Software Agents for Computer Vision: a Preliminary Discussion

F. Arcelli, M. De Santo, S. Di Salvo
{arcelli, desanto, sdisalvo}@dia.unisa.it

DIIIE - Dipartimento di Ingegneria dell’Informazione e Ingegneria Elettrica
Università di Salerno - 84084 Fisciano (Salerno), Italy

Abstract

In this paper, we start a preliminary discussion about the development of software agents developed for Computer Vision. Such agents must be heterogeneous and adaptive in nature to satisfy increasing requests for integration-cooperation capability exhibited by current Computer Vision applications (e.g., biomedical imaging and image retrieval on the Web). We give a brief introduction to the overall architecture design and we focus on lowest level agents, called W-bots (Working Software Robots). We define the nature of such agents and describe their ways of interaction, and their adaptive capabilities. Finally, we present an example of the proposed architecture utilizing VDM++ to specify agents as active objects.

1. Introduction

Software architectures of current Computer Vision (CV) applications, like biomedical imaging or image retrieval on the Web, should rely upon a twofold level of performance:
- the ability to collaborate with the user, helping him to clearly specify his goal and supporting him in the most of implementation choices;
- the effectiveness in accomplishing CV tasks.

In order to meet the first requirement, a change in the way people interact with such applications is needed. Direct manipulation where the user initiates and monitors all tasks explicitly is to be abandoned. Instead, the user should be involved in a cooperative process in which human and computer agents, both initiate communications, monitor events, perform tasks.

Considering the second requirement, a suitable approach is the realization of each CV task by means of a set of cooperating modules available and executed in a distributed environment.

As a matter of fact, a Vision Task (VT), realized by means of the simple connection of independent modules suffers of a degree of inflexibility, dramatically decreasing the overall performance of its execution. On the contrary, a model of integration/cooperation among modules allows relaxing single modules constraints thus achieving best result [1].

With integration/cooperation aim, several burdening requirements are, nevertheless, to be taken into account; among them: the heterogeneity of the modules, the dynamic behavior of open distributed environment, the heterogeneity of data representations related to different CV tasks, the limited communication bandwidth available in current systems.

Although cooperative models have been well studied in Distributed Artificial Intelligence [2], this approach has been poorly investigated in Computer Vision community. Previous works addressing C.V. techniques and algorithms integration problem have followed two main streams:
- traditional approaches, typical of the Computer Vision community;
- pure DAI approaches, typical of research in AI field.

The first approach is based on a rigid integration of defined sets of modules and conduces to the realization of systems devoted to one specific task. Obviously, according to this approach, little changes imply the redefinition of the whole system.

The second approach is based on agents [2, 3]. In [4], for instance, all the strategies, tactics, hypothesis, data etc. related to a C.V. task coexist in a single agent. In such agents also the cooperation among different techniques (that can be implemented by means of routine libraries) takes place. In [5], instead, the agents are communicating rather than cooperating ones; in fact they present a priori fixed communication patterns among them, making the whole society a static one. In such agents either data, or inferential and algorithmic
computation, are uniformly represented by means of facts and rules and metarules.

We have found unsatisfactory both these approaches for the design of an integrated software system which engages and helps end users in accomplishing CV goals; that is why we introduce a completely new one. In such a framework, our approach is a hybrid one, in the sense that it provides for architecture suitable to manage the dynamic requests of current CV systems without forgetting the algorithmic characteristic of most part of the CV techniques. The approach separates interface, domain and strategic information from operative ones, thus introducing modularity in computations and making the system suitable for the elaboration of different concurrent computations.

The rationale behind our research can be summarized in the following assumptions.

From a general point of view such software system is conceived as based on agent software architecture [3, 6]. This architecture is composed of different and heterogeneous layers of cooperating adapting agents and exploits an asynchronous, non-coupled modality of communication. The choice of a heterogeneous layered architecture allows modeling the «actions» an integrated system has to perform, by means of agents with their decisional, operative and cooperative capabilities. The possible cooperation patterns among agents are not fixed in advance, but are related to the problem at hand.

Therefore, our approach answers to the needs of techniques integration/cooperation, and of flexibility in the organization of the resolution of given problems, that are found in Distributed Integrated Computer Vision Systems. This approach generalizes previous research, but in the meanwhile introduces completely novel features. In particular, asynchronous and uncoupled communication [7, 8] is supported by means of shared object spaces so as to achieve dynamic characteristics and reduced communication bandwidth. In the present paper, we get a particular focus on the lowest level agents of this architecture, called W-Bots; these agents execute autonomously but can interact and cooperate to reach a global objective.

The paper is organized as follows. In Section 2, we give a brief description of the general architecture and of the different kinds of agents involved in it. In Section 3 we present the features of the lower level agents, the different types of messages they can exchange and their capabilities. Different execution modalities: standard, dynamic and adaptation are discussed. In Section 4 an example of low level vision task is presented and discussed. Finally we conclude and trace new research directions.

2. The general architecture

From an abstract point of view, three different layers of cooperating adapting agents compose our software architecture. The agents of different layers have different roles and present different capabilities. To this heterogeneity in functionality's, naturally corresponds an heterogeneity in the models chosen to represent different kinds of agents; such models, in fact, will show different degrees of sophistication and complexity. Figure 1 shows the discussed architecture.

Each agent of the top layer, called interface software robot (Interbot), behaves as a personal assistant that cooperates with the end-user on the definition of a CV goal. It is responsible of the management of man/machine interface, of the global comprehension and satisfaction of the goal, of the concurrent behavior of the entire application. It uses abstract reasoning: situation evaluation, planning in tasks terms, problem solving, and so on. According to the well-known metaphor of the assistant [9,10], Interbots become gradually more effective as they learn the user’s interests (application field for instance), habits and preferences.

The agents of the middle and the bottom layers, called processing software robot (Probot), provide a distributed management of the CV tasks.

To distinguish between structural and operational aspects, as desirable in Distributed Systems [11], we use two families of probots:

1. domain processing agents (D-bots), subdivided in levels corresponding to CV tasks, each responsible to manage and control the concurrent behavior of a part of an application (a CV task), that is the task organization. To this aim, they are responsible of planning activity, problem solving, and individuation of resource and of possible cooperation among different modules, operating-net initialization, and so on.

2. working processing agents (W-bots), subdivided in levels and sublevels; these last corresponding to different abstraction levels of the techniques used in the resolution of a CV task. W-Bots are used to implement separate CV algorithms and techniques that cooperate actively on the resolution of a problem. Each W-Bot is responsible of the concurrent behavior of a single module, in the sense that it can take advantage of other agents (subagents), working concurrently with it.

Basic communication actions are performed in non-asynchronous, decoupled way, through shared spaces [8]; the composition of such basic actions can give rise to different modalities of cooperation. All kind of information exchanged in the system, either message or
perceptual data, has the form of an object, instance of a particular class; this provides support for user defined data and for existent types extensions. The use of Object Oriented philosophy allows the definition of constraints on the internal structure of the objects, that will be preserved, independently of where, in the network, they will be processed. Further, modules can be implemented with generic operations, while objects will provide for the actual code for the operations to execute, through the mechanism of late binding.

Thanks to the communication features and to the introduction of special objects, called interaction protocols, the agents can dynamically accept to cooperate with other agents, not fixed in advance, according to a given protocol.

In such a context each computation involves one interface agent, one or many domain processing agents and one or many groups of working processing agents, each group being controlled by one domain processing agent. These groups constitute the operating net, whose configuration, initially individuated by the D-Bots, by means of their «initial task configuration», can dynamically change during the computation.

Hence, each problem, proposed to the system, is resolved by means of interacting working processing agents. In our architecture, interface and domain processing agents can be assimiled, respectively to assistant [9] and cognitive agents [3], already largely treated in Artificial Intelligence literature. Working processing agents, instead, are particular software agents [6], tailored to the needs of Computer Vision; they are critical, because they are the agents who really carry out the tasks and because of their distribution in the environment. The architecture is highly flexible and allows the dynamic development of applications, even in open distributed environment.

3. Working processing agents

Working agents represent the modules that actually operate on images, apply techniques and execute algorithms.

Peculiar features of W-Bots, are the following:

- heterogeneity -W-Bot could be designed and developed by different groups, could implement different programming techniques, could be persistent or not;
- distribution - in the sense that W-Bots can be located on different machines and can be executed in a distributed Environment, as a consequence, their availability changes with the operation of resources addiction and removal and the ability to reach them is subjected to the successful behavior of the underlying network. Distribution introduces delay in communication and the necessity to manage time-out mechanisms to avoid deadlocks;

semi-autonomy - in the sense that each W-Bot is subjected to a Domain-bots or to another W-Bot, responsible also of its initialization. In particular, the initialization step of a W-Bot, from a D-Bot, interests the social knowledge of the W-Bot (that is information about other W-Bots with whom interactions can take place) and possibly operative information (such as parameters values or operative modes). Each W-bot, nevertheless, can infer «knowledge» locally. In this view, the achievement of a local solution constrained by a-priori knowledge (resource knowledge and goal assessment related knowledge) available at the D-Bot level is allowed;

- structuration - each W-Bot can use or generate lower sublevel W-Bots to execute well defined parts of its work, giving rise to a hierarchy of concurrently elaborating W-Bots. We distinguished between Intralevel hierarchy and Interlevel hierarchy. The first is a hierarchy extended in a single elaboration level; that is all the W-Bots in father-son relation belong to the same level. The second is a hierarchy extended from an elaboration level to another; two W-Bots in father-son relationship belonging to different levels have to be under the control of two different D-Bots. Such D-Bots would be in communication and will transmit information to the W-Bots they control, with the aim of permitting direct communication between them;

- unique skill and duty - a W-Bot does not need to show strategic or planning capabilities; in fact the problem it resolves and the policy of resolution it applies are fixed in advance. This means that each W-Bot is characterized by a precise skill. Each W-Bot, however, exhibits decision capabilities, suited to choose the operating mode, operating capabilities, relative to the consequent elaboration and cooperation capabilities, to choose and implement cooperation schemes.

In this direction, in fact, the ability of direct cooperation among W-bots, will avoid the use of a unique common coordinator for communication, synchronization, exploitation of cooperation patterns and resolution of cooperation conflicts, thus decreasing the possibility of having bottlenecks.

The characteristic of cooperation patterns of being dynamically dictated by the actual W-Bot involved in a computation, for the implementation of a certain skill, and not fixed a priori by the skill itself, is the key for dynamic integration and usage of new techniques, without their revision ahead of system construction.

In our system, such semi-autonomous agents are realized as distributed active object; they are instances of subtypes of a general type (W-Bot) that defines all the properties and functionality's common to all the W-Bots.
3.1 W-Bot communication

Before exploring the W-Bots capabilities, let’s introduce the following basic communication aspects. As a member of a society, each W-Bot can communicate with other agents. The communication is realized sending and receiving different kinds of structured messages, to and from other agents, namely:

- **Regulation Messages** (RM), representing local knowledge and control from D-Bots.
- **Social Messages** (SM), representing information about the acquaintances of a W-Bot for communication, cooperation and migration purposes.
- **Data Messages** (DM), representing the means through which an agent propagates the data it knows.
- **Expectation Messages** (EM), representing commands and “models” from higher level W-Bots.
- **Observation Messages** (OM), representing evidences from lower level/sublevel agents sent as answers to observation messages.
- **Cooperation messages** (CM) representing requests of advice or results from peer W-Bots.

3.2 W-Bot capabilities

As above remarked each W-Bot shows operative, cooperative and decisional capabilities. As regards to operative capabilities, we found useful to introduce a more comprehensive characterization of the Working Agents. Therefore, from the operational point of view, the working agent j at level i, W\textsubscript{i,j} is an object characterized by:

- a skill applying process $T^{i,j}$ that, according to the kind of technique implemented, can be transformational or inferential;
- a set of parameters $p_k^{i,j}$ of $T^{i,j}$ (random variables), representing a way to adapt the procedure to the current data;
- input data $I^{m,i,j}$ to the skill applying process $T^{i,j}$ - sent out from other agents in the same hierarchy (distributed problem resolution) or in other hierarchies (cooperation) (See fig.2). In CV field, in dependence to the level and sublevel of the W-Bots, input data can be of different kinds: maps, 2D-descriptions (e.g., segments), symbolic descriptions, etc.;
- output data $D^{n,i,j}$ produced by $T^{i,j}$ - representing observations to send to higher levels in the hierarchy or cooperative data to send to other hierarchies or simply filtered data directed toward lower levels;
- the state of the process;
- a cost functions $Q_{ij}$, to be minimized in order to obtain $D_{ij}$ of high quality;
- other features - such as knowledge on other agents, expectations and so on.

For what concerns cooperative capabilities, we observe that the basic messages, together with the basic acts of communication, can be composed in complex schemes, giving rise to the so-called Interaction Protocols [12]. In this way, each W-Bot will be able to dynamically exploit different coordination/cooperation paradigms. In fact an Interaction Protocol can be thought as a framework defining the kind of dialogue that must be realized in a cooperation. So a pair of agents cooperates exchanging and synchronizing on the same Interaction Protocol object.

For what concerns decisional capabilities, a W-Bot can decide to go in execution according to one among the following operative modalities: standard, dynamic, adaptation standard, adaptation dynamic.

This functionality can be by-passed if the commitment, coming from D-Bot regulation messages or other W-Bot expectation messages, provide explicit indications about the operative mode to apply. Otherwise, a W-Bot uses some criteria to select the operative mode.

In relation, to the choice between standard and dynamic mode, a W-Bot can decide according to performance demands established by the commitments. For example, in the case of strong demands the W-Bot can decide to elaborate in dynamic mode. The dynamic mode will prevent the generation of subagents trying to involve acquaintances in the W-Bot elaboration.

Ortogonally with the choice between dynamic and standard mode, the W-Bot can decide if execute according to the adaptation modality or not. For example if the quality of the output, would not satisfy the commitments, an adaptation modality could be decided.

Given this ortogonal, we have:

1. In standard mode a W-Bot uses agents which belong to its, possibly multilevel, hierarchy to solve subproblems. In general, a W-Bot can exploit agents outside its hierarchy, just asking cooperative information from them.
2. In dynamic mode a W-Bot can take advantage of other W-Bots, outside of its hierarchy, making them subagents of itself.
3. In adaptation modality, the W-Bot aims to parameter calibration. This calibration, so that the W-Bot has a high fitness in performing its CV work, is achieved by energy manipulation [13].

3.2.1 Standard Mode. In standard mode a W-Bot uses its generated agents to solve subproblems. As a consequence of interactions inside the hierarchy, the W-Bot receives: output of the lower sublevel; that is intermediate or final observations coming from subagents elaboration as answer to expectation acts; output of the higher level; that is data filtering toward the bottom of the hierarchy.

For what concerns agents outside the hierarchy of the W-Bot in question, in general they can contribute to the problem resolution of the W-Bot just sending cooperative information. However, if suitably initialized by a D-Bot, they can actively participate to the problem resolution of the W-Bot acting as W-Bot subagents. This behavior remains unknown to the W-Bot. The condition for this is that such agents have to be of the same types of the W-Bot subagents.

3.2.2 Dynamic Mode. A W-Bot can modify its own implementation on condition that its skill applying process is appropriately structured and it possesses appropriate social knowledge.

In the dynamic mode, the W-Bot has to prevent the generation of some of its subagents and to involve other pre-existing agents in its elaboration. Such agents would elaborate the same tasks of the prevented subagents of the W-Bot, possibly showing difference in the method, implementation or performance. They, after a suited contact, migrate in the system, changing elaboration context and becoming subagents of the W-Bot in question. Naturally, these mobile subagents have to be acquaintances of the W-Bot. Finally, Social Messages will be sent toward the migrating W-Bot, to guide its functioning inside the new configuration and in the
opposite direction to introduce the features of the new subagent in the new configuration.

3.2.3 Adaptation Modality. A W-Bot is conceived as having the ability to modify its own elaboration parameters with the aim of self-adaptation to the current computation. The operational flow to provide an optimal set of parameters is called adaptation modality. It is beyond the scope of the present paper to discuss in detail the adaptation mode. In the following, we give a brief sketch of the driving ideas.

From a general point of view, the adaptation mode uses energy manipulation concept [13] to achieve optimal quality. Energy represents a value that indicates the global degree of quality of the data calculated by the W-Bot; it is valued as a weighted composition of quality features. Quality features, at their time, are computed applying a set of function to a datum; they are used to value how good calculated data are. Because data calculated by a W-Bot are functions of its parameters, the energy of a W-Bot is a function of the parameters.

Therefore, each W-Bot, for which the possibility to operate in adaptive mode is foreseen, has to be provided with knowledge concerning the parameters to regulate skill applying process phase and with specific criteria to estimate data quality.

The strategy used for local adjustment of parameters can be based on the computation of a probabilistic measure, risen from a global optimization criterion, called fitness of the W-bot or the strategy can be based on the computation of a fuzzy measure.

The parameters, which are involved to gain a global optimization criterion, are the following:

\( \gamma : \) the W-Bot regulation knowledge for parameters optimization; \( \rho : \) Regulation; \( \xi : \) Data; \( \nu : \) Expectation; \( \Psi : \) Observation; \( \omega : \) Cooperation.

All these parameters, except the first one, correspond to different kinds of messages transmitted among the W-Bots. The W-Bot achieves a global optimization criterion by interaction with its local environment i.e. with other connected W-Bots. It is possible to assign to the parameters involved, a particular weight according to the different types of message which are transmitted (regulation, data, expectation, observation, cooperation). Hence each single message contributes to gain the global optimization criterion in a different way, in relation to its weight. This weight related to each parameter can be obtained through a degree of membership of its value to a fuzzy set.

In fig. 3 we present the characteristic cycle followed by the W-Bot during adaptation mode.

3.2.4 Fuzzy Approach. Fuzzy features can be defined via fuzzy sets [16], where a fuzzy set can be defined as the following mapping on a domain \( X \):

\[ \chi_A : X \rightarrow [0,1] \]

such that to any element \( x \) belonging to the domain \( X \), we can associate a membership function \( \chi_A(x) \) in the fuzzy set \( A \). The elements 0 and 1 define the membership or not membership of each element to the set \( A \).

In this way a fuzzy set can be written also as

\[ A = \frac{x_1}{\psi_1} + \frac{x_2}{\psi_2} + ... + \frac{x_n}{\psi_n} = \sum x_i/\psi_i \]

In our context, consider for example the parameter Expectation, where its domain \( X \) is given by \{ \( x_1, x_2, x_3, x_4, x_5, x_6 \) \} corresponding to six different expectation messages. Then we may define the fuzzy set low-expectation over that domain as \( x_1/1, x_2/1, x_3/0.5, x_4/0.4, x_5/0, x_6/0 \) which will represent one of our fuzzy feature.

In the same way we can define some fuzzy set with the corresponding membership function for the other parameters. Hence we can propose an optimization criterion starting by considering the generated fuzzy set and applying some given operators (T-norms). The choice of the operators and the involvement of all the parameters are determined according to a given heuristic.

4. An Example

Before presenting an example relevant for the Computer Vision field, with the aim of showing the application of our architecture, we anticipate that the example is specified by means of the formal object oriented concurrent specification language VDM++.

VDM++ is the language we have chosen also for the specification of the kernel of integration/interaction of the agents. The kernel specification has resulted in a schema of active/passive object types respectively modeling the W-Bots and the communication among them. So, each agent or object in an application, is instance of a type derived from a type in the kernel. The chosen example shows an application of the previous ideas; in fact it describes a possible computation configuration in our architecture realized to satisfy an edge detection goal on an image captured from a camera.

We assume that the entire computation is initialized by a single D-Bot.

The initial task configuration realized by such a D-Bot involves 3 W-Bots; namely a Camera W-Bot (C), a Pre-processing W-Bot (PRE), and an Edge Detector W-Bot (ED). The descriptions of the module we use follow the lines drawn by Murino et al. in [13].

The Camera W-Bot simulates an optical sensor device; it is dedicated to the updating caption of the image from the outside world. The Pre-processing W-Bot is devoted to the elimination of noise from the captured image. The Edge Detector W-Bot gives a description of the image in terms of straight lines.
Beside the interaction with the D-Bot, these W-Bots communicate accordingly to the following patterns: ED communicates with PRE; PRE communicates with C; C communicates with the outside world.

In the following we give a brief description of each W-Bot and we underline the reasoning paths followed by the W-Bots to attain a global optimization of the results by means of local adaptation. The Camera W-Bot allows a real camera to be connected to the system; it takes care of the regulation of the caption parameters to improve the quality of the captured image.

The following parameters are kept into account: Focus (F), Aperture (A), Black Level (BL), Electronic Gain (EG).

The quality features computed by the W-Bot to state the quality of the captured image are:

Tenengrad (TN), that is based on the Gradient G(x,y) on each point of the image and giving a singular integral value that indicates gradient information on the whole image; Flatness (F), that is based on the gradient too; it gives information about high contrast and low contrast areas; Entropy (E) that is a measure of the uniformity of the image histogram; Lowest/Highest Saturation (LS, HS), that counts the number of low/high saturated pixels. It depends on the choice of the lowest and highest gray levels (LG, HG); Modes Changes (MC), that is based on the approximation of the image histogram by means of Gaussian curves; in particular it counts the number of curves whose amplitude is less than 10% of the maximum amplitude.

All these measures, excepted for the first are to be minimized to obtain an image of high quality. The Pre-processing W-Bot (PRE) is the agent devoted to the noise elimination. Based on the Perona-Malik filter, the algorithm that it implements is iterative. It updates the intensity of each pixel using the intensities of the 4 neighbors and a diffusion coefficient. The regulated parameter is the number NI of iterations. The quality features PRE computed are the same of the Camera W-Bots; so the total quality of its result is the same, also if the used weights are different. For what concerns the Edge detector W-Bot (ED) its main task is to split the image received from the preprocessing W-Bot in different parts. These parts are utilized to feed lower level Edge Extractor W-Bots (EE), and to compose edge information coming from such EE W-Bots with the objective of producing the best straight-line scene. ED W-Bot has no parameters to adjust. The EE W-Bots, in our case, are generated by ED or involved by the D-Bot in the computation. So, their number is variable; each of them takes care of the edge extraction on a different part of the image; the edges extracted by the Edge Extractors W-Bots are stored in a Hough Space. Edge Extractor W-Bots are based on Canny filter. They try to detect edges on an image, using gradient information and applying a hysteresis mechanism by means of two thresholds S1 and S2. These thresholds constitute the parameters regulated by the module to improve their results. The quality features computed by EE W-Bots are: the number of chained Edges (EN), the number of Edge points (EPN); the number of long edges (LEN); the number of connected edges (CEN).
At the beginning of the computation, even if ED, PRE and C are started, only C is really active; PRE is waiting for a captured image and ED is waiting for an improved image resulting from PRE. After the first image is obtained using default parameter settings, PRE can begin its elaboration on such an image. In the...
meanwhile C goes on, calibrating its parameters and capturing different images of the scene, that will be taken on count by PRE in a deferred time. PRE works on the images knowing the quality degree of each image. After performing its task on such images, it makes them available to ED. At the same time PRE computes the quality of its result. In the case of not satisfaction it regulates its own parameters and sends commitments to C, driving and restricting C possible parameters settings. After an image is processed by PRE, it is ready for the Edge Detection. Therefore, it is split in different areas that will be taken into account by different EEs. The results of them are then composed, possibly with results of other edge images of the same scene, resulting in an ever-improved reconstruction. If the total quality (of the image) or the partial quality (of a given subimagine) computed by the ED is low, commitments are sent to PRE (for instance focus commitments) to improve the quality of the whole or of a part of the image.

This way to proceed allows a user to dispose of an early reconstruction in a short time and of ever better reconstructions at later times. The global outcome, resulting from the W-Bots local adaptation, converges toward a good solution, even if possibly it is not the "best". Further the W-Bots are well employed, in the sense that they don't loose time, and the non-coupled asynchronous mechanism allows each W-Bot to compute at its own time, completely disregarding from the other W-Bots times. Figure 4 reports a schema of the W-bots involved in the discussed example together with VDM++ implementation.

5 Conclusions

Our approach to the development of a software architecture for Computer Vision generalizes previous research [11, 12] defining an architecture of intelligent agents allowing dynamic adaptive cooperation among heterogeneous software entities. We introduced a layered architecture involving Interface Agents, Domain Agents and Working Agents, which shows no limits to the amount of cooperation without to impose a priori connections among agents. In this paper we focused our discussion on Working Agents outlining their internal structure and their way to dynamically and adaptively cooperate for the solution of given tasks. The use of VDM++ to specify W-Bots and the kernel of integration/cooperation among them, allowed us to rigorously test our ideas about W-Bots modes of interaction and adaptation. We are currently developing a more complete kernel and testing it with real CV applications, in particular from adaptive capability point of view.

References