Associazione Italiana per l'Intelligenza Artificiale

Atti del Workshop su
Proceedings of the Workshop on

Intelligenza Artificiale, Visione e Pattern Recognition

Artificial Intelligence, Vision and Pattern Recognition

a cura di Antonio Chella
Donato Malerba

Lunedì, 24 Settembre 2001
Dipartimento di Informatica (CAMPUS)
UNIVERSITÀ DEGLI STUDI DI BARI

Atti del Workshop su
Proceedings of the Workshop on

Intelligenza Artificiale, Visione e Pattern Recognition

Artificial Intelligence, Vision and Pattern Recognition

Bari, 24 Settembre 2001
Dipartimento di Informatica, Università di Bari

a cura di Antonio Chella
Donato Malerba

Comitato Organizzatore
Antonio Chella, Università di Palermo e CERE-CNR, Palermo
Donato Malerba, Università di Bari

Comitato Scientifico
Giovanni Adorni, Università di Parma
Virginio Cantoni, Università di Pavia
Antonio Chella, Università di Palermo e CERE-CNR, Palermo
Luigi P. Cordella, Università di Napoli
Vito Di Gesù, Università di Palermo
Arcangelo Distante, IESI-CNR, Bari
Donato Malerba, Università di Bari
Vito Roberto, Università di Udine
Guido Tascini, Università di Ancona

Sito web
http://www.di.uniba.it/~aiia/ws1/

Prefazione

Un agente intelligente che opera nel mondo reale, sia esso un robot o un sistema software, deve poter percepire e interagire con oggetti in movimento, persone, ed altro, in ambienti non strutturati e non predicibili. Per eseguire il proprio compito in maniera appropriata, l'agente deve essere in grado di *comprendere* il proprio ambiente di lavoro, nel senso che deve poter generare una descrizione interna ad alto livello delle proprie percezioni.

La generazione di questa descrizione richiede elaborazioni direttamente connesse ai dati sensoriali, mediante tecniche tipiche del *Pattern Recognition*; richiede inoltre la generazione di ricostruzioni 2D e 3D, mediante tecniche tipiche della *Visione Artificiale*, e richiede infine elaborazioni di natura simbolica tipiche dell'*Intelligenza Artificiale*.

L'obiettivo ambizioso di creare un agente in grado di comprendere il proprio ambiente può quindi essere conseguito solo mediante uno sforzo congiunto di tecniche, di metodologie e di esperienze relative a diversi settori avanzati della ricerca.

Il workshop si colloca tra le iniziative del **Congresso AI*IA 2001**; il suo scopo è quello di riunire i ricercatori italiani impegnati nei settori dell'Intelligenza Artificiale, della Visione e del Pattern Recognition ed interessati ad esplorare le possibilità di interazione e di *ibridizzazione* delle varie teorie e metodologie, al fine di porre le basi per nuovi agenti intelligenti che interagiscano ad alto livello con il mondo che li circonda.

Gli argomenti del workshop hanno riguardato principalmente gli approcci di natura metodologica e applicativa relativi ai seguenti temi:

- Tecniche di IA nel Pattern Recognition
- Tecniche di IA nella Visione Artificiale
- Architetture cognitive di Visione Artificiale
- Generazione di rappresentazioni simboliche a partire da dati sensoriali
- Tecniche di ragionamento su dati sensoriali
- Sistemi simbolici per la visione artificiale
- Modelli cognitivi per il fuoco di attenzione

L'auspicio è che il workshop, anche attraverso una discussione vivace, contribuisca a rafforzare le collaborazioni tra i ricercatori delle aree interessate.

Un ringraziamento particolare va al Comitato di Programma del workshop, a quanti hanno inteso partecipare inviando un contributo e ai relatori invitati che hanno accettato di contribuite al confronto scientifico sui temi del workshop. Uno speciale ringraziamento va all'ideatrice dell'evento, Floriana Esposito.

Antonio Chella
Donato Malerba

Bari, Luglio 2001

Preface

An intelligent agent which operates in the real world, be it a robot or a software system, should be able to perceive and interact with moving objects, people, etc., in unstructured and unpredictable environments. To accomplish its task, the agent must be able to understand its workspace, meaning that it should be able to generate an internal high-level description of what is perceived.

Generating this description requires sensory data processing by means of Pattern Recognition techniques. It also requires the generation of 2D and 3D reconstructions by means of Computer Vision techniques and lastly it requires symbolic computation which is typically provided by Artificial Intelligence techniques.

Building an agent able to understand its environment is a demanding task that can be achieved only by means of the joint application of techniques, methodologies and experiences from several advanced research fields.

The workshop is among the events of the **Congress AI*IA 2001**. Its aim is to encourage the cooperation between Italian researchers working in the fields of Artificial Intelligence, Computer Vision and Pattern Recognition, who are interested in exploring opportunities for cross-fertilization of theories and methodologies and laying foundations for new intelligent agents that interact with the surrounding world at a high level.

Papers presented at the workshop mainly concern methodologies and applications for the following topics:

- AI Techniques in Pattern Recognition
- AI Techniques in Computer Vision
- Cognitive Architectures in Computer Vision
- Generation of symbolic representations from sensory data
- Reasoning Techniques on sensory data
- Symbolic Systems for Computer Vision
- Cognitive Models for Focus of Attention

We hope that the workshop may contribute to strengthen the cooperation among researchers of the afore-mentioned disciplines.

We wish to express our thanks to the Scientific Committee of the workshop, all who submitted papers for the presentation, and the invited speakers who have agreed to contribute to the scientific debate on the workshop topics. Special thanks are due to the real 'brain' behind this event, Floriana Esposito.

Antonio Chella
Donato Malerba

Bari, July 2001

Indice
Table of Contents

Relazione Invitata
Invited Talk

Articoli
Regular Papers

Programma del Workshop
Schedule

8:30	–	9:10	**Registrazione / *Registration***
9:10	–	9:20	**Apertura / *Opening remarks*: ** *A. Chella and D. Malerba*
9:20	–	10:10	**Relazione Invitata / *Invited talk*: ** *W. Burgard* Experiences with two Interactive Museum Tour-guide Robots
10:10	–	10:50	**Architetture per Robotica Mobile / *Architectures for Mobile Robotics*** *A. Chella, M. Cossentino, I. Infantino, R. Pirrone* A Vision Agent in a Distributed Architecture for Mobile Robotics *E. Menegatti, E. Pagello* Cooperative Distributed Vision for Mobile Robots
10:50	–	11.10	**Pausa caffè / *Coffee break***
11:10	–	13:10	**Tecniche di IA per la Navigazione di Robot /*AI Techniques for Robot Navigation*** *G. Cicirelli, T. D'Orazio, A. Distante* Vision-Based Behaviors for Autonomous Mobile Robot Navigation *P. Remagnino, N. Monekosso, O. Hatoum, Y. Halabi* Learning Robot Navigation in a Virtual World *A. Micarelli, E. Sangineto, G. Sansonetti* Natural Indoor Landmark Recognition for Robot Navigation *G. Adorni, L. Bolognini, S. Cagnoni, M. Mordonini, A. Sgorbissa* Designing an Omnidirectional Vision Sensor for Autonomous Navigation *A. Chella, R. Sorbello, S. Vitabile* A Bayesian Approach for Mobile Robot Navigation in Time-Variable Environments *A. Chella* Anticipative Representation in Robot Vision
13:10	–	14:40	**Pausa Pranzo / *Lunch break***
14:40	–	15:30	**Relazione Invitata / *Invited talk*: ** *S. Sclaroff* (Title to be defined)
15:30	–	16:10	**Tecniche di IA per l'Analisi di Immagini /*AI Techniques for Image Analysis*** *R. Cucchiara, C. Grana, M. Piccardi* Iterative fuzzy Clustering for Detecting Regions of Interest in Skin Lesions *G. Tascini, L. Regini, P. Puliti, P. Rogani* Genetic Evolution of a LVQ Classifier for Image Sequence Analysis
16:10	–	16:30	**Pausa caffè / *Coffee break***
16:30	–	17:10	**Architetture per l'elaborazione dell'immagine /*Architectures for Image Processing*** *A. Gentile, J.L. Cruz-Rivera, D. Scott Wills, F. Sorbello* Real-Time, Low Level Image Processing on SIMPil – An Embedded SIMD Architecture *A. Degli Esposti, A. Micarelli, E. Sangineto, G. Sansonetti* Multimedia Content Based Information Retrieval
17:10	–	17:30	**Architetture Cognitive in Visione /*Cognitive Architectures in Computer Vision*** *V. Roberto* Comprensione del Movimento e Apprendimento Percettivo
17:30	–	18:30	**Tecniche di IA per l'analisi di immagini di documento /*AI Techniques for Document Image Analysis*** *M. Aiello* Document Image Analysis via Model Checking *L. Caponetti, G. Castellano, A.M. Fanelli* Document Region Segmentation and Classification Using a Neuro-Fuzzy System *O. Altamura, F. Esposito, D. Malerba* Learning to Correct the Layout extracted from Document Images
18:30	–	18:40	**Discussione e Note conclusive / *Discussion and closing remarks***

Experiences with two Interactive Museum Tour-guide Robots

Wolfram Burgard

Institut für Informatik, Albert-Ludwigs-Universität Freiburg
Universitätsgelände Flugplatz, 79110 Freiburg
burgard@informatik.uni-freiburg.de
http://www.informatik.uni-freiburg.de/~burgard

Abstract. In this talk we will describe the software architecture of an autonomous, interactive tour-guide robot. We present a modular, distributed software architecture, which integrates localization, mapping, collision avoidance, planning, and various modules concerned with user interaction. The approach does not require any modifications to the environment. To cope with the various challenges in dynamic and ill-structured environments, the software relies on probabilistic computation, on-line learning, any-time algorithms, and distributed control. Special emphasis has been placed on the design of interactive capabilities that appeal to people's intuition. For several times, mobile robots were successfully deployed in a densely populated museum, demonstrating reliable operation in hazardous public environments, and raising the museum's attendance by more than 50\%. In addition, people all over the world controlled the robot through the Web.

Document Image Analysis via Model Checking

Marco Aiello

Institute for Logic, Language and Computation, and
Intelligent Sensory and Information Systems
University of Amsterdam
Plantage Muidergracht 24 1018 TV Amsterdam, The Netherlands
aiellom@ieee.org

1 Introduction

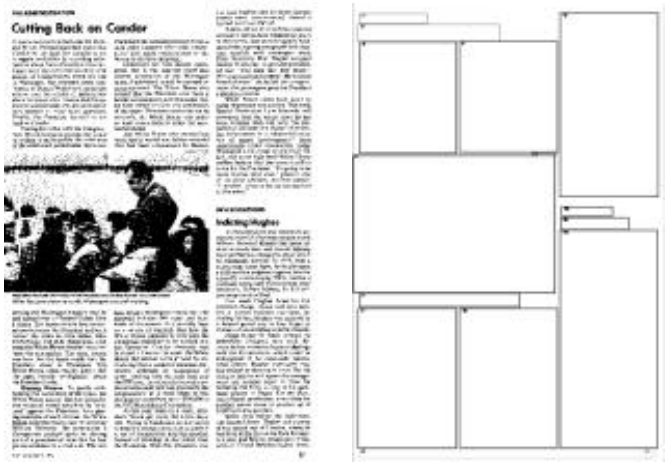
When Dave placed his own drawing in front of the 'eye' of HAL—in 2001: A Space Odyssey—HAL showed to have correctly comprehended and interpreted the sketch. "That's Dr. Hunter, isn't it?" [9]. But what would have happened if Dave used the first page of a newspaper in front of the eye and started discussing its contents? Considering HAL a system capable of AI, we expect HAL to recognize the document as a newspaper, to understand how to extract information and to understand its contents. Finally, we expect Dave and HAL to begin a conversation on the contents of the document.

Here we present a methodology based on model checking, which has been successfully experimented on an heterogeneous collection of documents [1, 11], to extract the content from images of documents. We focus on mechanically generated documents, in contrast with hand-writing and sketches. Using terms better-known to the image processing community, we are interested in logical structure detection in the context of document image analysis.

Document image analysis is the set of techniques involved in recovering syntactic and semantic information from images of documents, prominently scanned versions of paper documents. An excellent survey of document image analysis is provided in [8] where, by going through 99 articles appeared in the IEEE's Transactions on Pattern Analysis and Machine Intelligence, Nagy reconstructs the history and state of the art of document image analysis. Research in document images analysis is useful and studied in connection with document reproduction, digital libraries, information retrieval, office automation, and text-to-speech.

There are two distinct tasks in document image analysis. The first has a syntactical goal consisting of the identification of basic components of the document, the so-called *document objects*. The second has a semantic goal consisting of the identification of the role and meaning of the document objects in order to achieve an interpretation of the whole original document. The syntactic information is synthesized in the *layout structure* of the document, while the semantic information goes under the name of *logical structure*. In the latter task, two subtasks are usually identified: logical labeling, and reading order detection. *Logical labeling* consists of the assignment to document objects of labels indicating their

role (page number, title, sub-title, etc.). *Reading order detection* aims at reconstructing the sequence of textual document objects in which the user is going to (or is supposed to) read the document at hand.



(a) (b)

Fig. 1. A document image (a) and its layout information (b).

The pattern matching and, more generally, the computer vision communities have been very active in the field, especially in tasks tied to the layout structure detection. On the other hand, logical structure extraction has only been dealt with in very restricted domains, like the interpretation of addresses on envelops. A fundamental step toward generality is to be found in [12]. Tsujimoto and Asada present an algorithm to extract the logical structure from two-column black and white images of documents with a simple known a priori layout. On the negative side, no properties of the algorithm are known, the framework can not be extended to any other type of document (especially if the layouts get intricate) and no use is made of the textual information available.

The technique we use for reading order detection is based on model checking. Model checking is one of the most successful applications of logic techniques in computer science due to a number of factors, including effectiveness and robustness [6, 5, 4]. All model checking systems known in the literature involve a temporal logic and a model with a finite number of states. The system which is being modeled is represented as a finite state transition system, and specifications are expressed in a propositional temporal logic. The model checker works by exploring all the states of the model, in this way it is possible to automatically check if the specifications are satisfied. The termination of model checking is guaranteed by the finiteness of the model.

In the next section, we illustrate how to transform document information into a formal spatial model, overviewing the methodology we propose. In Section 3, we illustrate a test case for the methodology in which we focus on the detection of the reading order. In Section 4, we give some concluding remarks and open directions for future research.

2 Document Images as Formal Models

Given a document image, we assume available its layout and logical labeling information, that is, we assume identified the basic entities of the document, the document objects, their location and their logical type. Considering, for example, the document image in Figure 1.a, we assume given the segmentation of the document as represented in Figure 1.b together with its labeling information.

The core of the methodology we propose lies in viewing the layout together with the logical labeling information, as a spatial model, while considering logical document rules as formulas of a specific logic. The process of extracting logical information is then defined as an instance of model checking. The formulas encoding document logical rules are checked against the model of a given document image. The states (document objects) and paths (totally ordered collections of document objects) satisfying these formulas are the logical structure extracted from the document image. In Figure 2, we summarize the methodology. On the top left, a document image is represented. Via image processing techniques one gets the layout and logical labeling information, [11]. Here we assume that it is given. This information is transformed into a spatial model, as we shall see next. The model is then used in `SpaRe`, our model checker. In our current setup, the formulas used for model checking are written by an expert, but it is easy to imagine that these could be directly written by the document author, or they could be learned automatically. For the first case, imagine the designer of a magazine to write down which are the formal rules he follows in editing his magazine. While for the second, think of having a set of journals to analyze and to provide the correct logical structure for a number of issues. The learning system attempts to mine the formal rules behind the journal. These rules could then be used to analyze the whole collection of the journal.

Let us analyze more precisely the transition from layout to a formal model. In the present context, the layout is a set of document objects together with geometrical information

$$DO = \{do \mid do = \langle id, x_1, y_1, x_2, y_2 \rangle\}$$

where id is an identifier of the document object and (x_1, y_1) (x_2, y_2) represent the uppermost-leftmost corner and the lowermost-rightmost corner of the bounding box of the document object.[1] In addition, we consider the logical labeling

[1] Superimpose a coordinate system to the document image. The leftmost uppermost corner has coordinates (0,0). The x axis spans horizontally increasing to the right, while the y axis spans vertically towards the bottom.

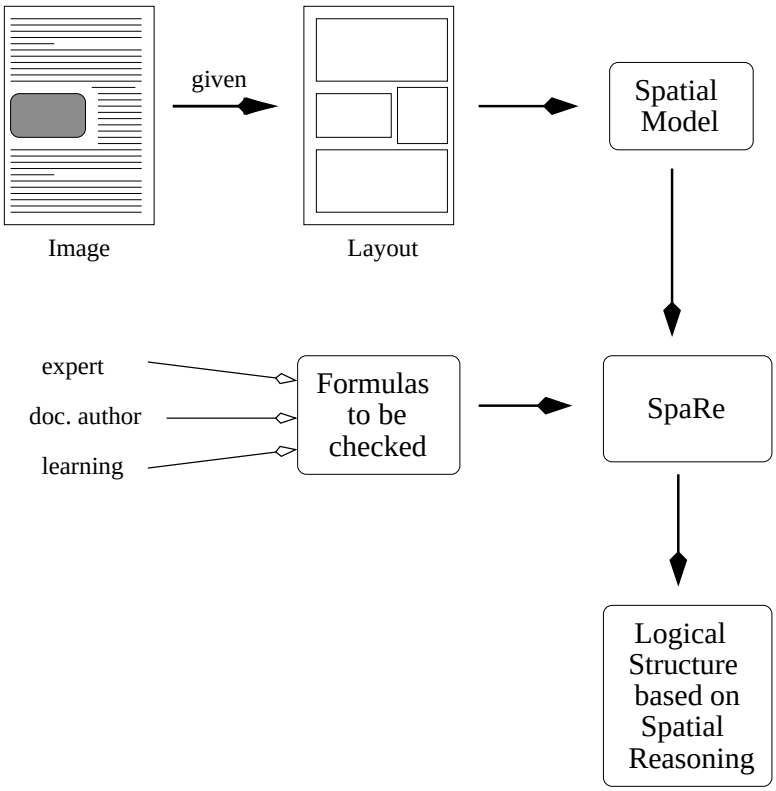


Fig. 2. The flow of information in document analysis as model checking.

information. Logical labels are associated with each document object and describe their function. Common examples are: title, subtitle, body, page number. Given a set of labels L, logical labeling is a function *lab*, typically injective, from document objects to labels:

$$lab: \ DO \rightarrow L$$

As for the formal model, we consider a special spatial kind: a *spatial model* is a tuple $\langle S, R, \nu, \rangle$, where S is a set of states, R is a set of bidimensional Allen's relations and $\nu : S \rightarrow L$ is a valuation function mapping states to proposition letters (the labels L in this case). The set of relations R consists of 13×13 relations, that is, the product of Allen's 13 interval relations [2] (precedes, meets, overlaps, starts, during, finishes, equals, and their inverses) on two orthogonal axes. The definition closely resembles the one of a rectangle model of Balbiani *et al.* in [3]. It is now easy to translate the layout and logical labeling information available for a document image into a spatial model, as we show next.

Definition 1 (spatial translation). A document image, i.e., a set of document objects DO and a labeling function lab, is *translated* into a spatial model $\langle S, R, \nu, \rangle$ by $\cdot^t$ in the following way:

1. For all $do \in DO$:

$$do^t \in S$$

2. For all $do^t \in S$:

$$\nu(do^t) = lab(do)$$

3. For all $do_i^t, do_j^t \in S$:

$$(do_i^t, do_j^t) \in R_k \text{ for some } R_k \in R$$

where R_k is given by Allen's relation on the x axis between the intervals $[x_{i1}, x_{i2}]$ and $[x_{j1}, x_{j2}]$, and by Allen's relation on the y axis between the intervals $[y_{i1}, y_{i2}]$ and $[y_{j1}, y_{j2}]$.

It is immediate to notice that $|S| = |DO|$ and $|R| = 13 \times 13$. Less obvious may be the fact that there is always a relation $R_k \in R$ among any two document objects. This follows from the fact that the Allen's relations are jointly exhaustive and pairwise disjoint, that is, given any two rectangles there is always one and only one bidimensional Allen relation holding among them. For example, a rectangle is equal on the x and equal on the y with itself, while the leftmost-uppermost and the rightmost-lowermost bounding boxes in Figure 1.b are precedes on the x and precedes on the y relation (in this order). If we look at the spatial model as being a directed graph, we notice that document object are nodes, labeled by the valuation function, that the graph is fully connected (there is a directed edge from any node to any node), and that for every edge connecting two nodes and denoting a bidimensional Allen relation, there is a converse edge denoting the inverse of the bidimensional Allen relation.

3 A test case

We have applied the proposed methodology to extract the reading order from an heterogeneous collection of documents. (For the experimental results and implementation choices see [1, 11], here we present the relation of the implementation with the methodology based on model checking.) First, we consider a sub-model of the spatial one defined in the previous section. We prune the model of many of its relations in R following the rule to keep only the relations that represent a "before in reading transition." Intuitively, we consider a document object to be before in reading of another one if it precedes or meets it on either axis, if it contains it or if it overlaps with it. For the full set of Allen's bidimensional relations that we consider to identify a before in reading relation we refer to [1]. Second, we regard the set of pruned relations R as a unique transition relation

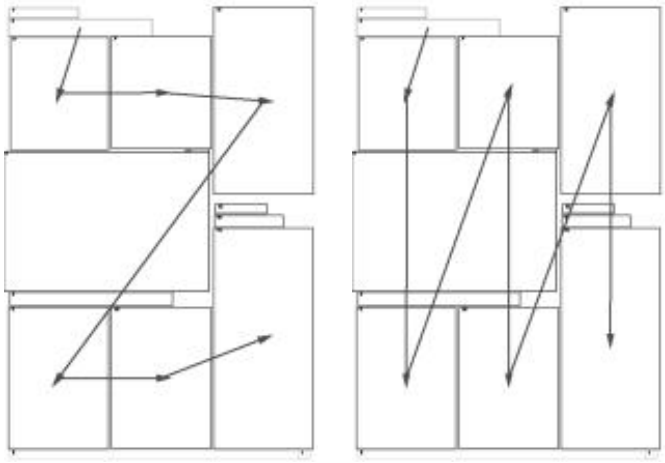
"before in reading." The new model just identified is a directed transitive graph (i.e., a directed graph, with the edge relation transitively closed). The modal logician may recognize in it the typical structure of an S4 model. Third, we consider the logic CTL to express rules to extract the reading order. In particular, the reading order for a page is given by the formula

$$EG((\texttt{title} \vee \texttt{body of text}) \wedge \neg\texttt{page no} \wedge \neg\texttt{caption})$$

for the complete syntax of CTL we refer to [5], here we just give an informal explanation. The formula states that there exists a path such that all elements of the paths are either labeled as title or body of text and are not captions and not page numbers.

In general, various paths in the model will satisfy a given CTL formula like the one above. Of all paths, we consider only those of maximal length. The driving assumption is that all textual document objects must contribute to the reading order. We further rank the remaining paths by considering the natural language probability of each given transition. By using shallow natural language processing techniques and retrieving the textual content of each document object associated with a state of the model, we assign a probability to each reading order found.

Experimental results and details of the natural language tools used are presented in [11]. For example, the result of the model checking for the document in Figure 1 are presented in Figure 3. Two reading orders are identified. The natural language processing then assigns a higher probability to the reading order on the right in Figure 3.



(a) (b)

Fig. 3. The result of model checking the document in Figure 1.

4 Concluding Remarks and Future Work

We have presented a new approach to the problem of logical structure detection in document image analysis. The prominent feature of the approach is its modularity. To extract different sorts of logical information from a document, one only needs to rewrite the modal formula to be checked and leave the whole architecture of the system untouched.

In the context of document image analysis, the presented proposal is the first to use Allen relations at the semantic level. Up to now Allen's relations have been only used as a feature descriptor (thus at the syntactic level of a layout property) [7, 10].

For future research there are various questions and directions open for investigation. The first question regards which is the appropriate logic to use for model checking with the most general spatial model as defined in Section 2. Given the sort of transitions in the model, bidimensional Allen relations, the appropriate starting point looks like a bidimensional generalization of Venema's logic of chopping intervals [13], but the issue is still open. The next challenge for us consists of identifying appropriate formulas to deal with independent reading orders. Independent reading orders arise in complex documents with independent portions of text. A typical example is the first page of a newspaper, where different unrelated news all appear together in the same page. A final important issue resides in automatically finding formulas describing logical rules of a document. Data mining and learning techniques seem the most appropriate to achieve this goal; the road to the solution is completely open for investigation.

References

1. M. Aiello, C. Monz, and L. Todoran. Combining Linguistic and Spatial Information for Document Analysis. In J. Mariani and D. Harman, editors, *Proceedings of RIAO'2000 Content-Based Multimedia Information Access*, pages 266–275, Paris, 2000. CID.
2. J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983.
3. P. Balbiani, J. Condotta, and L. Fariñas del Cerro. A model for reasoning about bidimensional temporal relations. In A. G. Cohn, L. Schubert, and S. Shapiro, editors, *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 124–130. Morgan Kaufmann, 1998.
4. A. Cimatti, E. M. Clarke, F. Giunchiglia, and M. Roveri. NUSMV: A New Symbolic Model Checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.
5. E. Clarke, O. Grumberg, and D. Peled. *Model Chekcing*. MIT Press, 1999.
6. E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.
7. S. Klink, A. Dengel, and T Kieninger. Document Structure Analysis Based on Layout and Textual Features. In *Fourth International Workshop on Document Analysis Systems*. IAPR, 2000.

8. G. Nagy. Twenty Years of Document Image Analysis in PAMI. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(1):38–62, 2000.
9. A. Rosenfeld. Eyes for Computers: How HAL Could "See". In D. Stork, editor, *HAL's Legacy*, pages 210–235. MIT Press, 1997.
10. R. Singh, A. Lahoti, and A. Mukerjee. Interval-algebra based block layout analysis and document template generation. In *"Workshop on Document Layout Interpretation and its Applications (DLIA99)"*, Bangalore, India, September 1999. http://www.wins.uva.nl/events/dlia99/final_papers/singh.pdf.
11. L. Todoran, M. Aiello, C. Monz, and M. Worring. Logical structure detection for heterogeneous document classes. In *Document Recognition and Retrieval VIII*, pages 99–110. SPIE, 2001.
12. S. Tsujimoto and H. Asada. Major Components of a Complete Text Reading System. *Proceedings of the IEEE*, 80(7):1133–1149, 1992.
13. Y. Venema. A Modal Logic for Chopping Intervals. *Journal of Logic and Computation*, 1(4):453–476, 1991.

Document Region Segmentation and Classification Using a Neuro-Fuzzy System

Laura Caponetti, Giovanna Castellano, and Anna Maria Fanelli

Department of Informatics, University of Bari
Via E. Orabona, 4
70126 Bari, Italy

Abstract. This paper presents a neuro-fuzzy system to document region segmentation and classification. The system performs two tasks. First it segments a document image into regions, using a fuzzy thresholding approach. After it labels each region using a neuro-fuzzy classifier on the basis of features describing the presence (or absence) of text lines in that region. Instead of processing the input gray level document image, the system works in the texture feature domain obtained, in a pre-processing phase, by computing local texture features.

1 Introduction

The aim of document image analysis is to convert document images to symbolic form for modification, storage, retrieval, reuse, and transmission. In such a field, document classification into regions, blocks of text and pictures, plays a fundamental role since it allows separate processing of different regions previously classified. In fact, once the regions of a document image have been classified, more specific techniques are needed. For example the text regions are separated in columns, paragraphs, text-lines, words and characters; then the individual words or characters are converted into a character code like ASCII. Instead the graphics regions, such as line drawings, are decomposed in primitives -strength lines, curve segments and so on- and then interpreted.

Several methods for document image segmentation and classification have been developed 1, 2, 3. In this work a neuro-fuzzy system is proposed that, starting from a document image, automatically produces a map of the connected components of the document, each of them labeled as text component or graphics component. The advantage of using neuro-fuzzy networks is that they provide a flexible framework with great potential for solving complex problems, like the classification of regions in a document image. Moreover, neuro-fuzzy systems are computationally simple and provide knowledge in form of rules that can have a direct interpretability 4.

The proposed system performs essentially two tasks. First it segments a document image into regions, using a fuzzy thresholding approach. After it labels each region

using a neuro-fuzzy classifier on the basis of features describing the presence (or absence) of text lines in that region. Instead of processing the input gray level document image, the system works in the texture feature domain obtained, in a pre-processing phase, by computing local texture features. In the following we describe in more details the fuzzy thresholding segmentation, the feature extraction, the neuro-fuzzy region classification and the experimental results.

Figure 1 shows the overview of the whole system.

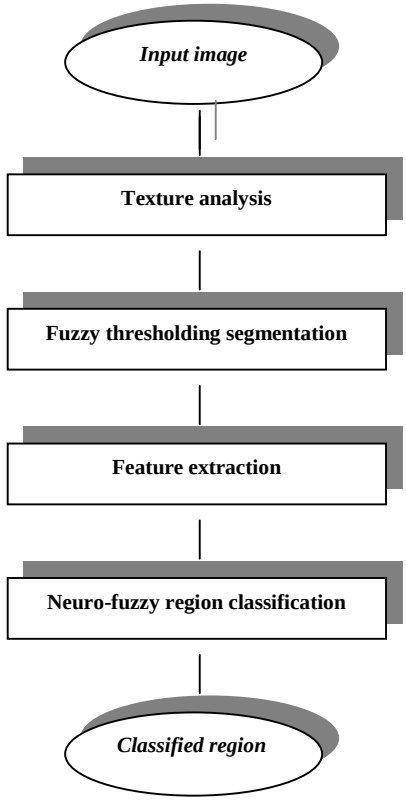


Fig. 1. Overview of the whole system

2 Fuzzy thresholding segmentation

Thresholding is a technique widely used in image segmentation. The objective of histogram thresholding is to determine a boundary value to partition the image space into meaningful regions. Generally the regions in a document image may be ill-defined, because of image data ambiguity, due to textured background, information

noise and so on. For this reason some approaches based on fuzzy set theory have been proposed. In 5 a method has been described based on the minimization of the image fuzziness in the intensity and spatial domain. In 6 classes of fuzzy entropies are constructed which are useful for image thresholding based on cost minimization.

Fuzzy interpretations of data images are based on the assumption that an $M \times N$ image G of L gray levels can be associated with an array F of fuzzy singletons $\mu(G(i.j))$, each denoting the value of membership of $G(i.j)$ to the set of gray levels.

In this work, we use the fuzzy compactness approach proposed in 7 that applies a distance measurement to texture feature images. More precisely, instead of processing the gray level document image, we consider the minimization of fuzziness in the texture feature domain, to select appropriate threshold value for segmentation.

To obtain the texture feature domain from the original document image, we compute local texture features as statistical variances of the image filtered with the Laws masks. Laws, in fact, in 8 introduced the notion of local texture energy as measure of texture features in the spatial domain. In other words, a fuzzy thresholding algorithm has been applied to the texture feature image, evaluated by means of Laws' texture energy measurement.

We denote by l_{max} and l_{min} the maximum and minimum texture energy values of the document image. The algorithm for fuzzy thresholding segmentation is summarized as follows:

1. Compute the membership function values in the texture feature image X

$$\mu(l) = S(l:a.b.c). \quad l_{min} \le l.b \le l_{max}$$

where S is an S-function defined as:

$$\begin{aligned} S(l:a.b.c) &= 0 && l \le a. \\ &= 2[(l-a)/(c-a)]^2 && a \le l \le b. \\ &= 1-[(l-c)/(c-a)]^2 && b \le l \le c. \\ &= 1 && l \ge c \end{aligned}$$

with $a = l_{min}$, $c = l_{max}$ and $b = (a+c)/2$ and width $\Delta b = b - a = c - b$.

2. Varying b from l_{min} to l_{max}, compute the linear index of fuzziness in the texture feature image X relative to the point b:

$$\upsilon_b(X) = \frac{2}{MN} \sum_l T_b(l) h(l)$$

where

$$T_b(l) = \min\{S(l:a.b.c).1 - S(l:a.b.c)\}$$

and $h(l)$ denotes the number of occurrences of the level l.

3. Take the thresholding value as the value of b providing the minimum value of $\upsilon_b(X)$.

Finally the flood-filling operator 9 has been applied to the thresholded image.

3 Feature extraction

The aim of the feature extraction is to represent each document region by a set of features useful to describe the presence (or absence) of text lines in that region.

A text line is a group of characters, symbols, and words that are adjacent and through which a straight line can be drawn. The dominant orientation of the text lines determines the skew angle. Text may also have a chosen orientation that is different from the page skew. In 10, 11 skew detection methods based on the Hough transform have been proposed. The Hough transform maps each point of the original $(x.y)$ plane to all points of the $(\rho.\theta)$ Hough plane describing the possible lines through $(x.y)$ with slope θ and distance from origin ρ. The dominant lines are found from peaks in the Hough space and thus the orientation. In 12 concepts related to human visual perceptual organization are used to state the classifier of a document region as a preattentive texture classification problem. Also in 12 the Hough transform is used to characterize text regions.

As pointed out in 10, 11, 12, text regions visually present well defined structures of lines with specific orientation and periodicity; on the contrary, graphics regions present uniformity without any specific structure. This means that only text regions present periodicity of the peaks values in the Hough space. Several techniques can be used to measure such a periodicity, as Fourier transform, autocorrelation, and power spectrum density.

In this work we use the Power Spectrum Density (PSD) coefficients as measure of periodicity: in fact, for large text regions, the PSD coefficients show a significant peak as a consequence of the frequency content in the region and for non-text the spectrum is almost flat. Then for each region a small number n of PSD coefficients is selected in the Hough space and used as inputs to the classifier of document image regions.

4 Neuro-fuzzy region classification

To classify each region encoded as a vector of *n* PSD coefficients, we have used a neuro-fuzzy network that implements a fuzzy classification system.

Let P be the number of input vectors $\bar{x}^p = \left(x_1^p \ldots\ldots x_n^p\right)$, $p = 1.\ldots\ldots P$ generated as described before and labeled as belonging to two classes: text (class C_1) and no text (class C_2). This set of labeled vectors will be referred to as training set in the fol-

lowing. The task of the classifier is to assign a given input vector $\bar{x}$ to one of the two classes based on its features values. To solve this classification problem we consider fuzzy rules of the following type:

$$R_k : \text{IF } \left(x_1 \text{ is } A_1^k\right) \text{AND...AND } \left(x_n \text{ is } A_n^k\right) \text{ THEN } \left(\bar{x} \in C_1 \text{ with degree } v_{k1}\right) \text{AND } \left(\bar{x} \in C_2 \text{ with degree } v_{k2}\right)$$

where R_k is the *k*-th rule ($1 \le k \le K$), $\{x_i\}_{i=1...n}$ are the input variables, $\{C_j\}_{j=1,2}$ are the output classes, A_i^k are fuzzy sets defined on the input variables, and v_{kj} are fuzzy singletons representing the degrees to which a region $\bar{x}$ belongs to classes $\{C_j\}_{j=1,2}$. Fuzzy sets A_i^k are defined by Gaussian membership functions

$$\mu_{ik}(x_i) = \exp^{-(x_i - w_{ik})^2 / 2\sigma_{ik}^2} \tag{1}$$

where w_{ik} and σ_{ik} are the center and the width of the Gaussian function, respectively.

Based on a set of *K* rules, the classification of an unknown region represented by the vector $\bar{x}^0 = \left(x_1^0 \ldots . x_n^0\right)$ is performed by the following inference mechanism:

1. Compute the activation strength of the *k*-th rule, for *r=1,…,K* , by means of product operator:

$$\mu_k\left(\bar{x}^0\right) = \prod_{i=1}^{n} \mu_{ik}\left(x_i^0\right) \quad k = 1 \ldots . K \tag{2}$$

2. Compute the membership degrees to each class as a weighted average:

$$y_j^0 = \frac{\sum_{k=1}^{K} \mu_k\left(\bar{x}^0\right) v_{kj}}{\sum_{k=1}^{K} \mu_k\left(\bar{x}^0\right)} \quad j = 1,2 \tag{3}$$

This yields to a fuzzy classification of the document region. To obtain a crisp classification, i.e. to classify the region into just one of the two classes, we select the class C_j. such that:

$$y_j^0. = \max\left\{y_1^0 . y_2^0\right\}$$

The fuzzy classifier can be view as a special 3-layer feed-forward neural network, as depicted in Figure 2. Units in the three layers have the following meaning:

1. Units in the first layer compute the membership values $\mu_{ik}(x_i)$ according to (1), thus each unit has two free parameters, i.e. the center and the width of the Gaussian function.
2. The second layer holds *K* units that compute activation strength of fuzzy rules according to (2); no free parameter is associated with such units.
3. The third layer provides the class membership degrees y_j, as defined in (3). Connections between the second and the third layer are weighted by the free parameters v_{rj}.

This neuro-fuzzy network encodes a set of fuzzy rules in its topology, and processes information in a way that matches the fuzzy inference scheme adopted. The weights of the network include the premise parameters w_{ik}, σ_{ik} and the consequent parameters v_{kj} of fuzzy rules.

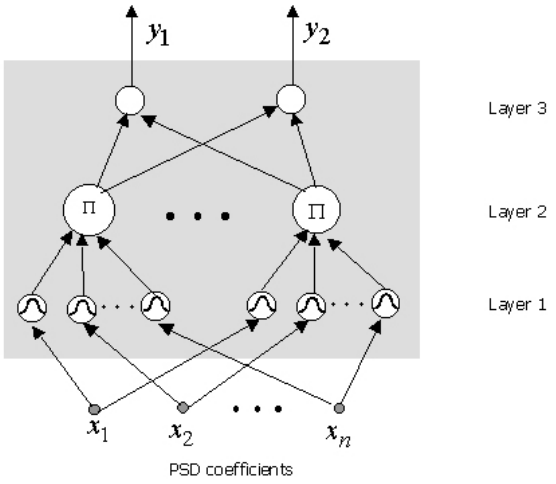


Fig. 2. The neuro-fuzzy network

The significance of converting the fuzzy classifier into a neural network lies in the possibility of finding fuzzy rule parameters taking advantage of the learning ability of neural networks. To define automatically fuzzy rule parameters on the basis of the available data, the neuro-fuzzy network is trained via a supervised learning algorithm based on a gradient-descent technique 13. To speed up learning, the network parameters are properly initialized according to the training data. Premise parameters (w_{ik}, σ_{ik}) are initialized by clustering the input space via the FCM algorithm 14. For each rule *k=1…K*, the center w_{ik} of the *i*-th Gaussian function is defined as the *i*-th coordinate of the *k*-th cluster center, and the width σ_{ik} is assigned to the value of the

cluster radius. The parameters v_{kj} are initialized by taking into account how much input vectors belonging to class C_j are covered by the *k*-th cluster:

$$v_{kj} = \sum_{\bar{x}^p \in C_j} \mu_k\left(\bar{x}^p\right) \Big/ \sum_{p=1}^{P} \mu_k\left(\bar{x}^p\right) \quad j = 1,2$$

By doing so, the structure as well as the initial weights of the neuro-fuzzy network are established. This corresponds to determine the number of fuzzy rules and the initial parameters of each rule. This initial fuzzy rule base is then finely tuned through the network learning to obtain the final fuzzy rule base.

5 Experimental results and conclusion

To test the effectiveness of the proposed system, 100 free format document images been considered. Among these images, we have selected 30 images containing only text regions, and 30 images containing only graphics regions as training set. The remaining 40 images, containing both text and graphics regions, have been used for the testing phase.

In both learning and testing phases, each image *G* has been filtered using two Laws masks, providing two filtered images G1 and G2. Then, a textured image *E* has been computed as energy of the difference between the image *G* and the average of the two images G1 and G2. The textured image *E* has been segmented into regions using the fuzzy thresholding segmentation technique described in Section 2. Each document region has been transformed in the Hough space and a vector of 10 PSD coefficients has been produced.

A neuro-fuzzy network with 10 inputs, 15 rule nodes and 2 outputs has been trained to correctly classify all the training images. The trained network has been used to classify the 40 testing images, providing a classification rate of 95,53%. Table I summarizes the classification results. It can be seen that misclassified regions correspond mainly to text regions classified as graphics regions. This is due essentially to the segmentation process that works badly when a small region is too near or included into a region of different type. To improve the classification rate, segmentation errors should be reduced. This can be obtained by improving the quality of the images (i.e. increasing resolution) to get better information inside the region to classify.

Table 1. Results of the testing image classification

No. of regions	Classification rate	No. of correctly classified regions	No. of misclassified text regions	No. of misclassified graphics regions
224	95.53%	214	7	3

Finally to give a qualitative evaluation of the proposed system, we provide the results for two testing images that differ in size, complexity and page skew. The considered images are depicted in figures 3a, and 4a. For each image, the regions found by the segmentation phase are reported in figures 3b, and 4b. Finally the classified regions are shown in figures 3c, and 4c, where regions classified as text are displayed in light gray level, while regions classified as graphics are displayed in dark gray level. It can be seen that in all of the three cases a good segmentation of the document image is obtained and all the identified regions are correctly classified into text regions and graphics regions.

Fig. 3a. Input image

Fig. 4a. Input image

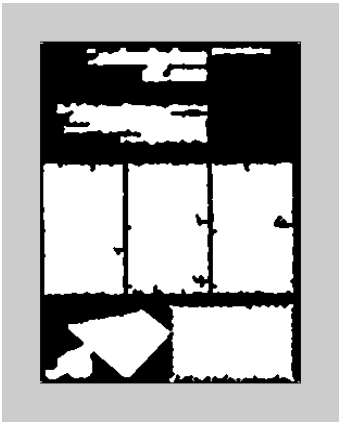


Fig. 4b. Connected regions of fig. 3a

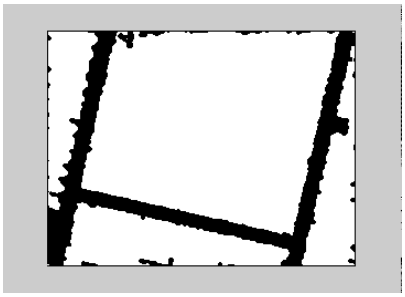


Fig. 4b. Connected regions of fig.4a

Fig. 5c. Classified regions of fig. 3b

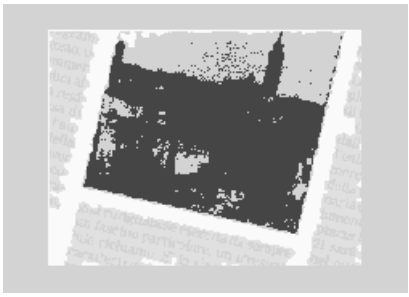


Fig. 4c. Classified regions of fig. 4b.

References

1. L. O'Groman and R. Kasturi: Document Image Analysis. IEEE Computer Society Press, 1995.
2. G. Nagy: Twenty Years of Document Image Analysis in PAMI: IEEE Trans. on Pattern Analysis and Machine Intelligence, 22(1):38-62, January 2000.
3. J.L. Fisher, S.C .Hinds, and D.P. D'Amato, A rule-based system for document image segmentation: Proc.10th Int. Conf. Patt. Recogn. (ICPR), Atlantic City, NJ, June 1990, pp.567-572.
4. B. Kosko: Neural Networks and Fuzzy Systems: a dynamical approach to Machine Intelligence, Prentice Hall, Englewood Cliffs, NJ, 1992.
5. S.K. Pal and A. Rosenfeld: Image enhancement and thresholding by optimization of fuzzy compactness. Pattern Recognition Letters, 7: 77-86, 1988.
6. S. Di Zenzo, L. Cinque, S. Levialdi: Image Thresholding Using Fuzzy Entropies, IEEE Trans on Systems, Man and Cybernetics*:* 28(1):15-23, February 1998.
7. W.P. Zhu, C.Z. Sun, J. You: Parallel textured segmentation using fuzzy set, Proc. of ICSP 1996, pp.1219-1222.
8. K.J. Laws: Textured image segmentation, Ph.D. Thesis, University of Southern California, January, 1980.
9. M. Sonka, V. Hlavac, R. Boyle: Image Processing, Analysis and Machine Vision, Chapman & Hall Computing, 1995.
10. S.N. Srihari and V. Govindaraju: Analysis of textual images using the Hough transform. Machine Vision Application, 2:141-153, 1989.
11. S.C. Hinds, J.L. Fisher, and D.P. D'Amato. A document skew detection method using run-length encoding and the Hough transform. Proc.10th Int. Conf. Patt. Recogn. (ICPR), Atlantic City, NJ, June 1990, pp.464-468.

12. M.I.Chacon Murguia : Fuzzy Neural Network for Document Region Classification using Human Visual Perception Features. Proc. International Joint Conference on Neural Networks (IJCNN'99), Washinghton, July 1999.
13. G.Castellano, A.M.Fanelli: Constructing compact fuzzy models from data. In Advances in Fuzzy Systems and Intelligent Technologies, Shaker Publishing, NL, 2000, pp. 187-196.
14. J.C. Bezdek: Pattern recognition with fuzzy objective function algorithms, Plenum, New York, 1981.

Anticipative Representations in Robot Vision (Short note)

Antonio Chella
Dip. Ingegneria Automatica Informatica
University of Palermo and CERE-CNR, Palermo

In order to build a deep understading of an observed scene, a robot should be able to build an inner representation of the perceived scene. This representation should contain some 3D information of the scene and it should be, observer independent. The inner representation is therefore a *cued* representation built up by computer vision processes in the Marrian tradition (see, e.g., Chella, Frixione, Gaglio, 2000).

The robot generally must also act in the perceived scene, in which in general there may be moving objects, other robots, persons, and so on. The robot must be also able to generate *anticipative* representations of perceived objects, i.e., action oriented representations in the sense of Gibson.

The observer robot therefore may be able to *imagine* possible evolving interaction processes between itself and the objects in the scene, through suitable inner simulations. The inner represenation is therefore not only a *cued* representation generated by current observations, but it may have *detached* capabilities to simulate possible evolutions of interaction among the robot and the perceived scene (Gärdenfors 2000).

As an example, let us consider a robot observing a work table in which there is a hammer and a nail. The inner representation allows the robot to build a 3D description of the scene as a CAD system, i.e., it models the geometric parameters of the hammer and the nail in the robot memory, by means of 3D primitives (e.g., superquadrics). By means of this 3D representation, the robot is able to know that there is a hammer in the scene, that there is also a nail, that the hammer is leftmost of the nail, etc. But the anticipative representation make the robot to simulate possible interaction with the objects, i.e., the robot can simulate to pick up the hammer and to use it with the nail in order to fix something. In this second kind of representation the robot is an agent that interact with the environment. The robot can therefore build an inner understanding of the objects in the scene. Similar operations holds when there is a nail and a long stick instead of the hammer. In this case, the anticipative representations allows the robot to simulate the hammering operation by using the stick instead if the hammer.

The paper describes in details how the inner and the anticipative representations may be modeled and integrated in a natural way by means of dynamic conceptual spaces (Chella, Frixione, Gaglio 2000). Examples of applications of anticipative representations are presented in the framework of skilled navigation of a RWI B21 robot.

Vision-based behaviors for autonomous mobile robot navigation

G. Cicirelli, T. D'Orazio, A. Distante
Istituto Elaborazione Segnali ed Immagini - C.N.R.,
Via Amendola, 166/5 - 70126 Bari (Italy)

Abstract

In this paper we describe two elementary behaviors which realize a goal-reaching task for an autonomous vehicle. The robot has to reach a door from every position of its environment, therefore it firstly needs to detect the door in the environment. Both behaviors, door-detection and door reaching, are based on visual information received by a TV-camera placed on the mobile robot. The module for detecting doors uses a supervised learning scheme for building the model of the door, in which the examples are particular views of the door, previously stored in form of image patterns. The adopted learning scheme is a neural network two-classes classifier. Since in the image the door can appear of different dimensions depending on the attitude of the robot with respect to the door, the detection of the door is performed by detecting its most significant components in the image. The door-reaching behavior, instead, has been developed associating the proper action to each encountered situation of the vehicle for reaching the considered target. A Q-learning algorithm is used to discover the optimal state-action rules (*optimal policy*). A few training trials are sufficient, in simulation, to learn the optimal policy since during the test trials the set of actions is initially limited. The optimal policy learned in simulation is then transferred on the real robot for the testing phase.

1 Introduction

In the last decade the machine learning community has mainly focused on the development of behavior-based robots [4]. These systems use behaviors as a way for decomposing the control policy needed to accomplish a task: the results are very robust with respect to the dynamics of real world environments and they are also simpler to develop, because each basic module can be built and verified separately. In fact a major advantage of the task decomposition into behaviors is that each module has to solve the perception, modelling, and planning problems in their simpler form, accounting only for what is relevant for its own particular goal. A further advantage is that behaviors can be independently developed and added to the control system whenever new capabilities are needed. Thus developing elementary behaviors can be considered the starting point for realizing complex robotic systems [9, 5]. Basic behaviors can often be thought and realized as direct associations (reactive control) between sensory data and actions. This association could be manually designed (at least for quite simple tasks) but they will be affected by the subjective skill of the designer. There is therefore an intense activity about techniques for automatically acquiring control strategies from suitable sets of examples (*supervised learning*) or from the evaluation of the current level of performance (*reinforcement-learning*).

In this work we describe two fundamental behaviors for autonomous navigation of a mobile vehicle: a goal-detection behavior and a goal-reaching behavior. The goals the robot has to reach are the doors of our office building. The module for detecting doors uses a supervised learning scheme for building the model of the door, in which the examples are particular views of the door, previously stored in form of image patterns. The adopted learning scheme is a neural network two-classes classifier trained with the well-studied BackPropagation (BP) algorithm. In fact the problem of recognizing a particular object can be seen as a classification problem. Techniques based on learning by examples have been successfully applied in wide areas of computer vision such as face-detection [7], hand-written character recognition [8], people detection [12, 13]. In our work the aim is to separate door views from image patterns that are not instances of the door. The doors of our building are red doors and are bounded by a black bar. Both color information and shape information are used for detecting doors and for distinguishing them from

other similar objects such as posters, cupboards, windows, etc. Since in the image the door can appear of different dimensions depending on the attitude of the robot with respect to the door, the detection of the door is performed by detecting its most significant components in the image. In [1] a system which detects people by their components is presented. Component-based approach for detecting objects is appealing because it allows to use geometric information concerning the objects and it can also detect objects partially occluded. Moreover it is often difficult to detect objects as a whole relying only on visual information. In [1] the authors compare different classifiers-based systems which combine the component classifiers in different ways and the full-body person detection system. They show that the performance of components-based classifier is better than the full-body person detector due to the component based approach. In our work we follow this idea so, instead of having a single machine trained on the whole views of the door, we use different classifiers each trained on a single component of the door. The fundamental components of the door are upper corners, top bar, left and right vertical bar. Since the neural network has been trained to detect each door component separately, a validating algorithm has been applied to verify that the detected components are in the proper geometric configuration of the door.

The door-reaching behavior, instead, has been developed associating the proper action to each encountered situation of the vehicle for reaching the door. The robot has to move close to the door from any position of the environment. The behavior uses qualitative information provided by a vision system about the vehicle's position with respect to the door. The use of visual sensors has been limited in literature due to the difficulties and the cost of processing visual data [15, 11]. Nonetheless vision can be very useful since it is able to detect distant goals and allows the acquisition of suitable behaviors for global and goal-oriented tasks [2, 10]. We have defined the state of the system in terms of specific geometric information about the door as it is seen in the images acquired by the color camera mounted on the robot. The behavior starts with no a-priori knowledge about its driving strategy, which is wholly learned from the interactions with the environment. Reinforcement learning techniques seem suitable for addressing this problem: the agent chooses an action, on the base of its current and past sensor data, in order to maximize over time a reward function measuring its own performance. In a simulation phase the agent learns the proper action for each encountered state, then the acquired policy is transferred on the real robot in order to test the learned behavior.

During test phase the two constructed behaviors are activated sequentially: first the door-detection behavior starts in order to search and recognize the door, then the door-reaching behavior can be applied. Door-detection becomes active every time the agent looses the door from the image. Notice that this situation happens rarely since the door-reaching behavior can be seen as a sort of tracking behavior. The door disappears from the image only in case of wrong actions.

Aim of this work is to describe how the considered problems of goal-detection and goal-reaching have been resolved and to describe the two different learning schemes applied for each behavior. Notice that the two described behaviors alone cannot solve the complex task of navigation. But these behaviors with additional ones, such as obstacle-avoidance, are single modules of a more complex behavior-based architecture which controls the vehicle during its complex task [3].

The paper is organized as follows. Section 2 describes the image pre-processing used in both behaviors. Section 3 describes how we have solved the problem of door-detection in images. Section 4 gives a full description of the door-reaching behavior. Section 5 show experimental results and therefore conclusions are drawn.

2 Color image pre-processing

The color images acquired by the camera of the robot are firstly pre-processed in order to make the door detection phase easier. The RGB images are transformed into HSI images. HSI (Hue, Saturation, Intensity) color space is more suitable than RGB since it separates the color components HS from the luminance component I and is less sensitive to illumination changes. Besides distances in the HSI space correspond to perceptual differences in color in a more consistent way than in the RGB space.

There are several ways to mathematically transform between the RGB and HSI color spaces. The conversion used in this paper is from [14]. It is done in two steps: firstly the RGB coordinates are transformed into the (I, V_1, V_2) using the formulas:

$$\begin{bmatrix} I \\ V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} \sqrt{3/3} & \sqrt{3/3} & \sqrt{3/3} \\ 0 & 1/\sqrt{2} & -1/\sqrt{2} \\ 2/\sqrt{6} & -1/\sqrt{6} & -1/\sqrt{6} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Then the HS values are obtained from the (V_1, V_2) values with the equations:

$$H = \arctan(V_2/V_1)$$

$$S = \sqrt{V_1^2 + V_2^2}$$

The HSI images are used by both modules of door-detection and door-reaching as it will be described in the following two sections.

3 Door-Detection

The doors of our building are red doors and are bounded by a black bar. Both color information and shape information are very useful for detecting doors and for distinguishing them from other similar objects such as posters, cupboards, windows, etc. The approach we have taken for detecting doors in static images is based on data classification. Since in the image the door can appear of different dimensions depending on the attitude of the robot with respect to the door, the detection of the door is performed by detecting its most significant components in the image. So, instead of having a single classifier trained on the whole views of the door, we use different classifiers each trained on a single component of the door. First neural classifiers detect the components of the door, then the system checks that the detected components are in a proper geometric configuration to asses if the door is in the image or not. Figure 1 shows a sample image taken by the camera of our robot. The camera is tilted up by 20° in order to capture always the top bar of the door. The tilt angle has been experimentally fixed. In figure 1 the principal parts of a door of our building are shown: upper left and right corner, top bar, left and right vertical bars. The system detects the components of the door in images evaluating 18×18 pixel windows extracted from the image. One classifier has been trained for detecting the right upper corner of the door (corner classifier). A second classifier has been trained for detecting the horizontal bar of the door (bar classifier).

Figure 1: An image of the door taken in our laboratory. The fundamental components of the door are pictured in figure.

During the detection phase each 18×18 pattern window of the image is tested by the corner classifier firstly as it appears in the image and secondly after a left-right mirroring. The bar classifier, instead, tests each pattern window and two others obtained by rotating the same window by 90° on left and right. Notice that because of the particular structure of the door, by using these simple tricks only two classifiers are needed for detecting five different constituent objects of the door: the corner classifier detects the upper left and right corner of the door; the bar classifier

detects sub-parts of the top bar and of left and right vertical bar of the door. Once the classifiers have examined the whole image, identifying candidate components, a verifying algorithm is applied for checking if the detected components are in the proper geometric configuration. In this way sparse false positives can be easily eliminated.

Each classifier has been realized by using a one-hidden-layer feed-forward neural network with sigmoid activation functions. The training algorithm is the standard error backpropagation with momentum [6]. The neural network is a fully connected network and it receives as input the H and S components of the 18×18 pattern window. Therefore the input layer of the network has 648 units; the number of units in the hidden layer has been chosen as a trade-off between classification speed and generalization. At this aim a validation set has been used for selecting this parameter. Finally the output layer consists of a single binary output unit.

4 Door-reaching

In this section we present the door-reaching behavior which is based on a reinforcement learning method. The only source of information is the image of the door captured by the camera mounted on the robot. The state of the system is defined considering the attitude of the door with respect to the robot. Three regions for the distance ($NEAR, MEDIUM, FAR$) and three for the orientation ($LEFT, FRONT, RIGHT$) have been considered. The combinations of these two sets of regions define the possible states of the vehicle. In particular the state s_t at time t is described by the couple (i, j) where $i \in \{LEFT, FRONT, RIGHT\}$ and $j \in \{NEAR, MEDIUM, FAR\}$. The couple $(FRONT, NEAR)$ is the goal state as the robot in this region is located near the door, therefore the navigation stops. The states of the vehicle are computed only by using the visual information about the door provided by the color camera placed on the robot. Knowing that the door is in the image, a binary image is obtained from the original one applying a threshold process. Then the contours of the white region are extracted and door slope and width are evaluated (see fig. 2). The slope of the horizontal line and the distance between the two vertical lines (i.e. the door width) are used to identify the region of the environment from which the image was taken or equivalently the region in which the vehicle is. The distance between the two vertical lines gives the information about the near, medium and far position of the vehicle. On the contrary the slope of the horizontal line gives information about the relative orientation of the robot with respect to the door. Therefore the regions of the environment, previously introduced, have been represented in terms of door's slope and width variations.

a) b) c)

Figure 2: a) A colored images of the door taken by the camera from the $(LEFT, MEDIUM)$ region; b) binary image obtained applying the threshold process; c) the extracted contour of the door.

The algorithm used to find the optimal association between actions and states (optimal policy) is the well known Q-learning algorithm [17]. Once the current system state relative to the door has been evaluated an action is selected by using an action selection function. The action space is discrete and contains eight actions corresponding to the absolute orientations that the agent can take in each state: $\mathcal{A} = \{0, 45, 90, ..., 315\}$ $(degrees)$. Performing one action means that the vehicle rotates until it is oriented towards the selected direction and translates until the current state changes. To learn the optimal state-action mapping a number of trials has been executed in simulation. A trial is the execution of a specified number of paths from different starting positions fixed in the environment. Each path terminates with a success if the goal-state is achieved or with

a failure if a collision occurs. The robot is rewarded only when it reaches the goal state, whereas it is penalized when it bumps into the surrounding walls (*delayed reward*).

5 Experiments

In next subsection we describe the example-based learning technique used for developing the door-detecting system. In particular we give details about the training phase of the component classifiers and the test results obtained on a large test set of images. Moreover in the second subsection we describe some experiments done in our laboratory for testing the door-reaching behavior by using our mobile vehicle Nomad200.

5.1 Door detecting system

For collecting positive examples for the component classifiers (corner classifier and bar classifier), a large number of images of the door of our laboratory have been taken from different positions, with different camera orientations and under different lighting conditions. The image size is $384 \times 288 pxl$. Each example of the training sets of the classifiers is a 18×18 pixel sub-image cropped by the original RGB image. The size of the example sub-image has been experimentally fixed considering the dimension of the door in different images in order to avoid the tedious scaling problem. Positive examples and some negative examples, for each classifier, have been manually extracted from the original images. Other negative examples have been obtained by using the bootstrap technique suggested in [16] which consists of a loop of some training and testing steps. In fact each network is trained on an initial training set then it is tested on images which do not contain any doors. Sub-images incorrectly identified as door components are added as negative examples to the previous training set and the process re-starts again. Notice that in a typical test on one image 98820 sub-images are tested by the network.

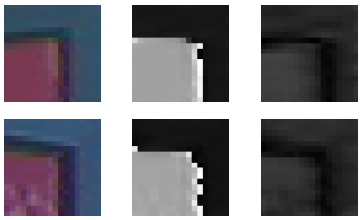


Figure 3: Some positive examples of the training set of the corner classifier. a) original example as RGB image; b) image of the Hue values of the example; c) image of the Saturation values of the example.

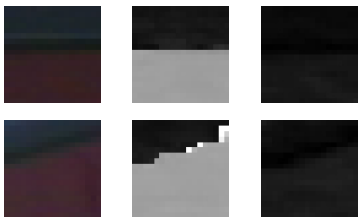


Figure 4: Some positive examples of the training set of the bar classifier. a) original example as RGB image; b) image of the Hue values of the example; c) image of the Saturation values of the example.

The final training set for the corner classifier contains 1906 positive examples and 2907 negative examples, 2106 of which have been obtained after 7 iterations of the bootstrap technique on 100 images which do not contain any doors. The positive examples for the corner classifier are 18×18 pixel windows containing the right upper corner of the door which have been manually extracted from the RGB images (see figure 3). In order to increase the number of positive examples the

left upper corner of the door has been also extracted from the image and left-to-right mirrored. The bar classifier, instead, detects sub-parts of the top bar of the door. In this case a number of positive examples can be extracted from each RGB image. In order to increase the number of negative examples the bootstrap technique has been applied on 65 images which do not contain any doors. The final training set for the bar classifier contains 3000 positive examples and 3161 negative examples (see figure 4). Each RGB example for both classifiers is color pre-processed obtaining the relative hue and saturation values which are passed to the neural network.

Once the component classifiers have been trained, the next step is to evaluate their performance on a test set of images which is distinct from the training sets. Firstly in order to detect a door in images a method for combining the results of the component classifiers must be defined. In our experiments we have considered a validating algorithm which confirms the detection of the door if at least three components of the door are correctly detected by the component classifiers. The detecting system starts testing on images by selecting 18×18 sub-images starting from the down left corner of the image and shifting the window to all pixel locations of the image. Notice that the images are acquired by the on-board camera of the vehicle which is tilted up by 20° in order to keep always the top bar of the door in the images. In this way the vertical lateral bars of the door appear always in the down region of the images. For this reason and in order to speed up the detection phase, testing starts from this region. The sub-images extracted from this region of the image are tested by the bar classifier looking for the vertical bars of the door. The number of vertical detections is counted and if it is above a threshold then the vertical window containing these detections is classified as a vertical bar of the door. After that the corner classifier starts testing selecting 18×18 sub-images starting from the image rows above the shortest detected vertical bar. For each window candidate for containing a corner of the door, we count the number of detections within a specified neighborhood measuring a confidence factor given by the sum of the scores returned by the neural network. If the number of counted detections is above a threshold and the sum of the relative scores is the highest, then that window location is classified as an upper corner of the door. Finally the top bar of the door is detected by using the described technique used for detecting the vertical bars of the door. In fact the bar classifier again continues testing, selecting 18×18 sub-images from the image rows within a neighborhood of the corner locations. In conclusion a door is detected if combining the single components detected by the component classifiers at least three components are in the proper geometric configuration of the door. Fig.5 is a graphical representation of the described procedure.

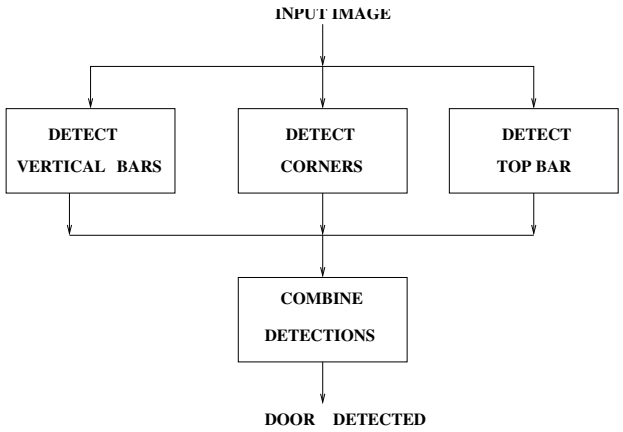


Figure 5: A graphical representation of the door-detecting system.

In order to evaluate the performance of the door detecting system 696 images containing a door have been acquired by different positions of different rooms of our office building. Applying the system on this set of test images has revealed a high detection rate of about 98.9%.

5.2 Door-Reaching

The trigger condition for the door-reaching behavior to become active is that the door is in the current image. Therefore the door-detecting system is charged to look for the door. Once the door has been detected the door-reaching behavior can start. Then the vehicle estimates its current state with respect to the door and chooses the optimal action for moving toward it according

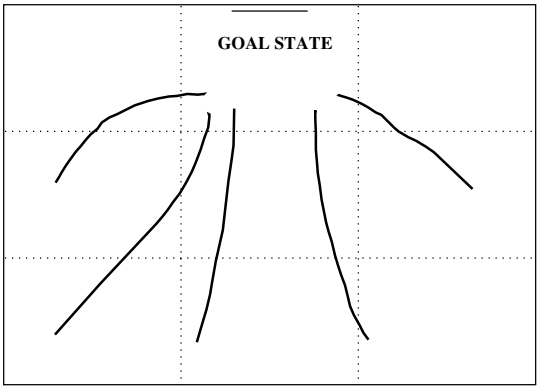


Figure 6: Real paths in the environment without obstacles.

to the policy learned in simulation. The optimal state-action association has been learned in simulation since a reinforcement learning method needs long learning time. Therefore learning the optimal policy directly in the real environment is impraticable whether for the long learning time and because dangerous situations can happen. However during navigation learning does not stop, but the system continues to update its policy because unforeseen situations can happen in the real environment. Figure 6 shows some real paths toward the door executed by our vehicle in a free-obstacle environment.

6 Conclusions

In this work we have presented two vision-based behaviors which realize the door-reaching task for an autonomous mobile vehicle. The robot has firslty to search the door in its environment and successively to move toward it until it is located adjacent to the door. Door-detecting has been realized by first detecting their constituent components and then verifying if they are in a proper geometric configuration. Recognizing the door by its components overcomes the fundamental problem of scaling which causes an increase in processing time. The realized door-detector performs very well. It is always able to detect the door in each room of our building. The door-reaching behavior has been realized by using an unsupervised learning approach. The robot learns the behavior associating the proper action to each encountered situation by a reinforcement learning method. The described behaviors constitute two bricks of a more complex architecture developed for our mobile robot.

References

[1] T. Poggio A. Mohan, C. Papageorgiou. Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):349–361, April 2001.

[2] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learning*, May/June 1996.

[3] G. Attolico, G. Cicirelli, and T. D'Orazio. Combining reactive behaviors for goal-oriented navigation. In *9th International Symposium on Intelligent Robotic Systems*, July 2001.

[4] R. Brooks. Intelligence without reason. Technical report, Massachusetts Institute of Technology, 1991. A.I.Memo 1293.

[5] P. Caironi and M. Dorigo. Training and delayed reinforcement in q-learning agents. *International Journal of Intelligent Systems*, 12(10):695–724, 1997.

[6] J.A. Freeman and D.M. Skapura. *Neural Networks Algorithms, Applications and Programming Techniques*. Addison Wesley, 1993.

[7] S. Baluja H. A. Rowley and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, Jan 1998.

[8] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In David Touretzky, editor, *Advances in Neural Information Processing Systems 2 (NIPS*89)*, Denver, CO, 1990. Morgan Kaufman.

[9] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, pages 311–365, 1992.

[10] S. Mahadevan, G. Theochaous, and N. Khaleeli. Rapid concept learning for mobile robots. *Autonomous Robots Journal*, 5:239–251, 1998.

[11] S.K. Nayar and T. Poggio. *Early Visual Learning.* Oxford University Press, 1996.

[12] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. *Int'l J. Computer Vision*, 38(1):15–33, 2000.

[13] C. Papageorgiou and T. Poggio. A trainable system for object detection. In *Computer Vision and Pattern Recognition*, pages 193–199, June 1997.

[14] Ioannis Pitas. *Digital Image Processing Algorithms.* Prentice Hall, 1992.

[15] M. V. Srinivasan and S. Venkatesh. *From living Eyes to Seeing Machines.* Oxford University Press, 1997.

[16] Kah-Kay Sung. Learning and example selection for object and pattern detection. Technical Report 1572, AI Lab, January 1996. Phd. thesis.

[17] C. J. Watkins and P. Dayan. Technical note - q-learning. *Machine Learning*, 8(3/4):323–339, 1992.

Iterative fuzzy clustering for detecting regions of interest in skin lesions

R. Cucchiara[1], C. Grana[1], M. Piccardi[2]

[1] D.S.I. University of Modena - Via Vignolese, 905 - 41100 Modena, Italy
{rita,grana}@dsi.unimo.it
[2] D.I. University of Ferrara - Via Saragat, 1 - 44100 Ferrara, Italy
mpiccardi@ing.unife.it

Abstract. Image analysis tools are spreading in dermatology since the introduction of dermoscopy (epiluminescence microscopy), in the effort of algorithmically reproducing clinical evaluations. Color-based region segmentation of skin lesions is one of the key steps for correctly collecting statistics that can help clinicians in their diagnosis. Nevertheless, an efficient and accurate region segmentation algorithm has not been proposed in the literature yet. This work proposes an iterative fuzzy c-means clustering algorithm based on PCA with the Karhunen-Loève transform of the color space. A topological tree is provided to store the mutual inclusions of the regions and then used to summarize the structural properties of the skin lesion. Preliminary experimental results are presented and discussed.

1 Introduction

Skin melanoma is a malignant pigmented skin lesion which needs particular medical attention. Its incidence as well as its associated mortality rate is rapidly increasing in developed countries. The consequences of a late diagnosis of malignant melanoma (MM) are very significant in terms of personal health, medical procedures and costs. Since it is not always easy for clinicians to distinguish between benign lesions and early MM, techniques like dermoscopy (DS, also called epiluminescence microscopy, ELM) have been developed to improve the diagnosis of pigmented skin lesions. DS is now a well recognised and established method to improve the clinical diagnosis of MM when used by dermatologists [1]. ELM is a non-invasive, in vivo technique which reveals features that are not perceivable by the dermatologist during his clinical observation. It renders the skin partially translucent and reveals the lower layers of the epidermis.

Systems for computer image analysis for melanoma diagnosis that use segmentation-based shape features have been developed [6]. However, as noted in [4], these features often do not correspond to known biological phenomena and do not model human interpretation of dermatoscopic imagery.

Most works in the literature concentrated on the separation of lesion from background, since it is obviously a critical preliminar step in any analytic procedure. Grayscale thresholding [7], color clustering [13], edge-finding [14] and non-linear

diffusion [4] have been proposed, while a metric for performance evaluation of the various techniques has been developed in [8]. Recently, it has been pointed out [4] that a segmentation stage should consider not only color similarity, but also spatial information.

Feature extraction has mostly focused on geometrical aspects of the lesion, and only a few works deal with other characteristics such as pigmented network [4,11] or border cut-off [3]. Even if color is a key feature in dermatoscopic diagnosis (the third of the ABCD criteria presented in [1]), not much effort was provided to adequately exploit color information.

In this paper we aim to provide an analysis of color regions, focusing our attention on their inclusion properties as suggested by the dermatologists' approach, and organize them in a structured representation based on a tree whose branches represent the inclusion property while nodes contain region informations.

2 Method

Our proposal is based on the work of P. Schmid [12] that proposed a fuzzy c-means segmentation over the two dimensional histogram of the first two components of the Karhunen-Loève transform of the color space. The only drawback of Schmid's proposal is that, lacking spatial information, the number of classes is automatically obtained by a statistical evaluation of the histogram and fails to identify small color areas in the original image, being their pixels included in larger surrounding clusters in the histogram. In this section, we shortly describe the whole process, focusing on our application.

The algorithm has been tested on ELM images, obtained by the use of an off-the-shelf dermatoscopic microscope and directly digitized to 640x480, 16 bit per pixel color images. The application software was developed with Microsoft Visual C++ on a standard Windows PC.

2.1 Preprocessing

In order to obtain a smoother outline for regions and to suppress details that are useless for the color area segmentation and sometimes only introduced by the acquisition process, we first convolve the image with a Gaussian kernel with standard deviation of one pixel. Then, since we want to obtain results similar to the dermatologist perception of color, we transform our images from the original RGB color coordinates into *CIE* $L^*a^*b^*$ color coordinates [9], a uniform color space in which equal distances mean, almost equal perceived chromatic difference. Working in a uniform color space gives us a more reliable metric for the clustering phase.

2.2 Karhunen-Loève transform

The diagnosis of pigmented skin lesions, regarding color properties, relies on no more then a dozen color classes, and this hints that a color reduction method can be

used having no worries about masking relevant properties. A common way used for the reduction of data, is to discard components with less discriminating power by projecting data onto principal components with the Karhunen-Loève transform, also called Hotelling transform or Principal Component Analysis [5]. This transform consists of the projection of the vectors to be reduced on the eigenvectors of their covariance matrix, computed using the following equations:

$$\mathbf{m}_x = \frac{1}{M}\sum_{k=1}^{M}\mathbf{x}_k \tag{1}$$

$$\mathbf{C}_x = \frac{1}{M}\sum_{k=1}^{M}\left(\mathbf{x}_k \cdot \mathbf{x}_k^T - \mathbf{m}_x \cdot \mathbf{m}_x^T\right) \tag{2}$$

where M is the number of data samples and $\mathbf{m}_x$ is the mean vector of the image. We define a matrix **A** whose rows are the eigenvectors of matrix **C** ordered by decreasing eigenvalue. The Karhunen-Loève transform of vector **x** is then defined by

$$\mathbf{y} = \mathbf{A} \cdot \left(\mathbf{x} - \mathbf{m}_x\right). \tag{3}$$

In our case the three components (L^*,a^*,b^*) are to be reduced to just two components and the samples are the image pixels. Processing with the Karhunen-Loève transform does not change the uniformity of the chosen color space [12].

We then discard the third component of the transform and rescale and quantize the first two components to 256 levels in order to store each transformed pixel in a byte-type variable. From this two components a 2D histogram is computed and used for the next steps.

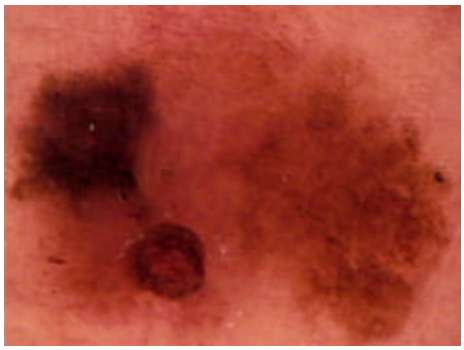


Fig. 1. Example melanocitic lesion, after the Gaussian filter preprocessing.

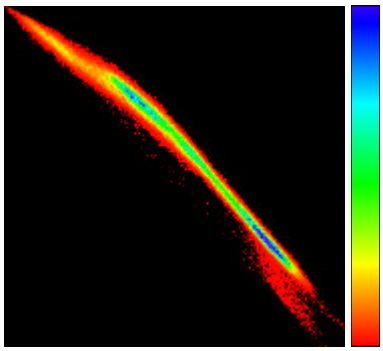


Fig. 2. 2D Histogram of the two KL components. Origin is top-left, the 1st component grows horizontally, the 2nd vertically. Pseudo color scale is shown on the right side of the image.

2.3 Fuzzy c-means clustering

The fuzzy c-means (FCM) algorithm is a robust clustering technique, especially efficient for the cluster center computation. c being the number of classes, we use the following two recurrent equations [12]:

$$U_{ik} = \left(1 + \sum_{\substack{j=1 \\ j \neq i}}^{c} \left(\frac{\left\| \mathbf{x}_k - \mathbf{v}_i \right\|^2}{\left\| \mathbf{x}_k - \mathbf{v}_j \right\|^2} \right)^{\frac{1}{m-1}} \right)^{-1} \tag{4}$$

$$\mathbf{v}_i = \frac{\sum_{k=1}^{M} (U_{ik})^m \cdot \mathbf{x}_k}{\sum_{k=1}^{M} (U_{ik})^m} \tag{5}$$

where U_{ik} is the fuzzy membership of $\mathbf{x}_k$ to class i and $\mathbf{v}_i$ is the i^{th} class center. The weighting exponent m defines the fuzziness of the membership values and in our application has always been set to 3.

Details of characteristics and properties of the FCM algorithm can be found in [10] and in [2].

2.4 Topological tree

The main idea behind structuring the region decomposition of the lesion in a tree derives from the observation that the pixels group easily into two clusters, one brighter, corresponding to the healthy skin and the other darker, corresponding to the lesion. Iterating the same concept on the lesions's interior could provide us with an informative view of the internal structure. Color is evidently the main clinical criterion in this type of analysis; however, a blind segmentation in multiple, differently colored areas could lead to segmentation into areas of poor interest (see Fig. 3, where the area corresponding to the healthy skin has been improperly split into two parts).

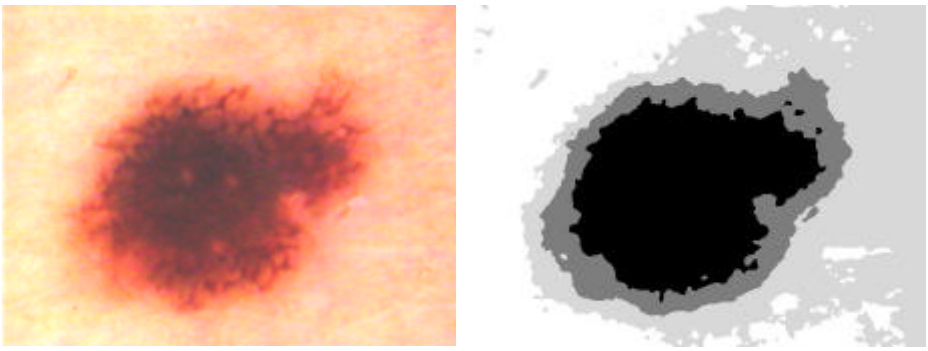


Fig. 3. Example of multiple-region color segmentation with no spatial constraints

We then should formally define two main concepts that will be used in the rest of the text.

Def. 1: we define *skin region of interest* (skin ROI) a set of pixels of the skin image exhibiting three properties: uniform color, connected pixels and significant area.

Given this definition, we must operatively define acceptance thresholds for the color uniformity, computed in the 2D histogram color space, for the area size and also for the degree of connectivity. About this aspect, some morphological operations can be exploited in order to obtain closure of regions with missing pixel and, at the same time, to disjoint loosely connected regions.

Def. 2: a *Topological Tree* (TT) is a tree whose nodes are skin ROIs and the arcs topological inclusion relationships between skin ROIs.

The tree construction follows a recursive procedure that starts with the analysis of the KL transform components to compute the 2D histogram relative to the region. By using fuzzy membership, it is possible to select two zones with strong belonging either to one or the other of two clusters ($U_{ik} > T_U$), and an intermediate region of non assigned pixels. We then consider the two "reliable" regions and their reciprocal position in order to select the next step of the algorithm. If it is possible to select an outer region, this is added to the tree while the other is further processed together with the non assigned pixels;otherwise, both regions are further processed. The intermediate area is added in turn to the region that will be analyzed at the next step.

By iteratively applying the FCM, a topological tree of the regions is obtained, which can contain an overlapping segmentation (because of the unclassified area) and that can be used to summarize the topological properties of the lesion itself.

A pseudo-code of the algorithm follows.

```
AnalyzeRegion (region R, node N)
{
    if (not StopCondition (R)) {
        [C1,C2] = FCM (R);
        [Cint,Cext] = VerifyInclusion ([C1,C2]);
        if (exists([Cint,Cext])) {
            Cres = R-Cint-Cext;
            Nnew = AddNodeToTree (Cext,N);
            for each C in ConnectedComponents(Cint)
                AnalyzeRegion (C+Cres,Nnew);
        }
        else {
            AnalyzeRegion (R-C1,N);
            AnalyzeRegion (R-C2,N);
        }
    }
    else
        AddNodeToTree (R,N);
}
```

3 Results

To evaluate the performance of our system, we randomly chose 16 lesions from a dermatologist's database (8 lesions are benign and 8 lesions are malignant melanoma), as shown in Fig. 4.

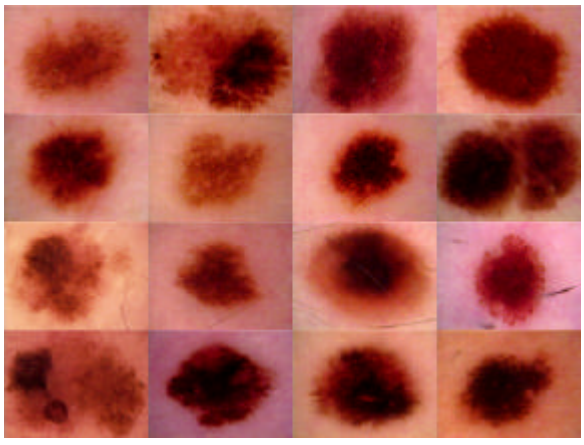


Fig. 4. Example lesions extracted from a dermatologist's database

Lesions were automatically segmented with 10 sets of thresholds for size, fuzzy membership and connectivity range and the best overall result was selected evaluating region significance by a dermatologist. Results are shown in Fig. 5.

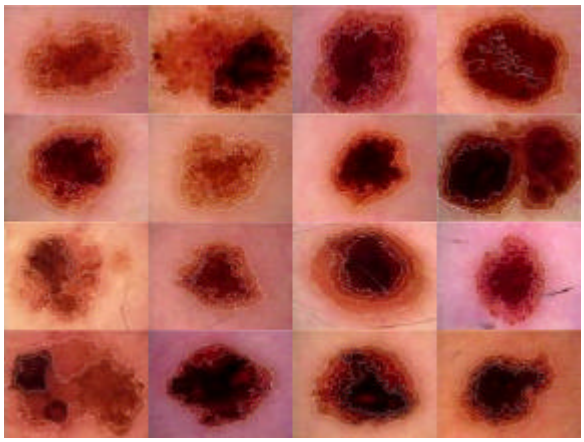


Fig. 5. Best automatic segmentation results

Overall performance was considered quite good based on dermatologist's opinion, finding it a useful hint about regions of interest. Cases of wrong segmentation were produced by usatisfactory evaluation of region inclusion, since regions should be considered included in some cases, even if they're not perfectly included from a topological point of view; different topologial crtieria are under evaluation. However, minor manual correction on the selection path often produces satisfactory results.

4 Conclusions

The main goal of this work was to present an iterative fuzzy clustering for segmenting significant color regions in skin lesion images.

The proposed algorithm aims to extract the skin ROIs by creating a topological tree (TT), whose nodes are skin ROIs and the arcs are topological inclusion relationships between skin ROIs.

The proposed technique presents some interesting novelties and is able to provide the user with a view of the lesion's structure. However, some preliminary tests show that the algorithm should still be refined; in particular, the topological inclusion definition and its use for building the TT, though flexible and promising, must be further investigated to prevent erroneous classification in some limit cases.

An interesting future direction is the study of a reliable metric to evaluate the results, starting from the metric presented in [8] and integrating dermatology's suggestions and knowledge.

Acknowledgements

The authors would like to thank Prof. Stefania Seidenari and Dr. Giovanni Pellacani from the *Dipartimento di Scienze Neuropsicosensoriali* of University of Modena for their help and for providing the test images.

References

1. Z. B. Argenyi, “Dermatoscopy (epiluminescence microscopy) of pigmented skin lesions,” Dermatologic Clinics, vol. 15, no. 1, pp. 79-95, Jan. 1997.
2. R.N. Dave, “Boundary Detection through Fuzzy Clustering,” Invited Paper, IEEE International Conference on Fuzzy Systems, San Diego, California, March 8-12, pp. 127-134,1992
3. G.R. Day, “How blurry is that border? An investigation into algorithmic reproduction of skin lesion border cut-off,” Computerized Medical Imaging and Graphics, 24 (2000), pp.69-72
4. Matthew G. Fleming, Carsten Steger, Jun Zhang, Jianbo Gao, Armand B. Cognetta, Ilya Pollak, and Charles R. Dyer. “Techniques for a structural analysis of dermatoscopic imagery.” Computerized Medical Imaging and Graphics, 22(5):375-389, 1998.
5. R.C. Gonzales and P.A. Wintz, Digital Image Processing. Reading, MA: Addison-Wesley, 1992.
6. A. Green, N. Martin, J. Pfitzner, M. O'Rourke, and N. Knight. “Computer image analysis in the diagnosis of melanoma”. Journal of the American Academy of Dermatology, Vol. 31, pp. 958-964, December 1994.
7. Gutkowicz-Krusin D, Elbaum M, Szwaykowski P, Kopf AW. Can early malignant melanoma be differentiated from atypical melanocytic nevus by in vivo techniques? Part II. Automatic machine vision classification. Skin Res Technol 1997;3(1):15-22.
8. G.A. Hance, S.E. Umbaugh, R.H. Moss, W.V. Stoecker, “Unsupervised Color Image Segmentation with Application to Skin Tumor Borders,” *IEEE Engineering in Medicine and Biology*, Vol. 15, No. 1, 1996, pp. 104-111.
9. J.K. Kasson, W. Plouffe, An analysis of selected computer inter-change color spaces, ACM Trans. Graphics 11 (4) (1992) 373–405.
10. Y. W. Lim and S. U. Lee. “On the color image segmentation algorithm based on the thresholding and the fuzzy c-means techniques”. Pattern Recognition, Vol. 23, No. 9, pp. 935-952, 1990.
11. Ph. Schmid and S. Fischer. "Colour Segmentation for the Analysis of Pigmented Skin Lesions". In *Proceedings of the Sixth International Conference on Image Processing and its Applications (IPA'97)*, vol. 2, pp. 688-692. The Institution of Electrical Engineers, Dublin, July 1997.
12. P. Schmid. "Segmentation of Digitized Dermatoscopic Images by Two-Dimensional Color Clustering". IEEE Transactions on Medical Imaging, 18(2):164-171, February 1999.
13. S.E. Umbaugh, R.H. Moss, W.V. Stoecker, G.A. Hance, “Automatic Color Segmentation Algorithms: With Application to Skin Tumor Feature Identification”, IEEE Engineering in Medicine and Biology, Volume 12, Number 3, September 1993, pp. 75-82.
14. L. Xu, M. Jackowski, A. Goshtasby, C. Yu, D. Roseman, S. Bines, A. Dhawan, A. Huntley, “Segmentation of Skin Cancer Images,” Image and Vision Computing, vol. 17, no. 1, 1999, 65-74.

Real-Time, Low Level Image Processing on SIMPil - an Embedded SIMD Architecture

Antonio Gentile[1,2], D. Scott Wills[1], Josè L. Cruz-Rivera[3], Filippo Sorbello[2]

[1]School of Electrical and Computer Engineering, Georgia Institute of Technology
Atlanta, Georgia 30332-0250
scott.wills@ece.gatech.edu
[2]Dip. Ingegneria Automatica e Informatica, University of Palermo
V.le delle Scienze, 90128 Palermo, Italy
{antonio.gentile, filippo.sorbello}@unipa.it
[3]School of Electrical and Computer Engineering, University of Puerto Rico in Mayagüez
Mayagüez, Puerto Rico 00681-9042
jcruz@ece.uprm.edu

Abstract. Current and future real-world workloads will be increasingly dominated by media-centric applications such as scene visualization and analysis, content-based video storage and retrieval, real-time imaging, and teleconferencing. This class of applications exceeds the computational capabilities of current microprocessor and digital signal processor (DSP) architectures. This paper presents the SIMD Pixel Processor (SIMPil) as a candidate architecture for real-time, low level image processing applications. SIMPil offers high performance, high energy efficiency computation across a broad range of image and video applications, spanning from spatial and morphological filtering to image and video compression. Simulation results indicate sustained operation throughput in the range of 100-1000 Gops/sec, thus supporting real-time execution of low level vision tasks and allowing for more room for subsequent higher level vision tasks.

1 Introduction

A new class of media-centric applications is emerging with increasing demands for portable computation and communication devices that handle video, electronic mail, teleconferencing, and speech recognition. These applications have computational requirements that make them unsuitable for implementation using conventional processing architectures, which poorly exploit the inherent data parallelism of these applications [1]. They also demand real-time processing of high bandwidth I/O streams that contain little temporal reference locality, which renders large (and expensive) data caches useless. Finally, media-centric applications are often realized in portable systems, thus imposing more restrictive size, weight, and power requirements than desktop computing. SIMD (Single Instruction Multiple Data) architectures are well suited for media-centric applications because of the tremendous amount of data parallelism available in the target workload. Processing elements are

distributed and co-located with the data I/O to minimize storage and data communication requirements.

This paper presents a suite of real-time image and video-processing programs implemented on the SIMD Pixel Processor (SIMPil), a SIMD architecture designed to support compact, area and power efficient processing for portable multimedia systems. This application suite, developed and evaluated through simulation, represents the target workload for future portable multimedia supercomputing systems. Computation workload, throughput, and memory storage over the selected workloads are considered in the analysis. Simulation results show that while I/O and computation throughput are difficult requirements in media-centric applications, systems based on the SIMPil architecture allow sustained performance on the order of 500-1500 Gops/sec.

2 Related Research

Several scientific supercomputers have been used for image processing applications. SIMD architectures like the Connection Machine models CM-1 [2] and CM-2 [3], the MasPar [4] and the GAPP [5], were designed for a more general set of applications and achieve performance at the expense of cost, poor data bandwidth, and lack of portability. MGAP [6] and ABACUS [7] are examples of fine grain parallel processing architectures that address portability issues. However, I/O operations remain a bottleneck for speed in these systems. Furthermore, since they are reconfigured for specific applications, they suffer from configuration latency as well as potentially low resource utilization. Specialized VLSI systems for image processing, such as the Silicon Retina [11] and FET-SEED [12] are systems in the opposite end of the design spectrum. These systems sacrifice flexibility and computing power for a more compact implementation at a lower cost. Application specific systems lack the programmability required in the fast-changing standards of multi-media.

Other architectures specialized for media processing are digital signal processors (DSPs) such as the TI TMS320C80 or TMS320C6000 families [8], and media processors like MPACT [9] or the latest SH-5 SuperH architecture [10]. Microprocessor manufacturers are also aware of the increased importance of media applications and have added multi-media extensions to their architectures. Both DSPs and microprocessors lack the computational power required to execute most media applications in real-time, and their designs are not well suited for the stream nature of media applications nor for the low power requirements of battery operated devices. Vector supercomputers are capable of delivering the computational demand of media applications. However, they are seldom designed for portability, low power, and low cost. While there are architectures, such as the IRAM [13], that targets portable multimedia supercomputing, they often do not have tight coupling between processor core and data source. Even with high-speed serial lines, there is still considerable communication distance to bring data to the processor core. External sensor data bandwidth remains a potential bottleneck for vector-like machines.

3 SIMPil Processor Array

The SIMPil architecture employs area-array I/O to access directly to the processors. The system operates on its target image-processing applications with a dense array of SIMD processing elements (PEs), harnessing the inherent data parallelism. While area-array I/O is not an essential element in a portable multimedia supercomputer, the SIMPil system employs area-array I/O as the main source of data I/O. Other fast peripheral I/O mechanisms are also suitable if the bandwidth requirements are satisfied. The SIMPil architecture consists of an array of SIMD processors, shown in Figure 1. Each processing element has a RISC-like data path with specialized units for SIMD operation and mechanisms for the area I/O data streams.

The major functional units are:

- ALU and barrel shifter
- Multiply-accumulator unit with double precision accumulator
- Three-ported general purpose and special registers
- Local memory (256 maximum words)
- Communication and serial I/O units
- Masking unit to control PE activity.

The instruction set architecture allows a single processing element (PE) to address a subarray of image sensors. Each processor incorporates an analog to digital converter to convert light intensities, incident on the sensors, into digital values. The SAMPLE instruction simultaneously collects all sensor values and makes them available for further processing. Processing elements are capable of communicating with their four nearest neighbors (North, East, West and South) by using a communication unit. While most SIMD architectures require a specialized set of registers associated with each of the four nearest neighbors (NEWS registers), the SIMPil architecture allows any entry in the register file to be the source or destination for its nearest neighbors.

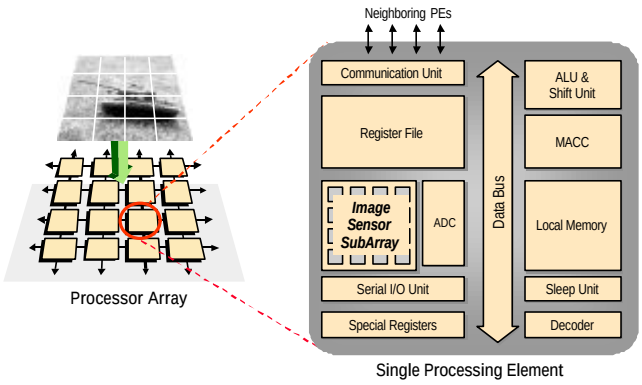


Figure 1. Block diagram of the SIMPil processor array and processing element

The SIMPil architecture has been implemented in two functional prototypes: *SIMPil8* and *SIMPil16*. The first prototype, SIMPil8, is an 8-bit implementation, designed to demonstrate the feasibility of direct coupling of a processing core with

sensor devices. The second prototype, SIMPil16, was designed as a complete digital-processing node to handle the workload of portable multimedia supercomputing.

A future target system is used in this paper for performance analysis. This system is able to deliver an unparalleled performance with 4096 PEs integrated into a single monolithic device running at 500 MHz. The operating clock frequency is chosen to ensure low power regime of operation for the digital circuitry. In the target system, a 256×256 array of image sensors is integrated with the system, with each PE directly mapped to a 4×4 pixel sub-array. This system can deliver a peak throughput of about 1.5 Tops/sec in a monolithic device, enabling image and video processing applications that are currently unapproachable using today's portable DSP technology.

Applications for the SIMPil system can be programmed using the SIMPil Simulator [14]. This software tool is an instruction level simulator, running under Windows95™. The SIMPil Simulator allows editing, assembling, executing, and debugging parallel applications in a single integrated workbench for the SIMPil architecture.

4 SIMPil Application Suite

To evaluate the set of architectural design choices, an application test suite has been developed and simulated using the SIMPil Simulator. These applications are selected among model problems to closely represent the target workload, to exercise key-design characteristics such as I/O and computational bandwidth, and to enable an accurate evaluation of the system design. This approach finds general consensus in the research community [15, 16], which suggests the use of specific test suites to evaluate SIMD processors.

Table 1. Application implemented in the suite used to evaluate SIMPil

Application Category	Applications
Image Transforms	Discrete Fourier Transform, Discrete Cosine Transform, Discrete Wavelet Transform, Image Rotation
Image Enhancement	Intensity Level Slicing, Convolution, Magnification, Median Filtering
Image and Video Compression	Quantization, Vector Quantization, Entropy Coding, JPEG Compression, Motion Estimation, MPEG Compression
Image Analysis	Morphological Processing, Region Representation, Region Autofocus, K-means Classification

The applications are selected to cover a wide spectrum of key tasks in image and video processing domains and are listed in Table 1. They include image transforms (discrete cosine transform, discrete wavelet transform, and discrete Fourier transform), image enhancement applications (intensity level slicing, spatial filtering, magnification), image analysis tools (edge detection, morphological filtering, region representation, and region classification), and compression standards for still-image (JPEG) and motion picture (MPEG). The suite of applications implemented on SIMPil is by far the largest reported in literature for an experimental system. All applications are described briefly in the next sections. Wherever possible, a comparison is offered against other parallel platforms in the appendix.

4.1 Image Transforms

Discrete Fourier Transform. A two-dimensional, complex DFT has been implemented based on a matrix multiplication algorithm. The original image is transformed row first then columns. The weight matrices are preloaded, and they are rearranged to support the nearest-neighbor communication scheme available on SIMPil. Fixed-point arithmetic is used to implement the algorithm. A comparison with other parallel architectures is offered in Table 4.

Discrete Cosine Transform. A 2D-DCT has been implemented on SIMPil using the row-column method, in which a one-dimensional DCT is applied to the rows and then columns. The importance of this application is particularly evident in real-time video compression and decompression, where DCT operations account for 25%-50% of CPU time without dedicated hardware supported. Differently from the DFT, the DCT remains in the real numbers domain. Cosine terms are used as coefficients and are preloaded into the system. Fixed-point arithmetic is used to implement the algorithm. A comparison with other parallel architectures is offered in Table 5.

Discrete Wavelet Transform. The discrete wavelet transform (DWT) is a mathematical tool, which converts 2D spatial image data into wavelet space for compression. The highly regular structure of the DWT makes it well suited for fine-grained parallel systems like SIMPil. Standard Daubechie's W_4 filter coefficients are used as the low and high-pass filter pair. A row-column process is used to filter and decimate rows first, and then columns. The entire sequence is repeated until the desired decomposition level is attained. In this implementation, the original image is decomposed into 61 bands at a rate of 0.4 Gpixels/sec. A comparison with other parallel architectures is offered in Table 6.

Image Rotation. A parallel rotation algorithm (ROT) has been implemented on SIMPil. A rotation angle is first calculated, and then a sequence of shear transforms is performed for each pixel. Because shear transforms operate in a single direction (either horizontally or vertically), they are efficiently implemented using horizontal or vertical shifts. Since shear transforms expand the image, and the system resolution is fixed to given system size, lossless image compression is first performed by creating virtual processing elements (VPEs) in the local memory of selected PEs to store its neighbor's pixel data. The shear transforms are then applied on the compressed data, and the rotated image is finally restored to the original resolution. Ninety-degree rotations can be seen as a special case rotation, which can be performed by row-column substitution. In the parallel implementation, SIMPil PEs are divided into concentric rings, and rotation is accomplished with the rings rotating around their axes of an angle proportional to the ring diameter. This ring rotation execution algorithm is more than a order of magnitude faster than a 90° rotation via shear transforms, and executes in about 20 μs. A comparison with similar 45° rotation implementations on other parallel architectures is offered in Table 7.

4.2 Image Enhancement

Intensity Level Slicing. Intensity Level Slicing (ISLICE) is used when the feature of interest is contained within a given gray level range. This application is a *point-operation,* where the operation is performed on each pixel without inter-processor communications. It typically involves an ALU operation followed by an assignment operation.

Convolution. Convolution-based filtering (CONV) has been implemented on SIMPil to perform different filtering operations, such as shadowing, smoothing, and edge detection. The result image of the convolution is accumulated and stored in the memory of each PE as 4x4 sub-images. The filter mask elements are broadcast one at a time to every PE. All calculations requiring the mask element are performed before the next element is broadcast. Each PE multiplies the mask element by the corresponding pixel values from the original image and accumulates the results to the corresponding result image pixel locations. Pixel values from the original image are moved to the PE that needs them for the current mask element. In this way, processing time scales linearly with the mask size. A comparison with other parallel architectures is offered in Table 8 for 3x3 masks.

Median Filtering. Median Filtering (MED) is useful to remove binary noise from an image while preserving spatial resolution. A 3x3 window is used for this implementation. The algorithm works by replacing each pixel in the image with the median value in the window. A comparison with other parallel architectures is offered in Table 9.

Magnification. Magnification and interpolation (MAGN) are operations typically performed in cameras and camcorders when a digital zoom is used. A 16x magnification by replication has been implemented on SIMPil. Inter-processor communications play a big role for this application, as 50% of the instruction executed are used to transfer pixel data among PEs. The computational bandwidth is 707 Mpixels/sec.

4.3 Image and Video Compression

Quantization. Quantization (QUANT) is an important stage of image and video compression standards, such as JPEG and MPEG. The implementation uses a default quantization matrix to generate the quantized DCT values. This fully parallel operation is performed on 8x8 arrays of pixels at a rate of about 8 Gpixels/sec and is independent of the image size.

Entropy coding. Entropy coding (HUFF) is performed on the quantized coefficients using the Huffman method. This process is characterized by lower system utilization (25%) because only one PE (of the four PE contributing to an 8x8 block) is active at any given time. The execution time of the coding process depends on the number of non-zero values in the quantized coefficient matrix. Each active processor works on a sub portion of the final bitstream, which is constructed in a serial post-processing stage. The output bitstream elements are coded at a rate of 4.6 Gpixels/sec.

Vector Quantization. In its basic scheme, vector quantization is a mapping function, which associates vectors of input data to a representative vector (*codevector*), chosen

within a previously learnt dictionary (*codebook*). A scalar index is then used to reference the chosen codevector in the dictionary and encode the input vector. The compression ratio depends on the cardinality of the codebook, usually much smaller than that of the input domain. In the 2D case, non-overlapping vectors are extracted from the input image by grouping a number of contiguous pixels, in order to retain available spatial correlation of the data. In the implementation on SIMPil, a 256-word codebook is used to achieve a 0.5 bit per pixel encoding of 256 gray-level images. The corresponding computational bandwidth is of the order of 1 Kops/pixel, or about 2 Gops/sec for video rate processing. A key enabling role is played by the toroidal structure of the interconnection network, which enables short communications between the processing elements in the opposite sides of the mesh. In this implementation, the input blocks are compared against the codebook in a parallel systolic fashion. This results in a large speedup and in a high degree of concurrency. A comparison with similar implementations on other parallel architectures is offered in Table 10. Performance figures are adjusted to account for the different implementation choices (codebook size, cost function, and image size.)

JPEG Compression. A sequential DCT-based JPEG encoding scheme has been implemented on SIMPil, building upon some of the applications described above. The color transformation stage converts the RGB components into their luminance and chrominance components. The forward DCT stage is then applied on image blocks. The resulting DCT coefficients are then quantized using the quantization tables defined by the JPEG standard. Finally entropy coding is performed on the quantized coefficients using a Huffman coding method. Only the luminance component of the image has been implemented on SIMPil. A post-processing stage is used to combine the simulator output into the JPEG compressed bitstream. JPEG compression is performed on SIMPil at a rate of 1.2 Gpixels/sec.

Motion Estimation. Motion Estimation (ME) is a core building block in several video compression standards, such as MPEG, H.263 and similar. The basic idea is to determine the motion vectors for the reference blocks in the current frame. A *macroblock* is a 16 by 16 pixel region, which is used as the basic block in the motion estimation process. For each reference block in the current frame, a search area in the previous frame is explored in order to find the closest matching block according to a selected error criterion. Three levels of parallelism are exploited by this implementation: (i) the computations of motion vectors for each reference macroblock can be calculated concurrently, as they are independent from one another; (ii) for a given reference block, all motion vectors can be calculated in parallel; (iii) the matching criterion, which determines the similarities among each pixel in the reference block and the corresponding one in the candidate block, can be computed in parallel. Motion estimation for 720x480 pixel images performs on SIMPil at a rate of 9 Mblock/sec. A comparison with other parallel architectures is offered in Table 10.

MPEG Compression. The MPEG-1 video encoder has been implemented on SIMPil as an example to demonstrate its effectiveness for real-time video compression. Each frame in the input stream is encoded differently, according to the different picture types (I-picture, P-picture and B-picture) as defined in the standard. I-pictures, P-pictures, and B-pictures are compressed on SIMPil at a rate of 27.5M, 5M, and 2.9M macroblocks/sec respectively, for an image size of 720x480 pixel. A typical I B B P B B P B B P B B P I B B P picture sequence can be compressed at a rate of 30 frames in

10msec, well within the real-time bound, thus demonstrating the system potential for real-time execution of the MPEG-1 compression.

4.4 Image Analysis

Morphological Processing. Morphological image processing refers to the study of the topology or structure of objects from their 2D spatial representation. In the general case, binary images are morphologically transformed by passing a *structuring element* over the image in a process similar to convolution. At each pixel position, a specified logical function is performed between the structuring element and the underlying image. Depending upon the size and content of the structuring element, different effects such as edge smoothing, edge detection (IED), and skeletonization (SKL) can be produced from erosion and dilation operations. A 3×3 structuring element is used to implement the morphological operations on SIMPil. The implementation of erosion and dilation is based on shifted versions of the original image. Edge detection and skeletonization execute on SIMPil at a rate of 43 Gpixels/sec, and 2.3 Gpixels/sec, respectively.

Region Representation. A quadtree based decomposition algorithm (*QTREE)* is used to subdivide the original image into smaller image quadrants. Near neighbor communications are utilized to remove data broadcast and to maintain minimal buffer storage. Region representation executes at a rate of 12 Gpixels/sec.

Region Autofocus. The region autofocus application (*REGION*) is used to isolate a small region of interest within a given image based on some selected features, and it provides a magnified version of this image for further analysis. Specifically, the region of interest is first identified based on the presence of the object signature, and the image is binarized. A quadtree-based region identification stage is then performed, to determine the position of the region of interest in the image. Finally, the region is enlarged to focus on the object of interest, using a magnification stage. Region autofocus is performed on SIMPil at a rate of 662 Mpixels/sec.

Classification Techniques. K-means clustering is a commonly used technique to segment large images into specific objects or areas of interest. The implemented algorithm takes as input the binarized image to be analyzed and the number of clusters to be constructed (K). The algorithm first identifies all possible pixel clusters in an image according to a particular threshold metric (*CLUSTERING*). All pixels that satisfy the threshold condition (*live* pixels) are grouped into N clusters based on a connectivity criterion, and then the clusters are labeled accordingly (*LABELING*). The toroidal interconnection network is utilized to allow efficient execution, as all PEs with live pixels can communicate with their neighbors in order to grow a cluster of pixels. The PEs can then collaborate in each centroid computation of the N identified clusters (*CENTROID*). Once the centroids for the N ($N > K$) clusters are determined, the algorithm must map these clusters into K new clusters (*CLASSIFICATION*). K clusters are chosen out of the N identified ones, and the remaining N-K clusters are mapped onto it. The centroids of these K clusters are denoted as seeds. The remaining N-K clusters are mapped by minimizing the distance between their centroids and the selected seeds. The implemented algorithm chooses the seeds randomly. After each one of the N-K clusters is remapped, the centroid of the newly expanded cluster is

calculated. This new centroid substitutes the original cluster's centroid as one of the *K* seeds. The algorithm continues until no centroid changes cluster. At the end, there will be *K* main-clusters that, taken together, will hold the *N* original clusters. Each of the *K* clusters will have a specific centroid calculated using all internal clusters.

5 Performance Analysis

This section describes the characterization and performance of the SIMPil architecture over the application suite. The applications have been simulated using the SIMPil simulator, described earlier in the paper. An instruction histogram is tabulated to determine the average workload for the applications.

5.1 Instruction Histogram

Each application is profiled in terms of the amount of cycles spent in the key functional types: arithmetic-logic-unit (ALU), memory (MEM), communication (COMM), PE activity control unit (MASK), and image loading (PIXEL). The ALU and MEM cycles are computation cycles while COMM and MASK cycles are necessary for data distribution and synchronization of the SIMD processor array. Scalar instructions operate in the array controller unit, and are considered the serial portion of the algorithm. They are executed concurrently to the vector instructions, which represent the parallel portion of the algorithm operated on the processor array.

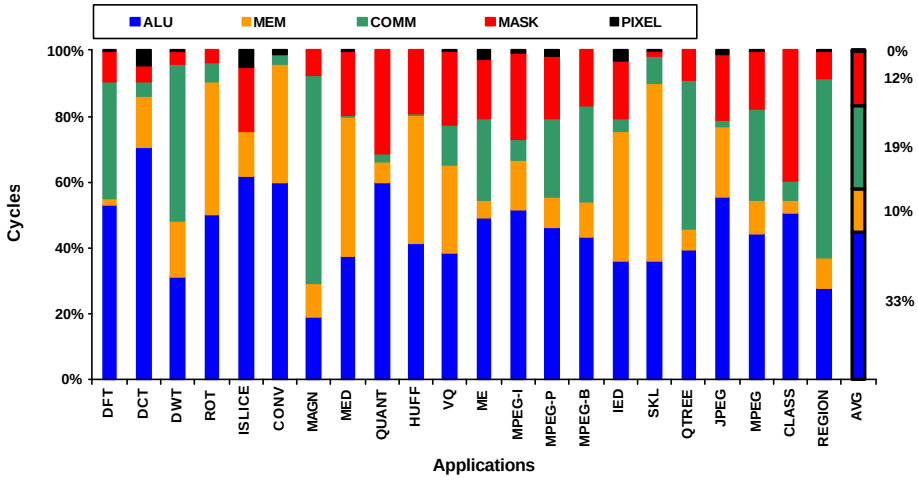


Figure 2. Workload characterization for the applications

The average workload for the entire application suite is depicted in Figure 2 As expected, sensitive shares of the execution time are expended in ALU and MEM operations. Latency for image loading is negligible, indicating that all selected applications are computationally intensive. The application suite shows a high degree of parallelism, as the serial component (Scalar) represents about 30% of the total

instruction count. The SIMD programming model harnesses data parallelism by delegating work to the processor array.

5.2 Computation Throughput

Table 2 summarizes performances for the implemented applications. A 4,096 PEs target system was used for the simulations, operating on 256x256 pixels, with a 4x4 pixel subarray per PE. For each application, the worst-case scenario was used. System utilization (U) is calculated as the average number of active processing elements and it is relative to the total system size. Execution time (t_{exec}) and sustained throughput (Th_{sust}) are computed with reference to the 500 MHz target platform as follows:

$$t_{exec} = \frac{C}{f_{ck}}, \qquad \text{and} \qquad Th_{sust} = \frac{IC \cdot U \cdot N_{PE}}{t_{exec}},$$

where f_{ck} is the clock frequency, C is the cycle count for a given application, IC is number of instructions issued during the application (i.e., the dynamic instruction count), and N_{PE} is the number of processing elements in the PE array.

Table 2. Application performance comparison

Application	Memory Size		Instruction Count	Total Cycles	System Utilization [%]	Execution Time [μs]	Sustained Throughput [Gops/s]
	PE [Byte]	System [KB]					
DFT	16	64	3,027	4,297	72%	9	1,043
DCT	198	792	2,401	2,725	99%	5	1,783
DWT	112	448	17,796	25,333	76%	51	1,092
ROT	262	1,048	423,795	443,585	33%	887	638
ISLICE	48	192	330	330	90%	1	1,850
CONV	80	320	1,233	1,397	100%	3	1,808
MAGN	64	256	2,799	4,107	89%	8	1,236
MED	180	720	10,759	10,783	86%	22	1,759
QUANT	294	1,176	3,639	3,687	74%	7	1,506
HUFF	490	1,960	4,282	4,349	20%	9	411
VQ	102	408	70,701	75,325	90%	151	1,735
ME	500	2,000	54,118	61,934	93%	124	1,658
MPEG-I	434	1,736	17,582	18,874	68%	38	1,302
MPEG-P	500	2,000	82,973	95,042	84%	190	1,495
MPEG-B	512	2,048	160,843	189,357	87%	379	1,508
IED	40	160	539	551	92%	1	1,840
SKL	40	160	8,476	8,856	99%	18	1,950
QTREE	30	120	1,719	2,226	97%	4	1,538
JPEG	490	1,960	10,514	10,953	59%	22	1,163
MPEG	512	2,048	3,695,427	4,321,176	86%	8,642	1,503
CLASS	60	240	43,838	46,278	67%	93	1,308
REGION	142	568	4,848	6,663	92%	13	1,364

The simulations show that all applications execute in lower millisecond range, suggesting that they can be combined in stages to form complex applications, and are

still able to execute at full frame rates (30-60 fps, or 15-30 ms). The high system utilization achieved further indicates that the available data bandwidth and parallel execution schemes in the SIMPil architecture are suitable for image processing applications. Memory usage on SIMPil is kept to the minimum required for stream processing, which can be quantified in most cases to up to ten times the amount of storage required for a single frame. Because computation is performed as images arrive on the area I/O, large buffering storage is unnecessary. For most of the applications, memory usage is contained within 700 KB for 256x256, 8 bpp images, to account for the dynamic range required by the computations.

Traditional SIMD systems such as the MASPAR II delivers 68 Gops/s processing throughput with 16K PEs running at 12.5 MHz. If a 40-fold increase in clock frequency alone is assumed, an estimated 2.7 Tops/s processing capability may be possible. However, data bandwidth in the global router (1.2 Gbit/s) may easily limit the estimated performance. The proposed SIMD focal plane system incorporates the focal plane area I/O to deliver high data throughput to the processor array, maintaining high processor utilization. Without a high bandwidth communication structure, the application workload will likely shift towards *COMM*, *MASK* and *Scalar* instruction types because processors must wait for data to arrive. The percentage of time spent on useful work (*ALU* and *MEM* instruction types) will reduce, and processing latency will increase beyond real-time (30 fps) bounds.

SIMPil performance largely benefits from the stream-oriented mapping of image data directly to each PEs. In fact, a considerable throughput is achieved by keeping the PEs busy most of the time thanks to the spatially mapped data. The application simulations show the potential of processing image and video data streams on a parallel SIMD environment in a focal plane. The SIMPil architecture provides a suitable operating platform for current and future semiconductor technology.

6 Conclusions

A suite of media-centric applications, selected from important image and video-processing tasks, are developed and evaluated through simulation. The application suite shows that media applications not only require high computational and data throughput, but that they can be implemented using small amounts of on-chip memories. This observation supports the stream-based model for media applications, which emphasizes the use of many simple processing elements over few complex instruction-level or thread-level processors. The results also indicate that coupling between the processing elements and area I/O is crucial to provide an efficient way of exploiting data parallel computation. In this paper, a focal plane was used as an example of the tight coupling needed between the processing elements and the area I/O. The performance evaluation shows that the architecture meets the computational and data bandwidth requirements of media applications. The results in this paper prove that the SIMPil system is a suitable architecture for real-time execution of low level image and videoprocessing tasks.

References

1 K. Diefendorff and R. Dubey. “How Multimedia Workloads Will Change Processor IEEE Computer, Vol. 30, No. 9, September 1997, pp. 43-45.

2 L. W. Tucker and G.G. Robertson, "Architecture and applications of the connection machine," IEEE Computer, vol. 21, No. 8, August 1988, pp. 26-38.

3 "Connection machine model CM-2 technical summary," Thinking Machines Corporation, ver. 51, May 1989.

4 MasPar (MP-2) System Data Sheet, MasPar Corporation, 1993.

5 W. F. Wong and K. T. Lua, "A preliminary evaluation of a massively parallel processor: GAPP," Microprocess. Microprog., vol. 29, No. 1, July 1990, pp. 53-62.

6 M. J. Irwin, R. M. Owens, "A Two-Dimensional, Distributed Logic Processor," *IEEE Transactions on Computers*, v. 40, n. 10, 1991, pp.1094-1101.

7 M. Bolotski, R. Armithrajah, W. Chen, "ABACUS: A High Performance Architecture for Vision," in Proceedings of the International Conference on Pattern Recognition, 1994.

8 C. P. Feigel, “TI Introduces Four-Processor DSP Chip.” Microprocessor Report, March 28 1994, pp.22-25.

9 P. Kalapathy. “Hardware-Software Interactions on MPACT,” *IEEE Micro*, 1997, pp. 20-26.

10 P. Biswas, A. Hasegawa, S Mandaville, M Debbage, A. Sturges, F. Arakawa, Y. Saito, K. Uchiyama, “SH-5: The 64-Bit SuperH Architecture,” IEEE Micro, Vol. 20, No. 4, July-August 2000, pp. 28-39.

11 C. Mead, *Analog VLSI and Neural Systems*, Addison-Wesley, Reading Massachusetts, 1989.

12 C. B. Kuznia, A. A. Sawchuk, and L. Cheng, "FET-SEED Smart Pixels for Free-Space Digital Optics Systems," Optical Computing. 1995 Technical Digest Series, vol. 10, 1995, pp. 108-110.

13 D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, K. Yelick, “A Case for Intelligent RAM,” IEEE Micro, v.17, n. 2, 1997, pp. 24-44.

14 SIMPil Home Page, http://www.ee.gatech.edu/research/pica/simpil

15 M. C. Herbordt, A. Anand, O. Kidwai, R. Sam, C. C. Weems, "Processor/Memory/Array Size Tradeoffs in the Design of SIMD Arrays for a Spatially Mapped Workload," in *Proc. Of Computer Architectures for Machine Perception*, 1997, pp. 12-21.

16 K. Preston, "The Abington Cross benchmark Survey," *IEEE Computer,* v. 22, n. 7, 1989, pp. 9-18.

Appendix: Comparison Tables

System features such as number of processing elements, clock rate, and image size are reported for the different systems in Table 3. Comparisons are presented in terms of task execution time, pixel rate, and pixel per machine cycle, to account for the different implementation technologies. SIMPil is found to offer competitive computation throughput (pixel/cycle), but the architecture can offer more by increasing the number of processing elements. With its efficient operation and programmable core, SIMPil provides real-time supercomputing for a large suite of media-centric applications.

Table 3. List of architecural features for several parallel systems

	APx	BLITZEN	CLIP4	DAP	GAPP	iCPA	Kestrel	MasPar1	MasPar2	MGAP	MorphoSys	MPP	RaPiD	SIMPil	SliM
PEs	256	16 K	9, K	16 K	12 K	64	64	16 K	16 K	16 K	64	16 K	16	4 K	5 K
Clock [MHz]	12.5	20	40	12.5	10	100	33	12	12.5	25	100	10	100	500	25
Image Size	512x 512	128x 128	96x 96	64x 64	96x 108	640x 480	256x 256	512x 512	512x 512	128x 128	256x 256	128x 128	256x 256	256x 256	100x 50

Table 4. Performance comparison for DFT for different parallel architectures

	Platform		
	APx	MGAP	SIMPil
Execution Time [μsec]	60,000	2150	9
Performance (pixels/cycle)	0.4	0.02	15
Performance (Mpixels/sec)	4	1	7,626

Table 5. Performance comparison for 8x8 DCT for different parallel architectures

	Platform					
	MGAP	Kestrel	iCPA	RaPiD	MorphoSys	SIMPil
Execution Time [μsec]	8,637	368,700	41,020	656	215	5
Performance (pixels/cycle)	0.08	0.01	0.08	1	3	24
Performance (Mpixels/sec)	2	0.2	7	100	305	12,025

Table 6. Performance comparison for DWT for different parallel architectures

	Platform		
	MasPar1	MasPar2	SIMPil
Execution Time [msec]	5,750	15	51
Performance (pixels/cycle)	0.004	1	3
Performance (Mpixels/sec)	0.05	18	1,293

Table 7. Performance comparison for 45° image rotation for different parallel architectures

	Platform		
	MPP	BLITZEN	SIMPil
Execution Time [μsec]	192	47	887
Performance (pixels/cycle)	9	18	0.1
Performance (Mpixels/sec)	85	351	74

Table 8. Performance comparison for 3x3 convolution for different parallel architectures

	Platform								
	CLIP4	DAP	MPP	GAPP	MGAP	SliM	iCPA	RaPiD	SIMPil
Execution Time [μs]	16,050	336	80	300	57	7	1,730	655	3

Performance (pixels/cycle)	0.01	1	41	4	12	29	2	1	47
Performance (Mpixels/sec)	1	12	410	35	287	714	177	100	23,456

Table 9. Performance comparison for 3x3 median filtering for different parallel architectures

	Platform			
	MGAP	SliM	iCPA	SIMPil
Execution Time [μsec]	13	9	1,229	22
Performance (pixels/cycle)	50	22	3	6
Performance (Mpixels/sec)	1,260	556	250	3,039

Table 10. Performance comparison for vector quantization for different parallel architectures

	Platform			
	MasPar1	Inmos	MasPar2	SIMPil
Execution Time [msec]	1,790	4,800	585	151
Performance (pixels/cycle)	0.002	0.003	0.04	1
Performance (Mpixels/sec)	0.02	0.06	0.45	435

Table 11. Performance comparison for forward motion estimation for different parallel architectures

	Platform		
	RaPiD*	MorphoSys*	SIMPil
Execution Time [μsec]	1,184	10,444	124
Performance (pixels/cycle)	0.6	0.1	1
Performance (Mpixels/sec)	55	6	529

* Results provided are based on an 8x8 block size

A vision agent in a distributed architecture for mobile robotics

Antonio Chella[1], Massimo Cossentino[2],
Ignazio Infantino[2], and Roberto Pirrone[1]

[1] Dipartimento di Ingegneria Automatica ed Informatica, Universitá di Palermo,
Viale delle Scienze, 90128, Palermo (PA), Italy
{chella, pirrone}@unipa.it
[2] Dipartimento di Ingegneria Elettrica, Universitá di Palermo,
Viale delle Scienze, 90128, Palermo (PA), Italy
{cossentino,infantino}@csai.unipa.it
http://www.csai.unipa.it

Abstract. An approach to the design and implementation of a vision agent inserted in a generic multi-agent architectures for mobile robotics is presented, that is based on the Unified Modelling Language. The main goal of the work is to provide a framework to perform a rigorous agent-based design process for cognitive architectures both in the case of a single robot, and in a multi-robot scenario. Details of the methodology, system implementation using FIPA-OS environment, along with real and simulated experiments are reported.

1 Introduction

In recent years, mobile robots have been involved in more and more complex tasks often requiring the collaboration among several individuals that in general differ in their skills, and in the way they perceive the external environment. In such a context, the research activity in the field of robotics has been mainly focused on the development of complex algorithms to accomplish the specific robotic tasks like path-planning, vision, localisation, and so on. From the architectural point of view, two different philosophies have been carried on: the reactive and the behaviour-based paradigms. We think that these approaches don't allow to manage very large problems like the case in which a single robot has to solve a very complex task, or when a fleet of robots cooperates to achieve a common goal. Nowadays, agent-based architectures are increasingly used to model more and more complex systems. This induces the designers to the introduction of software engineering principles in developing such systems. Starting from the previous considerations, our work aims to propose a novel methodology for the design of multi-agent robotic architectures using the Unified Modeling Language. The methodology has been applied to the cognitive architecture previously developed by some of the authors, that could be viewed as an extension of the behaviour-based approach. Particularly, the proposed methodology uses behaviour-based philosophy as a part of a wider process which begins with the

requirements analysis for the whole system, identifies agents, and defines behaviours also by means of classical FSA diagrams. The agents defined in such a way are deployed on the required hardware platforms, thus allowing both single robot and multi-robot scenarios. The paper is arranged as follows. Section 2 deals with the overall description of the multi-agent architecture; section 3 explains the design methodology; section 4 reports experimental results, while in section 5 some conclusions are drawn.

2 Description of the Architecture

From the cognitive point of view, in our approach we refer to the architecture of fig. 1. In this structure it's possible to devise three main components: the perception, which is responsible to map the stream of raw data in an intermediate form, that in turn is provided to the cognitive component where the symbolic computation and, in general, deliberative behaviors of the system are located. The cognitive part can also support perception with some hints aimed to refine the perceptive process, and focus the attention on those external stimuli that are judged to be more useful for the current task completion. The third component is the actuation one, which communicates with the other two, in order to drive the robot hardware during perception tasks, and in attention focusing. The perception-action link allows also reactive behaviours. Some of the authors already presented this architectural structure [16–18]. Its main goal is to go beyond the classical behaviour-based model, and to provide the robot with true symbol grounding capabilities due to the intermediate representation of sensory data, that is used to instantiate pieces of knowledge at the symbolic component [9, 19]. . Through this mechanism the robot is able to act in a deliberative fashion more effectively.

It's possible to address this issue from the point of view of the holonic enterprise paradigm for agent-based software design introduced by Brennan and Ulieru in [11]. Like in the case of holonic enterprises, our approach suggests a possible abstraction from the single robot architecture to a multi robot team: the robot that is itself a multi agent system, can be viewed as a single agent in the multi robot context in which it cooperates with the others in order to reach the goals of the entire system.

3 The Design Methodology

Design has been performed using an extension of the AODPU (Agent-Oriented Design Process with UML) methodology [5–8] that is particularly useful in the case of robotic software architectures. The process is an iterative one, and

in a single iteration we could find the following phases:

1. Requirements analysis
2. Agents identification
3. Definition of the agents' structure

4. Description of the behaviours.

The agent structure identification and the behaviour description phases can be viewed as mutually dependent and cyclically performed to define the agents implementation. At the end of this process all the requirements are fixed (for this iteration) and the implementation can start.

Moreover we consider that:

1. the interactions between the agent and the world are a series of communication acts;
2. the agent can achieve its scope through its own knowledge and functionalities;
3. the knowledge of each agent can be increased using the communications with other agents or the real world;
4. the behaviours of a single agent are implemented through a series of tasks.

This schema could be suitable to deal with a robot that navigates in an environment with obstacles, has a target to reach and could receive orders by an user: for example a new target is designated by a vocal command or a pointing at gesture of the user's arm [12, 13]. In the rest of the paper we'll assume this scenario as the reference one in order to perform experiments.

3.1 Definition of the agents' structure

Agents' structure can be provided through class diagram. In order to perform experiments, we selected FIPA-OS [14, 15]as target architecture for implementation. This choice is essentially based on the wide diffusion of this programming environment in the agent software community. Moreover, using Java ensures portability and a very efficient thread management. Inside FIPA-OS an agent is represented as a class, and it has to accomplish to several tasks regarding both communication with other agents, and performing its own duties. Tasks are represented as subclasses of the agent class. Following this approach, each use-case that has been identified as an agent is represented as a new class. The tasks of the agent are implemented through subclasses (one for each task). With such a structure each agent can play his own role in the system organization using his own tasks (performed by the methods of its subclasses), knowledge (attribute of the agent class) and interactions with other agents (messages are sent/received by dedicated handlers tasks). Tasks of a same agent interacts in a conventional object-oriented way without message exchanging but through methods invocation and attributes access [10] therefore a new task (being a subclass of the agent class) could be initialized or destroyed during runtime, each task could access the knowledge (attributes) of the agent it belongs and so on.

4 Experimentation

Experimental phase has been performed using both a software simulator and a real robot equipment. Simulator was needed to easily implement multi-robot scenarios. Two experiments have been set up: a prey-hunter competition using the

simulator, and target reaching in the real case. In both experiments, robots were provided with obstacle avoidance capabilities. All the implemented behaviours are quite simple because our study was mainly focused on testing architecture implementation rather than developing high quality solutions to accomplish the robot's tasks. In particular, we were interested to stress multi-platform communication features of the FIPA-OS environment, and to cope with its lack of real-time control capabilities. Our robot was a K-Team Koala equipped with IR sensors, and controlled by a PC through a serial link. Vision was provided by a calibrated camera looking at the action field, and reporting localisation information to the rest of the system. In order to test distribution of agents software across multiple platforms, the camera was connected to a separate PC running also the vision agent's code. Obstacle avoidance was simply implemented by processing IR sensors readings in order to detect obstacle proximity. Then the robot follows obstacle's contour until it has free path to reach the goal. Path planning consists of a series turn and go straight movements that are computed starting from vision data. In what follows, a typical simulation experiment as long as the implementation of the vision agent will be reported in detail.

4.1 Prey-Hunter Simulation

Simulations were implemented using a Java GUI displaying a scaled action field, along with the obstacles layout. Robots are displayed using their 2D outline: in our simulation we used a Koala and a Kephera model. A ring around the Koala was displayed too, representing the range of the IR sensors. In fig. 1 is reported a hunting sequence; the Kephera model plays the role of the prey and it's programmed to perform random trajectories. It's to be noted that the only difference between simulation and real experiments is the implementation of the core methods for the vision and motion control agents. Their communication interface with the rest of the system is the same as in real robot tests. Simulation takes into account also time delays due to physical inertia. This approach allows us to make an extensive use of simulation in order to stress our design process.

4.2 Vision in Target Reaching

This section describes the process of localisation of the Koala robot during his task, in order to give useful feedbacks to the planning component [1]. We use a fixed CCD camera, connected to a computer, viewing the scene (see fig. 2). The software component implemented can run on a different machine from that runs the rest of the system, communicating to it by socket over the local net. In this way we have the possibility of performing the vision task in real time without adding high computational costs to the whole system. The valuable capabilities of the vision agent in the whole system are:

1. to individuate and segment the Koala robot also in contrasted and irregular backgrounds;

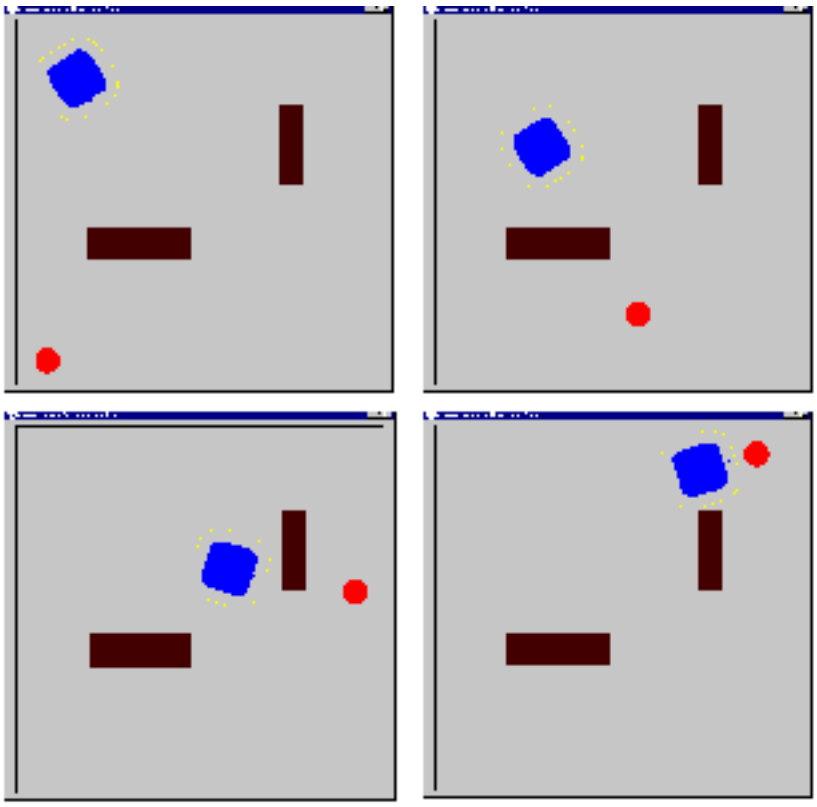


Fig. 1. A sequence of the prey-hunter experiment

2. to perform a estimation of the position of the robot by camera images;
3. to interpret the sequence of movements of the robot giving information of the direction followed by it.

The implemented computer vision task can be decomposed in three main steps:

1. localisation of the robot on image by low-level image processing on the single frame;
2. estimation of the 2D location of the robot on the floor;
3. reconstruction of the 3D position of the robot .

4.3 Localisation of the robot on image

The position of the robot on image is calculated by simple low-level image processing operations. The current frame is subtracted to the previous (grey level

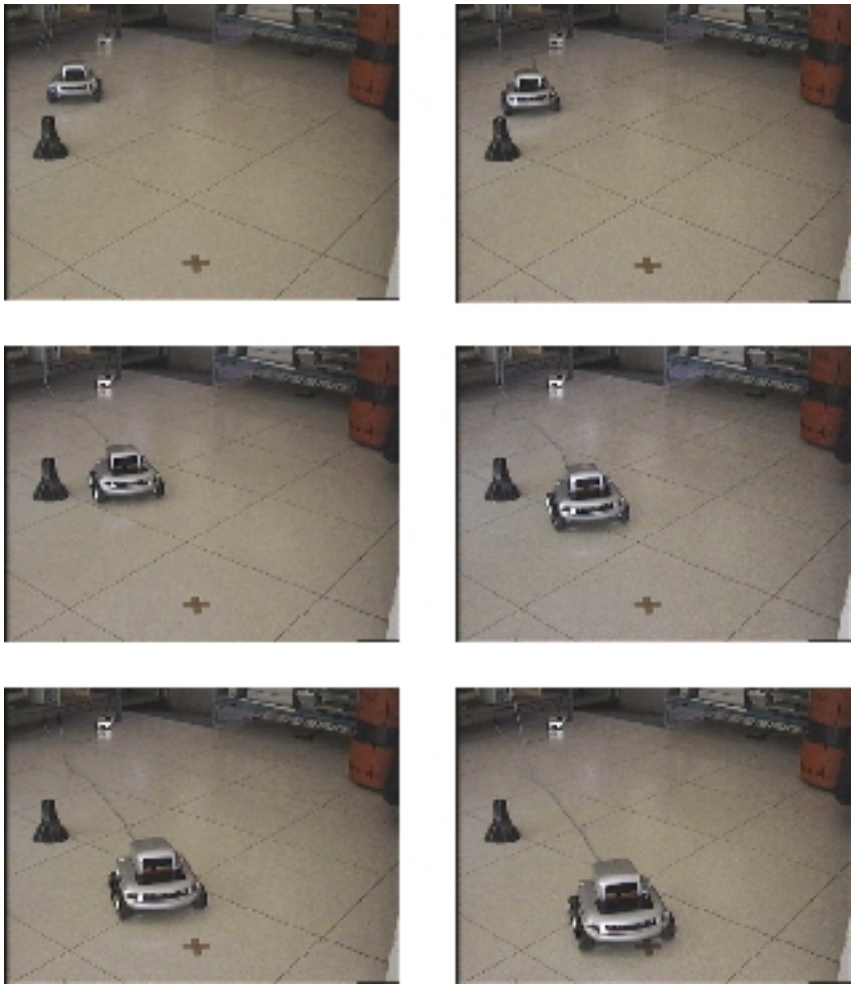


Fig. 2. Some frames of a video sequence in which the Koala robot perform obstacle avoidance (black object) and target localisation (cross painted on the floor)

images), obtaining the pixels related to moving objects in the viewed scene. If there are more than one object moving, Koala shape is selected using colour and texture features [2]. Naturally, some standard filtering operations are performed to reduce noise. Moreover, a corner detector is applied in the area of the image representing Koala shape to have feature points to tracking (see fig. 3). The estimation of the position of the robot on the floor it is based on this tracked points.

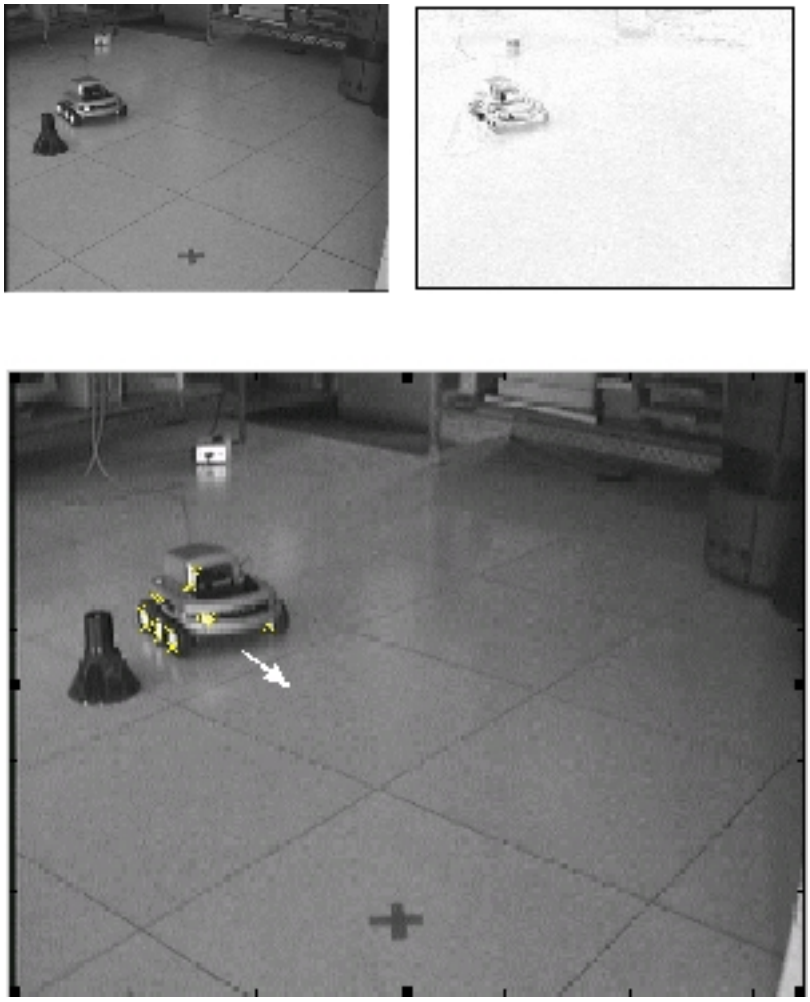


Fig. 3. (a) (b) Localisation of the shape of the moving robot by subtracting current frame to previous. (c) Tracking point used to estimate the position of the Koala robot and reconstructed direction of the movement referred to previous processed frame

4.4 Estimation of the 2D 3D location

The position of the robot respect a reference system is estimated using the homography between image plane and floor [3, 4]. A generic 3D point X generates the point w on image:

$$\lambda \, \mathbf{w} = \mathbf{P} \, \mathbf{X} = \mathbf{K} \, [\mathbf{R}|\mathbf{t}] \, \mathbf{X} \tag{1}$$

if the 3D points are on a plane (for instance Z=0), the transformation is simplified to a 3x3 matrix H:

$$\lambda\ \mathbf{w} = \mathbf{H}\ \mathbf{X}^P = \mathbf{K}\ [\mathbf{r^1}|\mathbf{r^2}|\mathbf{t}]\ \mathbf{X^P} \tag{2}$$

where H is the homography matrix, decomposable on calibration 3x3 matrix K, and 3x3 matrix has the first two columns of the rotation matrix R and the translation vector t. X and w are indicated using homogeneous coordinates. H is estimated using detected points belonging to the floor during a preliminary framework that also includes a calibration process: a grid placed in front of camera is used to obtain the calibration matrix K and fixes the rotation and translation referred to a reference system (see fig. 4).

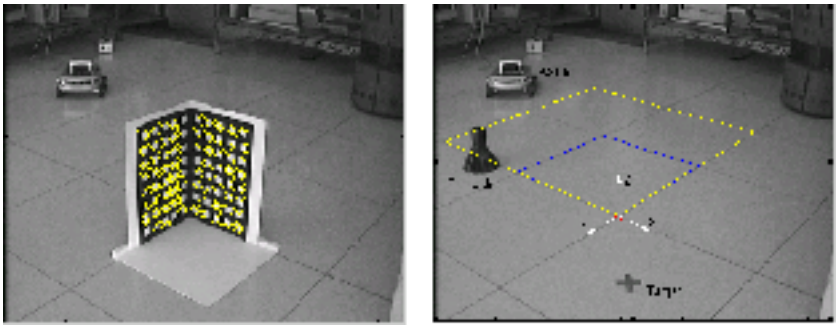


Fig. 4. (a) Calibration process using a grid; (b) point of the floor are used to estimate homography matrix

The tracked points on image are translated in 2D coordinates using estimated homography. The exact 3D position is recovered using the known real dimensions of the koala robot and the data coming from calibration framework. The estimated 2D coordinates of the robot and the direction of the detected movements are communicate by a message to the system every time are calculated.

5 Conclusion

A novel methodology for the design of multi-agent robot architectures including also vision agents is presented that extends the classical behaviour-based approach. It shall be showed that it can be profitably used both in the case of a single robot design, and in a multi-robot scenario. The methodology has been implemented using a FIPA compliant, and the experimental results are very encouraging. We are currently extending the methodology towards automatic code generation for a great part of the agents' implementation. In particular, regarding the agents' structure, we are looking at the classical categorisation reported by Russel and Norvig.

References

1. Chella, A., Gaglio, S., Guarino, M. D., Infantino, I.: An artificial high-level vision agent for the interpretation of the operations of a robotic arm. In proc. 5^{th} Int. Symp. on Artificial Intelligence, Robotics and Automation in Space, Noordwijk, 1999.
2. Chella, A., Di Gesu, V., Infantino, I., Intravaia, D., Valenti, C.: A Cooperating Strategy for object Recognition. In Proc. Of Int. Workshop on Shape, Contour and Grouping in Computer Vision, R. Cipolla, D. Forsyth (eds), Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1998.
3. Faugeras, O.: Three-Dimensional Computer Vision. MIT Press, Cambridge, MA, 1993.
4. Horn B.P.K., Robot Vision, MIT Press, Cambridge, MA, 1986.
5. Jennings N.R., On agent-based software engineering, Artificial Intelligence 117 (2000), 277-296
6. OMG Unified Modeling Language, version 1.3, June 99, Object Management Group document ad/99-06-08, available from http://cgi.omg.org/ docs/ ad/ 99-06-08.pdf
7. Chella A., Cossentino M., Lo Faso U., Applying UML use case diagrams to agents representation, Convegno AI*IA 2000, Milan, Sept. 2000.
8. Chella A., Cossentino M., Lo Faso U., Designing agent-based systems with UML, IEEE International Symposium on Robotics and Automation, ISRA'2000, Monterrey, Mexico, Nov. 2000.
9. Arkin R., Behavior Based robotics, The MIT Press, Cambridge, Massachussets, London, England, 1998.
10. Odell J., Objects and Agents: how do they differ?, on-line at: www.jamesodell.com/publications.html.
11. Ulieru M., Walker S. S., Brennan R. W., The holonic enterprise as a collaborative information ecosystem, to appear in Proc. of Autonomous Agents 2001 workshop Holons: Autonomous and Cooperative Agents for Industry.
12. Ardizzone E., Chella A., Pirrone R., Alfano F., An Architecture for Automatic Gesture Analysis, in V. Di Ges, V., Levialdi, S., Tarantino, L. (eds.) Proc. of the Working Conference on Advanced Visual Interfaces AVI 2000, May 23-26 2000, Palermo, Italy, ACM Press, 205-210.
13. Waldherr S., Thrun S., Romero R., A gesture-based interface for human-robot interaction, Autonomous Robots, 9, (2), 2000, 151-173.
14. O'Brien P., and Nicol R, FIPA - Towards a Standard for Software Agents, in: BT Technology Journal, 16, (3), 51-59, 1998.
15. Poslad S., Buckle P., Hadingham R., The FIPA-OS Agent Platform: Open Source for Open Standards, in: Proc. of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents, UK, 355-368, 2000.
16. Chella A., Frixione M., Gaglio S., Understanding dynamic scenes, Artificial intelligence, 123, (2000), 89-132.
17. Chella A., Gaglio S., Pirrone R., Conceptual representations of actions for autonomous robots, Robotics and Autonomous Systems, 34, (2001), 251-263.
18. Chella A., Frixione M., Gaglio S., An architecture for autonomous agents exploiting conceptual representations, Robotics and Autonomous Systems, 25, (1998), 231-240.
19. Russel S., Norvig P., Artificial Intelligence: A Modern Approach, Prentice Hall Int. Ed., 1995.

Learning to Correct the Layout Extracted from Document Images

Oronzo Altamura Floriana Esposito Donato Malerba

Dipartimento di Informatica, Università degli Studi di Bari,
via Orabona 4, I-70126 Bari – Italy
{altamura, esposito, malerba}@di.uniba.it

Abstract. Layout analysis is the process of extracting a hierarchical structure describing the layout of a page. In the document processing system WISDOM++ the layout analysis is performed in two steps: firstly, the global analysis determines possible areas containing paragraphs, sections, columns, figures and tables, and secondly, the local analysis groups together blocks that possibly fall within the same area. The result of the local analysis process strongly depends on the quality of the results of the first step. In this paper we investigate the possibility of supporting the user during the correction of the results of the global analysis. This is done by allowing the user to correct the results of the global analysis and then by learning rules for layout correction from the user sequence of actions. Preliminary experimental results on a set of multi-page documents are reported.

1 Background and motivations

Many approaches have been proposed in the literature for the extraction of the layout structure from document images. Traditionally, they have been classified as *top-down* or *bottom-up* [15]. In top-down methods, the document image is repeatedly decomposed into smaller and smaller components. Typical examples are the RLSA [17] and the projection profile cuts [11]. In bottom-up methods, basic layout components are extracted from bitmaps and then grouped together into larger blocks on the basis of their characteristics. Neighborhood line density [6], connected component analysis [5] and minimal-cost spanning tree [14] are some examples.

Another dimension of classification of layout analysis methods is the required amount of *explicit knowledge*, here intended as the body of facts and principles that humans apply when they are asked to decompose a page. The RLSA and the projection profile cuts are two prototypical examples of layout analysis algorithms that require a limited amount of knowledge expressed as a few numerical parameters. This view is at variance from that expressed by Tang *et al*. [16] according to which top-down methods are knowledge-based while bottom-up are data-driven. In fact, in the literature it is possible to find several examples of both top-down and bottom-up knowledge-based approaches. For instance,

publication-specific grammars are used in Gobbledoc [10] to describe all legal page formats allowed for a given publication. Geometric trees [3], a kind of hierarchical document layout models, are used to decompose single-page documents and to extract their logical components at the same time. Form Definition Languages [6] were also proposed to express models of the layout used by a Form Dividing Engine that decomposes document images. These are all examples of top-down approaches that require declarative knowledge on the possible layout structures. Some examples of bottom-up knowledge-based approaches are the rule-based method proposed by Fisher *et al*. [4] and the document spectrum (docstrum) method [12] that determines the structural blocks by means of some criteria expressed as rules of a production system.

In their seminal work, Nagy, Kanai and Krishnamoorthy [9] distinguish three levels of knowledge in the layout structure of a document:

- *Generic* knowledge (e.g., type base lines of a word are collinear).
- *Class-specific* knowledge (e.g., no text line is lateral to a graphical object).
- *Publication-specific* knowledge (e.g., maximum type size is 22 points).

Knowledge used in bottom-up layout analysis is necessarily different from that used for top-down processing: it is much less document specific. In addition, knowledge used in top-down approaches is typically derived from the relations between the geometric and the logical structures of specific classes of documents.

In WISDOM++ (http://www.di.uniba.it/~malerba/wisdom++/), a document image analysis system that can transform paper documents into either HTML or XML format [1], only generic knowledge is required by the layout extraction and analysis module. The applied page decomposition method is hybrid, since it combines a variant of the RLSA [13] to segment the document image and a bottom-up layout analysis method to assemble basic blocks into larger components called *frames*. The layout analysis is done in two steps:

1. A *global* analysis of the document image in order to determine possible areas containing paragraphs, sections, columns, figures and tables. This step is based on an iterative process, in which the vertical and horizontal histograms of text blocks are alternatively analyzed in order to detect columns and sections/paragraphs, respectively.
2. A *local analysis* of the document to group together blocks that possibly fall within the same area. Three perceptual criteria are considered in this step: *proximity* (e.g. adjacent components belonging to the same column/area are equally spaced), *continuity* (e.g. overlapping components) and *similarity* (e.g. components of the same type, with an almost equal height).

Generic knowledge on rules and typesetting conventions is used only in the bottom-up phase, especially in the local analysis step. Experimental results reported by Altamura *et al*. [2] proved the effectiveness of the approach on images of the first page of papers published on either conference proceedings or on journals. However, performance degrades when the system is tested on intermediate pages of multi-page articles, where the structure is much more variable due to the presence of formulas, images, drawings that can stretch on more than one column, or are quite close. The main source of the errors made by the layout analysis module is in the global analysis step, while the local analysis step performs satisfactorily when the result of the global analysis is correct.

In this paper we investigate the possibility of supporting the user during the correction of the results of the global analysis. This is done by means of two additional system facilities:

1. the user can correct the results of layout analysis by either grouping or splitting columns/sections automatically produced by the global analysis;
2. the user can ask the system to learn grouping/splitting rules from his/her sequence of actions correcting the results of the layout analysis.

In the following section, a description of the layout correction operations is reported, and the automated generation of training examples is explained. Section 3 briefly introduces the learning system used to generate layout correction rules and presents some preliminary experimental results.

2 Correcting the results of the global analysis

Global analysis aims at determining the general layout structure of a page and operates on a tree-based representation of nested columns and sections (Figure 1). At the end of the global analysis process, the user can see only those sections and those columns that have been considered atomic, that is, not subject to further decomposition (Figure 2). The user

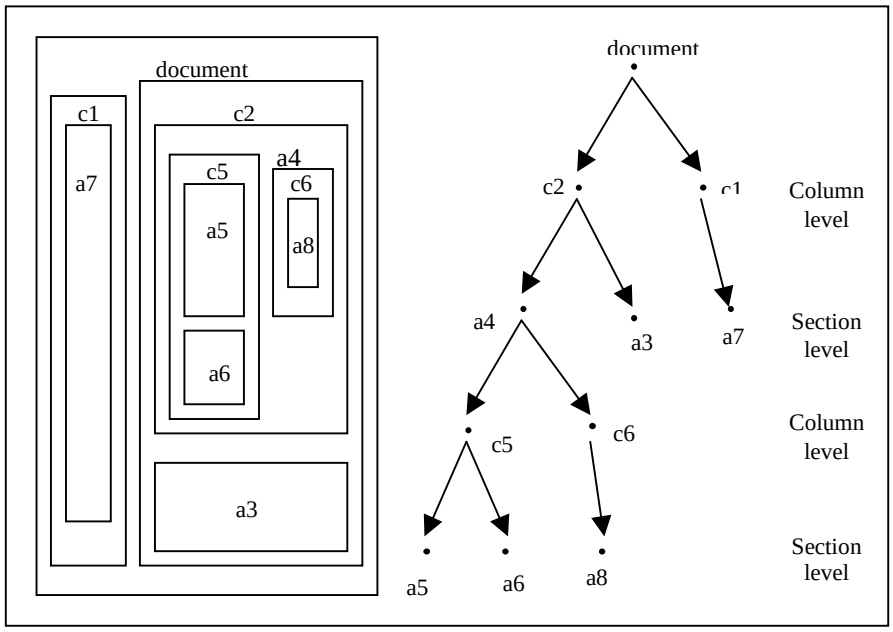


Fig. 1. Tree structure of the columns and sections determined by WISDOM++.

can correct this result by means of three different operations:

- Horizontal splitting: a column/section is cut horizontally.
- Vertical splitting: a column/section is cut vertically.
- Grouping: two sections/columns are merged together.

The cut point in the two splitting operations is automatically determined by computing either the horizontal or the vertical histogram on the basic blocks returned by the

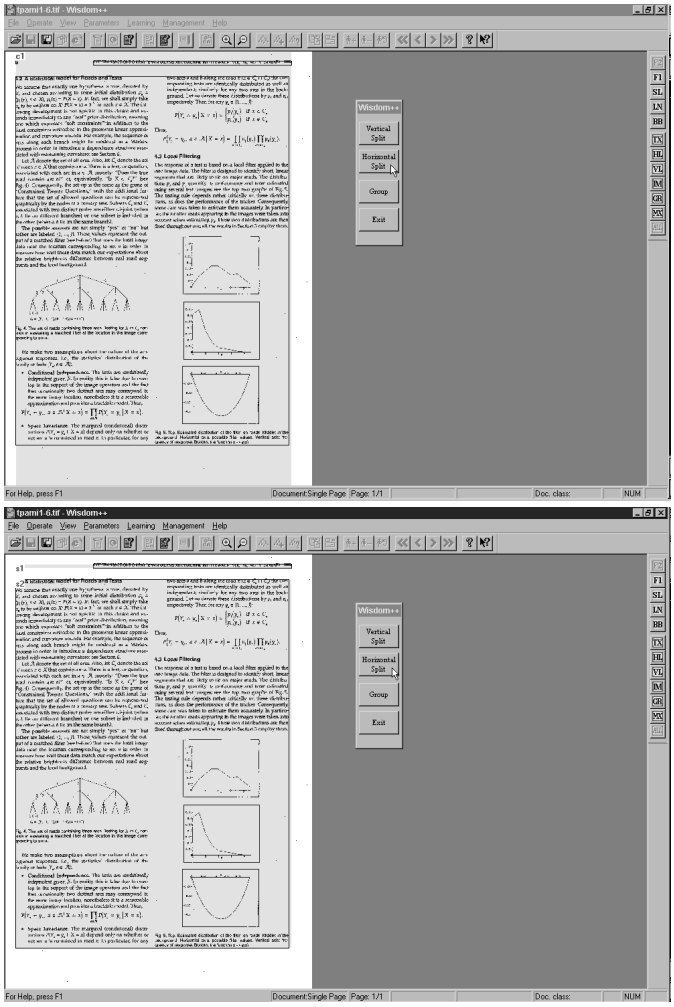


Fig. 2. Results of the global analysis process: one column (*above*) includes two sections (*below*). The result of the local analysis process is in reported the background.

segmentation algorithm. The horizontal (vertical) cut point corresponds to the largest gap between two consecutive bins in the horizontal (vertical) histogram. Therefore, splitting operations can be described by means of a binary function, namely *split(X,S)*, where *X* represents the column/section to be split, *S* is an ordinal number representing the step of the correction process, and the range of the split function is the set {*horizontal*, *vertical*, *no_split*}.

The grouping operation, which can be described by a ternary predicate *group(A,B,S)*, is applicable to two sections (columns) *A* and *B* and returns a new section (column) *C* whose boundary is determined as follows. Let ($left_X$, top_X) and ($bottom_X$, $right_X$) be the coordinates of the top-left and bottom-right vertices of a column/section *X*, respectively.[1] Then:

$$left_C = min(left_A, left_B),\ right_C = max(right_A, right_B),$$
$$top_C = min(top_A, top_B),\ bottom_C = max(bottom_A, bottom_B).$$

Grouping is possible only if the following two conditions are satisfied:

1. *C* does not overlap another section (column) in the document.
2. *A* and *B* are nested in the same column (section).

Immediately after each splitting/grouping operation, WISDOM++ recomputes the result of the local analysis process, so that the user can immediately perceive the final effect of the requested corrections and can decide whether to confirm the correction or reject it.

From the user interaction, WISDOM++ implicitly generates some training observation describing when and how the user intended to correct the result of the global analysis. An example is reported in the following:

```
split(c1,s)=horizontal, group(s1,s2,s)=false,
split(s1,s)=nosplit, split(s2,s)=nosplit ←
   step(s)=1,
   type(s1)=section, type(s2)=section, type(c1)=column,
   width(s1)=552, width(s2)=552, width(c1)=552,
   height(s1)=8, height(s2)=723, height(c1)=852,
   x_pos_centre(s1)=296, x_pos_centre(s2)=296,
   x_pos_centre(c1)=296,
   y_pos_centre(s1)=22, y_pos_centre(s2)=409,
   y_pos_centre(c1)=426,
   on_top(s1,s2)=true,
   part_of(c1,s1)=true, part_of(c1,s2)=true,
   no_blocks(s1)=2, no_blocks(s2)=108, no_blocks(c1)=110,
   per_text(s1)=100, per_text(s2)=83, per_text(c1)=84.
```

This observation describes the first correction applied to a page layout where two sections and one column where originally found (Figure 2). The horizontal splitting of the column is the first correction performed by the user (see Figure 3), as described by the

[1] The origin of the coordinate system is in the top left-hand corner; the abscissa increases from the leftmost to the rightmost column, while the ordinate increases from the uppermost to the lowest row.

first literal, namely *step(s)=1*. This column is 552 pixels wide and 852 pixels high, has a center located at the point (296,426), and includes 110 basic blocks and the two sections *s1* and *s2*, which are one on top of the other; the percentage of the area covered by text blocks enclosed by the column is 84%. It is noteworthy that the multiple-head clause above also describes that the two sections *s1* and *s2* should be neither split (literals *split(s1,s)=nosplit* and *split(s2,s)=nosplit*) nor grouped (literal *group(s1,s2,s)=false*) at the first correction step. Many other literals, such as *group(c1,s2,s)=false*, *group(s1,c1,s)=false*, *group(c1,c1,s)=false*, have not been generated, since they do not represent admissible groupings according to the two constraints specified above. All literals in the head of the clause are called *examples* of the concepts *split* and *group*. They can be considered either positive or negative according to the learning goal.

3 Learning rules for layout correction

Rules for the automated correction of the layout analysis can be automatically learned by means of the system ATRE. The learning problem solved by ATRE can be briefly

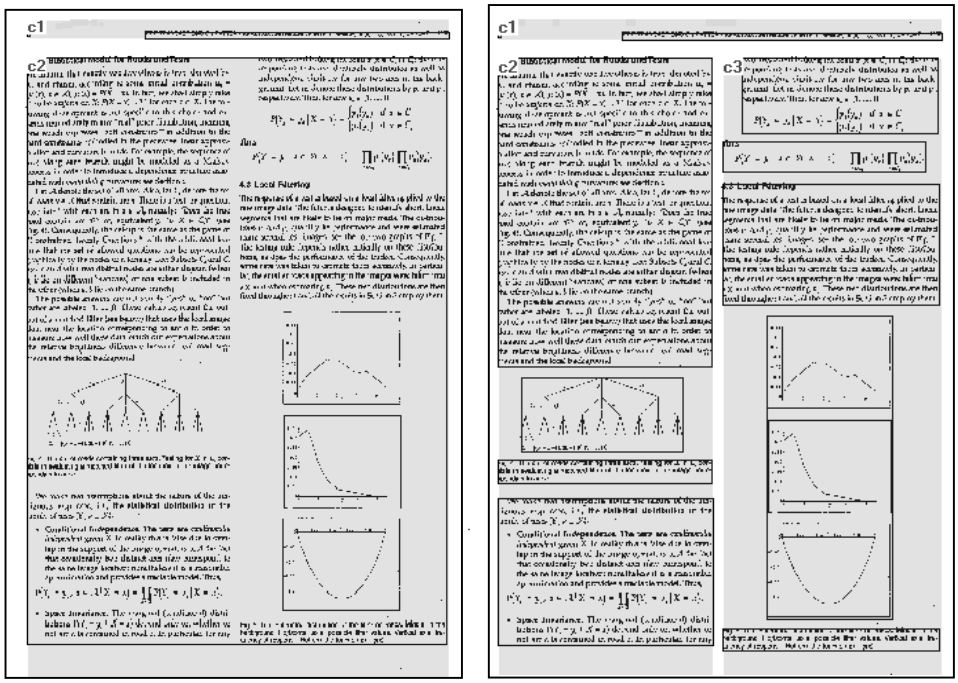


Fig. 3. Horizontal split of the column (*left*) and vertical split of column c2 (*right*). The result of the layout analysis process is in the background.

formulated as follows:

Given

- a set of concepts C_1, C_2, .., C_r to be learned,
- a set of training observations O described in a language L_O,
- a user's preference criterion *PC*,

Find

a (possibly recursive) logical theory T for the concepts C_1, C_2, …, C_r, such that T is complete and consistent with respect to O and satisfies the preference criterion *PC*.

Details on the learning algorithm implemented in ATRE are reported in [7,8].

In the context of the global analysis correction, the set of concepts to be learned are *split(X,S)=horizontal*, *split(X,S)=vertical*, *group(X,Y,S)=true*, since we are interested to find rules predicting both when to split horizontally/vertically a columns/section and when to group two columns/section. No rule is generated for the case *split(X,S)=no_split* and *group(X,Y,S)=false*.

The preference criterion *PC* is a set of conditions used to discard some solutions and favor others. In particular, we prefer shorter rules that explains a high number of positive examples and a low number of negative examples.

To prove the applicability of this approach we considered four multi-page documents. They are long papers published in the IEEE Transactions on Pattern Analysis and Machine Intelligence, issue of January 1996. The distribution of pages per document is reported in Table 1.

Table 1. Distribution of pages and examples per document.

Name of the document	***No. of pages***	***No. of horizontal splits***	***No. of vertical splits***	***No. of groupings***	***Total no. of examples***
TPAMI1	14	6	6	8	916
TPAMI2	8	9	7	1	458
TPAMI3	15	8	6	1	395
Total (training)	37	23	19	10	1769
TPAMI4 (testing)	13	6	5	3	984

We used the first three papers to train the system and the fourth paper for testing. The number of training observations corresponds to the final, corrected layout of each page (i.e., 37) plus the number of intermediate global layout structures, which are subject to corrections (i.e., 52). The total number of examples in the eighty-nine training observations is 1769, which correspond to the total number of literals in the multiple-head clauses. Given the set of concepts to be learned, only 52 out of 1769 examples are positive, which correspond to actual corrective actions performed by the user (vertical/horizontal splitting or grouping). The average number of corrections performed by the user is 1.41 (i.e., 52/37) per page. In fact, some intermediate pages of multi-page documents are the most critical and may require several operations to correct the column/section structure.

ATRE generated the following set of rules corresponding for the three concepts to be learned:

```
1.  split(X1,S)=horizontal ← step(S)∈[1..1],
        height(X1)∈[873..875], type(X1)=column
2.  split(X1,S)=horizontal ← step(S)∈[1..1],
        height(X1)∈[848..875], x_pos_centre(X1)∈[279..304]
3.  split(X1,S)=horizontal ← x_pos_centre(X1)∈[431..439],
        y_pos_centre(X1)∈[467..571], height(X1)∈[606..814],
        step(S)∈[3..4]
4.  split(X1,S)=horizontal ← type(X1)=column,
        per_text(X1)∈[11..63], step(S)∈[2..5]
5.  split(X1,S) = horizontal ← step(S)∈[1..1],
        height(X1)∈[398..398], width(X1)∈[262..262]
6.  split(X1,S)=horizontal ← height(X1)∈[814..875],
        to_right(X3,X1), width(X3)∈[269..269],
        step(S)∈[3..3]
7.  split(X1,S)=vertical ← type(X1)=column,
        no_blocks(X1)∈[524..543], step(S)∈[2..2]
8.  split(X1,S)=vertical ← height(X1)∈[789..830],
        step(S)∈[2..2], width(X1)∈[540..552]
9.  split(X1,S)=vertical ← type(X1)=section,
        height(X1)∈[154..154], step(S)∈[4..4]
10. split(X1,S)=vertical ← width(X1)∈[548..552],
        y_pos_centre(X1)∈[444..700],
        x_pos_centre(X1)∈[290..304], step(S)∈[1..2]
11. split(X1,S)=vertical ← step(S)∈[2..4],
        height(X1)∈[28..28], width(X1)∈[539..539]
12. split(X1,S)=vertical ← type(X1)=column,
        width(X1)∈[539..539], step(S)∈[3..3],
        height(X1)∈[820..820]
13. split(X1,S)=vertical ← type(X1)=section,
        width(X1)∈[541..541], x_pos_centre(X1)∈[284..284],
        step(S)∈[1..1]
14. split(X1,S)=vertical ← type(X1)=column,
        no_blocks(X1)∈[2..2], width(X1)∈[539..548],
        step(S)∈[3..3]
```

```
15.  group(X1,X2,S)=true ← width(X1)∈[9..9],
        type(X2)=column, step(S)∈[1..1]
16.  group(X1,X2,S)=true ← type(X1)=column,
        height(X1)∈[151..154], step(S)∈[2..2],
        type(X2)=column
17.  group(X1,X2,S)=true ← width(X1)∈[548..548],
        y_pos_centre(X1)∈[631..665], step(S)∈[1..3],
        type(X2)=section
18.  group(X1,X2,S)=true ← y_pos_centre(X2)∈[124..124],
        x_pos_centre(X1)∈[323..334], step(S)∈[2..3]
19.  group(X1,X2,S)=true ← width(X2)∈[7..150],
        no_blocks(X1)∈[4..5], step(S)∈[4..4]
20.  group(X1,X2,S)=true ← step(S)∈[6..6],
        on_top(X2,X1)=true, type(X2)=column
```

The interpretation of these rules is straightforward. For instance, the first rule states that «at the first correction step, columns with a height between 873 and 875 pixels should be horizontally split», while rule 20 states that «at the sixth correction step, an area[2] on top of a column should be grouped with that column». This rule involves the relation *on_top* and could be generated only by learning systems that operate on first-order logic descriptions, such as ATRE.

Some of the induced rules (e.g., 5, 9, 11, 12, 13, and 15) are clearly specific and have been generated by the system to explain a limited number of examples. Specificity of rules are due to two factors: firstly, the limited number of positive examples used in the training set, and secondly, the fact that ATRE is asked to generate a *complete* set of rules, that is a set of rules that explain *all* positive examples. However, other rules generated by ATRE are quite general, like rule 4, which states that columns with low percentage of text (i.e., including text and figures) should be split horizontally after the first correction step.

WISDOM++ uses the induced rules to automatically correct a page layout every time a document image is processed. This operation is quick and totally transparent to the user. Results on the test examples are reported in Table 2. *Omission* errors occur when a rule for a given concept is not applied when it should, while commission errors occur when a rule is applied when it should not.

Table 2. Commission and omission errors performed by rules of various concepts.

Rule for	***No. omission errors***	***No. commission errors***
split(X,S)=vertical	2	1
split(X,S)=horizontal	3	2
grouping(X,Y,S)=vertical	3	0

[2] In this case the area is necessarily a column, since users can only group two columns or two sections.

Unfortunately, the induced set of rules missed all grouping operations, while it was able to correct some page layouts by performing horizontal and vertical splitting six out of the eleven expected times. Three unnecessary splits were also performed. Pages in which all errors occurred were five of the thirteen test pages, and in particular eight of the eleven errors concerned two pages (41 and 51) where the correction process was quite complex.

4 Conclusions

This work presented a preliminary application of machine learning techniques to the problem of correcting the result of the global layout analysis process in WISDOM++. The learning problem to be solved has been introduced and the first-order logic representation of the correction actions performed by the user has been illustrated. Experimental results on a small set of multi-page documents showed that the proposed approach is able to capture relatively simple layout corrections. Inaccuracy for complex process can be mainly attributed to the limited size of training documents. A more extensive experimentation is planned to confirm these initial conclusions. A further research issue to be investigated concerns the application of a learning system devised to solve classification problems, like ATRE, to a typical planning task, like that of performing a sequence of actions that lead to the desired goal. Finally, we intend to investigate the problem of incrementally refining the set of rules generated by ATRE when new training observations are made available.

Acknowledgments

The authors wish to thank Teresa Pinto for her help in conducting the experiments. This work is in partial fulfillment of the research objectives set by the IST -1999-20882 project COLLATE (Collaboratory for Automation, Indexing and Retrieval of Digitized Historical Archive Material) funded by the European Union (`http://www.collate.de/`).

References

1. O.Altamura, F.Esposito and D. Malerba (1999). Wisdom++: An Interactive and adaptive document analysis system. *Proc. of Fifth International Conference on Document Analysis and Recognition* , pp.366-369.
2. O.Altamura, F.Esposito and D. Malerba (2001). Trasforming paper documents into XML format with WISDOM++. *International Journal on Document Analysis and Recognition* (in print).
3. A. Dengel and G. Barth (1988). Hight level document analysis guided by geometric aspects. International Journal of Pattern Recognition and Artificial intelligence, 2, pp. 641-655.

4. J.L. Fisher, S.C. Hinds, and D.P. D'Amato (1990). A rule-based system for document image segmentation. *Proc. of the 10th Int. Conf. on Pattern Recognition*, IEEE Computer Society: Los Alamitos, CA, pp. 567-572.
5. L.A. Fletcher and R.Kasturi (1988) Arobust Algoritm for Text String Separation from Mixed Text/graphics Images. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol 10, n.6, pp910-918.
6. K. Kubota, O.Iwaki and H.Arakawa (1984) Document understanding system. *Proceedings of the 7th ICPR*, pp.612-614
7. D. Malerba, F. Esposito, and F.A. Lisi (1998). Learning recursive theories with ATRE, in H. Prade (Ed.), *Proceedings of the 13th European Conference on Artificial Intelligence*, 435-439, John Wiley & Sons, Chichester, England.
8. D. Malerba, F. Esposito, F.A. Lisi, and O. Altamura (2001). Automated Discovery of Dependencies Between Logical Components in Document Image Understanding. *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, Seattle (to appear).
9. G. Nagy, J.Kanai and M.Krishnamoorthy (1988). Two complementary techniques for digitized document analysis. *Acm Conference on Document Processing Systems*.
10. G. Nagy, S. Seth and M. Viswanathan (1992) A Prototype Document Image Analysis System for Technical Journal, *IEEE Computer*, vol. 25, no. 7, pp.10-22.
11. G.Nagy, Sharad C. Seth and S.D. Stoddard (1986) Document Analysis with an expert system. In *Pattern Recognition in Practice II* pp.149-159 E.S.Gelsema and L.N.Kanal(Editors) Elsevier Science Publishers B.V.
12. L. O'Gorman (1993). The document spectrum for page layout analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1162-1173.
13. F.Y. Shih, and S.-S. Chen (1996). Adaptive document block segmentation and classification. *IEEE Trans. on Systems, Man, and Cybernetics - Part B*, vol. 26, no. 5, pp. 797-802.
14. A. Simon , J.C. Pret and A.P. Johnson(1997) A fast algorithm for bottom-up layout analysis. *IEEE Transaction on Pattern Analysis and Machine Intelligence* , vol.19, n.3, pp.273-277
15. Srihari and G.W. Zack (1986) Document Image Analysis. *Proceedings of the 8th ICPR*, pp.434-436
16. Y.Y. Tang, C. De Yan and C.Y. Suen(1994) Document processing for automatic knowledge acquisition. *IEEE Transaction on Knouledge and Data Engineering*, vol.6, n.1, pp.3-21
17. K.Y. Wong, R.G. Casey, and F.M. Wahl (1982). Document analysis system. *IBM Journal of Research Development*, vol. 26, no. 6, pp. 647-656.

Cooperative Distributed Vision for Mobile Robots

Emanuele Menegatti, Enrico Pagello[†]

Intelligent Autonomous Systems Laboratory
Department of Informatics and Electronics
University of Padua, Italy
[†] *also with* Institute LADSEB of CNR
Padua, Italy
emg@dei.unipd.it

Abstract. Multiple robot systems in which every robot is equipped with a vision sensor are more and more frequent. Most of these systems simply distribute the sensors in the environment, but they do not create a real Cooperative Distributed Vision System. Distributed Vision System has been studied in the past, but not enough emphasis has been posed on mobile robots. In this paper we propose an approach to realize a Cooperative Distributed Vision System within a team of heterogeneous mobile robots. We present the two research streams which we are working on, along with theoretical and practical insights.

1 Introduction

Mobile robots are more and more fitted with vision systems. The popularity of such sensors arises from their capability of gathering a huge amount of information from the environment surrounding the robot. Nowadays, the relatively low cost of the required hardware allows to equip every robot of a mobile robot team with a vision system. An alternative approach is to control a robot team with a centralized vision system, i.e. a unique camera that monitors the whole environment where the robots move. This has been applied in well structured and relatively small environments [2], but it is unfeasible for large environments.

If a camera is mounted on every robot of the team, each robot can gather a more detailed information on its surrounding and the system is more versatile. In fact, fixed cameras positioned in *a priori* location in the environment limit the flexibility and robustness of the system. If something happens outside the field of view of the fixed cameras, the system cannot see this event. If we have cameras mounted on mobile robots, the system can send a robot to inspect the new location of interest.

Mounting a camera on each robot distributes the sensors in the environment, but this is not enough: we aim to the creation of a real Distributed Vision System. A Distributed Vision System requires not only a set of cameras scattered in the environment, but also the sharing of information between the different vision systems.

In the following we will prefer the term *Vision Agent* instead of "vision system". The term Vision Agent emphasizes that the vision system is not just one of the several sensors of a single robot, but that it interacts with the other vision systems to create an intelligent distributed system.

Fig. 1. Our team of heterogeneous robots

2 Previous Works

Our work has been inspired by the work of Ishiguro [4]. He proposed an infrastructure called *Perceptual Information Infrastructure* (PII). In his paper, he proposed an implementation of the PII with a Distributed Vision System (DVS) composed by static Vision Agents, i.e. fixed cameras with a certain amount of computational power. The cameras, strategically placed in the environment, navigate a mobile robot. The robot is not autonomous, in the sense that it needs the DVS to navigate, but it has a certain amount of deliberative power, in the sense that it decides which Vision Agent provides him the more reliable information on the surroundings. The vision algorithms of the Vision Agents are really simple, because of the assumption that every Vision Agents is static.

A parallel but independent work is the one of Matsuyama [5]. Matsuyama explicitly introduced mobile robots in its Cooperative Vision System. In the

experiments presented, he used active cameras mounted on a special tripod. The active cameras were pan-tilt-zoom cameras modified in order to have a fix view point. This allowed the use of a simple vision algorithm, not very different from the case of static cameras. As far as we know, no attempt has been tried to realize a DVS with truly mobile robots running robot vision algorithm.

3 The aim of our work

Our aim is to introduce a real Mobile Vision Agent in the DVS architecture, i.e. to apply the ideas and the concepts of Distributed Vision to a mobile robot equipped with a camera.

The domain in which we are testing our ideas is the RoboCup competitions. We are on the way to create a Distributed Vision System within a team of heterogeneous robots fitted with heterogeneous vision sensors. We want to create a dynamic model of the environment, which can be used by mobile robots or humans to monitor the environment or to navigate through it. The model of the environment is built fusing the data collected by every Vision Agent. The redundancy of observers (and observations) is a key issue for system robustness.

4 Implementation

4.1 Two VAs mounted on the same robot

The first implemental step is to realise a Cooperative behavior between two heterogeneous vision agents embodied in the same robot. Exploiting the knowledge acquired in our previous research [7], we want to create a Cooperative Vision System using an omnidirectional and a perspective vision system mounted on the same robot. The robot is our football player robot, called Nelson that we entirely built starting from an ActivMedia Pioneer2 base (see the web page www.dei.unipd.it/~robocup). The omnidirectional vision system is a catadioptric system composed by a standard colour camera and an omnidirectional mirror we designed [6].The omnidirectional camera is mounted on the top of the robot and offers a complete view of the surroundings of the robot [1]. The perspective camera is mounted in the front of the robot and offers a more accurate view of objects in front of it. These two cameras mimic the relationship between the peripheral vision and the foveal vision in humans. The peripheral vision gives a general, and less accurate, information on what is going on around the observer. The foveal vision determines the focus of attention and provides more accurate information on a narrow field of view. So, the omnidirectional vision is used to monitors the surroundings of the robot to detect the occurrence of particular events. Once one of these events occurs, the Omnidirectional Vision Agent (OVA) send a message to the Perspective Vision Agent (PVA). If the PVA is not already focused on a task, it will move the robot in order to put the event in the field of view of the perspective camera. This approach was suggested by our previous researches presented in [3].

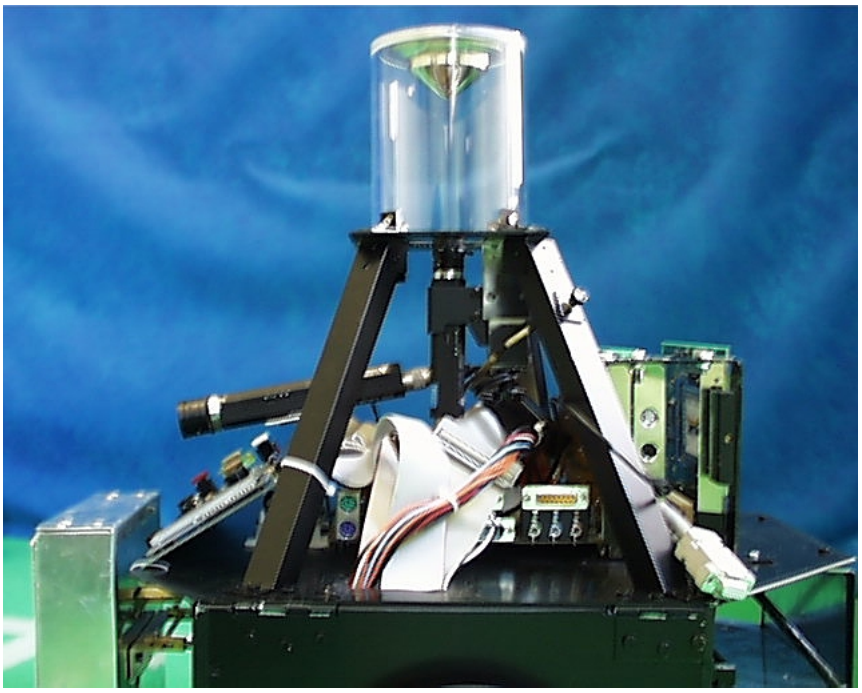


Fig. 2. A close view of the vision system of Nelson. On the left, the perspective camera. In the middle, pointed up-ward the omnidirectional camera

Experiments on such a system are running and they will provide more insight on the cooperation of the two heterogeneous vision agents.

4.2 Coordination of several VAs mounted on different robots

Another stream of research is the creation of a Cooperative Distributed Vision System for our team of football player robots. Our aim is to implement the idea of the Cooperative Object Tracking Protocol proposed by Matsuyama [5]. In the work of Matsuyama the central notion is the concept of *agency*. An agency, in the definition of Matsuyama, is the group of the VAs that see the objects to be tracked and keeps an history of the tracking. This group is neither fixed nor static. VAs exit the agency, if they are not able to see the tracked object anymore. A new VA can joint the agency as soon as the tracked object comes in its field of view. To reflect the dynamics of the agency we need a dynamic data structure with a dynamic role assignment. Let us sketch how the agency works using an example draw from our application field: the RoboCup domain. Suppose to have a team of robots in the field of play. Each robot is fitted with a Vision Agent. None of the Vision Agent is seen the ball. In such a situation no agency exists. As soon as a Vision Agent see the ball, it creates the agency

sending a broadcast message to inform the other Vision Agents the agency has been created and it is the master of the agency. After this message a second message follows, telling the other Vision Agents the estimated position of the ball. All the other Vision Agents maneuver the robots in order to see the ball. Once a Vision Agent has the ball in its field of view, it asks permission to joint the agency and send to the master its estimation of the ball position. If this information is compatible with the information of the master, i.e. if the new Vision Agent has seen the *correct ball*, it is allowed to joint the agency.

The described algorithm has been realised by Matsuyama with his fixed vied point cameras. His system was composed of four pan tilt zoom cameras mounted on a special active support in order to present a fixed view point. The system is able to track a radio controlled toy car in a small closed environment. As mentioned before, in such a system there is not a truly mobile agent. Moreover the vision algorithm used is typical of static Vision Agents. In fact, this is a smart adaptation of the background subtraction technique.

Our novel approach is to implement the Cooperative Object Tracking Protocol within a team of mobile robots equipped with Vision Agents. This requires a totally new vision approach. In fact, the point of view of the Vision Agent changes all the time. The changes in the image are due not only to the changes in the world (as in the Matsuyama testbed), but also to the change of position of the Vision Agent. Therefore, we need a vision algorithm able to identify the different objects of interest and not only to reveal the objects that are moving. Moreover, we have to introduce a certain amount of uncertainty in the estimation of the position of these objects, because the location of the Vision Agents is not known exactly anymore and there are errors in the determination of the relative distance between the objects the Vision Agents.

To explain these issues, let us come back to our RoboCup example. Above we said that if a new Vision Agent sees the ball, it sends a message to the master that checks if it has seen the correct ball. In a RoboCup match there is just one ball, but sometimes what a robot identifies as a ball is not the correct one. This can result either because the robot sees objects resembling the ball, and erroneously interprets them as a ball (like spectators hands or reflex of the ball on walls), or because it is not properly localized and so it reports the ball to be in a fallacious position.

To cope with the uncertainty in the objects position, every Vision Agent transmits to the master the calculated ball position with a confidence associated to this estimation. The master dispatches to the other robot a position calculated as an average of the different position estimations, weighted by the confidences reported by every Vision Agent (if there is more than one Vision Agents in the agency).

Especially in the described dynamic system, the master role is crucial in the correct functioning of the agency. The master role cannot be statically assigned. The ball is continuously moving during the game. The first robot that sees the ball will not have the best observational position for long. So, the master role must pass from robot to robot. The processes of swapping the master role is

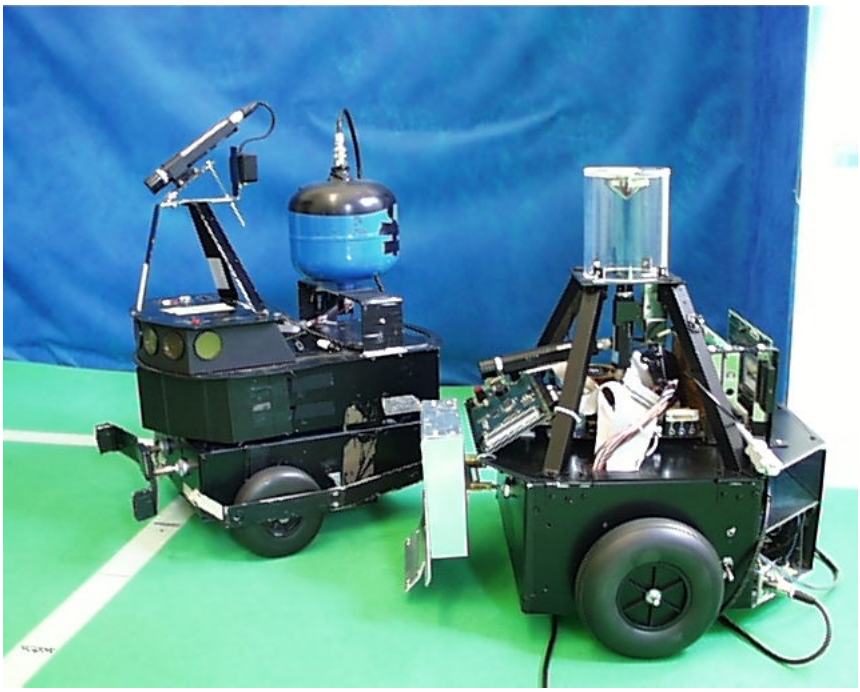


Fig. 3. A close view of two of our robots. Note the different vision systems

critical. If the master role is passed to a robot that sees an *incorrect ball* all the agency will fail in the *ball tracking* task.

The simplest solution could be to pass the master role to the robot with the highest confidence on the ball position. This means to shift the problem to identify a reliable confidence function. This makes sense, because the confidence function will be used for two services that are two sides of the same coin. In fact, if a robot is correctly localized and correctly calculates the relative distance of the ball, it will have strong weight in the calculation of the ball position. Given this, it can reliably take the role of master.

The confidence function The confidence function ψ_{abs} associated to the reliability of the estimation of the absolute ball position is a combination of several factors. It has to account for the different aspects that contributes to a correct estimation of the ball position. In fact, the position of the ball in the field of play is calculated by a vectorial sum of the relative distance of the ball from the robot and the absolute position of the robot in the pitch. So, the confidence of the estimation of the absolute position of the ball is the sum of the confidences function associated to the self-localisation, ψ_{sl}, and of the confidence function associated to the estimation of the relative position of the ball with respect to the robot, ψ_{rel}.

$$\psi_{abs} = \psi_{sl} + \psi_{rel} \quad (1)$$

The self-localisation process uses the vision system to locate landmarks in the field of play. The process is run only by time to time and if the landmarks are visible. Between two of these process the position is calculated with the odometers. This means that the localisation information degrades with time. The confidence function associated with the self-localisation is the result of the following contribution:

- type of vision system (perspective, omnidirectional, etc.);
- a priori estimated absolute error made from the vision system in the calculation of the landmarks position;
- time passed after the last self-localisation process;

The relative position of the ball with respect to the robot is calculated as in [6]. The confidence function in this process presents the following contribution:

- type of vision system;
- distance from the ball;

At the moment the exact definition of the confidence function is under testing. The experiments will tell us how much every contribution should weight in the final function.

5 Conclusion

In this paper we presented the two research streams we are following to implement a Cooperative Distributed Vision System.

In this paper we proposed to realise the DVS with heterogeneous mobile Vision Agents. We suggested a way to fuse the information coming from two heterogeneous Vision Agents mounted on the same robot. Regarding the problems introduced by the mobile Vision Agents, we suggested a way to cope with the uncertainty introduced in the localisation of the objects of interest.

At the time of writing experiments are running on such a systems providing theoretical and practical insight.

Acknowledgments

We wish to thanks the student of the ART-PD and Artisti Veneti Robocup teams who built the robots. This research has been partially supported by: the EC TMR Network SMART2, the Italian Ministry for the Education and Research (MURST), the Italian National Council of Research (CNR) and by the Parallel Computing Project of the Italian Energy Agency (ENEA).

References

1. A. Bonarini. The body, the mind or the eye, first? In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup99: Robot Soccer World Cup III*, volume 1856 pp. 210-221 of *LNCS*. Springer, 2000.
2. J. Bruce, T. Balch, and M. M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)*, volume 3, pages 2061 – 2066, October 2000.
3. S. Carpin, C. Ferrari, E. Pagello, and P. Patuelli. Bridging deliberation and reactivity in cooperative multi-robot systems through map focus. In M.Hannebauer, J. Wendler, and E. Pagello, editors, *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*,, LNCS. Springer, 2001.
4. H. Ishiguro. Distributed vision system: A perceptual information infrastructure for robot navigation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI97)*, pages 36–43, 1997.
5. T. Matsuyama. Cooperative distributed vision: Dynamic integration of visual perception, action, and communication. In W. Burgard, T. Christaller, and A. B. Cremers, editors, *Proceedings of the 23rd Annual German Conference on Advances in Artificial Intelligence (KI-99)*, volume 1701 of *LNAI*, pages 75–88, Berlin, Sept. 13–15 1999. Springer.
6. E. Menegatti, F. Nori, E. Pagello, C. Pellizzari, and D. Spagnoli. Designing an omnidirectional vision system for a goalkeeper robot. In *Proceeding of RoboCup 2001 International Symposium*, 2001.
7. E. Menegatti, E. Pagello, and M. Wright. A new omnidirectional vision sensor for the spatial semantic hierarchy. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM '01)*, July 2001.

Genetic Evolution Of A Lvq Classifier For Image Sequence Analysis

Guido Tascini, Luca Regini, Paolo Puliti, Patrizio Rogani.

Istituto di Informatica, Università di Ancona,
via Brecce Bianche, 60131 Ancona.
e-mail: tascini@inform.unian.it

Abstract. A basic task in computer vision and robotics is the classification of images describing an environment in which an agent must move. Neural Networks can be used to solve such problems. However when images belong to filmed sequences some issues arise regarding the optimal selection of the network training set and size. In this paper we propose a genetic algorithm that is able to automatically perform such selection throughout evolution. The evolution process is done according to a recently introduced approach that combines multiobjective programming and uniform design to find a set of solutions uniformly scattered over the Pareto frontier. The final result of the genetic algorithm is a population of LVQ Neural Networks in which every individual represents an optimal compromise solution for a given trade-off between the classifier accuracy and size.

1. Introduction

Recognition of images taken from a given environment is, from a pattern recognition perspective, the problem of finding a proper mapping from a set of random variables to a finite set of labels. Labels identify regions in the environment while random variables are the observed images. When we posses one or more filmed sequences of the environment then we have some instances of these random variables. In these cases the problem can be formulated as one of supervised classification where we have to deduce the proper mapping given a set of previously labelled examples. A main drawback of supervised classifiers is the strong negative influence of incorrectly labelled training samples ("noisy" samples or outliers). In general also classifier size plays an important role because it means a faster classification of an unknown sample: usually classification is *O(s)*, where S is some characteristic classifier size. Moreover these algorithms require that the exact number of labels should be known a priori. While there are some problems whose domain is clearly decomposable in a single way, most real-life applications have many possible choices for the number of clusters [4] depending on the performance required to the classifier in terms of accuracy, quickness of response, resources needed, ability to generalise and so on. This is exactly the case when we are dealing with classification of images taken from filmed sequences. The most obvious choice of using a nearest neighbour classifier with a reference set containing all the samples from the training set becomes impracticable even in the case of a film of a small indoor environment that typically contains some thousands of frames. One can of course exploit the high correlation of adjacent frames trying to select a reduced and representative reference set from the training set. This may be done in two way: by *editing-condensing techniques* or by *adaptive learning techniques*. Using editing condensing procedures the resulting reference set is a subset

of the training set while there is no explicit relations between the two sets when using adaptive learning procedures. The aim of editing condensing techniques is two-fold: improving the accuracy of the classification by removing samples located in overlapping acceptance surfaces (*editing techniques*) and reducing the computational effort required to find the nearest neighbour(s) (*condensing techniques*). A different approach consists of adaptive learning. When adaptive learning is adopted the training samples are used to tune a fixed number of codebooks or *prototypes* with a particular learning algorithm. Now the reference set is not usually a subset of the training set and is called *the codebooks* or *prototypes set*. An adaptive non parametric learning method, and the one we use in this work, is Kohonen's LVQ [2][8][9]. Each LVQ codebook vector defines a region in the observation space following a 1-nearest neighbour rule and hence LVQ prototypes are Voronoi vectors that specify a partition of the observation space into a set of Voronoi cells. In the training phase a set of already classified past observations is used to fine tune the locations and the decisions of the Voronoi vectors. After an initialisation procedure vectors are then computed by a gradient search type algorithm. Specifically, an observation is picked randomly from the past observations; if the decision of the closest Voronoi vector and the decision associated with the new observation agree then the Voronoi vector is moved in the direction of the observation else it is moved away. It can be shown [8] that if T is the number of observations and K the number of codebook vectors as K and T go to infinity while K/T goes to zero the decision regions of the LVQ converge to the optimal decision regions. In this paper we present our experiences in implementing a classifier for images in filmed sequences that is built using LVQ networks and genetic algorithms. We first briefly discuss about previous work in genetic optimisation of classifiers then we expose our methodology.

2. Related Work

Despite the vast amount of literature already produced on global optimisation of neural networks and other classifiers using genetic algorithms, parameter tuning by hand seems to be yet the state of the art, for example in backpropagation [5] and for Kohonen's self-Organising Map [6]. Statisticians estimating parameters for classifiers (the statistical equivalent of neural network training) face a similar problem. Some global optimisation procedures have been tested: simulated annealing and stochastic approximation, but little mention is made to genetic algorithms, except for optimising k-nearest neighbour [7]. Some other methods for optimising LVQ have been based on incremental approaches [1] which are still local error-gradient descendent search algorithms on the space of networks or dictionaries with different size. An interesting methodology is the one proposed by Perez and Vida [3]. This method adds or takes codevectors after presentation of the whole training sample, eliminating those that have not been near any vector in the training sample, and adding as new codevector a vector from the training sample that has been incorrectly classified, and, besides is the most distant from all codevectors belonging to its same class. This approach has the advantage of not relying on threshold parameters, but it still has the problem of being a local search procedure, that optimises size step by step; and besides, it relies on heuristics for the initial weights.

3. Genetic Evolution Of An Lvq Neural Network.

A colour image classifier is obtained using three LVQ networks where each and every one of them take as input a different RGB component of the patterns. While training is executed separately for each network, during classification only the majority vote of the three network, when existing, is considered. Typically images are pre-processed to limit the computational effort required. For the moment we have been using patterns of 60x60 pixels whose intensity is computed as the average intensity of the corresponding area in the original image. Even if the chosen pre-processing is kept quite simple for practical reasons this doesn't invalidate the merits of the proposed methodology. The asymptotic properties of LVQ already discussed in the introduction guarantee that accurate classifications can be obtained with a relatively low number of codebook vectors. Moreover LVQ ability to approximate the probability distribution of observation space patterns can be conveniently used to exploit the high correlation of images belonging to a same place in the environment. Given a region in the filmed sequence and supposing that no outliers are present in the film (this condition can be easily fulfilled by excluding frames with unexpected variations i.e. a person passing by) a representative codebook for the region is obtained directly from LVQ training procedure simply associating the same label to all frames in the corresponding sub-sequence. Even if LVQ can considerably simplify the task of building a classifier for images taken from films we must solve other issues; namely we have to identify regions in the environment together with their bounding frames in the sequence. The number of regions equals the number of Voronoi cells and thus defines the size of the classifier and has a strong influence on its accuracy while the choice of boundary frames determines the "shape" of the Voronoi tessellation of the observation space and so affects both the classifier accuracy and its generalisation ability. Thus we are confronted with a problem where every candidate solution can be quantitatively evaluated only in respect to classifier size whereas effects on classification ability can be just estimated qualitatively or determined explicitly only experimentally. In these conditions training the network by hand is a tedious and excessively time consuming task: just to mention one case we took about six days to get a classifier with 95.2% accuracy and 147 codebook vectors for a filmed indoor environment composed by 2500 frames. Using genetic algorithms we have been able to train an optimal network in comparable time by a completely automatic procedure. In this paragraph we discuss general aspects of the proposed methodology. In paragraph 4 we give a detailed and algorithmic description of the procedure.

3.1 Hybrid Optimisation.

As cited by Yao [12] genetic algorithms are extremely slow when used to directly compute weights of a neural network. This happens because they are global optimisation methods while many neural networks training procedures are derived from gradient descendent search methods. On the other hand genetic algorithms are not affected from the problem of getting stuck in local minima so, given enough time, they are able to behave well in some difficult learning tasks where common neural training fails. We can still combine benefits of both paradigms, global optimisation of

genetic algorithms and quick local search of traditional neural training, if we let the genetic algorithm to search only for initial parameters of the network so to use them as a starting point to standard neural training. Following this approach we use the genetic algorithm to select the number of regions in the filmed sequence, the start and end frame for each region, LVQ learning rate α and the number of training steps. During the evolution process this parameters are then used to build and train each LVQ network that composes the population being optimised.

3.2 Training And Test Sets.

To evaluate correctly the fitness of the LVQ classifier we need to test it on patterns which don't belong to the training set. To do so we extract a subset S_{TEST} of images from the original filmed sequence S_O .We use S_{TEST} to evaluate the classifier's fitness while we use $S_{TRAIN} = S_O - S_{TEST}$ to train it. We select images with a uniform random sampling procedure so to ensure maximum independence between training and test set. We found that a ratio $i_{TEST} / i_O = 25\%$, where i_O is the number of images in the original sequence, i_{TEST} the number of images in the test sequence, is more than adequate.

3.3 Genome Encoding And Genetic Operators.

We encode each LVQ network in a binary genome scheme composed by $i_{TRAIN}+10$ bits, where i_{TRAIN} is the number of images in the training sequence. The first i_{TRAIN} bits encode regions in the filmed sequence together with their start and end frame , bits from $i_{TRAIN}+1$ to $i_{TRAIN}+5$ encode the learning rate α and the last five bits encode the number of training steps. The first i_{TRAIN} bits are interpreted as follows: if bit *i* is set to one then frame *i* in the sequence is a bounding frame of a region, if bit *i* is set to zero and bit *b (b<i)* is the first bit set to one that immediately precedes bit *i* while bit *e (e>k)* is the first bit set to one that immediately follows bit *i* then all frames $k \in [b,e]$ are interpreted as one region and they are assigned the same label. In a few words each region is encoded in the genome scheme by a sequence of bits set to zero delimited by two bits set to one; the leftmost bit position is the start frame of the region while the rightmost bit position is the end frame of the region. So if the total number of bits set to one in the first i_{TRAIN} bits of the genome scheme is *n* we have defined *n-1* different regions in the filmed sequence and hence we have to train a network with *n-1* codebook vectors. In this way we are able to specify variable size neural networks with a fixed length genome. As recommended by Kohonen et al. [9][10] an appropriate number *Ts* of training steps for an LVQ network is found by the rule:

$$Ts = K * (\ n° \text{ of codebook vectors}),$$

where $K \in [50,300]$ is an integer constant. The same authors suggest to use a learning rate always smaller than 0.1; anyway we found experimentally that smaller learning rates are preferred so we let $\alpha \in [0.01,0.05]$. We didn't represent these parameters by a real value encoded genome because small variations in both of them have an insignificant role on the training process. The choice to quantize both α and *K* with 5 bits has proven adequate.

We use double point crossover to combine the first i_{TRAIN} bits in the genome and single point crossover for bits $[i_{TRAIN}+1, i_{TRAIN}+5]$ and bits $[i_{TRAIN}+6, i_{TRAIN}+10]$. Mutation is a simple bit flipping operator.

3.4 Individual Score.

To compute the score of each individual we must first decode the genome to get the starting parameters of every neural network. Then three LVQ networks are created and trained accordingly so that each network receives in input a different RGB component of the given patterns. Successively we randomly select from the test set *Tr* patterns that are fed to the networks to determine the number *Ac* of correct answers. An answer is considered correct only if the majority vote of the three networks equals the label of the submitted test pattern. *Tr* should increase with the complexity of the learning task; we have seen that the qualitative rule $Tr = 30\% \cdot i_{TRAIN}$ is sufficient to get a correct estimate of the network classification accuracy. To compute an appropriate score we must take into account both accuracy and size of the classifier. As a first tentative score function we used:

$$F1 = Ac/(Tr \cdot Log_{10} Ncv)$$

where *Ncv* is the number of codebook vectors in the network. *F1* express accuracy normalised by size and so can be viewed as a measure of the classification power of the network. We employed *F1* on a problem for which we had already trained by hand an LVQ classifier. The resulting evolved network (tab. 2) was about 23% smaller but also significantly less accurate. We can partially ascribe this result to a wrong choice for *F1* that doesn't take into account a measure of problem complexity. As a matter of fact we must agree that also the problem of finding an optimal classifier for images taken from filmed sequences is posed with a certain degree of arbitrariness. Some tasks require fine localisation, others demand for a quick and approximate answer; meaning of "*optimal*" changes accordingly: in the first case we are more interested in accuracy, in the latter we are more interested in size. This means that the problem we treat admits an infinity of optimal solutions each one depending on a different trade-off between accuracy and size. This also implies that an approach based on multi-objective optimisation is more suitable to the nature of the problem than one based on a single score function.

3.5 Multi-objective programming using genetic algorithms and uniform design.

Multi-objective programming has already been used several times with genetic algorithms: a comprehensive discussion is found in the paper written by Fonseca and Fleming [14]. A common defect of these methods is their inadequacy in finding a set of solutions that is uniformly distributed over the Pareto frontier. Leung Y.W. and Wang Y. recently introduced a new approach that is based on uniform design [13]. Uniform design is an experimental design method whose main objective is to sample a small set of points from a given set of points, such that the sampled points are uniformly scattered. Here we very briefly describe only those aspects of uniform design that are relevant to our work and we refer the readers to [13] for more details. Let there be *n* factors and *q* levels per factor. When *n* and *q* are given, uniform design

selects q combinations out of q^n possible combinations, such that these combinations are scattered uniformly over the space of all possible combinations. The selected q combinations are expressed in term of a uniform array $U(n,q)=[U_{i,j}]_{q\times n}$ where $U_{i,j}$ is the level of the *j*th factor in the *i*th combination. It has been proven that *U* is given by:

$$U_{i,j} = (i\ \sigma^{j-1} \bmod q) + 1$$

where σ is a parameter given in tab. 1.
A well known score fitness function to guide evolution of the population toward the Pareto frontier is:

$$fitness = w_1 f_1(x) + w_2 f_2(x) + \ldots + w_M f_M(x)$$

where $f_1, f_2, \ldots, f_M$ are the *M* objective functions in the multi-objective optimisation problem and $w_1, w_2, \ldots, w_M$ are nonnegative weights such that $w_1 + w_2 + \ldots + w_M = 1$. Different values for the weights vector specify different search directions and lead the algorithm to different optimal solutions on the Pareto frontier. The main pillar in Leung and Wang work is the adoption of uniform design to compose multiple fitness functions such that their directions are scattered uniformly toward the Patero frontier in the objective space.

4. Algorithm Description.

The evolution process of the LVQ classifier is relying on five main aspects: fitness functions composition, initialization, mating, mutation and selection.

4.1 Fitness Functions Composition.

We compose *D0* fitness functions of the form:

$$fitness_i = w_{i,0}\frac{Ac}{Tr} + w_{i,1}\frac{Ncv}{i_{TRAIN}}.$$

$W_{i,j} = (w_{i,0}, w_{i,1})$ is a weight vector determined using uniform design:

$$w_{i,j} = U_{i,j} / (U_{i,0} + U_{i,1})$$

where *Ui,j* is the generic element belonging to the uniform array $U(2, D0) = [U_{i,j}]_{D0\times 2}$ as seen in the previous paragraph. Every fitness function so defined expresses a different trade-off between the LVQ classifier accuracy and size.

4.2 Population Initialisation.

We create an initial population of G individuals whose number of codebook vectors, defined in the first i_{TRAIN} bits of each chromosome, is varying with continuity in [4,200]. Bits controlling learning rate and number of training steps are generated randomly.

4.3 Mating.

Mating is described by the following procedure:

1. Set the offspring population to the empty set. Set $j = 0$.
2. Select the best parent chromosome in respect to the *jth* fitness function.
3. Select randomly another parent chromosome.
4. Select randomly an integer $\Delta \in [1, i_{TRAIN}/2]$
5. Set $i = 1$
6. Select randomly an integer $C_1 \in [1, \Delta]$
7. Let $C_{i+1} = C_i + \Delta$
8. If $C_{i+1} > i_{TRAIN}$ then $j = j+1$; iterate from step 2 until there have been less then $D0$ pairings; else exit.
9. Generate two sons using double crossover on bits $[1, i_{TRAIN}]$ with application points C_i and C_{i+1} .Use single point crossover on bits defining learning rate and number of training steps. Add the resulting offspring to the offspring population. $i=i+1$. Jump back to step 7.

4.4 Mutation.

Mutation is performed by three simple bit flipping operators which are executed with a fixed probability. Each operator acts only on one of the following bit sequences in the chromosome: $[1, i_{TRAIN}]$ or $[i_{TRAIN+1}, i_{TRAIN} + 5]$ or $[i_{TRAIN} + 6, i_{TRAIN} + 10]$.

4.5 Selection.

Selection is performed selecting G individuals among offspring and parent populations. Based on each of the $D0$ fitness functions, we select the best $\lceil G/D0 \rceil$ or $\lfloor G/D0 \rfloor$ chromosomes, such that the total number of selected chromosomes is G.

5. Results

As a preliminary test we evolved a population of LVQ to classify a filmed sequence of a small indoor environment composed of 2500 frames. We used a population of $G = 40$ individuals and we chose $D0=7$ different search directions. We let the algorithm run for 600 generations; results are showed in tab.2. We can immediately notice that fitness function $f3$ was able to find a classifier with an higher accuracy and a smaller size than the one trained by hand. We can also see that accuracy and size generally grows going from $f0$ to $f6$ with the notable exception of $f6$ that produces a worse classifier than $f5$. This is due to overfitting of the resulting classifier; the worst accuracy so obtained is compensated by the higher size since $f6$, with respects to $f5$, weights size more than accuracy. The evolution process took about 4 days on a Pentium IV class processor.

Numbers of levels per factors	Numbers of factors	σ
5	2-4	2
7	2-6	3
11	2-10	7
13	2	5
	3	4
	4-12	6
17	2-16	10
19	2-3	8
	4-18	14
23	2,13-14,20-22	7
	8-12	15
	3-7,15-19	17
29	2	12
	3	9
	4-7	16
	8-12,16-24	8
	13-15	14
	25-28	18
31	2,5-12,20-30	12
	3-4,13-19	22

Table 1. Values of parameters for different number of factors and different number of levels per factor.

Fitness function	*LVQ accuracy*	*LVQ size*
Hand trained LVQ	95.2	147
First tentative f	93.4	113
f0	89.6	81
f1	91.1	95
f2	94.2	122
f3	95.6	138
f4	96.7	160
f5	97.5	174
f6	97.3	196

Table 2. Comparison of evolution with various fitness fucntions.

6. Conclusions

We have proposed a methodology, based on genetic algorithms, for evolving automatically an LVQ classifier for images taken from filmed sequences. The algorithm produces several LVQ networks, each one for a different trade-off between accuracy and size, so that the user can choose the one that best suits his needs. Even if the first results are promising we need to pursue further experimentation to see how the algorithm behaves with bigger and more complex environments. Some investigation should also be done on the influence of different pre-processing methods [11].

REFERENCES

[1] *Alpaydim, A.;* "GAL: Networks that grow when they learn and shrink when they forget." Technical Report TR-91-032,International Computer Science Institute, May 1991.

[2] *Schmidbauer, O.; Tebelskis, J.* **"**An LVQ based reference model for speaker-adaptive speech recognition". IEEE International Conference on Acoustics, Speech, and Signal Processing, 1992, Volume: 1 , Page(s): 441 -444

[3] *Perez, C.P.; Vida, E.* "Constructive design of LVQ and DSM classifiers". New Trends in Neural Computation; 1993.

[4] "The number of clusters". Published in comp.ai.neural-nets,Adapted Mar 16, 1996, from the SAT/STAT User's Guide(1990) and Sarle and Kuo(1993).Copyright 1996 by SAS Institute Inc, Cary, NC, USA, 1996.

[5] *W.Schiffman; M. Joost;R.Werner*. "Optimization of the backpropagation algorithm for training multilayer perceptrons". Technical report, University of Koblens, Institute of Physics, 1994

[6] *F.Murtagh; M.Hernandez Pajares.* "The kohonen self-organizing map method: An assessment". Technical report, European Space Agency/UPC,1993.

[7] *Jr.James Kelly; A.Lawrence.* "Hybridizing the genetic algorithm and the k nearest neighbour classification algorithms". In Richard K.Belew and Lashon B.Booker, editors, ICGA91, san Mateo, CA, 1991. Morgan Kaufmann.

[8] *A.LAVigna.* "Nonparametric Classification Using Learning Vector Quantisation". Phd thesis, TR 90-8. University of Mariland.

[9] *Teuvo Kohonen*. "The self organizing map". IEEE proceedings vol 78 n.9 1990 pag.1464

[10] *T.Kohonen; J.Kangas; J.Laaksonen; K.Torkkola*. "LVQ-Pack: The learning vector quantization program package, version 2.1". LVQ Programming Team Technical Report. Laboratory of Computer and Information Science, Helsinki University of Technology, Helsenki, Finland (1992).

[11] *Trygve Randen; J.H Husoy*. "Multichannel filtering for image texture segmentation". Technical Report. Department of Electrical and Computer Engineering. Rogaland University Center.

[12] *Xin Yao*. "A Review of Evolutionary artificial neural networks". International Journal of Intelligent Systems, 8:539-567

[13] *Leung, Yiu-Wing; Wang, Yuping.* "Multiobjective programming using uniform design and genetic algorithm", IEEE Trans. Syst., Man, Cybern. Aug 2000.

[14] *Fonseca, C.M.; Fleming, P.J.* "Multiobjective optimization and multiple constraint handling with evolutionary algorithms-Part I: Unified formulation" IEEE Trans. Syst., Man, Cybern. Jan 1998.

Learning Robot Navigation in a Virtual World

P. Remagnino, N. Monekosso, O. Hatoum, and Y. Halabi

Digital Imaging Research Centre,
School of Computing and Information Systems,
Kingston University, United Kingdom
{p.remagnino,n.monekosso}@king.ac.uk
http://www.king.ac.uk/dirc

Abstract. Virtual and real world can be combined to train LEGO-based robotic platforms (LEGObots) to solve visual tasks. LEGObots roam on an empty floor controlled by a software agent that learns the optimal path in a cluttered virtual terrain. Machine learning and Computer Vision techniques drive the learning: a camera, the eye of the LEGObot, films the top of the scene, while a software agent processes the scene top view and controls the LEGObot through an infrared port. This is our first step towards a fully embodied robotic platform, able to find the optimal path in a virtual environment, and transpose its experience to a real terrain.

1 Introduction

Machine Learning techniques are necessary to learn complex tasks, a model can not possibly be precompiled when robots are required to solve real problems in the real world. Uncertainty and data incompleteness are the main problems. The approach we take is to simplify the learning task of the robotic platform by instructing it in a virtual domain, and then testing the effectiveness of the virtual training in a real situation. In this paper we will present some preliminary results with a virtual domain and how

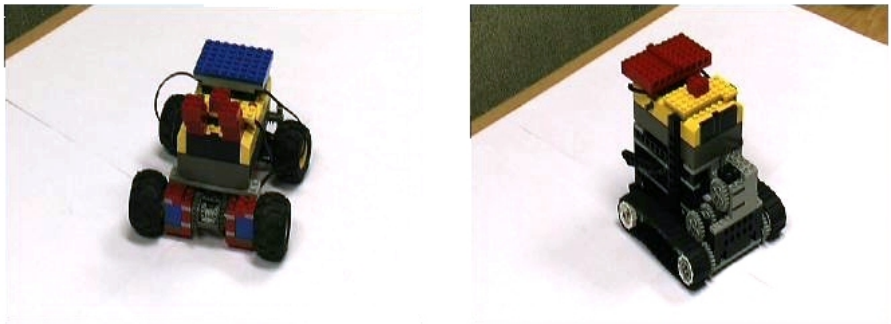


Fig. 1. Two designs of LEGObot

a robotic platform can be instructed to solve visual searches in a partially known environment. The robotic platforms were built using the LEGO MINDSTORM kit [2]. Figure 1 illustrates two examples of robotic design. In our preliminary experiments virtual environments (virtual floors) are represented by paintings (Figure 2) with a start and a goal state (usually top-left and bottom-right corner respectively). The user can create obstacles in the virtual environment (the painting) by marking the selected area with a tool developed in our laboratory. The Q-learning technique [3] was employed to

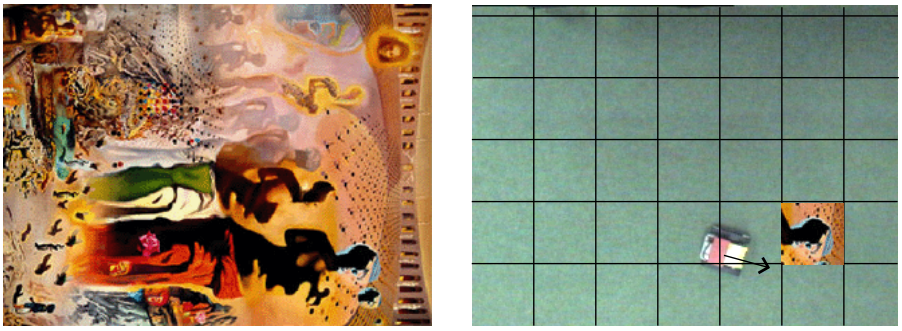


Fig. 2. The virtual floor: a painting of Dali, and the field of view in front of the LEGObot

train the robot. It was tested with a quality table (discretised terrain) and work is under way to use a standard back-propagation neural network [4] to approximate the quality landscape. The software agent learns how to control the two motors of the LEGObot to guide it to the goal state. The available actions are represented by the degrees of freedom of the motors mapped into the four cardinal directions, while the robot field of view is a square region of interest (the cell) in front of the robot, as shown in Figure 2. Future experiments will include the use of 3D virtual worlds (built using JAVA 3D) and more complex and articulated tasks.

2 The System

A considerable amount of time was devoted to build the system infrastructure, to provide a reasonable test-bed for experiments. The system consists of one camera connected to a personal computer which in turn uses an infrared transmitter to control the LEGObot. The software was entirely implemented in JAVA, with a fairly complex hierarchy of classes which form the basis of the main agent class. The agent class consists of three layers (all JAVA classes) including the communication, algorithm and reasoning modules. Each module is independent but all contribute to the learning of the optimal path. At present communication is limited to the acquisition of visual data and the control of the direction of motion, however the design can easily incorporate future developments, in particular the communication between agents for co-operation and co-ordination of actions. The algorithm module is limited to the implementation

of the reinforcement learning technique, including the Q-learning and a preliminary version of the neural network implementation. The reasoning module has not been developed yet, however, its role is clear, it will be used to reason about the environment, to schedule tasks, and choose the most suitable algorithm. Stochastic scheduling will be used to choose the optimal task and related strategy given the current agent state and environment conditions.

At present an agent is pervasive being represented by one thread in a multithreaded main process controlling remote pieces of hardware. However, ongoing work is under way to embed the agent into a low cost robotic platform.

3 An overview of Reinforcing Learning

Reinforcement Learning (RL) is a machine learning technique that allows an agent to learn by trial and error which action to perform by interacting with the environment. Models of the agent or environment are not required. At each discrete time step, the agent selects an action given the current state and execute the action, causing the environment to move to the next state. The agent receives a reward that reflects the value of the action taken. The objective of the agent is to maximise the sum of rewards received when starting from an initial state and ending in a goal state. One form of RL is Q-Learning [5]. The objective in Q-learning is to generate Q-values (quality values) for each state-action pair. At each time step, the agent observes the state s_t, and takes action a. It then receives a reward r dependent on the new state s_{t+1}. The reward may be discounted into the future, meaning that rewards received n time steps into the future are worth less by a factor γ^n than rewards received in the present. Thus the cumulative discounted reward is given by (1)

$$R = r_t + \gamma r_{t+1} + \gamma^n r_{t+2} + \cdots \tag{1}$$

where $0 \leq \gamma < 1$. The Q-value is updated at each step using the update equation (2) for a non-deterministic Markov Decision Process (MDP)

$$\begin{aligned} \hat{Q}_n(s_t, a) \longleftarrow\ & (1 - \alpha_n)\hat{Q}_{n-1}(s_t, a) + \\ & \alpha_n(r_t + \gamma \cdot max_{a'}\hat{Q}_{n-1}(s_{t+1}, a')) \end{aligned} \tag{2}$$

where $\alpha_n = \frac{1}{1+visits_n(s_t,a)}$. Q-learning can be implemented using a look-up table to store the values of Q for a relatively small state space. Neural networks are also used for the Q-function approximation.

4 Optimising the path in a virtual environment

A LEGObot was placed on the floor of the laboratory and its actions controlled by the RL algorithm to learn the optimal path. At first the classic RL was adopted by tessellating the floor into a grid of virtual cells, and the convergenced proved using merely the positional information of the robot. The current position of the robot was detected with the aid of visual routines, extracting the robot shape (blob) using chromatic information followed by a region growing algorithm to estimate the centroid of the LEGObot

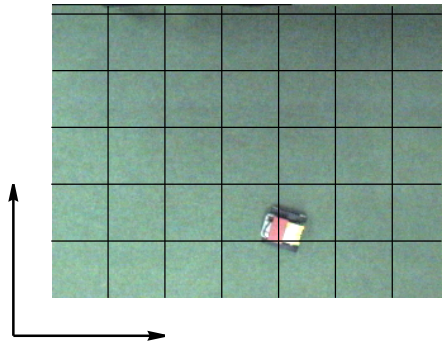


Fig. 3. The robot and the virtual grid, with the co-ordinate system.

with respect to the grid coordinate system (as shown in Figure 3). In a second stage the agent controlling the robot was only allowed to use the visual information in front of it, that is the cell immediately in front of the robot given the current direction of motion. Different types of grid size were tested. The idea is to move one step towards a neural networked approach, where backpropagation is driven by the visual information, rather than positional information [1].

The frames in Figure 4 were extracted from the optimal sequence, run by the LEGObot trained to learn the cheapest path from the starting position (top left corner) to the goal position (bottom right corner). The final frequency map, that is the map indicating the most frequented cells in the grid, is shown by the following in Figure 5 and Figure 6. The map shows that the robot successfully reached the goal state using visual information provided by two different virtual floors (Dali's painting and London underground, the Tube).

5 Two virtual environments

The technique has been tested with a number of different virtual environments. As mentioned ealier on in the text, at present, virtual environments are merely virtual floors, represented by images of paintings or maps.

In order to display the results of a training session until convergence (that is RMS varying little - residual noise - below a given threshold) each cell shows four arrows, indicating the four main directions of motion (north, east, south and west). Each arrow has a colour indicating the quality (Q-value) of the state and action pair. The range varies between -1 and +1, and the colour of an arrow respectively between red and yellow, darker being closer to zero. The zero value (black colour) corresponds to maximum entropy, that is no decision can be made between the positive (yellow) and negative (red) quality of the pair to solve the problem in hand.

Figure 7 illustrates one of the used mazes. Figure 8 shows the frequency map indicating that the path was correctly optimised from the top-left corner to the bottom-right corner, the goal green in colour. Figure 9 renders the quality landscape using the

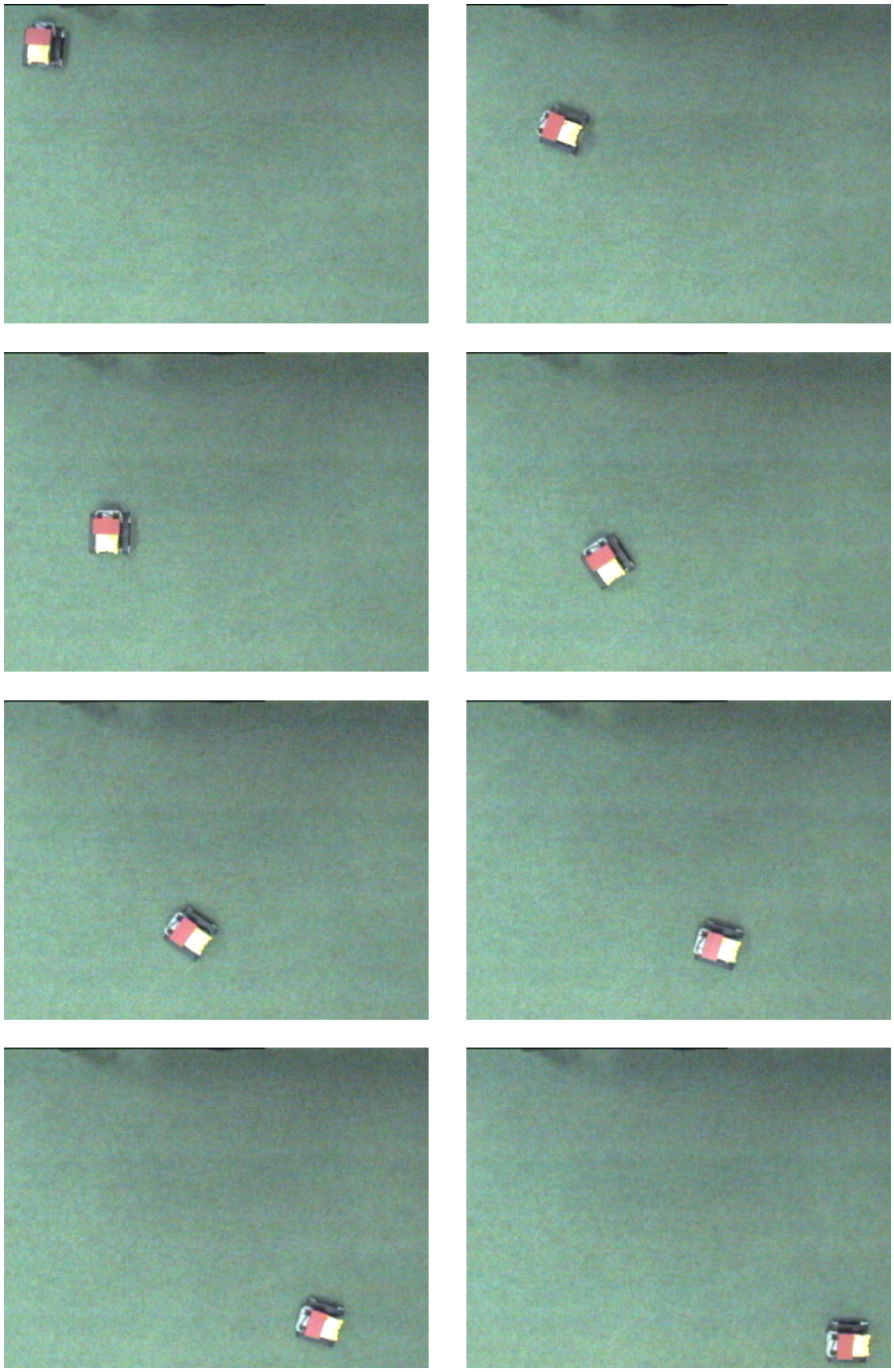


Fig. 4. The optimal sequence run by the LEGObot

Fig. 5. The Dali virtual floor and the optimal path.

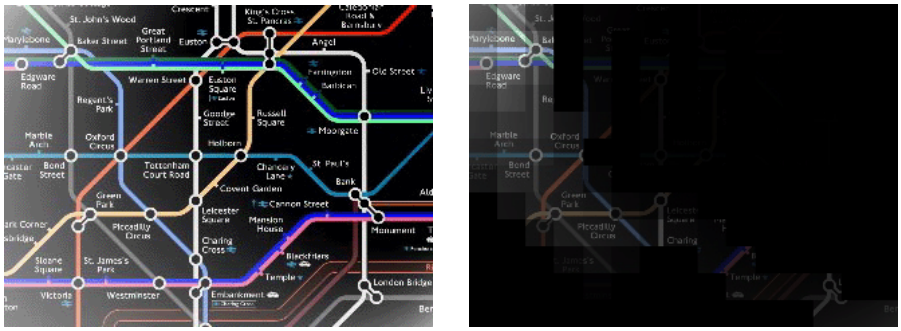


Fig. 6. The Tube virtual floor and the optimal path.

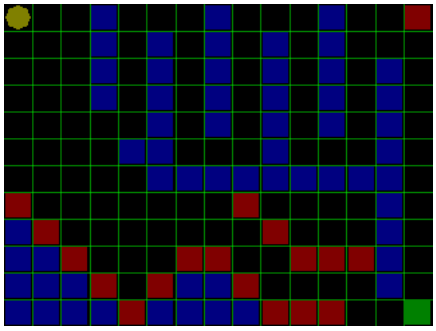


Fig. 7. One of the used mazes.

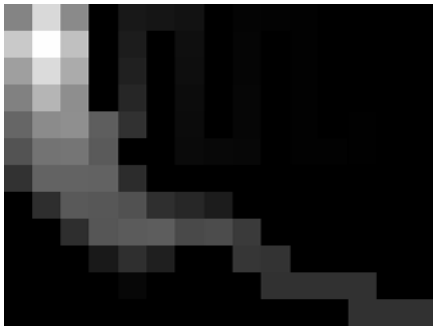


Fig. 8. The frequency map.

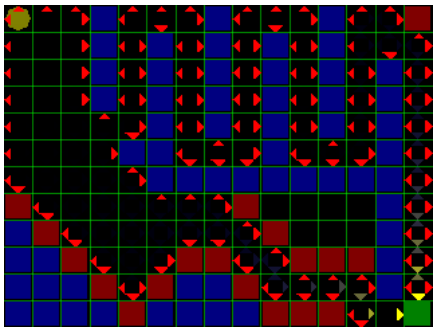


Fig. 9. The Q-values.

colouring explained earlier on in the text. Figure 10 shows another example, on the left

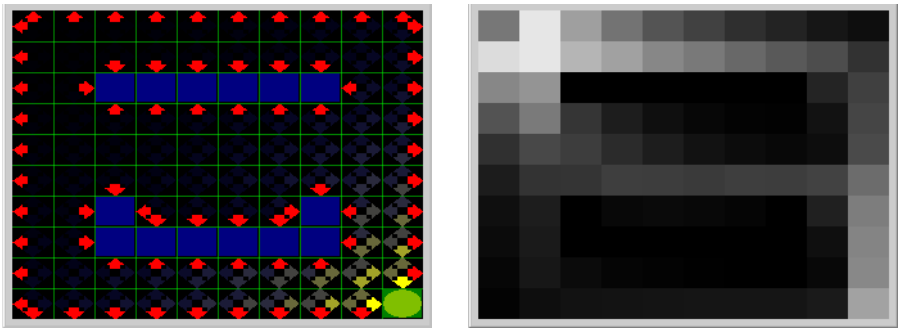


Fig. 10. Another example of maze and the frequency map.

the quality values and on the right the actual frequency map, after convergence. Several training sessions were tested with different virtual floors and a different number of epochs varying from 100 to 1000. Convergence was proven to occur only after 20 epochs. Figure 11 illustrates the convergence for one of the cells to four asymptotic values, one for each one of the different main directions.

6 Discussion

We presented the prototype of a system, which implements a low cost mobile platform controlled by an agent-based software module. The agent perceives the environment through a camera connected to the computer, and remotely moves the robot through an infrared port. The agent makes use of the standard Q learning technique, however instead of using directly positional information, it uses the field of view of the LEGObot

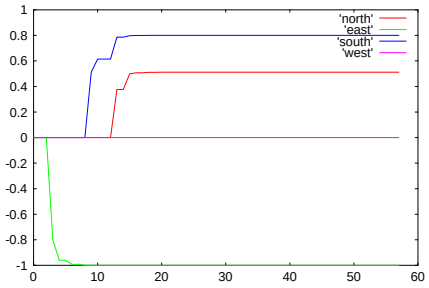


Fig. 11. Example of convergence of q value for a typical grid cell.

(the cell in front of the robot) as the state and the four cardinal directions as possible actions (mapped onto the LEGObot motors). An efficient JAVA implementation of the Q learning allows the rapid learning of the optimal path from a starting position to the goal position. The system is still at prototype stage, even though it was written in a modular and highly extensible way. Future work will include the use of more robotic platforms to solve the same visual task, and a design to embed the system into the robotic platform itself.

References

1. Parag Batavia, Dean Pomerleau, and Charles Thorpe. Applying advanced learning algorithms to alvinn. Technical Report CMU-RI-TR-96-31, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, October 1996.
2. J.B.Knudsen. *The unofficial guide to LEGO MINDSTORM Robots*. O'Reilly, 1999.
3. R.S.Sutton and A.G.Barto. *Reinforcement Learning: an introduction*. MIT Press, 1998.
4. S.Haykin. *Neural Networks, a comprehensive Foundation*. Prentice Hall, 1999.
5. C. J. C. H. Watkins. *Learning with delayed rewards*. PhD thesis, University of Cambridge, 1989.

Natural indoor landmark recognition for robot navigation

A. Micarelli, E. Sangineto, G. Sansonetti

Laboratorio di Intelligenza Artificiale
Dipartimento di Informatica e Automazione
Università "Roma Tre",
via della Vasca Navale 79, 00146 Roma, Italy.
e-mail: {micarel,sanginet,gsansone}@dia.uniroma3.it

Abstract. In this paper, we address the problem of how to recognize natural landmarks for robot navigation (in an unknown office-like environment) given a description of the actual robot neighbourhood acquired through a ring of ultrasonic sensors. Such a proximity information is coded in a local map describing obstacles around the robot by means of occupied cells. The problem lies in recognizing, starting from the map, patterns of places that we want to utilize as landmarks in the building a global topological map of the whole explored environment. We show two different techniques. The first one relies on a wavelet representation of the local map and in the cross-correlation of wavelet coefficients of the input case with old cases extracted from a library, while the second one exploits an abstract description of the patterns through geometric constraints. About the latter, the experimentation work is still in progress, but we can already show very good results based on a large number of trials.

1 Introduction

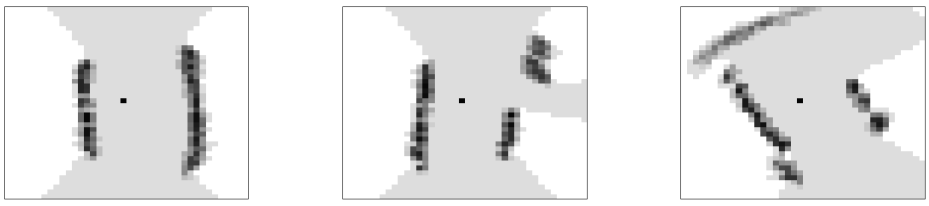
In literature (see, for example, [3]), *landmarks*, in robot navigation field, are distinct features that a robot can recognize from its sensory input. There are two main type of landmarks: *natural* and *artificial* ones. In the following we assume the below Borenstein definition [3]:

> *natural landmarks* are those objects or features that are already in the environment and have a function other than robot navigation; *artificial landmarks* are specially designed objects or markers that need to be placed in the environment with the sole purpose of enabling robot navigation.

Generally, "natural" landmarks are man-made objects, and they work best in highly structured environments, like corridors, manufacturing floors, hospitals... On the other side, artificial landmarks usually have an easy recognizable shape or colour, or they may include additional information such as in the form of bar

codes. Moreover, landmarks can be used either to help robot localization (if their position in the environment is well known), or to give important information to progressively construct a global knowledge about the unknown environment the robot is exploring.

In this work we propose some pattern recognition techniques to recognize natural indoor landmarks in an unknown environment with the purpose to use them in the building of a global map of the environment itself. The robot architecture we take into consideration [9] is equipped with a ring of ultrasonic rangefinders giving a proximity information of the near obstacles. These data, with their uncertainty degree, are coded with fuzzy values of cell occupancy in a local map (presently 40×40 cells) having the robot in its centre. Indeed, the interaction between the ultrasonic sensors and real-world environments is very complex, and there are several sources of uncertainty when an ultrasonic rangefinder is used in determining the position of an object. Consequently, fuzzy logic was chosen as the tool to process sensor data. Here we assume that this process is done in a sub-level of the robot software architecture and we refer to [9] for details. In Figure 1 are represented two typically fuzzy maps describing robot neighbourhoods, taken in different positions of the robot itself during its exploration of a corridor. Different gray levels in the images represent different fuzzy values. Pixels (x, y) with darker gray levels correspond to lower values $M(x, y)$ of membership to the set of empty cells E. Finally, white pixels are unexplored regions, with a fuzzy value of membership to E equal to 0. Indeed, we remember that, in fuzzy logic, the principle of "tertium non datur" does not hold, so the maximal uncertainty about the membership of a cell c to set E of empty cells does not imply, of course, that c surely belongs to the set of occupied cells. We assume here that these maps are our knowledge representation input.



(a) (b) (c)

Fig. 1. Three examples of fuzzy maps representing proximity information computed by integrating data coming from all the ultrasonic sensors of the robot ring taken in different positions of the robot exploration journey: (a) image representing a passage, (b) image representing a crossing, (c) image representing a corridor corner. The black pixel in the middle is R, the robot position, while different gray-levels correspond to different fuzzy values of $M(x, y)$. Finally, white pixels represent unexplored regions.

The problem we deal with in this paper, is the classification of the type of the robot office-like environment starting from its neighbourhood proximity information described by an occupancy fuzzy map. We distinguish among the following classes: Passage, Corner, T-Junction, End Corridor, Open Space, Bottleneck; with the possibility to extend the class catalog.

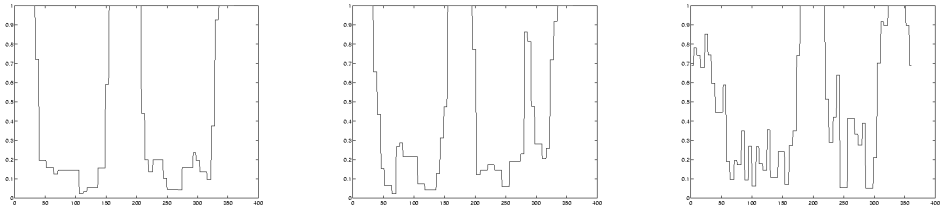
The recognized patterns are natural landmarks. Our aim is to use them in building a *global topological map* of the whole environment in which the robot acts (this part of the robot architecture is still not realized). Usually, a topological map is described through a planar graph, in which nodes represent different regions and edges the adjacency relation between regions, while a *metric map* is generally described using a grid [6]. Our fuzzy maps are examples of *local* metric maps. We refer to [3, 11, 4] for a comparison between topological and metric maps in robot navigation. Here we only remark that topological maps can be preferable to metric ones when the whole environment is very large due to their more abstract and compact representation that allows more efficient planning, that they do not require accurate determination of the robot position for a long period of time and that they can better interface with a natural language plan ("go straight along the passage, turn right at first crossing..."). On the other side, building a topological map requires a good recognition of places, independently of the point of view and of the possible ambiguity of sensorial information [11].

We explored two different solutions. The first exploits shape representation of the robot neighbourhood through a wavelet encoding, while the second is based on the extraction of pre-determined geometric features and on their comparison with constraint models. The rest of the paper is articulated as follows. Section 2 shows the first (chronologically speaking) solution and Section 3 the second one, explaining the reasons that induced us to follow a different approach. Section 3.1 shows some experimental results (very promising) and Section 4 draws final conclusions.

2 A Solution Based on Wavelet Representation

Often, the first step, in a wavelet (or Fourier Descriptors) based representation of a bi-dimensional shape, consists in its linearization, i.e., in describing the shape (usually a silhouette) through a sinusoidal function [5].

In our case, first of all we transform the fuzzy map (M) in a polar coordinate system, centered in $R = (19, 19)$, the position of the robot in M. More exactly, given a map M in which $M(x, y)$ is the fuzzy value representing the degree of membership of the cell (x, y) to the set E of empty cells, the *World Mark* $W(M)$ of M is represented by a Cartesian reference frame in which the horizontal axis represents the directions around the robot (ranging in $[0, 360[$) and the vertical axis represents the fuzzy range $[0, 1]$. A point (θ, μ) in $W(M)$ states that, in M, starting from R, and following the direction θ, the cell (let us call it $c' = (x, y)$) with the highest value of membership to E is such that $M(x, y) = \mu$. Figure 2 shows the World Marks of the fuzzy maps of Figure 1.



(a) (b) (c)

Fig. 2. Three examples of World Mark representations corresponding to the fuzzy maps of Figure 1: (a) the World Mark of Figure 1 (a), (b) the World Mark of Figure 1 (b), and (c) the World Mark of Figure 1 (c).

It is now possible to encode $W(M)$ as a signal, providing its *time-frequency* (or spice-frequency) representation by using a wavelet transform [8].

Wavelets are mathematical functions that satisfy certain requirements and are used in representing signals or other functions. This idea is not new. However, the innovative aspect behind wavelets is to analyze according to scale (dependent inversely on frequency). In other terms, these functions enable to split data into different frequency components, and then to study each component with a resolution matched to its scale. Thus, they have advantages over traditional Fourier methods in analyzing physical situations where the signal contains discontinuities and sharp spikes. This makes wavelets an interesting and useful tool for image representation and analysis.

The wavelet analysis procedure lies in adopting a wavelet prototype function, called *mother wavelet*. Temporal analysis is carried out with a contracted, high-frequency version of the mother wavelet, while frequency analysis is carried out with a dilated, low-frequency version of the same wavelet. Since the original signal or function can be represented in terms of a wavelet expansion (using coefficients in a linear combination of the wavelet functions), data operations can be performed using just the corresponding wavelet coefficients. And if the best wavelet adapted to data at hand is selected, or the coefficients below a threshold truncated, data are sparsely represented. This sparse coding makes wavelets an excellent tool in the field of data compression as well.

One startling aspect of wavelet methods is officered by the extremely fast (compared with similar techniques) algorithms that compute wavelet transforms. The discrete wavelet transform (DWT) turns a discretized signal into a set of discrete wavelet coefficients. The *fast DWT* or *pyramid algorithm* is due to Stephane Mallat [7] which applayed identical algorithms of signal processing literature to wavelets. The fast DWT consists of three components: low-pass and high-pass filters L and H and an operation called *dyadic decimation*. Dyadic decimation removes every odd member of a sequence, halving its overall length. For sake of

clarity, we assume that the filter operations include dyadic decimation so that applying L means apply the low-pass filter followed by dyadic decimation. The DWT operates recursively by taking a sequence of values, applying L and H and then re-applying the same procedure to the result of the L filter (see figure 3).

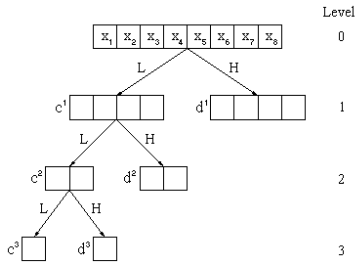


Fig. 3. The fast DWT.

We denote our sampled signal by the vector x of length $N = 2^j$. Then, the j-iteration of the algorithm produces two sets of coefficients: the discrete wavelet coefficients at scale j

$$d^j = HL^{j-1}x$$

and the smoothed coefficients at scale j

$$c^j = L^j x$$

for $j = 1, ..., J$. The computed coefficients in these sequences form the wavelet decomposition of the measured signal x. The coefficients d^j can be interpreted as the *details* of the signal x at coarser and coarser resolutions as j is increased, while the coefficients c^j represents the *approximations*. Another way of viewing this is to regard d_j as the detail lost when moving from one representation c^{j-1} to a smoother one c^j. In essence, the fast DWT takes x and transforms it to $c^J, d^J, d^{J-1}, \cdots, d^1$. The filters employed in the fast DWT are well-known to the signal processing community and referred as *quadrature mirror filters*.

Thus, for each pattern to recognize, we produce a set of fuzzy map examples (using a software simulator [1]) and, by means of the fast DWT, compute the corresponding set of wavelet coefficients c_3. Then, for each example, the wavelet coefficients so produced are stored in a case library. Indeed, we organize our recognition system in a Case Based Reasoning (CBR) architecture [2]. In particular, each case in the library is a couple of the type $\langle C, L \rangle$, where C is a vector representing the coefficients of a pattern example, and L is the landmark associated with. At run-time, the fuzzy map M_i, produced by the i-th sensor reading (at i-th step of the robot exploration journey), is first transformed in its world mark representation $W(M_i)$; after, from this, by using the fast DWT, we extract the vector C_i of the coefficients of $W(M_i)$ wavelet transform. Finally, C_i is sequentially matched with all the cases in the system library. The

matching between C_i and the candidate case $\langle C_j, L\rangle$, retrieved from the library, is performed through a standard cross-correlation technique between the two coefficient vectors C_i and C_j.

One of the most significant advantage of CBR over other Artificial Intelligence techniques is that it allows the case library to be developed incrementally. It is so possible, for our signal classifier, to augment its library with a new case $\langle C', L'\rangle$ when the robot meets new unknown situations C' (for instance, obstacles...) and a human expert, during the system learning phase, gives a landmark L' classification.

There is still not a systematic experimentation of the method, but initially results are promising, with a good accuracy and a low time-consuming. [8].

3 A New Solution Based on Geometric Constraints

Nevertheless the good experimental results of the solution presented in the previous Section, and in an efficiency and in an accuracy point of view, recently we designed and implemented an alternative way, no more based on the using of the World Mark representation but directly extracting geometric features from the original local map. The reasons induced us to try a different approach were been the following.

First of all, we want to avoid the need to compute the wavelet transform at each sensor reading (for each obtained local map), that is a time-consuming operation. In this way it is possible to augment the whole architecture efficiency or, possibly, augment of at least a magnitude order the resolution of the local maps (and, thus, the recognition accuracy) without abandon real-time system responses, that is an obvious issue in robot navigation architectures.

Moreover, the World Mark representation leads to a metric information loss, because in the representation (θ, μ) there is no trace of the original distance of the cell c' (see Section 2) from R. Nevertheless, accurate metric information can be useful when we want to extend our landmark pattern set. For example, the pattern for the landmark Bottleneck, not present in the set of landmarks available in the first solution, is characterized by a different width of the free (empty) space in front and behind the robot. Moreover, in the second solution, we want to introduce the possibility to accurately recognize the presence of obstacles (for example human beings or other "objects" near the robot), which can be distinguished from wall corners or corridor crossings because of the smaller size and different displacement of the occupied regions in their local map.

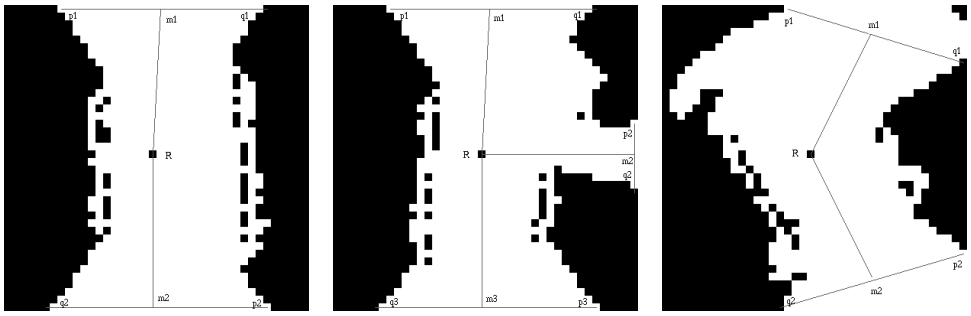
For these considerations, in these last months we are investigating a new solution with a different representation of the geometric knowledge. We explain it starting from the input representation, then we show the modellistic apriori description and finally how these two reperesentations are matched together.

In the input description, first of all, we binarize the local fuzzy map applying to it a suitable threshold. Consequently, from now on, we assume that: $M(x, y) \in \{0, 1\}$, instead of: $M(x, y) \in [0, 1]$, with the meaning that $M(x, y) = 1$ if the cell (x, y) is empty.

We remark that this thresholding operation is not in contrast with the initial fuzzy logic representation of the map. Indeed, we used fuzzy logic only in the phase of sensor data acquisition and fusion, when we build M integrating different ultrasonic readings for each cell (x, y) and we must deal with the uncertainty produced by the ultrasonic sensor model (see [9]). When the fusion of the data of all the readings of the ultrasonic ring is done, we can assume a security threshold to classify a cell as empty or occupied.

The second step in the input representation consists in the analysis of the borders of M, starting from the high-left corner (cell $(0,0)$) and proceeding in a clockwise order (but, of course, the starting point and the order choices are indifferent with respect to our technique). During this analysis we build a segment $s_i = (p_i, q_i)$ when it is possible to link the points p_i and q_i (following the boards of M in a clockwise manner) with a chain of empty cells (Figure 4). We use two threshold, thE and thO: the first to be sure that all s_i are such that $|s_i| \geq thE$, while the second to allow small interrupts (less or equal to thO) to the chain of empty cells linking p_i and q_i to deal with noise. Then, we call m_i the median point of s_i and r_i the segment obtained linking m_i with R.

In this way we obtain a radial geometric frame F. If n is the number of segments r_i ($i \in [0, n-1]$) belonging to F, we say that the *topology class* of F is n and we indicate this as: $TC(F) = n$. The others attributes of F are the length of the segments s_i and the magnitude of the angles $\alpha_i = m_i \hat{R} m_{i+1 \bmod n}$, so we can represent F with the triplet $\langle n, \{\alpha_i\}, \{|s_i|\}\rangle$.



(a) (b) (c)

Fig. 4. Three examples of input representations through geometric frames. They are extracted, respectively, from the fuzzy maps of Figure 1: (a) representation of the local map of Figure 1 (a), (b) representation of the local map of Figure 1 (b), (c) representation of the local map of Figure 1 (c). White pixels correspond to empty values of M and black ones to occupied or unknown cells.

We now explain the modellistic apriori knowledge representation. A pattern for a landmark is characterized by a fixed number of segments r_i that define its

topology class and by some constraints on the possible ranges for angle magnitudes (α_i) and segment lengths ($|s_i|$).

For example, the topology class of Passage, Corner and Bottleneck is 2, the topology class of End Corridor and Open Space is 1, and the topology class of T- Junction is 3. Moreover, *range constraints* for Passage angles are:

$$\alpha_1, \alpha_2 \in [180 - \theta_1, 180 + \theta_1], \tag{1}$$

while, for Corner:

$$\alpha_1 \in [270 - \theta_2, 270 + \theta_2], \tag{2}$$

$$\alpha_2 \in [90 - \theta_3, 90 + \theta_3], \tag{3}$$

where $\theta_1, \theta_2, \theta_3$ are constants tuned in learning phase. We refer to [10] for a complete treatment of range constraints as a tool to represent abstract geometric patterns.

A pattern P for a landmark is represented by the triplet: $\langle n, \{A_i\}, \{S_i\}\rangle$, where n is its topology class ($TC(P) = n$), while $\{A_i\}$ and $\{S_i\}$ are, respectively, a set of range constraints on the magnitude of the angles and on the length of the segments s_i of the landmark.

Given such representations of, respectively, the input image and the landmark patterns, the automatic recognition system is organized, as in the first solution, in a CBR architecture, but with a different similarity function (in Section 2 we adopted a cross-correlation among wavelet coefficients). If F is a frame extracted from the local map as above described, and $\langle P, L\rangle$ is a case retrieved from the system library, where P is the pattern for the landmark L, the system recognize F as an instance of P if both belong to the same topology class and all the attributes of F satisfy the ranges in P. Otherwise, it suggests, as output, the case with the closest pattern. We remark that now, differently since the previous wavelet-based solution, each case in the system library is no more an example of input situation but a *class* describing all the input situations belonging to a landmark pattern. This is the reason for which we speak of "F is an instance of P" instead of "F is similar to P".

More exactly, we define, first of all, a penalty function between a range constraint C ($C = A_j$ or $C = S_j$), and an attribute value v ($v = \alpha_i$ or $v = |s_i|$) as follows:

$$pen(v, C) = \begin{cases} 0 & if v \in C \\ \min\{|v - k_1|, |v - k_2|\} & otherwise, \end{cases} \tag{4}$$

where $C = [k_1, k_2]$. Equation (4) means that we assume to pay no penalties if v belongs to the range C, otherwise the value of the penalty is the distance between v and the closest extreme of C. We can now define the penalty function between a pattern $P = \langle n, \{A_j\}, \{S_j\}\rangle$ and a frame $F = \langle n, \{\alpha_i\}, \{|s_i|\}\rangle$ as:

$$Pen(F, P) = \begin{cases} \infty & if TC(F) \neq TC(P) \\ \delta & n = TC(F) = TC(P), \end{cases} \tag{5}$$

where:

$$\delta = \min_{k \in [0,n-1]} \sum_{i=0}^{n-1} \beta_1 pen(\alpha_i, A_{i+k \bmod n}) + \beta_2 pen(|s_i|, S_{i+k \bmod n}). \quad (6)$$

β_1 and β_2 are two constants used to weigh the importance of, respectively, angle and segment range constraints in (6). Presently, we have $\beta_1 = \beta_2 = 1$.

Suppose now that $\langle P', L' \rangle$ is the case in the library minimizing $Pen(F, P)$. If F and P' belong to the same topology class and all the attributes of F satisfy the ranges in P', then this means that:

$$\exists \overline{k} \in [0, n-1] : \forall i \in [0, n-1] : \alpha_i \in A_{i+\overline{k} \bmod n} \wedge |s_i| \in S_{i+\overline{k} \bmod n}. \quad (7)$$

(7) implies that in (6) $\delta = 0$. In this case, the system answers that the landmark L' was recognized as the place corresponding to the local map in input. We call this situation as an *exact recognition*. Otherwise, it warns that it can not exactly recognize the input and, if $TC(F) = TC(P')$, it suggests L' as the landmark closest to the present situation.

3.1 Efficiency and Experimental Results

To build a frame F from a map M we need to analyze all the cells in the boards of M, with a computational cost of $O(n)$, where the number of total cells of M is n^2. Moreover, if n_1 is the topology class of F, and P is a pattern retrieved by the case library, we need, in the worst case, n_1^2 constraint verification operations to minimize $Pen(F, P)$. There is no indexing method to search in the case library, so all the patterns must be retrieved sequentially. The whole retrieval phase worst case cost is, then, $O(mn_1^2)$, where m is the number of cases of the system library (namely, the number of patterns). Since we have a very restricted number of landmarks (patterns) and of topology classes, we have that $n_1, m \ll n$ (presently $m = 6$ and $n_1 = 3$). For this reason, the whole landmark recognition procedure, the frame F building and its comparison with the cases of the library, has a worst case complexity of $O(n + mn_1^2)$, that we can assume of the order of $O(n)$.

Experimental results on a Pentium III, 450 MHz, 256KB cache, gave a consuming time of no more than 0.11 seconds for the whole procedure, included output graphic displacement operations.

In our tests we used a simulator to produce local maps representing ultrasonic information of the robot. The robot simulator software [1] allows to draw a global map of an office-like environment and to trace in it a trajectory. After that, it creates a sequence of local maps for each discretized position of the robot trajectory. These local maps are computed taken into account the model of the ultrasonic sensors used by the robot [9]. We have one-two hundreds of maps in average each trajectory. These maps are the input of our system tests.

Experimentation phase is still in progress, but we have already tested the architecture with 4 different environments, with a total of 1012 different maps. We obtained 998 exact recognitions (following the definition given in Section 3) and

only 14 errors, with a Precision ratio (defined as the number of $correctanswers/totalanswers$) of 98.6%.

Finally, also if we have still not done a precise comparison, we can surely state that the recognition solution based on constraints is more accurate and faster with respect to the wavelet based one in which the former mistakes much less and is more computationally efficient.

4 Conclusions

In this paper we have presented a pattern recognition problem applied to robot navigation. The proximity information coming from an ultrasonic ring is coded in a local map of the robot neighbourhood and we want to exploit it to understand in what kind of place the robot is. We showed two different solutions, the second one still in progress. Results are very encouraging: we are planning a more systematic experimentation phase, but we can already state that, in front of 1012 classification tests, we obtained, with the second solution, a Precision ratio of 98.6%.

References

1. *Nomad 2000 user's guide.* Nomadic Technologies, 1996.
2. Aamodt, A., and Plaza, E. Cased-Based Reasoning: fondational issues, methodological variations and system approaches. *AI Communications 7. No. 1* (1994), 39–59.
3. Borenstein, J., Everett, H. R., and Feng, L. *Navigating mobile robot: sensors and techniques.* A. K. Peters, Ltd., Wellesley, MA, 1996.
4. Fabrizi, E., and Saffiotti, A. Extracting topology-based maps from gridmaps. In *Proceedings of International Conference on Robotics and Automation, San Francisco* (2000).
5. Haralick, R. M., and Shapiro, L. G. *Computer and Robot Vision.* Vol 2. Addison–Wesley Publishing Company, 1993.
6. Latomb, J. C. *Robot motion planning.* Kluwer Academic Publishers, Boston MA, 1991.
7. Mallat, S. G. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence 11. No 7* (1989), 674–693.
8. Micarelli, A., Neri, A., Panzieri, S., and Sansonetti, G. A case-based approach to indoor navigation using sonar maps. In *Sixth International IFAC Symposium on Robot Control (SYROCO 2000)* (2000).
9. Oriolo, G., Ulivi, G., and Vendittelli, M. Fuzzy maps: a new tool for mobile robot perception and planning. *Journal of Robotic Systems 14. No. 3* (1997), 179–197.
10. Sangineto, E. *Automatic Image Classification by Geometric Abstraction. Ph.d Thesis in Computer Engineering.* Dipartimento di Informatica e Sistemistica dell'Università di Roma "La Sapienza". In press., 2001.
11. Thrun, S. Learning metric-topological maps for inddor mobile robot navigation. *Artificial Intelligence 99* (1998), 21–71.

Multimedia Content-based Information Retrieval

Alberto Degli Esposti, Alessandro Micarelli, Enver Sangineto, Giuseppe Sansonetti

Laboratorio di Intelligenza Artificiale
Dipartimento di Informatica e Automazione
Universita' degli Studi "Roma Tre"
Via della Vasca Navale 79, 00146 Roma, Italy.
e-mail: alb.fradegl@tiscalinet.it , {micarel, sanginet, gsansone}@dia.uniroma3.it

Abstract. In this article we show an architecture which processes multimedial material, subdividing it on the basis of its own semantic contents. We propose an innovatory approach, whose methodology introduces an abstraction level, which has the task to study the relationships among the morphological attributes in a systematic way, before estimating the content. This analysis tries to unify the descriptors information and to gather them into structures that we call over-regions, which represent particular configurations of the objects to be recognized. The purpose is to facilitate the tasks reserved to the higher abstraction levels, down to the elements on which they must work won't be low level attributes, but particular configurations in which those attributes place themselves. The format given to this analysis will have a case-based reasoning nature

1 Introduction

In this article we show an architecture which process multimedia material, subdividing it on the basis of its own semantic contents. We propose an innovatory approach, whose methodology introduces an abstraction level, which has the task to study the relationship among the morphological attributes before estimating the content. The application domain is the soccer. Although soccer is a specific environment, it is sufficiently complex to test general-purpose attitude of our architecture. Let's look quickly the low level analysis (morphological).

The first operation is to compute frame by frame the colours histogram. This operation allow us to evaluate if the camera is observing a field part or a player face or the fans. Doing this we can avoid to perform operations which looking for the ball when the cameras are not looking the play.

After that, we can analize a significant action. To manipulate the images we use 2 descriptors: the first one observes the colours, the second the movements (ball movements and players movements). The Colour descriptor segments the image by searching the two team players positions. It is not possible to observe the ball movement directly, because the camera motion tend to maintain the ball in the middle of the frame that cause the presence of apparent movements. This forces us to look for fixed points in the images to eliminate the apparent movements.

When the goal is not framed is hard to choose a reference, but when we can locate this object, unique in his shape in this domain, we can try to refer all motions (ball an player motions) to this object, eliminating the camera movements. This searching task is semiautomatic to deal with the case in which the goal is not present or is difficult to locate it. When the reference is located, we follow his edge and we vectorialy subtract his motion to all frame pixels. The ball is located manually in the first frame and followed automatically in the rest of the sequence. After that, the case based structural analysis starts.

The first structure which is searched for is the "ball holder": the last player that has substantially changed the ball direction. Knowing his position we observe his coloured shirt to decide the kind of action observed. The last structure observed is the "shot on goal".

2 A structured approach for the semantic segmentation

In order to propose this kind of approach we drew our inspiration from the biological systems working, trying to imitate particularly the working mechanisms of the human visual system. Professor Ennik Kandel's (Nobel Price for Medicine in the year 2000) works have been an inspiration source. In those works he illustrates how the image, impressed on the retina, undergoes 3 processing steps before the brain recognizes the represented object. In the first processing step the colour, shape and movement components are estimated. These three feelings remain separated until the specific cerebral cortex cells unify them into one only perception. The final step of recognizing the object works on perception, not on feelings. This schema seams to suggests the use of a structured approach to improve the performance of the system.

3 A new architecture

We show an application realized by ourselves, as an example of the utilization of the structural analysis. We have built up a system which analyses some filmed sequences pulled out of a soccer game, explaining and recognizing specific observed events. Events like goals and highlights are recognized by the system, so that it can build up a file from which afterwards it will be possible to retrieve information either submitting textual queries, or using queries in a film form, in which some events are represented similar to those we would like to find inside the file.

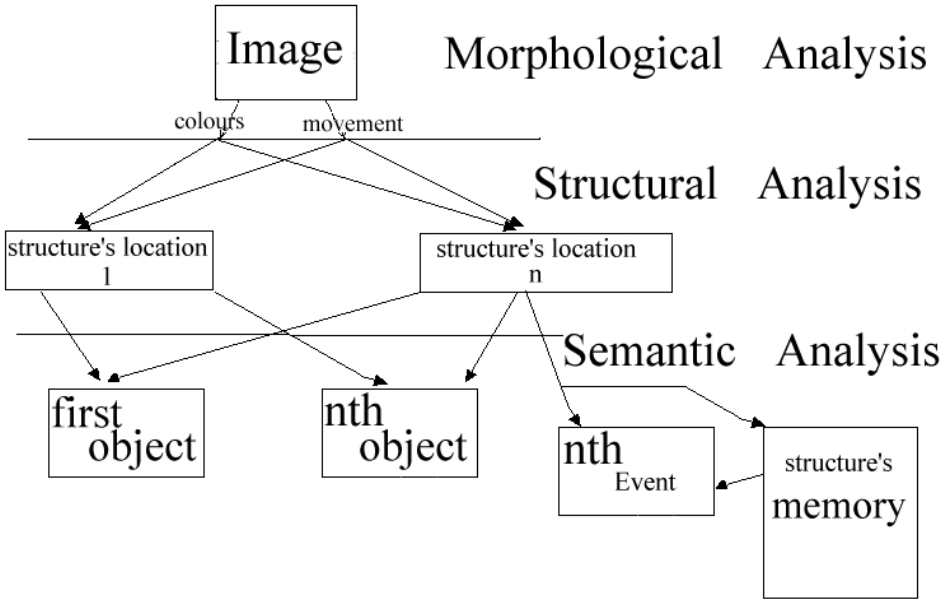


Fig. 1. A New Architecture

Let us first of all analyze how the morphological analysis is realized. The first operation to be performed for each frame is to calculate its image histogram. This operation allows us to evaluate whether the considered film sequence is observing the field or the detail of a player's face, or the terraces of the stadium.

In particular observing the histogram, if we notice the majority of a clear green component (which can be realized in the HSV space, checking if the value of H has a peak in the space [0.2, 0.4]) which helps us to know whether the scene that we are observing must be analysed as a part of the game or not. Doing this way, for instance, we can avoid the operations which look for the presence of the ball when we are observing the terraces of the stadium.

After making the fact clear that the observed video sequence is part of the game, we work in the following way. To understand the images we use 2 descriptors: one which observes the colour and one which observes the movement of the ball and describes its tracks. The colour descriptor works in the following way: it performs a conversion of the image from the used colour space into the HSV space. Down to the particular nature of the images of the whole HSV space, the only component we need to identify the colours is hue, since the shirts of the two teams unlikely have the same chromatic tonality value. A similar reasoning can also be used as far as the possibility is concerned to distinguish the shirts from the field (the authors are not aware of the existence of a team using a green shirt); therefore it will be enough to observe the H component in order to distinguish the players from the field.

Quite another matter is what concerns the possibility to distinguish the black and the white colours. These colours can not be included in the HSV space, if you do not observe the I components of saturation and of value. In particular, what has a very low value (<30) is considered as black, what, on the contrary, has a value higher than 200 is considered as white. Summing up, the operation which performs the colour analysis is the following one: it observes the histogram, sets out a series of quantization thresholds and quantizes the colours which are present inside the image. A correction step follows where the values of white and of black are identified also observing the V component. What results after these analysis is a matrix whose values range from 0 to 255 (the lowest quantization is 2) plus two check values (-1 for the white and -2 for the black). This quantization operation, based on the observation of the histogram, allows us to

simplify very much the threshold within which we shall consider the colours as similar: all the colours which have the same quantized value are put together in the same region.

```
function Colours_region_labeling(RGB_matrix) returns  labeled _Matrix
local Variables : hsv_Matrix
quantized_Matrix
similarity_threshold
hsv_matrix =  hsv2rgb (RGB_matrix)
if (game_action (RGB_matrix)= true)   % game_action is true if the hue histogram has a peak in
% the region between 0,2  0,4 whic is the green playing field
then
quantized_matrix ← huequantization(hsv_matrix)
labeled_Matrix ← labeling (quantized_matrix, similarity_threshold=0 )

else labeled _Matrix ←black_ Matrix   %   black_ Matrix is a costant that identifies a frame that
%shouldn't be analized
```

```
function huequantization(RGB_matrix) returns a hsv_Matrix
local Variables : threshold array of the quantization threshold
hsv_Matrix
hsv_Matrix ← rgb2hsv(RGB_Matrix)
threshold ← histogramthresholding (hsv_Matrix )
hsv_Matrix←quantization(hsv_Matrix,threshold)
```

```
function  histogramthresholding(hsv_Matrix)  returns   thresholds
local Variables : hist
filtered
filtered ← green_filer(hsv_Matrix)        % delete all the regions not surrounded by green
hist  ← Makehist(hsv_Matrix)
thresholds ← hist_analysis(hist)                % try to found the colours of the teams who are
playing
```

Fig. 2. An analized action

Afterwards, we perform 2 filterings, 1 to let green pass, 1 to let white pass. The filtering to let the green pass eliminates all the regions which are not surrounded by the green colour: as a result, everything not being on the field is eliminated. The filtering to let the white pass, on the contrary, has the aim of pointing out the presence of the goal.

The movement of the ball is handled in the following way: due to the continuous movements made by the television camera in order to follow the actions, we are compelled to search inside the images for some fixed reference points in order to try to motocompensate this movement. If we were not reasoning in a space in which the movements are observed to a fixed reference point, we could crushingly misrepresent the game situations. When the goal is not framed it is difficult to choose a fixed point, on the contrary, when inside the image it is possible to point out the presence of this object unique in this extent, we can try to refer all the movements to this fixed point.

We shall work in the following way: we take the image, pull out of it the white components and perform an operation of pulling out the skeleton.

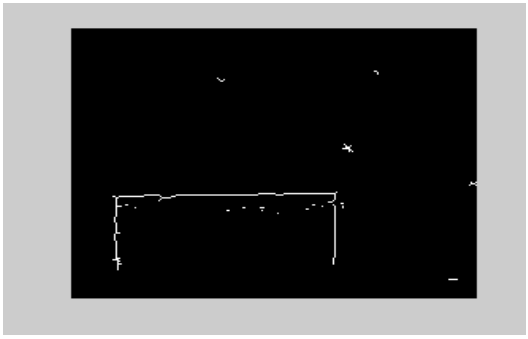


Fig. 3. Scheletonization

Afterwards, we perform an operation of correlation with the cases in storage: this operation will determine its location. Then, we look for the presence of the edge of the goal post and of the cossbar within all the following frames. We choose to plot this object because this element is very seldom stopped up by other objects and, if it is present within the image, it is very easy to point out its presence. After determining the movements of the camera, we subtract these movings from the movement of the ball.

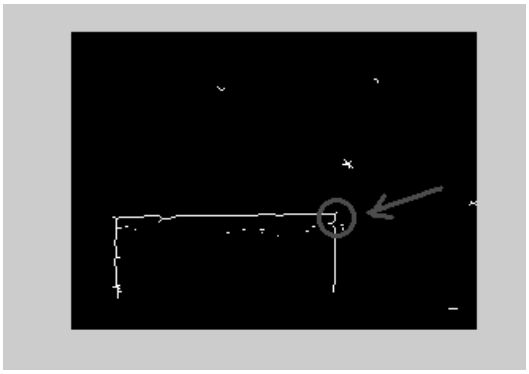


Fig. 4. Goal edge

The structural analysis therefore follows. The first structure which is searched for within the image is the location of the two teams on the field, in such a way as to understand which one of the two teams is attacking or defending. This kind of structure, as you can see, has the characteristic to be able to be detected, on the only base of the analysis of the colour descriptor. Afterwards, a semantic estimator will call back a correlation about the colour of the shirts identifying the teams on the field.

The second structure which is searched for is the player which in that moment is called ball holder. This structure analyzes which player has substantially changed the ball direction last or has made "contact" with the ball. In fact, this structure is obtained analyzing both the movement and the colour descriptor. Knowing this kind of structure, we can understand a lot of things: which sort of action we are observing, if the ball holder was before a player with a white shirt and afterwards he becomes a player with the coloured shirt. A semantic descriptor will be able to detect that the coloured team has interrupted the action of the white team. Let us take into consideration another case. The present ball holder has received a pass from the preceding ball holder and now he gives the ball back with a course having an angle equal and opposite to that of the initial pass: these two players have performed the so called triangle.

```
    new_ball_holder is a case-based function, that looks at the change of  trajectory
and  at the player's position in the near field

    function
change_ball_holder(new_ball_holder,old_ball_holder,old_old_ball_holder,Ball_tra
jectory)  returns nth_event
          local Variables :
       %  interpret the event looking at colours of the old and new ball holder
        new_ball_holder_team   ← team(actual_ball_holder, new_ball_holder)
        %interpret if the ball hoder has changed team
        if    (new_ball_holder_team  =  true)
         nth_event ← intercept % the ball was intercepted
        else
          nth_event          ←kind_of_event          (new_ball_holder,old_ball_holder,
old_old_ball_holder,  Ball_trajectory)
          %    kind_of_event is a function that looks the correlation of the present
event
          %  with past event,
```

```
    Function  Ball_holder(Ball_trajectory,  actual_ball_holder,near_field)  returns  the  ball_holder  at  the
nth Frame  if (Ball_trajectory(frame_N-1) - Ball_trajectory(frame_N)) > threshold
        Then actual_ball_holder ← new_ball_holder(Ball_trajectory, near_field)
      end
```

So, we can go on in defining structures through whose interpretation it is possible to understand some even very complex events of the game. The last structure we take into consideration is the so called course of the ball as to the goal, trough the analysis of this structure we can detect the presence of some shots at the goal or of some goals.

4 Conclusions

In this article we show an architecture which processes multimedial material, subdividing it on the basis of its own semantic contents. We propose an innovatory approach, whose methodology introduces an abstraction level, which has the task to study the relationships among the morphological attributes in a systematic way, before estimating the content. This analysis tries to unify the descriptors information and to gather them into structures that we call over-regions, which represent particular configurations of the objects to be recognized. Sperimental results shows encouraging results.

Bibligraphy

[1] Kandel E. Schwartz H. Jessel T. M. Principles of Neural Science. Mc Grow Hill 2000.

[2] Salambier P., Marquès F. Region-Based Representations of Image and Video Segmentation Tools for Multimedia Services. IEEE Transactions on Circuits and Systems for Video Technology 9(8):1147-1169. 1999.

[3] Rowe N. , Frew B. Automatic Classification of Objects in Captioned Photograhs for Retrieval Intelligent Information Rerieval. Acm press 1998.

[4] Hong Jiang Z. Chien Yong L. Video Parsing, Rerieval and Browsing:An Integrated and Content-Based Solution. Intelligent Information Retrieval. Acm press 1998

[5] Adelson E. H. Chew ,Huttenlocher D. An Efficently Computable Metric for Comparising Polygonal Shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 13(3) :209-216 1985.

[6] Duda, R. , Hart P. Pattern Recognition and Scene Analysis. New York: Wiley 1973.

[7] Aamodt A. and Plaza E. Case-based Reasoning: Foundational issues, methodological variations and system approaches. AICOM 7(1): 39-59,1994

[8] Grimnes M. and Aamodt A. A Two Layer Case Based Reasoning Architecture for Medical Image Understanding 1998.

[9] Tulving E. Organisation of Memory. Academic Press 1977.

[10] Campbell F. and Green D. Optical and Retina Factors Affecting Visual Resolution. J. Physiol 181:576-593, 1965.

[11] Clancey W. J. Heuristic Classification. Artificial Intelligence 27: 289-350, 1985

[12] Kolodner J. L. Maintaining Organization in a Dynamic Long-Term Memory. Cognitive Science 7:281-328, 1983.

[13] Koton P. Using Experience in Learning and Problem Solving. MIT, Laboratory of Computer Science, October 1988.

[14] Micarelli A., Neri A. and Sansonetti G. A Case Based Approach to Image Recognition. Proc. of the Fifth European Workshop on Case Based Reasoning (EWCBR 2K) Trento, Italy 2000.

[15] Micarelli A., Neri A., Panzieri S. and Sansonetti G. A Case Based Approach to Indoor Navigation Using Sonar Maps. Proc. of the Sixth International {IFAC}. Symposium on Robot Control (SYROCO 2000), Vienna, Austria, 2000.

[16] Schank R. C. Dynamic Memory: a Theory of Reminding and Learning in Computers and People. Cambridge University Press, Cambridge, UK. 1982

Designing an Omnidirectional Vision Sensor for Autonomous Navigation

Giovanni Adorni[1], Luca Bolognini[2], Stefano Cagnoni[2], Monica Mordonini[2], Antonio Sgorbissa[12]

[1]D.I.S.T. – Università degli Studi di Genova – Via all'Opera Pia 13, 16145 Genova
[2]D.I.I. - Universita` degli Studi di Parma – Parco Area delle Scienze 181A, 43100 Parma

Abstract. In this paper we focus on the design of an omnidirectional vision sensor which is meant for different navigation and self-localization tasks. More precisely, the paper describes in details the guidelines that we followed for the design of the omnidirectional vision sensor, reporting also some considerations about using the device as a part of a stereo system to locate obstacles in semi-structured environment through a perspective removal algorithm.

1 Introduction

Among the sensors that can be adopted for autonomous navigation and localization in indoor and outdoor environments, vision systems are probably the ones that offer the best compromise between cost, flexibility and richness of information content. The ever-growing computer performances, along with the ever-decreasing cost of video equipment, allows nowadays for the development of robots that heavily rely on vision for navigation and self-localization, even if the fusion of data coming from different sensory sources (ultrasounds, laser range-finders) is still very important in many cases in order to guarantee a safe behavior and the real-time responsiveness of the system.

In particular, as robots are moving to less and less structured environments, the need for sensory systems that provide a global, even if sometimes rough, description of the surrounding environment is increasing. This is particularly important in real-world applications, in which robots are required to operate in a dynamically and quickly changing environment. The above-mentioned considerations have stimulated growing interest in omnidirectional vision systems.

Omnidirectional vision systems are usually based on a catadioptric sensor, consisting of an upwards-oriented camera that acquires the image reflected on a convex mirror hanging above it. Anyway, a number of such sensors are available and, consequently, it is possible to devise a large number of approaches to omnidirectional vision [1].

Besides providing the widest possible field of view, omnidirectional vision systems can obviate the need for active cameras, that require complex control strategies. However, they suffer from two main limitations. The main limitation is that the near field is partially obstructed by the reflection of the camera on the mirror, which makes the central part of the image useless. This limitation is particularly severe,

because resolution decreases radially, and therefore it is maximum in the central part of the image, and also because such a region is also the least distorted by the combined effect of the camera lens and the mirror. A further limitation is that the accumulation of the camera and system distortions makes it quite difficult either to find the resulting distortion law and to compensate it, or to design a mirror profile that can achieve a good trade-off between width of the field of view, image resolution and distortion.

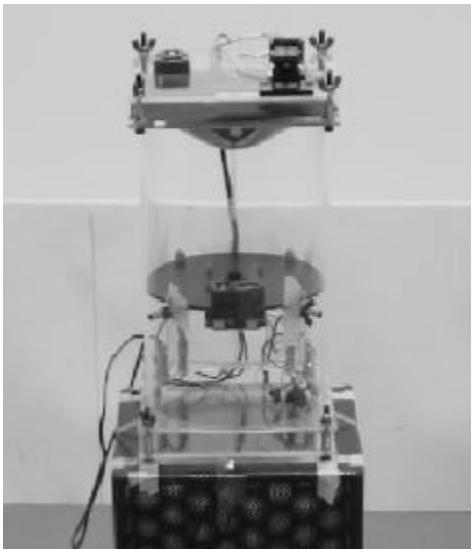


Fig. 1. HOPS vision system.

The paper describes in details the design process that we have followed and the strategies that have been adopted in order to overcome these limitations, and in particular the 1) introduction of discontinuities in the first order derivative of the profile of the mirror in order to shrink areas of the image which contain useless information and 2) the analysis of profiles which are well suited for a sub-set of significant navigation and self-localization tasks.

The software tools that we developed as a support to the design of such a vision system are discussed too (software for generating mirror profiles with different characteristics, simulation of the vision system by means of rendering programs, efficient perspective removal algorithms for a generic mirror whose profile is known etc.). These software tools have been important in the design of HOPS (Hybrid Omnidirectional/Pin-hole Sensor), a vision system prototype that consists of an omnidirectional sensor coupled with a standard CCD camera (see fig. 1). HOPS was designed with the main intent of providing autonomous robots with the most useful features of both omnidirectional and traditional vision systems. Some experimental results of an obstacle detection method that was implemented for use with the hybrid sensor are discussed.

2 Designing the profile of the mirror

One of the most important issues in the design of an omnidirectional vision system is choosing the geometry of the reflecting surface of the mirror. Such a procedure has been completely automated by developing a software program (MirrorForge) which allows to generate different mirror profiles depending on the requirements of the application and on the physical restrictions of the robot for which the system is meant (maximum field of view, encumbrance of the mirror, etc.). Next, in order to test the different profiles, we use a rendering program, which allows to create three-dimensional photo-realistic models of the environment and to generate an image of the scene from the view point of a camera opportunely set in the environment. Even if the program that we use does not allow to model the camera in details (i.e., it is possible to choose between different perspective and orthogonal cameras with varying field of view, but it is not possible to set the focal length of the camera) it allows to define a surface as a reflecting one (i.e. the convex mirror) and to analyze the effects of the image distortion when the camera is pointing upwards to the convex mirror. This allows to model a catadioptric vision system with a sufficient accuracy for our purposes.

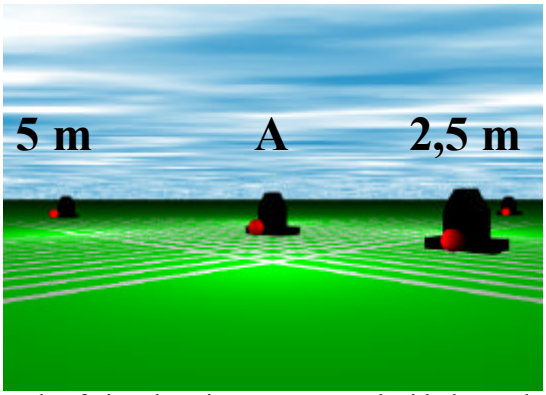


Fig. 2. An example of virtual environment created with the rendering program.

Fig. 2 shows an example of virtual environment. The robot in the middle (A) is the one with the omnidirectional vision system; a grid of intersecting white lines is painted on the floor (the grid will be used to test the algorithms for perspective removal which will be introduced in the following; four robots are displaced in the environment at different distances to test the field of view of the catadioptric system and the effect of distortion. Finally, since one of the application for which the system is meant is the RoboCup competition, in this example each robot has a red ball by its side.

Different mirror profiles have been proposed in literature [2]; in particular the solution proposed by [3] allows to design a mirror which preserves the geometry of a plane perpendicular to the axis of symmetry of the reflecting surface (i.e. isometric mirror): the mirror acts as a computational sensor, capable of providing distortion-free images automatically, thus eliminating the need for further processing (not

considering the distortion caused by the camera lens). Other researchers propose the use of multi-part mirrors [4], composed of a spherical mirror part (which allows better resolution in the proximity of the robot for the precise localization of significant features in the environment) and a conic mirror part which allows to enlarge the field of view of the system (i.e. to perceive objects at a larger distance with a loss in resolution).

Obviously, designing the profile of a mirror always requires to define the requirements of the particular application for which the catadioptric vision system is meant; thus a rule for generating the optimum mirror profile does not exist (i.e. a vision system which is meant only for obstacle avoidance probably does not require the robot to recognize objects at a large distance; the opposite is true for a localization system which relies on the recognition of significant features distributed on the walls or radial straight lines extracted from the surrounding environment [5].

Once the requirements for a particular application have been specified, designing a mirror profile with the desired characteristics needs to solve a differential equation which can be inferred by applying the laws of linear optics. However, for the current version of the MirrorForge program, we chose a simplified approach, since our main purpose is to design a profile according to the following guidelines:

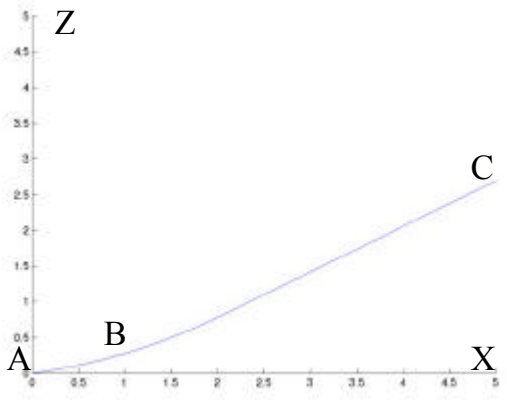


Fig. 3. A mirror profile generated by MirrorForge.

1. we aim at reducing the encumbrance of the mirror by minimizing its radius. This helps also reducing the weight of the mirror; if one considers that the whole vision system is composed by a camera and a plexiglass cylinder with the mirror on top of it, by reducing the weight of the mirror (and, if possible, the height of the cylinder) we increase the mechanical stability of the whole system, stabilizing the acquired images as a consequence. Notice also that, for a given focal length of the camera, if we decrease the radius of the mirror we need also to decrease the distance between the camera and the reflecting surface and, consequently, the height of the plexiglass support.
2. we focus on the design of profiles which allow to reduce the area of the image containing useless information, i.e. the reflection on the mirror of the robot itself.

According to these guidelines, the prototype on which we are currently working is very similar to the one in fig. 3. The mirror is obtained as a revolution surface by revolving the curve around the z-axis, it has a radius of 5 cm and it is about 2.69 cm high. It can be noticed that the curve can be decomposed in two part: the first part (from point A to point B) can be modeled as an arc of circumference, while the second part (from point B to point C) is a straight line. The tangent to the circumference in point B has the same slope of the straight line, preserving the continuity in the first order derivative. When we consider the resulting surface (see fig. 4), it is straightforward to see that it is composed of a conic mirror which is jointed to an "almost spherical" mirror.

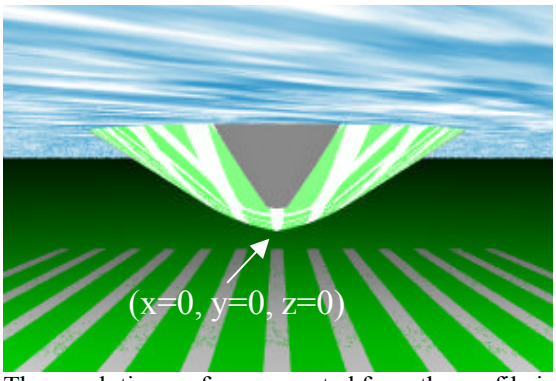


Fig. 4. The revolution surface generated from the profile in fig. 3.

As we stated before, the conic part allows to detect objects which are very far from the robot: this is very useful for localization purposes or to detect objects of interest (other robots, openings, signs on walls, etc.). Obviously, the resolution decreases as we consider objects which are very far from the robot, allowing only an approximate perception of the distance of interesting features. However, if the robot wants to focus its attention on a particular feature, the system provides sufficient information for the robot to move towards it, until it is able to perceive its geometric properties (position and shape) more accurately .

The spherical part allows detecting close objects with a higher resolution, and it is therefore well suited for obstacle detection and avoidance.

As we anticipated before, we aim also at reducing the part of the image containing useless information. Since this area is situated in the center of the image and corresponds to the reflection of the robot itself on the central part of the mirror, we can ask MirrorForge to introduce a discontinuity in the gradient of the revolution surface in correspondence of point (x=0, y=0, z=0). In fig. 4 it can be easily seen that, since the first order derivative of the curve in correspondence of (x=0, z=0) is not equal to zero, the resulting revolution surface will have a discontinuity in (x=0, y=0, z=0). By varying the slope of the tangent in (x=0, z=0) in the curve we obtain different revolution surfaces; i.e. we purposely introduce a distortion in the reflected image with the effect of shrinking the central area which contains useless information. In figg. 5a, 5b, and 5c images obtained with three different surfaces are shown.

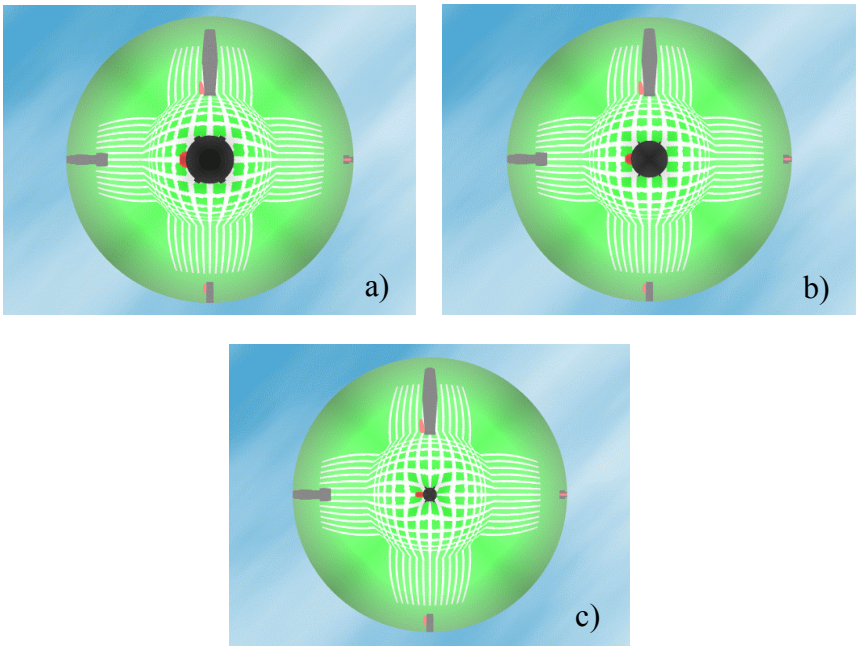


Fig. 5 a) The virtual environment reflected on a mirror with no discontinuities in (x=0, y=0, z=0). b) The virtual environment reflected on a mirror with a discontinuity in (x=0, y=0, z=0). c) The virtual environment reflected on a mirror with a discontinuity in (x=0, y=0, z=0).

3 Real-time obstacle avoidance through stereo-disparity computation

The main purpose of our omnidirectional vision system is to provide information for

1. safe navigation and real-time obstacle avoidance.
2. self-localisation.

In the following, we will focus on point 1, which we investigated both in the RoboCup environment and in the more general scenario of service robotics. Notice that, in the RoboCup domain, objects can be easily identified since they are painted with unique, well identifiable colors. Thus, in this scenario, we have to deal with the two major problems related to variations in lighting and to occlusions, which sometimes make it a quite difficult task to build a color histogram from the images acquired by the vision system. However, in the more general scenario of Service Robotics applications in indoor and outdoor environments, we have to deal with a larger set of problems, since the robot is asked to perform tasks within an environment which is partially or totally unknown and, in most cases, not structured. For this kind of applications, our past experience with HOPS (Hybrid

Omnidirectional Pin-hole vision System) leads us to further explore the approach to obstacle detection through the computation of stereo-disparity between images acquired by an omnidirectional vision system and a standard camera.

The idea of using inverse perspective mapping for obstacle detection was first introduced in [6]. If one applies the inverse perspective mapping transform with respect to the same plane to a pair of stereo images, everything that lies on that plane looks the same in both views, while everything that does not is distorted differently, depending on the geometry of the two cameras through which the stereo pair is acquired. This property is particularly useful for tasks in which a relevant reference plane can be easily found. This is the case for navigation, either for vehicles traveling on roads [7] or for indoor-operating autonomous robots [8][9]. Three steps are usually required to detect obstacles based on stereo vision:

1. Application of the inverse perspective transform to each of the two images;
2. Subtraction of one image from the other one to evidentiate differences;
3. Re-mapping of the regions where obstacle candidates can be found on at least one of the acquired images to label pixels either as obstacle or free space.

Stereo vision is usually obtained by two cameras slightly displaced from each other or through the acquisition of two images from a single sensor that can move to simulate the availability of two cameras displaced as above. The sensors can be traditional cameras or even omni-directional sensors [8].

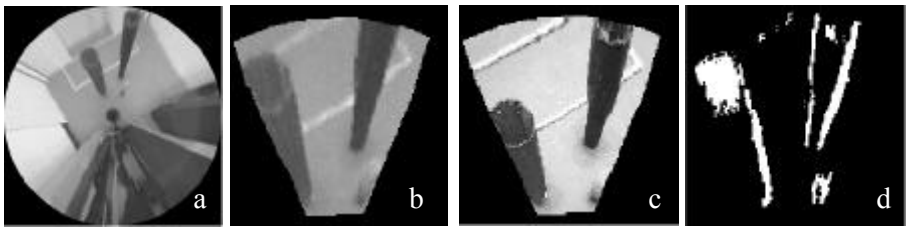


Fig. 6. a) image acquired by the omnidirectional vision system b) the part of the image which is considered for stereo disparity (after perspective removal) c) the image acquired by the traditional camera (after perspective removal) d) difference between b and c (after some filtering).

As we stated before, HOPS extracts three-dimensional information using a hybrid approach based on an omni-directional sensor and a traditional camera. In this case the "stereo disparity" is not constant, since the angle between the optical axis of the camera and the plane that is tangent to the mirror surface is different for each point of the mirror. Furthermore, with the traditional camera positioned on the axis of the omnidirectional sensor, disparity might be close to zero in a few points. Positioning the pin-hole sensor off the omnidirectional sensor axis, as shown in fig. 1, ensures that a different perspective be obtained from the two sensors for all points. Such a geometry makes navigation tasks less manageable, as the two sensors do not share any axis of their respective reference frames. However, the tilt angle of the upper camera is different from any of the view angles under which the scene reflected by the

mirror is seen by the camera of the omnidirectional sensor, at least in the image regions whose resolution allows for a reliable processing. This demonstrates that, even in case in which the optical axis of the upper camera intersects the axis of the omnidirectional sensor, such a geometry allows for stereo vision-based obstacle detection to be performed.

Because of these considerations, it becomes very important to opportunely displace the traditional camera and the omni-directional vision system in order to obtain a significant stereo disparity between the two images. Once again, our virtual environment appears to test different configuration of the system, since we need to find a good compromise between reducing the encumbrance of the system and increasing the stereo-disparity. Similarly to what we did with the omnidirectional vision system, we simulate the perceptions of the standard camera by declaring a new camera object in the virtual environment, and we execute the operations required for obstacle detection (IPT, difference between the images, filtering) on the virtual images in order to test different configurations and finally to obtain cues for the design of the system. Fig. 7 shows the image in fig. 5c after perspective removal.

4 The inverse perspective transform

The problem dealt with by the inverse perspective transform (IPT) consists of computing a correspondence function C that maps the pixels (with coordinates belonging to image I, onto points of a new image P that shows a bird's view of a reference plane. From the view obtained by applying C(I) information about the relative positions of objects (e.g., distances between objects) that lie on the plane can be trivially extracted.

If all parameters related to the geometry of the acquisition systems and to the distortions introduced by the camera were known, the derivation of C could be straightforward [6][7][9]. However, this is not always the case, most often because of the lack of a model of the camera distortion. In most cases, assuming an ellipsoidal model for the camera lens allows for an easy empirical derivation of the related parameters.

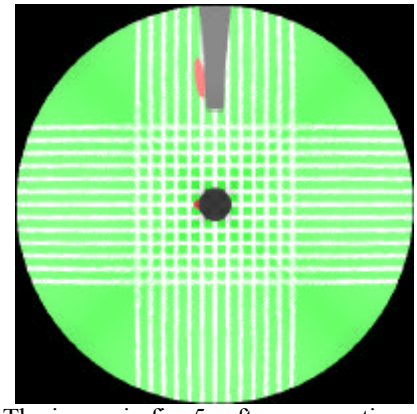


Fig. 7. The image in fig. 5c after perspective removal.

When computing the IPT for the catadioptric omnidirectional sensor, further problems arise. It should be noticed that, on the one side, the problem is complicated by the non-planar profile of the mirror, on the other side the symmetry of the device geometry is such that it is enough to compute the restriction of C(I) along a radius of the mirror projection on the image plane to be able to compute the whole function. However, in catadioptric mirrors, that are usually made of polished steel, several manufacturing flaws are likely to be found, as regards both the actual shape of the mirror profile and the smoothness of the mirror surface.

The mirror used in the HOPS prototype is no exception, and is characterized by a few slight ring-like "bumps". These flaws derive from the manufacturing process and, despite being almost unperceivable at the touch, they produce significant local distortions in the reflected image. In addition to such flaws, that are luckily regular enough to be included in the radial model of the mirror profile, a few other minor "random" flaws can be found scattered on the mirror surface. This irregularities require that the IPT be derived empirically, or at least that the ideal model be corrected in the flawed points. Therefore, C(I) was derived both from the ideal profile model and from empirical corrections to such a model.

However, the new mirrors that we are currently developing will be made of a plastic material through a different manufacturing process. This, on the one side, will help us reducing the weight of the whole vision system, which therefore will become suitable to be used even on smaller and lighter robots than the ones we are currently using. On the other side, the new manufacturing process will help us to obtain a mirror surface which is closer to the mathematical model that has been previously fed to the rendering program. As a consequence, the same set of points which have been used to define the mirror profile in the virtual world can be fed into the IPT algorithms in order to remove perspective in the images acquired by the real robot. IPT will be computed from the ideal model with a very good approximation, thus minimizing the need for subsequent empirical calibrations.

The whole process for designing the HOPS vision system is summarized in fig. 8.

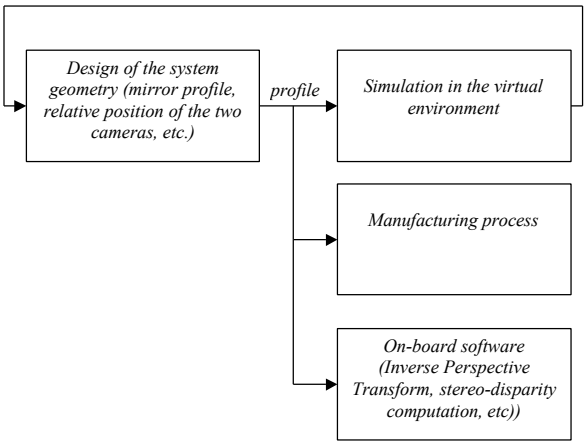


Fig. 8. Designing HOPS.

5 Conclusions

In this paper we discussed the guidelines that we have followed for the design of a catadioptric vision system which is meant for different indoor and outdoor applications: in particular we focused on the navigation problem, which we face by computing the stereo-disparity between images acquired by two sensors (an omnidirectional vision system and a standard camera) after having computed their projection on a reference plane through a Inverse Perspective Transform algorithm. While describing the design process (design of the geometry of the system, simulation in a virtual environment, manufacturing process, design of the on-board software), we outlined some characteristics that we consider important: low encumbrance and weight, high fidelity in the real mirror surface in replicating the ideal model, etc. A future work, we aim at dealing with the self-localization problem; moreover, we aim at investigating the use of two omnidirectional vision systems for stereo-disparity computation. This would allow us to detect obstacles all around the robot body (currently only a limited area in front of the robot is considered, because of the limited field of view of the standard camera), thus implementing a sort of "visual range-finder" which could improve the navigation and real-time obstacle avoidance skills of the robot.

References

1. S. K. Nayar, Omnidirectional vision, Robotics Research. 8th International Symposium, pp. 195-202, 1998.
2. Y. Yagi, S. Kawato, S. Tsuji, Real-time omni-directional image sensor (copis) for vision-guided navigation, IEEE Transactions on Robotics and Automation 10 (1), pp. 11-22, 1994.
3. R. A. Hicks, R. Bajcsy, Reflective surfaces as computational sensors, IEEE Workshop on Perception for Mobile Agents – Proc. Of the 1999 IEEE Conference on Computer Vision and Pattern recognition (CVPR99), IEEE Computer Press, Piscataway, NJ, 1999.
4. A. Bonarini, The body, the mind or the eye, first?, in: M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup 99-Robot Soccer World Cup III, Springer Verlag, Berlin, pp.210-219, 2000.
5. L. Delahoche, C. Pegard, B. Marhic, P. Vasseur, A navigation system based on an omni-directional vision sensor., in: Proc. IEEE Int. Conf. On Intelligent Robots and Systems, pp. 718-724, 1997.
6. H. A. Mallot and H. H. Ulthoff and J. J. Little and S. Bohrer, Inverse perspective mapping simplifies optical flow computation and obstacle detection, Biological Cybernetics, Vo 64, pp. 177-185, 1991.
7. K. Onoguchi and N. Takeda and M. Watanabe, Planar Projection Stereopsis method for road extraction, IEICE Trans. Inf. & Syst., Vo. E81-D, No. 9, pp. 1006-1018, 1998.
8. C. Drocourt and L. Delahoche and C. Egard and C. Cauchois, Localization method based on omnidirectional stereoscopic vision and dead-reckoning, Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 960-965, 1999.
9. G. Adorni and S. Cagnoni and M. Mordonini, An efficient perspective effect removal technique for scene interpretation, Proc. Asian Conf. on Computer Vision, pp. 601-605, 2000.

A Bayesian Approach for Mobile Robot Navigation in Time-Variable Environments

A. Chella [1,2], R. Sorbello[3] and S. Vitabile[2]

[1] Dipartimento di Ingegneria Automatica ed Informatica
University of Palermo, Italy
e-mail: chella@unipa.it
[2] CE.R.E. – Centro di Studio sulle Reti di Elaboratori
Consiglio Nazionale delle Ricerche, Palermo, Italy
e-mail: vitabile@cere.pa.cnr.it
[3] Dipartimento di Ingegneria Elettrica
University of Palermo, Italy
rosario@csai.unipa.it

Abstract. We present an architecture for mobile robot navigation based on Bayesian Networks. The architecture allows a robot to plan the correct path inside an environment with dynamic obstacles. Interactions between the robot and the environment are based on a powerful vision agent. The results of simulations, showing the effectiveness of the approach, are described. Finally, the integration of the proposed approach with the Georgia Institute of Technology Mission Lab Software is outlined.

1 Introduction

A goal directed robot, navigating in a time-variable, unstructured environment, needs to compute the optimal path between its current position and the goal position. The mobile robot has to incorporate deliberative reasoning capabilities in order to process abstract plans and include decision-making mechanisms for critical situation. However, it also needs reactive skills to deal with different unexpected events, like dynamic objects along its path. In recent years many approaches have been proposed for integrating planning and reactive control [1], [2], [3], [4], [5], [6]; however these approaches, generally, do not consider time-variable environments.
In this paper, we analyze the navigation problem of a robot, equipped with the only vision sensor in a time-variable environment.

The navigation problem can be viewed as the problem to decide among many competing action hypotheses. Bayesian Networks are a powerful tool to resolve this problem [7], [8], [9], [10]. In this respect, the visual information, coming from the external environment, is incorporated as evidence that guides the robot decision-making process among the different action hypotheses [11], [12], [13], [14].

Therefore, a Vision Agent, designed to accurately deal with the real features of its sensor, allows robot and time-variable environments interaction updating robot

evidence during the navigation task. In order to validate this approach, we developed a simulation software that allows us to define the environment topology with fixed and dynamic obstacles and the features of the Vision Agent.

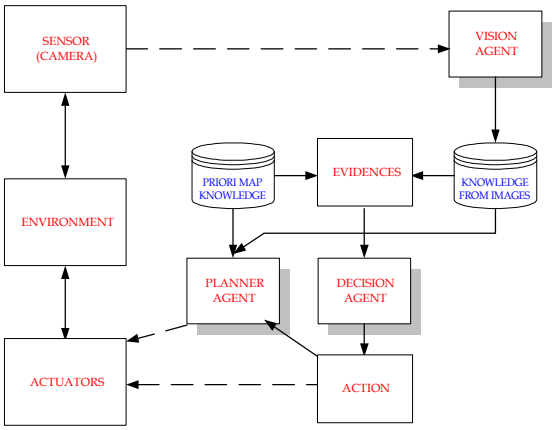


Fig. 1. The Block Diagram of the Agent.

2 Overview of the Robot System

The developed system is divided in some independent blocks interacting each other (see Figure 1).
The block, indicated as “knowledge from images”, is the main and essential source of information for the robot, acquired through the camera. The single snapshot image is analyzed from the Vision Agent that picks the salient aspects of the observed scenes updating the knowledge base of the robot. The block, indicated as “priori map knowledge”, is the *a priori* knowledge about the environment, e.g. position and extension of the present fixed obstacles.
These two types of knowledge, one acquired and one *a priori*, constitute the database that gives the evidence information as input for the decision-making process.
A decision agent, based on the developed Bayesian network, chooses the action that is executed in a determined time-slice on the basis of the probabilities generated by the network. Therefore, the robot, in each time-slot, will always execute the highest probability action.

2.1 The Vision Agent

Once the images have been acquired, the Vision Agent elaborates the information that can be useful to the robot. The main functionalities of this Vision Agent are the following:

- dynamic obstacle detection;
- obstacle dimensions and shape recognition;
- speed and direction obstacle determination, with the respect to the relative robot position;
- memorization of the obstacle information by a suitable tag.

This information is affected by errors as we will show later. The Vision Agent is fully described in details in [6].

2.2 The Planner Agent

The Planner Agent, on the basis of the *a priori* knowledge and of the knowledge acquired through the Vision Agent, allows the robot to navigate inside time-variable environments. The agent, other than avoid obstacle to reach the goal, is also able to replan a new path every time a failure is observed. In this way the robot deals with time-variable environments. In particular, in the current implementation, this agent is based on the well-known A* algorithm [16].

2.3 The Decision Agent

The environment, where the autonomous robot navigates, is time-variable and therefore it is necessary to turn out the more convenient action for the present analyzed map configuration.

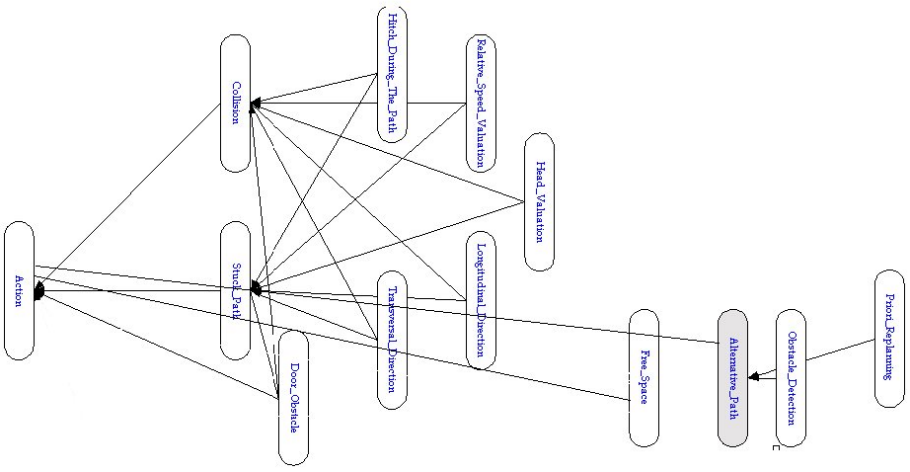


Fig. 2. The adopted Bayesian Network structure.

The Bayesian Network decisions are closed to the following robot actions: *Waiting, Replanning, Going Around* and *Escape.* The robot can choose between these actions to manage all the situations that can happen during its navigation:

- **Waiting**: the robot decides this action until the detected obstacle along its path, goes away from its cone of visibility;
- **Replanning**: the robot plans an alternative path from its current position;
- **Going Around**: the robot tries to go around the identified obstacle by constructing a path that passes sideways the obstacle (see Figure 3 for an example);
- **Escape**: the robot uses this maneuver to avoid a collision with the obstacle.

Figure 2 shows the adopted Bayesian Network with its nodes and relations. For instance the node "Alternative_Path", exhibited in the Figure 2, is closed to the presence of one or more paths for the robot other than the planned ones.

The features of the node and the related Condition Probability Table (CPT) are described in Table 1 and Table 2, respectively. As shown in Table 2, the environment observation (Obstacle_Detection) has an higher impact rather than the *a priori* knowledge (Priori_Replanning). Similar tables have been defined for the other nodes of the network.

Table 1. The features of the node "Alternative_Path".

Parents	Priori_Replanning, Obstacle_Detection
Descendents	Action
Values	TRUE, FALSE

Table 2. The Condition Probability Table (CPT) of the node "Alternative_Path".

Priori_Replanning	Obstacle_Detection	True	False
True	True	0.9	0.1
True	False	0.6	0.4
False	True	0.8	0.2
False	False	0.1	0.9

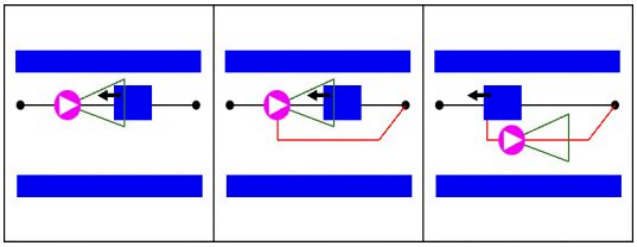


Fig. 3. Movement of the going around action with a dynamic obstacle.

3 Simulated Tasks

The visual apparatus of the robot is a virtual camera placed on the top of its head. The camera owns one main feature: the Cone of Visibility. Ideally, it is modeled by a cone, whose vertex lies on the camera while the base is coplanar to the plan. It is delimited by two parameters: the camera lens aperture and the depth of field.

We introduce also a coefficient error that models all the imperfections of a real camera. This coefficient is a linear function of the distance. The error coefficient influences the input evidences of the Bayesian network. When the robot meets a dynamic obstacle, the knowledge base is suitably updated with the new information. The algorithm used to find the dynamic obstacle is based on the idea to detect the first vertex appeared inside the Cone of Visibility. This information, containing the position of obstacle, the geometry and the type of movement, is converted in the input evidence pattern of the Bayesian Network.

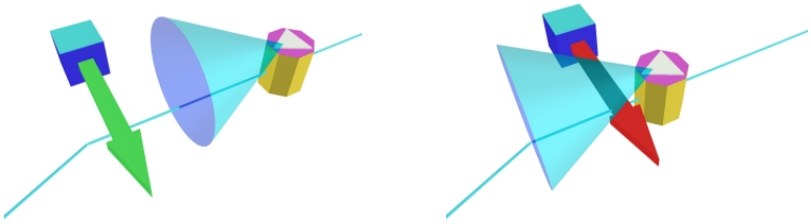


Fig. 4. On the left the robot with a not dangerous obstacle, on the right the robot with a dangerous obstacle.

The simulation environment is a rectangular room of variable dimensions. Inside the room we can place, beyond the starting-point and the goal-point, different types of obstacles with very precise characteristics. These obstacles are divided in:

Fixed Obstacles: cylindrical or cubic obstacles, free to be scalable in the two axis coplanar to the base of the map-room. They are motionless, and the robot knows their position and dimension at the beginning of the simulation, because they constitute the *a priori* knowledge for the robot.
Dynamic Obstacles: they are structurally identical to the previous ones, but they have the peculiarity to move with constant speed inside the environment following a constant user defined path. The robot does not have *a priori* knowledge about these obstacles.
Cyclical Dynamic Obstacles: the movement of these obstacles is along a closed path. Also for this type of obstacle the robot does not have *a priori* knowledge.
Door Dynamic Obstacle: this obstacle is a particular case of Cyclical Dynamic Obstacle: it is a cyclical obstacle whose path only consists of two waypoints. The doors have a traslational speed and the robot will not have *a priori* knowledge of this obstacle.

Figure 4 shows the robot with two dynamic obstacles: in the first case there is no collision and the robot goes on along its path, in the second one there can be a collision so the robot stops its run.

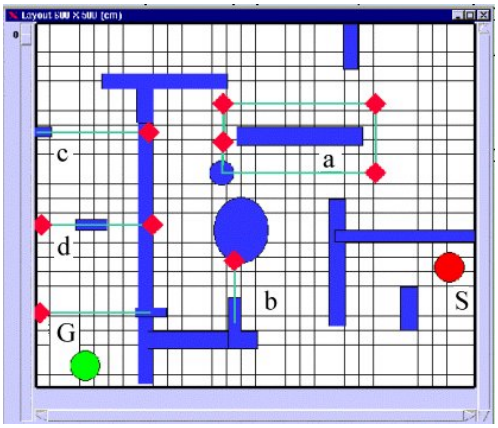


Fig. 5. A snapshot of simulated the environment: **S** and **G** are, respectively, the source and goal point, **a** is one cyclical obstacle, **b**, **c** and **d** are door.

4 Experimental Results

We have been carried out many simulation tests in order to verify the goodness and robustness of the proposed solution for robot navigation in a time-variable environment.

Table 3. The nodes activated by the perceptions of the Vision Agent.

Relative_Speed_Valuation (M,m,0)	9,3; 9,3; 81,4
Longitudinal_Direction	18,6
Transversal_Direction	18,6
Head_Valuation (V,L)	50
Hitch_During_the_Path	18,6
Door_Obstacle	18,6
Obstacle_Detection	**81,74**
Priori_Replanning	74,3
Free_Space	18,6

Our current simulator is based on the libraries of the Netica tool for the Bayesian Network simulation [17] and on the MesaGl libraries for environments design [18].

Figure 5 shows a snapshot of a map constituted from different types of obstacles. The robot, starting from the S source point must reach the G goal point; on its path there are different kinds of dynamic obstacles.

The first robot path is shown in Figure 6. The robot follows the planned path until it detects the obstacle **b** that represents a failure in its knowledge. Table 3 shows the nodes without parents and activated by the perceptions of the Vision Agent. Each node has its own probability in order to take into consideration the errors of the vision system. Table 4 shows the related output values of the Bayesian Network with a probability worked out from the network and connected with each of the possible actions. The consequently chosen action is the **Replanning** action that gives the new path as shown in Figure 6.

Table 4. The output values of the Bayesian Network when the robot detects the first obstacle.

Waiting	14,1
Replanning	**65,2**
Going Around	11,4
Escape	9,3

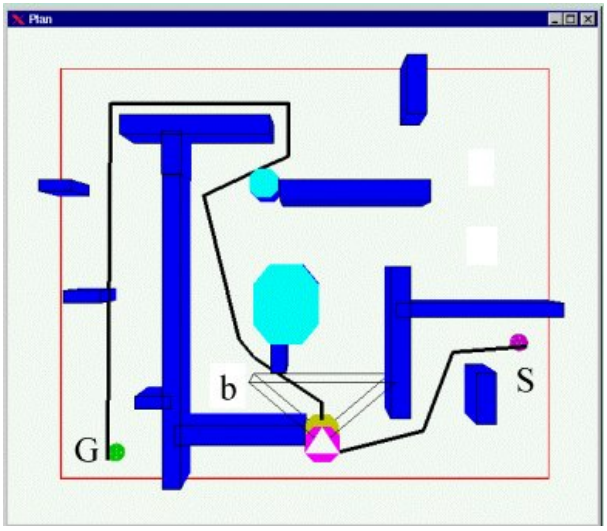


Fig. 6. The robot detects the first obstacle along its path.

Table 5 and Table 6 show the input nodes and the related output values of the Bayesian Network when the robot detects the door (obstacle **c**) as shown in Figure 7. The consequently chosen action is the **Waiting** action.

In both cases the robot chooses the most coherence action inspired to the human behavior in similar situations.

Table 5. The nodes activated by the perceptions of the Vision Agent.

Relative_Speed_Valuation (M,m,0)	9,7; 80,6; 9,7
Longitudinal_Direction	19,1
Transversal_Direction	80,6
Head_Valuation (V,L)	80,6
Hitch_During_the_Path	19,1
Door_Obstacle	**80,6**
Obstacle_Detection	19,4
Priori_Replanning	74,3
Free_Space	80,6

Table 6. The output values of the Bayesian Network when the robot detects the door obstacle.

Waiting	**46,4**
Replanning	-21
Going Around	-22,7
Escape	-9,69

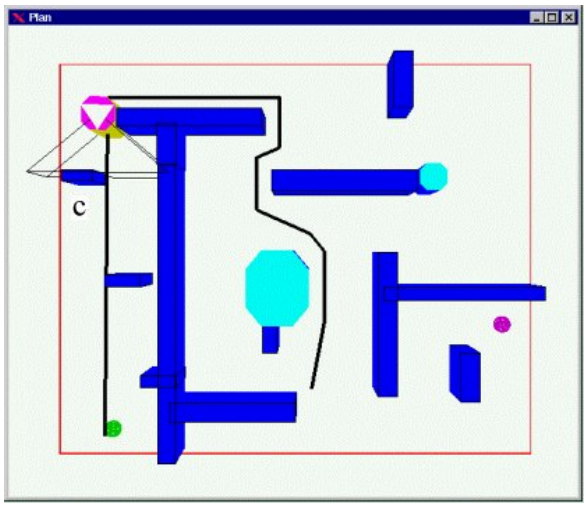


Fig. 7. The robot detects the door along its path.

5 Current Works

Our efforts are currently aimed to integrate the proposed software tool with the Mission Lab software, a software tool for multi-agent robots developed at the Georgia Institute of Technology [20]. We are developing a complete test-bed software for a new

paradigm of the dynamic coordination of multi robots agents in time-variable environments.
We have built a framework for testing a new architecture based on the Metaphor of Italian Politic. The purpose of this Metaphor is to maximize the behaviors of robots colonies on the assumption that the robots colonies autonomously are able to get organized with a dynamic structure. The robot are also able to assign an executive political government to 3 robots of the colonies. The roles robots are divided into 4 category: PCM (President of Minister Council), MC (Communication Minister), MD (War Minister) and NORMAL (the others robots).

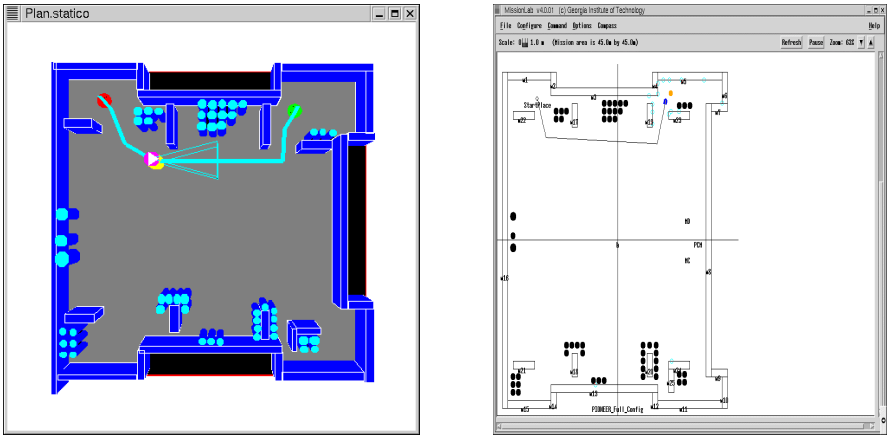


Fig. 8. The testing Bomb_Map on the CSAI Lab environment (left) and on the Mission Lab environment (right).

The use of this Metaphor has to allow the robots to autonomously get organized so that they can complete dangerous mission. During the testing phase, we have used a scenario with a number between 10 and 20 robots. This robots team has to get organized and defuse the bombs that they will find inside the environment in a smaller time and smaller number of robots lost and destroyed.
Figure 8 shows the movements of the robot inside an unknown environment. The testing Bomb_Map can be viewed in both the new CSAI Lab environment and in the Mission Lab environment.

6. Conclusions

The use of a Bayesian Network has allowed us to manage time-variable environments. The robot is able to easily navigate with dynamic obstacles and flowing doors. Our current work aims to integrate the CSAI Lab software with the Mission Lab software.

Moreover, we are testing the proposed approach in real environments, through the use of a RWI B21 robot equipped with a Vision Agent based on color analysis for object detection [15] and on the Harris algorithm for detection of the angle of the obstacles [19].

References

1. R.C.Arkin, Motor schema based navigation for a mobile robot, in: Proceeding of the IEEE International Conference on Robotics and Automation, 1987, pp.264-271.
2. R.C.Arkin, Motor schema mobile robot navigation, Int. J. Robot. Res. 8(4) (1989) 92-112.
3. R.C. Arkin, Integrating behavioral, perceptual and world knowledge in reactive navigation, Robot Autonomous Syst. 6 (1990) 105-112.
4. R.C. Arkin Behavior-Based Robotics, MIT Press, Cambridge, MA, 1998.
5. D. Lyons, A.Hendrix, Planning for reactive robot behavior, in: Proceeding of the IEEE International Conference on Robotics and Automation, Nice, France, 1992, pp.2675-2680.
6. S. Vitabile, F. Torterolo, F. Sorbello (1998). "*A Neural Multi-Agent Architecture for an Autonomous Mobile Robot Control*", Proc. of X Italian Workshop on Neural Nets, Vietri sul Mare (SA) – Italy, pp. 383-388, Springer-Verlag.
7. Zadeh,L. (1973). Outiline of a new approach to the analysis of complex systems and decision processes. IEEE Trans. Syst. Man Cybernet. 3 (I), 28-34.
8. Levitt,T. T Binford, G. Ettinger and P. Gelband (1989). Probability-based control for computer vision. In: Image Understanding Workshop, Palo Alto, CA, 23-26 May 1989. Morgan Kaufman, San Mateo, CA, 355-369.
9. Pearl J. (1988). Probabilistic Reasoning in Intelligent Systems: Networks for Plausible Inference. Morgan Kaufman, San Mateo, CA.
10. Rimey, R.D. and C.M. Brown (1994) Control of selective perception using Bayes nets and decision theory. Internat. J. Computer Vision 12, 173-207.
11. S. Jun, K. G. Shin, "A Probabilistic Approach to Collision-Free Robot Path Panning", Proc. IEEE Int. Conf. Robotics and Automation, pp. 220-224, Philadelphia, USA, 1988.
12. H. Asoh, S. Akaho, Y. Motomura, "Statistical Inference and Inference in AI", Journal of Japanese Society for Artificial Intelligence, Vol. 12, No. 2, pp. 196 -203, 1997
13. Y. Motomura, "Learning Situated Prior Probability for Optimum Bayesian Discrimination", JNNS '97.
14. Y. Motomura, "Learning of Neural Bayesian Network for Office-Conversant Robot", JSAI Technical Report SIG-CII-9601-04, pp. 22-25, 1996.
15. S. Vitabile, G. Pilato, F. Pullara, F. Sorbello (1999). "*A Navigation system for Vision-Guided Robots*", Proc. of 10° International Conference on Image Analysis and Processing, Venice – Italy, pp. 566-571, IEEE Computer Society Press.
16. Hart, P.E., Nilsoon, N.J. Raphael, B. (1968). A Formal Basis for the heuristic determination of minimum cost path. IEEE Transactions on System Scienze and Cybernetics, SSC-4(2):100-107.
17. http://www.norsys.com
18. http://www.opengl.org
19. C. Harris and M. Stephens. A combined corner and edge detector. In Proc. Alvey Conf., pages 189--192, 1987.
20. http://www.cc.gatech.edu/aimosaic/robot-lab/research/multi-agent.html

Comprensione del Movimento e Apprendimento Percettivo

Vito Roberto
Machine Vision Lab
Universita' di Udine

Introduzione

Fasi di apprendimento percettivo sono utili nei sistemi artificiali per sintetizzare un modello dell'ambiente esterno, che sia robusto ed efficace per raggiungere gli scopi desiderati. Il presente contributo intende approfondire alcune di queste fasi relativamente a problemi di analisi e comprensione del movimento.
Questi ultimi, per la complessita' dei compiti da assolvere – ricordiamo l'inseguimento di oggetti, (tracking), la stabilizzazione, la ricostruzione di forme 3D da sequenze – sono particolarmente interessanti in domini applicativi avanzati come la sorveglianza, la guida automatizzata, il telerilevamento.

Affronteremo in particolare il problema del tracking che verra' scomposto in una fase di focalizzazione dell'attenzione ed una di inseguimento; metteremo in risalto l'esigenza di rendere robusta la fase attenzionale eliminando caratteristiche stimate poco affidabili: sara' proprio un'ulteriore fase di apprendimento non supervisionato a fornire adeguate risposte alle esigenze sollevate.

Focalizzazione dell'attenzione

L'analisi del moto deve fornire stime attendibili dei parametri di posizione e movimento di un oggetto in moto relativo rispetto ad un sensore, a partire da sequenze d'immagini. Il primo problema e' individuare zone 'rilevanti' (features) dei singoli fotogrammi (frames) ed estrarle in modo automatico per mezzo di 'operatori di interesse' (interest operators), che abbiano requisiti di selettivita', invarianza alle rotazioni e stabilita' al rumore. Numerose soluzioni sono state proposte in letteratura gia' per il caso d'immagini statiche (Moravec, Harris&Stephens, Noble); di una di queste (Harris&Stephens) si forniranno risultati applicativi, che e' opportuno discutere in modo critico. Se infatti le zone di interesse apparenti vengono riconosciute, queste tuttavia registrano come rilevanti anche le zone di occlusione o di rumore presenti nella scena; un altro serio problema e' la sensibilita' alla soglia di attenzione che va fissata a-priori.
Si tratta in realta' di due aspetti di un problema solo: quello della robustezza dell'analisi, che e' un problema piu' vasto di quello della stabilita' al rumore, investendo anche altre ambiguita' di interpretazione dovute ad occlusioni ed ombreggiature, ma anche all'inadeguatezza dei modelli del moto e della fotometria che vengono adottati.

La fase di tracking

Un'idea interessante per ottenere maggiore robustezza e' sfruttare la ridondanza dei dati offerta dalla stessa sequenza da analizzare; viene cosi' rimossa una limitazione intrinseca al metodo degli operatori di interesse, e cioe' il fatto di operare sui frame come se fossero indipendenti, senza sfruttare la correlazione che il moto fornisce loro. In altri termini, una fase di tracking effettuata in modo accurato puo' aggiungere robustezza alle stime gia' realizzate.
L'approccio ben sperimentato che si puo' seguire per il tracking e' quello della 'Sum of Squared Differences' (SSD) noto anche dalla stereovisione. La disparita' – che nel caso in esame porta alla stima del campo di moto – viene stimata calcolando il minimo residuo quadratico tra il livello di grigio medio all'interno di una finestra del frame corrente, e quello predetto da un modello su una finestra del frame successivo. Allo scopo, e per le ragioni sopra accennate, occorre adoperare un modello del moto piu' elaborato, come ad es, il modello affine.
Si giunge in tal modo a 'seguire' (tracking) ciascuna delle features da un frame al successivo attraverso l'intera sequenza; ma resta l'esigenza di selezionare le features 'buone' da quelle spurie, che permangono anche dopo che e' stata effettuata la fase di inseguimento.

Il ruolo dell'apprendimento

Siamo ora in condizione di discriminare le due classi di features attraverso una fase di apprendimento non supervisionato. Infatti possiamo raccogliere una statistica dei residui calcolati nella fase di tracking, cioe' delle stime di quanto una feature in un frame si discosta dalla stessa su un frame successivo. Il problema diviene quello di una stima statistica robusta: dall'analisi della distribuzione osservata di ciascun residuo lunga la sequenza, siamo in grado di individuare le feature spurie in quanto causano 'outliers' rispetto alla distribuzione attesa dei residui. Questa si puo' dimostrare essere una gaussiana; in letteratura esiste un metodo detto X-84 (Hampel, et al.), il quale fornisce uno stimatore della distribuzione dei residui ed una regola che consente di individuare e scartare gli outliers.
Verranno presentati i risultati dell'applicazione del metodo X-84 all'analisi del movimento per la sequenza 'Artichoke' del noto database d'immagini del CMU-VASC. I risultati dimostrano l'efficacia del metodo sia nel preservare le feature effettivamente buone, sia nello scartare quelle effettivamente spurie.
Ora che il tracking viene effettuato solo su features attendibili, e' senz'altro piu' attendibile ogni successiva fase di comprensione del movimento che su di esse si basa.
A tale scopo viene illustrata una fase di stabilizzazione: il moto del sensore viene 'compensato' avendone stimato correttamente i parametri; all'immagine viene di conseguenza applicata una trasformazione (warping) che ne ripristina i valori di livello di grigio al frame precedente, ad eccezione, ovviamente, di quelli che al successivo risultino usciti dal campo visuale.

Conclusioni

I problemi che rendono ambiguo un processo visivo, sebbene noti, sono a volte tali da rendere un sistema percettivo inefficace o addirittura inapplicabile a contesti reali. Nel presente contributo si e' posto l'accento sulle esigenze di robustezza nel caso di sistemi per l'analisi e la comprensione del movimento.
Risultati promettenti si ottengono introducendo una fase di apprendimento all'interno delle usuali fasi di analisi del moto. Nel caso da noi presentato l'apprendimento consiste nel raccogliere la distribuzione dei residui nel tempo per poi applicare un metodo di statistica robusta (X-84), con lo scopo finale di

discriminare caratteristiche affidabili da altre ritenute spurie. Ne risulta una mappa delle caratteristiche affidabili da poter usare in successive fasi della comprensione del movimento.
Crediamo che il caso in esame non sia isolato ed abbia analoghi in contesti e per applicazioni diverse. Ad esempio, l'uso sempre piu' ampio che si fa delle reti neurali nella visione preliminare (early vision) realizza gli stessi scopi tramite approcci diversi solo in apparenza. Le reti, infatti, possono essere riguardate come macchine attenzionali robuste, in grado di estrarre in modo efficiente poche caratteristiche a partire dai dati osservati, avendo effettuato una fase di apprendimento che parta dai dati e permetta loro di autoconfigurarsi.
Riteniamo dunque che sia lecito generalizzare e ipotizzare che fasi di apprendimento sono realmente fondamentali nei sistemi percettivi operanti in universi aperti, ambigui e dinamici.