} = (a_1, a_2, \ldots, a_k)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_l)$ are two lists of blobs ordered with decreasing areas, a comparison is made between them (and the associated windows), by means of the simple cost function:

$$f_c = \min_{M=1,\ldots,N} \left(\sum_{i,j}^{M} \sqrt{|\mathbf{c}_i - \mathbf{c}_j|^2} + |A_i - A_j| \right) \qquad (4)$$

where c_i is position of centroid of the i-th blob, A_i the value of its area, $M = \min(k, l)$, and N the number of different choices of M elements among those in $\mathbf{a}$ and $\mathbf{b}$.

However, tracking objects in a complex environment means dealing with partial or total occlusions: a solution for these problems is proposed in [6]: an object going out from the part of the scene tracked by the sensor is not immediately forgotten. Instead, its probability to be present is decreased. Only when this probability decreases under a given threshold, the object is no more considered. Therefore, to the i-th tracked blob is associated a vector $\vec{v}_i = (A_i, \mathbf{c}_i, p_i)$ containing not only its area A_i and centroid position $\mathbf{c}_i$ but also its probability p_i to be in the scene. The last value is updated following the formula

$$p_i \rightarrow p_i + -\alpha \qquad (5)$$

where α is a small fixed value, p_i is limited to [0,1], and the sign in the formula is chosen according to the permanence (+) or the missing (-) from the scene. So, if we have a ship disappearing from the scene, and later reappearing, it can be recognized as the same and correctly tracked, and similarly when dealing with partial occlusions, making the system able to track occlusions.

2.5. 3D Motion Estimation by Sequence Analysis

The successive step in the algorithm is the extraction of 3D motion and position parameters of each target: 2D information, in fact, are not sufficient to control a real environment, since different (and possibly colliding) trajectories may give the same or similar projection on the image plane.

3D motion extraction can be computed by means of theoretically precise, local closed form expressions found in literature (e.g. [8]). However, these procedures present strong ambiguities whereas noisy data is provided. It has been instead proved that a global solution of the motion equations could provide, via an explicit depth computation and an iterative procedure, a much more robust method [10]. The proposed iterative procedure is based on a least square method, able to estimate the actual velocities of the bodies in the scene. With good data this algorithm is strongly convergent provided the initial estimate of motion parameters is not too far wrong. It has the disadvantage that –even in the case of perfect data– it may not converge to the right solution. This algorithm, directly implemented on the image sequence of a ship moving in our environment, does not give good results. More than half of the points do not converge to the same velocity vector on the image plane. The algorithm works well, instead, on synthetic images.

Our improvement to this algorithm starts from the observation that it allows, even if the object tracked is rigid, different calculations for different parts of the same object. This choice may induce errors due to the noise superimposed to the actual image. So, if a depth value in the neighbourhood of the examined pixel is wrong, it may affect the correct calculation of the motion parameters of all the neighbourhood. Following the same notation:

$$u = (Wx - U)z - B(1 + x^2) + Cy + Axy \qquad (6)$$

$$v = (Wy - V)z - A(1 + y^2) - Cy - Bxy \qquad (7)$$

where X, Y, Z are the cohordinate of a point in the real world projecting to a pixel x, y in the image ($x = X/Z$, $y = Y/Z$). Moreover, $z = 1/Z$, u and v are the x and y components of the optic flow, U, V and W are the components of the translational motion and A, B and C those of the angular one along the X, Y and Z axis respectively.

Equation (6,7) can be solved by an iteration procedure:

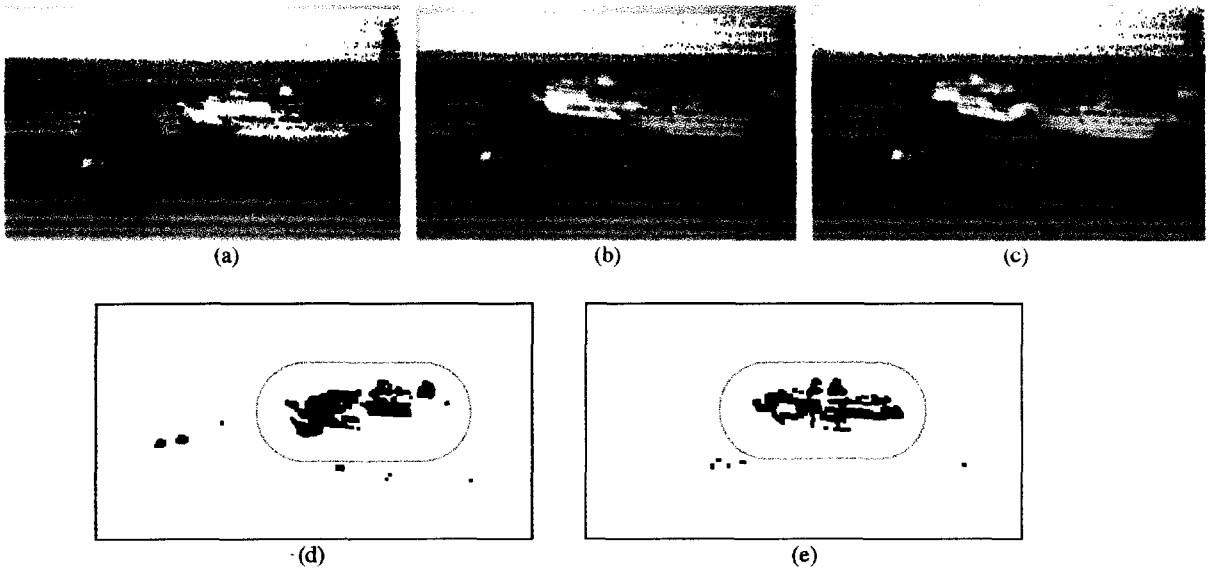

Figure 4. (a-c) A sequence of harbour scene showing a moving ship as well as other objects: a partial occlusion of the target is also observed; (d-e) the moving target tracked by the proposed procedure.

1. an inital estimate of the motion parameter is introduced for a small (5x5) neighbourhood of the starting pixel;
2. the cohordinate z for each pixel is estimated through a least square procedure on the equation (6,7);
3. new motion parameters are recomputed starting from this values of z;
4. goto 2 until stable.

To minimize noise effects, we choose for all the points of a given objects the same value of z, with two simple possiblities based on the z values computed for each pixel in the first iteration: *(a)* the z value in the central pixel; *(b)* the average z value. The first choiche is the straightforward one, and makes the algorithm converge on almost all the points. By means of this choice we are able to retrieve with good accuracy the 3D motion parameters from our sequences.

3. EXPERIMENTAL RESULTS

The whole procedure has been tested on two different frame sequences, the first one taken in the harbour of Livorno, and the second one in the harbour of Civitavecchia, both in Italy. Since Italy has a large amount of coast areas and harbours, there is a consistent interest in developing a system like the one we propose. Results are very good, either on the qualitative side (tracking of moving ships, controlling of the environment), either on the quantitative one (retrieving the 3D position of each target). Frames from the Civitavecchia sequence are shown as processing steps throughout the paper and here we report in fig. 4 three frames with the moving regions (blobs) obtained after all the segmentation steps. Similar results have been obtained for the Livorno sequence.

REFERENCES

[1] J.L. Crowley and Y. Demazeau, "Principles and techniques for sensor data fusion", *Signal Processing*, Vol. 32, No. 1-2, pp. 5-27, 1993.

[2] J.R. Bergen, P.J. Burt, R. Hingorani and S. Peleg, "A three-frame algorithm for estimating two-component image motion", *IEEE Trans. Patt. Anal. Machine Intell.*, PAMI-14, No. 9, pp. 886-896, Sept. 1992.

[3] Z. Zhang and O.D. Faugeras, "Three dimensional motion computation and object segmentation in a long sequence of stereo frames", *International Journal of Computer Vision*, Vol. 7, pp. 211-241, 1992.

[4] F. Leymarie and M.D. Levine, "Tracking deformable objects in the plane using an active contour model", *IEEE Trans. Patt. Anal. Machine Intell.* , Vol.PAMI-15, No. 6, pp. 617-634, June 1992.

[5] D.G. Lowe, "Robust model-based motion tracking through the integration of search and estimation", *International Journal of Computer Vision*, Vol. 8, pp. 113-122, 1992.

[6] J.P. Gambotto, "Segmentation and interpretation of infrared image sequences", in *Advances in Computer Vision and Image Processing*, JAI Press Inc., 1988.

[7] B.K. Horn and B.G. Schunck, "Determining optical flow", *Artificial Intelligence*, Vol. 17, pp. 185-203, 1993.

[8] H. Nagel and W. Enkelmann, "An investigation for estimation of displacement vector fields from image sequences", *IEEE Trans. Patt. Anal. Machine Intell.*, Vol. PAMI-8, No. 9, pp. 565-593, Sept. 1986.

[9] B.G. Shunck, "Image flow segmentation and estimation by constraint line clustering", *IEEE Trans. Patt. Anal. Machine Intell.*, Vol. PAMI-9, No. 10, pp. 1010-1027, Oct. 1989.

[10] G. Scott, "Global and local interpretation of moving images", *Pitman Publishing*, London, 1988.

[11] R.M. Haralick and L.G. Shapiro, "Computer and robot vision", *Addison Wesley*, New York, 1993.