ELSEVIER

Journal of Systems Architecture 43 (1997) 317–325

JOURNAL OF SYSTEMS ARCHITECTURE

A real-time oriented system for vehicle detection [1]

Massimo Bertozzi *, Alberto Broggi [2], Stefano Castelluccio [2]

Dipartimento di Ingegneria dell'Informazione, Università di Parma, I-43100 Parma, Italy

Abstract

Starting from the analysis of the specific requirements of automotive on-board equipments, this work presents a system for vehicle detection in a monocular video sequence. The system is composed of a pipeline of two different engines: PAPRICA, a massively parallel architecture for the efficient execution of low-level image processing tasks, improved by the integration of a specific feature for direct data I/O; and a traditional serial architecture running medium-level tasks aimed to the detection of the vehicles' position in the sequence. A preliminary version of the system has been demonstrated on MOB-LAB land vehicle, while a fully optimized one is currently under tuning in laboratory.

Keywords: Vehicle detection; Parallel architecture; Serial architecture

1. Introduction

This work presents a vision-based system for vehicle detection in images acquired from a moving vehicle. The main target of this work is to determine a system (hardware and software) able to detect obstacles (other vehicles) on rural roads running sufficiently fast to face the extremely hard real-time constraints imposed by automotive applications.

Since the main goal is to reach real-time performance, a system based first on the detection of edges and then on their interpretation would require an extremely powerful hardware engine. Moreover, the match of each edge against a sufficiently high number of different models (vehicles, trees, houses,...) would require a computational power that traditional

* Corresponding author. Email: bertozzi@ce.unipr.it.

[1] This work was partially supported by CNR Progetto Finalizzato Trasporti under contracts 93.01813.PF74 and 93.04759.ST74.

[2] Email: {broggi,castellu}@ce.unipr.it.

PII S1383-7621/0165-6074(96)00106-3

architectures cannot provide at a low cost. The cost of the system is in fact a key parameter in automotive applications, where the cost of the road- and obstacles-detection system should be comparable to that of other on-board equipments. It is for this reason that the use of *general-purpose* massively parallel systems (for example the Warp [5,1] or the MasPar MP-2 [13] implemented on Navlab [17] at CMU [4,10]), although highly indicated for the implementation of the low-level portion of the processing, has not been considered.

In the literature few different approaches have been considered for automotive obstacle detection, each of them relying on different hardware characteristics:

- systems based only on vision [11,6,12,7]; and
- systems where vision is aided by other sensors [20,9].

Obviously the fusion of data coming from many different sensors makes the detection of obstacles easier and more reliable, but unfortunately the cost of the system is proportional to the number of different sensors installed on the vehicle. Moreover, the use of *active* sensors, such as a range finder, causes a pollution of the environment. Conversely the approach discussed in this work is based only on *passive* sensors (cameras); furthermore, an extension to this system, consisting on the integration of another camera for stereo vision, is currently under study and evaluation.

Moreover, the system must be able to support the real-time processing of image sequences. Thus, for the maximization of the performance, it should overlap the processing of the current frame with the acquisition of the following frame from the camera.

For this reason a specific hardware architecture has been considered. It is divided into two concurrent engines working in pipeline:

- a massively parallel one, devoted to the data acquisition and low-level processing, optimized for the efficient handling of binary images; and
- a traditional serial architecture, based on a Sparc processor, devoted to the medium- and high-level steps of the processing. This subsystem is also in charge of sending warnings to the driver, as well as controlling the whole system.

Although the selected computer architecture matches the generic image processing steps (low-level vision implemented on *massively parallel* systems, while medium- and high-level steps on traditional *serial* systems), real-time performance can be obtained only through the use of simple and effective algorithms.

Since a generic obstacle (vehicle) has a rectangular shape, the goal of the system described in this work is to extract from the incoming image all the possible rectangles that may represent vehicles on the path. The preliminary image filterings and the detection of the features of interest (corners of the rectangles) are performed by the massively parallel system PAPRICA, by means of a mathematical morphology-based computational paradigm. Conversely, the following selection of the meaningful corners, the grouping of 4 corners representing a rectangle (or trapezoid), and the final validation of the determined polygons are performed on the serial system.

Due to its simplicity, the following high-level interpretation of the results is currently performed by the same sequential system. In fact, in the current version of the system only warnings are issued to the driver through a set of leds, and no vehicle control is left to the vision system as well as no navigation and planning processes are currently integrated: a simple led-based control-panel is the output device currently installed on MOB-LAB, the test land vehicle integrating the results of the activities of the Italian research units within the PROMETHEUS project.

The following section justifies the choice of the hardware platforms and details the two engines; Section 3 presents the algorithm for vehicle detection, while Section 4 discusses the performance of the system in terms of both computational time and output quality; finally Section 5 ends the paper with some concluding remarks.

2. The computing architecture

Since the main bottlenecks of real-time vision-based systems are the high I/O bandwidth required for the data acquisition [15] and the high computational power needed to process a great amount of pixels, the chosen computing architecture is composed of a massively parallel system (PAPRICA) with the capability to acquire images directly into its local memory.

The PAPRICA system [3,2,8] has been designed as a specialized coprocessor to be attached to a general purpose host workstation, which acts as the medium-level processor. In the current implementation, the PAPRICA board (a single 6U VME board) is connected to a Sparc-based workstation on a standard VME bus.

PAPRICA comprises 5 major functional parts:

(1) the Program Memory (storing up to 256 k instructions),
(2) the Image Memory (up to 3 MBytes),
(3) the Processor Array (a 2D matrix of 256 Processing Elements (PEs)),
(4) the Frame Grabber device, and
(5) the Control Unit.

The current prototype of the PA is composed of an array of 4×4 ICs, each of them containing a sub-array of 4×4 PEs. The PA is then a 16×16 square matrix of 1-bit PEs each one with full 8-neighbors connectivity; each PE has an internal memory composed of 64 bits; a single 16-bit memory fetch takes 150 ns, while the PA cycle time is 250 ns.

The frame grabber device, interfaced directly to PAPRICA image memory, has two different operating modes: single field or full frame. In the first mode only a single image field is acquired (512×256 pixels, 8 bit/pixel corresponding to 256 gray levels), and the acquisition time is 20 ms, while the synchronization time can range from 0 to 20 ms; in the second mode a full frame (2 fields, 512×512 pixels, 8 bit/pixel) is acquired: the acquisition time is 40 ms, while the synchronization time ranges from 0 to 20 ms.

Fig. 1 presents a block diagram of the entire hardware system.

3. The vehicle detection algorithm

3.1. The low-level processing

The incoming image (a single field, 512×256 pixel) is acquired and loaded directly into PAPRICA image memory. PAPRICA system provides a mechanism for the emulation of pyramidal interconnections between processors without any computational overhead: the image is in fact undersampled to obtain a working resolution of 256×256 pixels. Then an average operation on a 3×3 neighborhood is performed prior to the gradient filter in order to dimin-

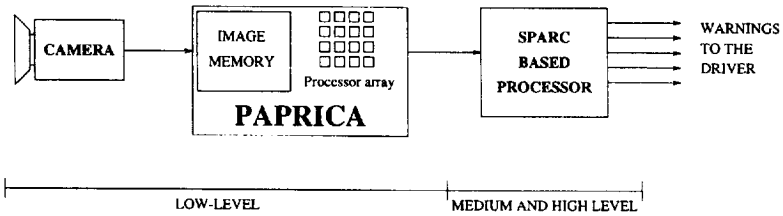


Fig. 1. Block diagram of the hardware system.

ish the influence of noise spikes. Then a threshold is applied for the binarization of the resulting image: in this case, due to the specific following processing, an accurate determination of the threshold is not necessary. In fact, a low threshold is used in order to keep a high amount of edges; the edges which do not correspond to rectangles will be deleted by the following medium-level step.

With this choice, the binarized image consists of wide edges, but a thinning filter [16] is not recommended: in fact, although reducing the lines' width, thus leading to a smaller number of black pixels (and thus to a faster subsequent processing), it would not permit to detect corners with a simple and traditional pattern matching technique. Fig. 2(b) shows the binarized version of Fig. 2(a). The low-level processing ends with the determination of 4 binary images indicating the presence of corners: Fig. 3 shows the four patterns used; these are not symmetrical; in fact, experimental tests have demonstrated that longer horizontal lines allow to detect also non perfectly regular rectangles (trapezoids). Tests on different road conditions have shown that the horizontal lines are more important and less noisy than vertical ones.

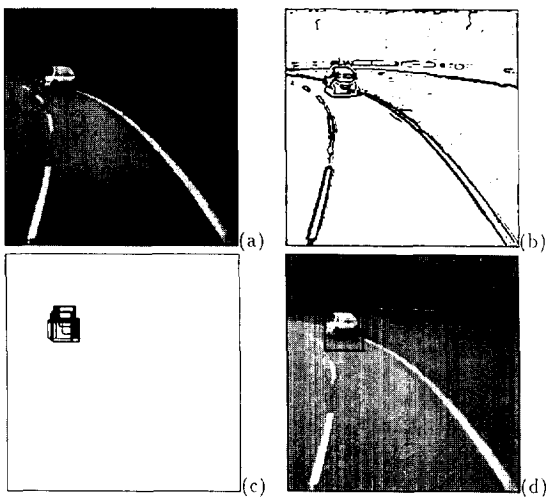


Fig. 2. (a) Original image, (b) binarized image, (c) rectangles detection, (d) final result.

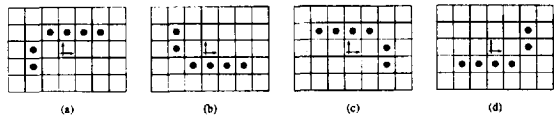


Fig. 3. Patterns used to detect the 4 corners of a rectangle.

3.2. The medium-level processing

As mentioned in the Introduction, obstacles' edges are here assumed as rectangles. The main goals of the medium-level computation are the following:

3.2.1. Detection of rectangles

The algorithm scans the whole image looking for upper-left corners (Fig. 3(a)). For each detected corner, it looks for a matching lower-left one (Fig. 3(b)), then for a lower-right one (Fig. 3(d)) and finally for an upper-right one (Fig. 3(c)). It is possible to avoid many computations reducing the number of rectangles before their complete construction: the side length can range only in a precise interval. This interval depends on the obstacle position within the image: due to the perspective effect induced by the specific acquisition conditions, rectangles detected in the lower part of the image correspond to vehicles close to the camera. Thus their size should be larger than the size of similar objects detected in the upper part of the image. Moreover, real obstacles are characterized by a specific aspect-ratio interval: this allows to reject a large amount of rectangles, providing a speed-up in the following steps of the processing. The rectangle construction procedure is sketched in the following pseudo-code:

```
for every upper left angle
  for every matching lower left angle
    lx=vertical side length
    if min_dim < lx < max_dim
      for every matching lower right angle
        ly=horizontal side length
        if (min_dim < ly < max_dim) and (min_ratio < lx/ly < max_ratio)
          for every matching upper right angle
            if (upper right angle) matches (upper left angle)
              then a rectangle is detected
          end for
      end for
  end for
end for
```

3.2.2. Rectangles validation

The validation procedure consists of checking, in the binarized image, for the presence of over-threshold pixels in the positions corresponding to the rectangle's sides. A side is considered valid *iff* at least half of the pixels representing it are actually present in the binarized image. Rectangles with at least one not valid side are rejected. The rectangle validation procedure is sketched in the following pseudo-code:

```
for every valid rectangle
  for every side
    if side is not 'present' in the binarized image
      then mark rectangle as not valid
  endfor
endfor
```

Obviously, in order to speed-up the computation, the validation procedure is interleaved with the rectangle construction process: conversely, for sake of simplicity of the presentation, the two steps have been here discussed separately.

3.2.3. Fusion step

Normally, for each obstacle the algorithm detects more than one rectangle, each one with small differences in position and size, as shown in Fig. 2(c). Although increasing the computational time (a larger number of rectangles must be detected and validated), this redundancy offers the advantage of reducing the risk of missing obstacles. Thus, a fusion algorithm is used to reduce the amount of redundant information, simplifying the following result interpretation. If two or more rectangles are sufficiently overlapped, they are replaced by a larger one, as in Fig. 2(d).

The fusion step is sketched in the following pseudo-code:

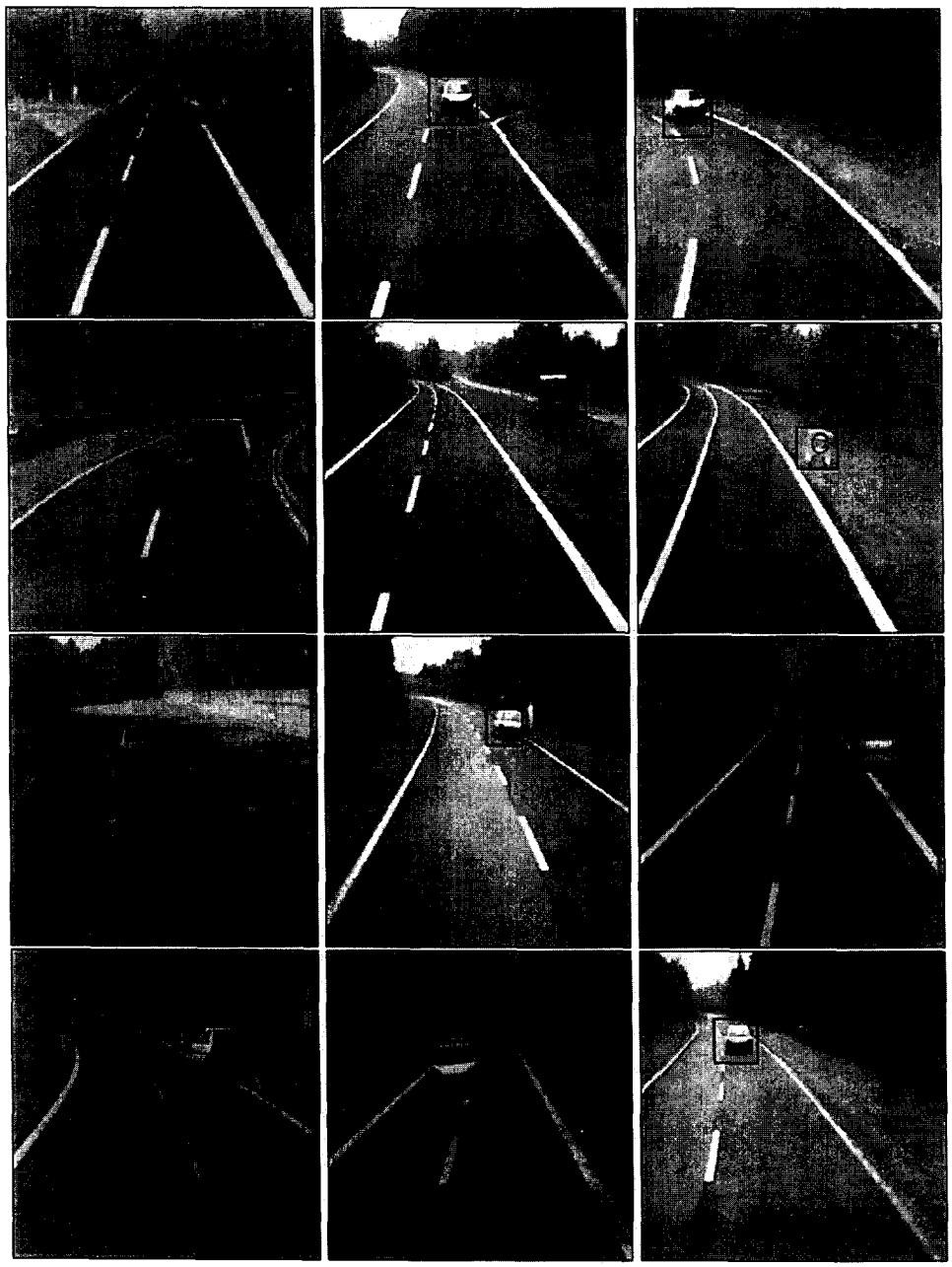


Fig. 4. Some examples showing the detection of obstacles with a rectangular shape.

```
for r1=every valid rectangle
  for r2=every different valid rectangle
    if overlap_ratio(r1, r2) > max_overlap_ratio
      new_rectangle → position=weighted_average(r1 → position, r2 → position)
      if (r2 is included in r1)
        new_rectangle → dim=r2 → dim
      else
        new_rectangle → dim=r1 → dim+r2 → dim+distance(r1, r2)
        mark r1 and r2 as not valid
  end for
end for
```

The overlap ratio of rectangle r1 over r2 is defined as the ratio between the area of the intersection between r1 and r2 and the area of r1. The weighted average is computed using the size of rectangles as weight. Since generally all the rectangles are heavily overlapped the fusion step is reduced to the growing of a single rectangle, including all the others.

The presented algorithm has been also extended to handle image sequences. In this case a sensible speed-up is achieved by the dynamic reduction of the area of interest [19,14], namely the region where the obstacle is supposed to be in the following frame of the sequence. It is reduced to a single image window where the obstacles have been previously detected: these windowing techniques [18,7] allow a fast tracking of the moving obstacle. Currently the region of interest is determined by taking the smaller bounding box including all the obstacles.

However, periodically or when no obstacles are found, the entire image is analyzed, in order to avoid the risk of missing new obstacles.

4. Performance analysis

The whole process has been tested on images representing several extra-urban double-lane roads, with painted road markings. Although the specific processing does not relay on the presence of road markings, it has been tested on structured roads since a different system, based on road markings detection, runs concurrently for the identification of the lane boundaries. Unfortunately road markings represent a noisy feature that slows down the medium-level phase: in fact the number of detected corners is increased as well as the number of required match-

Table 1
Timings for vehicle detection in different conditions

Operation	Road conditions	Time	Speed
Low level	–	265 ms/frame	3.77 Hz
Medium level	Straight road with one car	1.21 s/frame	0.82 Hz
	Straight road without cars	0.631 s/frame	1.58 Hz
	Curve road with one car	1.97 s/frame	0.50 Hz
	Curve road without cars	0.576 s/frame	1.74 Hz
	Junction with cars	1.63 s/frame	0.61 Hz
	Junction without cars	2.33 s/frame	0.43 Hz
	Mixed route with one car	1.38 s/frame	0.72 Hz

ings. Nevertheless, generally the rectangles generated by a successful marching do not comply with the size requirement mentioned in Section 3.2. For this reason, the final amount of valid rectangles is not affected by the presence of road markings, although sensibly slackened.

The test images has been acquired from the MOB-LAB land vehicle during the final demonstration of the PROMETHEUS Eureka Project at Mortefontaine track, Paris, October 1994. Fig. 4 presents a few results of the processing of images representing a vehicle on the road: the resolution of the incoming gray-level images is 256×256, 8 bit/pixel.

The algorithm has been tested in different conditions: without obstacles, on straight and curved roads, in different illumination conditions, on junctions. The more critical behaviour corresponds to the presence of shadows on the path, inducing the appearance of a considerable number of edges in the binarized image.

Table 1 presents the timings of the low and medium level processings: the former does not depend on the specific image, while the latter is strongly influenced by the amount of extracted features (corners). Table 1 shows that on straight and curve roads without vehicles the algorithm runs faster than in the same conditions but with vehicles on the path. This is due to the overhead caused by the rectangles validation phase. On the other hand, on particular kinds of images with a lot of details (edges) the number of rectangles grows, thus requiring a longer processing. In this case the presence of one vehicle decreases the computational time thanks to the windowing techniques.

5. Conclusion

In this work a system (hardware and software) for vehicle detection has been presented, focusing on the specific requirements for automotive integration: the cost of the system must be kept low while the computational power must be sufficiently high to process images in real-time. A two-engine pipeline architecture is used for the concurrent low-level and medium-level processing of incoming images: a massively parallel system (interfaced directly to the acquisition sensor) and a traditional sequential one. A few examples are discussed, showing the detection of obstacles in different road conditions. An optimised version is currently under development and testing in laboratory and will be integrated on the new PAPRICA prototype which is going to be installed on MOB-LAB.

References

[1] M. Annaratone, E. Arnould, T. Gross, H. Kung and J. Webb, The Warp computer: Architecture, implementation and performance, *IEEE Trans. Comput.* 36 (12) (1987) 1523–1538.

[2] A. Broggi, G. Conte, F. Gregoretti, C. Sansoè and L.M. Reyneri, The PAPRICA massively parallel processor, in: *Proc. MPCS – IEEE Internat. Conf. on Massively Parallel Computing Systems*, Ischia, Italy (IEEE Computer Society – EuroMicro, 1994) 16–30.

[3] A. Broggi, G. Conte, F. Gregoretti, C. Sansoè and L.M. Reyneri, The evolution of the PAPRICA system, *Integrated Computer-Aided Engineering J.*, special issue on massively parallel computing 4 (1) (1995).

[4] J. Crisman and C. Thorpe, SCARF: A color vision System that tracks roads and intersections, *IEEE Trans. Robotics Automation* 9 (1) (1993) 49–58.

[5] J.D. Crisman and J.A. Webb, The Warp machine on Navlab, *IEEE Trans. Pattern Analysis Machine Intelligence* 13 (5) (1991) 451–465.

[6] E.D. Dickmans and B.D. Mysliwetz, Recursive 3-D road and relative ego-state recognition, *IEEE Trans. Pattern Analysis Machine Intelligence* 14 (1992) 199–213.

[7] V. Graefe and K.-D. Kuhnert, Vision-based autonomous road vehicles, in: I. Masaki, ed., *Vision-based Vehicle Guidance* (Springer, Berlin, 1991) 1–29.

[8] F. Gregoretti, L.M. Reyneri, C. Sansoè, A. Broggi and G. Conte, The PAPRICA SIMD array: Critical reviews and perspectives, in: L. Dadda and B. Wah, ed., *Proc. ASAP'93 – IEEE Computer Society Internat. Conf. on Application Specific Array Processors*, Venezia, Italy (IEEE Computer Society – EuroMicro, 1993) 309–320.

[9] A. Jain, T. Newmann and M. Goulish, Range-intensity histogram for segmenting LADAR images, *Pattern Recognition Lett.* 13 (1992) 41–56.

[10] T.M. Jochem and S. Baluja, A massively parallel road follower, in: M.A. Bayoumi, L.S. Davis and K.P. Valavanis, eds., *Proc. CAMP'93 – Computer Architectures for Machine Perception*, New Orleans (IEEE Computer Society, 1993) 2–12.

[11] T.M. Jochem, D.A. Pomerleau and C.E. Thorpe, MANIAC: A next generation neurally based autonomous road follower, in: *Proc. Internat. Conf. on Intelligent Autonomous Systems: IAS-3*, Pittsburgh, PA, 1993.

[12] S.K. Kenue and S. Bajpayee, LANELOK: Robust line and curvature fitting of lane boundaries, in: *Proc. SPIE – Mobile Robots VII*, Vol. 1831 (1993) 491–503.

[13] MasPar Computer Corporation, Sunnyvale, CA, *MP-1 Family Data-Parallel Computers*, 1990.

[14] H. Neumann and H. Stiehl, Toward a computational architecture for monocular preattentive segmentation, in: R.G. Hartmann and G. Hauske, eds., *Parallel Processing in Neural Systems and Computers* (North-Holland, Amsterdam, 1990).

[15] D.A. Pomerleau, *Neural Network Perception for Mobile Robot Guidance* (Kluwer Academic Publishers, Boston, 1993).

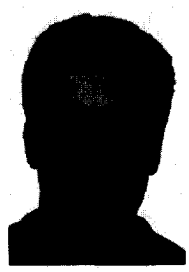
[16] A. Rosenfeld, A characterization of parallel thinning algorithms, *Inform. Control* 29 (1975) 286–291.

[17] C.E. Thorpe, ed., *Vision and Navigation. The Carnegie Mellon Navlab* (Kluwer Academic Publishers, 1990).

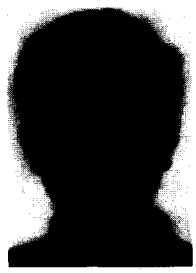
[18] M.A. Turk, D.G. Morgenthaler, K.D. Gremban and M. Marra, VITS – A vision system for autonomous land vehicle navigation, *IEEE Trans. Pattern Analysis Machine Intelligence* 10 (3) (1988).

[19] J.M. Wolfe and K.R. Cave, Deploying visual attention: the guided model, in: A. Blake and T. Troscianko, eds., *AI and the Eye* (1990) 79–107.

[20] M. Xie, L. Trassoudaine, J. Alizon and J. Gallice, Road obstacle detection and tracking by an active and intelligent sensing strategy, *Machine Vision and Applications* 7 (1994) 165–177.



Massimo Bertozzi, born 1966, received the degree in electronic engineering from University of Parma, Italy in 1994, discussing a thesis on Petri nets. He is currently a Ph.D. candidate in Information Technology at Dipartimento di Ingegneria dell'Informazione of University of Parma. His research interests are in parallel processing and architectures, and their application to computer vision problems.



Alberto Broggi received the Dr. Ing. degree in electronic engineering from the University of Parma, Italy, in 1990 and the Ph.D. degree in 1994 from the same University, working on a coordinated project with the Polytechnic of Turin, Italy. In 1994 he joined the Dipartimento di Ingegneria dell'Informazione (Faculty of Engineering) of the University of Parma as a researcher (faculty position). His main interests include the development of low-cost vision-based hardware architectures and algorithms for real-time image processing for vehicle navigation, and the study of the mathematical morphology computational model and its application to special purpose VLSI parallel architectures. He acted as a Guest Editor for a special issue of the international journal (*Real-Time Imaging Journal*) in the area of real-time vision computer architectures; he is currently coordinating the "Engineering Complex Computer Systems" Minitrack (within the Advanced Technology Track) of the 30th IEEE Hawaii International Conference on System Sciences (HICSS-30). Recently he was invited to join the Editorial Board of Real-Time Imaging Journal as an Area Editor in the field of "Computer Architectures for Real-Time Imaging".

Stefano Castelluccio received the Dr. Eng. degree in electronic engineering from Università di Parma, Italy, in 1995, discussing a thesis on parallel digital image processing. He is currently co-operating with Research & Development staff of RIA1 Vacuum S.p.a. in the field of digital mass spectrum analysis. His research interests include parallel processing and architectures, and their application to computer vision problems.