A Purposive Computer Vision System: a Multi-Agent Approach

Reinaldo A. C. Bianchi[1] Anna H. R. C. Rillo[2]

[1]*Division of Automation and Artificial Intelligence - Laboratory of Integrated Systems*
[2]*Department of Computer Engineering and Digital Systems*

Politechnical School - University of São Paulo - Brazil.
Av. Prof. Luciano Gualberto, travessa 3, 158.
05508-900 São Paulo, SP, Brazil.
E-mail: rbianchi@lsi.usp.br, arillo@pcs.usp.br

ABSTRACT

This paper describes a purposive computer vision system for visually guided tasks and a Multi-Agent architecture used to model it. In this architecture, the vision system's purpose is decomposed into a set of behaviors, which are translated into specific tasks. Purpose, behaviors and tasks, as well as the relationship among them, are modeled using a Multi-Agent approach: purpose is modeled by a society of autonomous agents, which communicate through a common language, each one responsible for a specific visually guided behavior; tasks are represented by basic agents, organized in a hierarchical structure. A description of a system that is being implemented as a testbed for the architecture is given, with some details on the implementation of the agents and its communication methods. Finally, a brief discussion about the use of the basic agents is done and research directions are proposed.

1. Introduction

The purposive paradigm for computer vision [1], [2] is a field of computer vision that believes that vision must be considered within the set of tasks an agent must accomplish. It tries to find in the purpose of the agent the constrains to solve the ill posed problem of vision. The purposive paradigm researchers believe that general purpose vision will arise from the organization of several different dedicated solutions to different visual tasks. So, the major goal of purposive vision research is how to organize solutions and define primitive tasks, focusing on architectures for integration of visual systems.

Seeking an answer to this problem, we look at Distributed AI and Multi-Agent Systems theory as a field where a formalism to describe behaviors and their relationship can be found. Distributed Artificial Intelligence (DAI) is defined by Bond and Grasser [3] as "the field of AI concerned with concurrency in AI computations, at many levels." Multi-Agent Systems (MAS), one sub-area in DAI, are concerned with the coordination of the behaviors of several autonomous intelligent agents to solve one or more goals.

In this context, this paper presents a purposive computer vision system for visually guided tasks based on Distributed AI theory. This system is being implemented using a distributed architecture where the vision system's purpose is decomposed into a set of behaviors, which are translated into specific tasks. Purpose, behaviors and tasks, as well as the relationship among them, are modeled using a Multi-Agent approach: purpose is modeled by a society of autonomous agents, which communicate using a common language, each one responsible for a specific visually guided behavior; tasks are represented by basic agents, organized in a hierarchical structure with autonomous agents on the top.

This Architecture has several distinguishing features, among them the possibility of cooperation between low and high processes, giving flexibility, modularity, and autonomy to the system, allowing the addition and deletion of agents responsible for behaviors, and facilitating the integration between system and environment.

Finally, this paper also presents some details on the implementation of the basic agents and the communication methods used by all agents.

2. Related works

Various works inspire the proposed architecture, among which we mention: Brooks' [4] *Subsumption Architecture*, where a system is composed of layers with

specific tasks, and where each layer interacts directly with the world through perception and action; Elfes [5] Distributed Control Architecture, where a system is divided into processing levels and where independent processes communicate through a blackboard; and Boissier and Demazeau works on the ASIC Multi-Agents Control Architecture [6] and the MAVI system [7], which integrate different visual modules using MAS theory.

Rivlin et al. [8] and Aloimonos [1] also inspired this work. Aloimonos [1] works raise a criticism to Brooks architecture, observing the lack of cooperation between low and high level processes. This criticism stimulated a large portion of the efforts aimed at the solution of these problems, resulting in the architecture presented below.

3. The multi-agent architecture

In this section a brief description of the architecture used in our system is made. In this architecture, a system is modeled with Autonomous Agents (AAs), each one responsible for a specific behavior, organized in a society with rules, where they have to communicate in order to achieve their goals. A single agent model is defined for all autonomous agents. Each AA communicates with other AAs in the society and also with Basic Agents (BAs), which are responsible for specific tasks. This system decomposition in agents with their behaviors is similar to the task decomposition proposed by Brooks [4].

In this society, each autonomous agent is connected to all others, through a decentralized communication network. A scheme of the this connection between the autonomous agents is presented in figure 1.

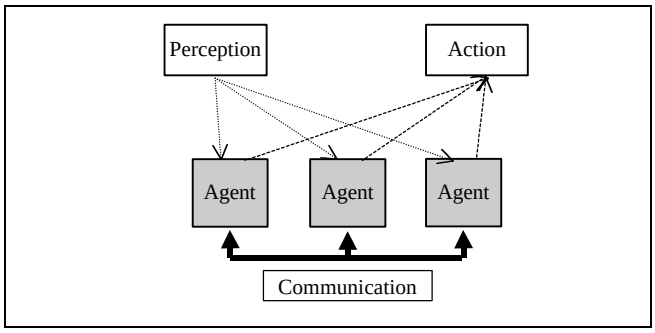


Figure 1 - Architecture Scheme.

The autonomous agents can be classified as the focus-agents described in Boissier and Demaseau [6], which are defined as the ones created in a vertical splitting of the system in its tasks, because each agent has a defined behavior. They are also related to the layers of the Brooks Subsumption Architecture, because they perceive and act directly in the world, as shown in figure 1.

The agents in the architecture are organized in a society with rules of behaviors and with an authority structure. This structure enables the agents to decide how the resources of the system are allocated, for example, in order to decide which agent should have the control of one resource at a certain moment. The manner this decision is made is dependent of the rules defined for the society in a specific implementation, and can be, for example, the result of a competition between agents.

A resource is defined as a part of the system that is shared by the agents, and that can be controlled by only one agent at a time. For example, a robotic manipulator in an assembly cell is a resource, and the drive system of a mobile robot is another. On the other hand, a fixed camera (and its data acquisition hardware and software) is not a resource, as all agents can have the images it captures at the same time. Yet, this camera could be a resource if acting in an active vision system, where each agent could compete to control the process of data acquisition.

The authority structure of a system is deeply related to the dependence and precedence of the autonomous agents behaviors in the society, and its definition is based on the study of the linearization of an activity plan, having the behavior of the autonomous agents as operators, the resources an autonomous agent needs as the pre-conditions of the operator, and the accomplishment of the system intentions as it goals.

The definition of how the autonomous agents can allocate system resources are left to the implementation of the system itself.

It is worth noticing that is the authority structure that makes this architecture related to the Subsumption architecture, allowing agents to suppress each other by taking away resources. Finally, the authority structure has a logic nature, while the communication network has a physical nature.

An empirical division of the system is proposed, based on Brooks' Task Decomposition [4] and Rivlin [8] to map intentions into behaviors, and based on Boissier and Demazeau [6], [7] to split behaviors in tasks.

The definition of an autonomous agent is one of the main features of the proposed architecture and is viewed below.

3.1. Autonomous Agents (AAs)

In this architecture the autonomous agents (AA) are modeled based on DAI-MAS theory, and its definition is strongly influenced by the work of Boissier and Demazeau [7].

An AA is defined as:

<Agent> ::= <rules> <other agents> <state of the world> <communication language> <basic agents> <decision capabilities>

Where:

<*rules*> are the behavior rules and the authority relation between the AAs in the society;

<*other agents*> are the other AAs in the society, and the topological information of the society;

<*state of the world*> is the minimal symbolic representation of the world that an agent need;

<*communication language*> is the communication language used by the AAs;

<*basic agents*> are the basic agents each AA has connection to;

<*decision capabilities*> are the capabilities an AA has to be able to decide which agent has the control of a resource in a certain moment.

This definition is used to model all AAs in the system, not mattering their behavior. Another essential feature related to agents in a DAI-MAS environment is their capability to communicate. That makes the definition of this communication language a very important topic, which is detailed in the next item.

3.2. AA's Communication Language

A Communication Language for the AAs is defined, based on the one described in Boissier and Demazeau [7], as:

<*interaction*> ::= <*nature*><*type*><*content*>

Where:

<*nature*> is the nature of the communication, which can be **decision** or **control**;

<*type*> is the type of communication;

<*content*> is the content of the message.

Messages having a **decision** nature are used to decide which agent will have the control of one system resource in a defined moment. It can be of four types:

request - used by an AA to request the control of a resource to another AA. The message indicates which agent is asking the control and the resource it wants;

agreed - used by an AA to agree with a requisition. This message is used to acknowledge the request, and does not transfer the control of the resource;

free - used to inform that an AA is willing to release a resource that is not needed anymore;

inform - used to transfer the control of a resource from an to another one. The message indicates which agent is transferring the control, if it is taking or giving the control, to which agent it is giving the control (if this is the case), about which resource is the transaction and the state of the resource.

When an AA sends a ***free*** message, meaning that it does not need the control of the resource any longer, several other AAs can request this control, which is then transferred to the one that has higher authority.

Control nature messages are used to add and delete AAs in the society. It has three types:

addNewAgent - used to add an agent to the society. Its contents are the name of the new agent and two lists, one with the name of the agents with higher authority and one with the name of the agents with less authority;

deleteAgent - used to delete an agent from the society;

acknowledge - used by the other agents in the society to acknowledge the insertion or deletion of an agent.

Table 3 summarizes the communication language defined for the autonomous agents.

<type>	**<content>**
addNewAgent	<name of the new agent> <above which agents> <below which agents>
deleteAgent	<name of the agent>
acknowledge	<name of the agent>
request	<name of the agent> <resource>
agreed	<name of the agent> <resource>
free	<name of the agent> <resource>
inform	<name of the agent> <give|take> <from|to which agent> <resource> [state]

Table 3 - Elements of the communication language of the autonomous agents.

3.3. Basic Agents (BAs)

In the proposed architecture, AAs communicate with a set of basic agents (BAs), which are responsible for specific tasks. These BAs are connected through a communication network and organized in a hierarchical structure with AAs on the top.

Part of the knowledge the AAs need to accomplish their behaviors, as visual and manipulation tasks, is located at these BAs.

In this manner, each AA interacts with several BAs, and the set of BAs one AA has contact can be completely different from one AA to another. Moreover, BAs connected to an AA can also interact among themselves and with other BAs in sets connected to other AAs.

Connections among Autonomous Agents and Basic Agents result in a communication network. This network is determined by a programmer, based on the study of an activity plan which has the basic agents as its operators, and its goal is to achieve the autonomous agents desired behavior.

The information (data and task requisitions) exchange among BAs and AAs is done through a message exchange.

4. Description of the multi-agent system

As a testbed for the architecture described here, a purposive computer vision system performing simple visually guided assembly tasks is being implemented on a

Flexible Assembly Cell [9] at the Escola Politécnica da Universidade de São Paulo. It is composed of several workstations, two robotic manipulators with 5 degrees of freedom, acquisition boards and cameras, with all the computers linked by a local Ethernet network. Figure 2 presents a schematic description of the Flexible Assembly Cell and figure 3 presents one photo of the cell.

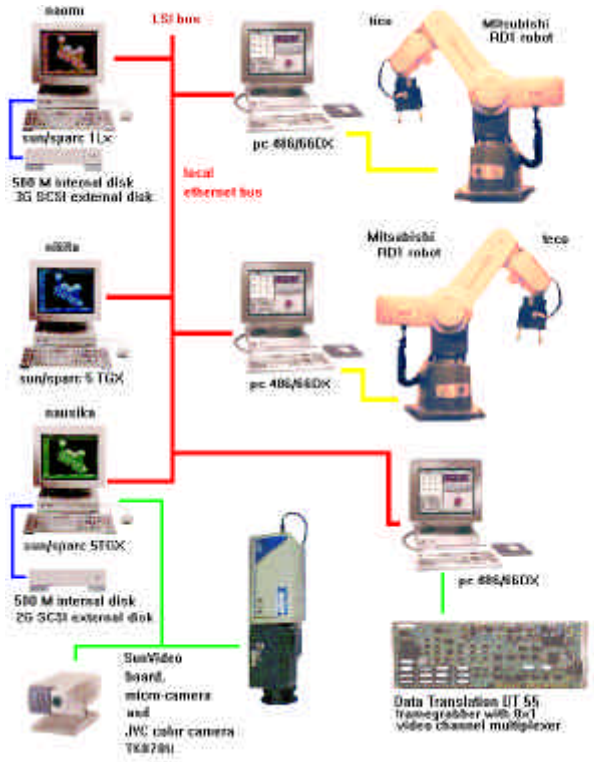


Figure 2: Schematic description of the Flexible Assembly Cell (by João Kogler)

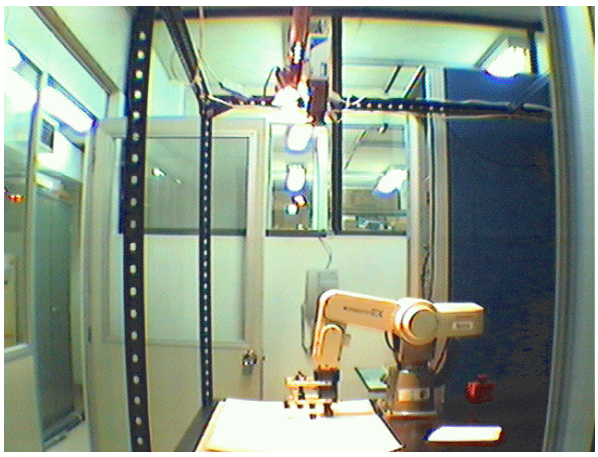


Figure 3: The Flexible Assembly Cell with one manipulator and one of the cameras at the top.

Whereas the chosen domain is the one of an assembly cell, the architecture can be applied to other domains, like to autonomous mobile robots.

The system, which uses one of the manipulators of the cell, is being implemented on several workstations (not related to the cell but in the Ethernet network), where the agents are executed as independent and parallel distributed processes, communicating through the cell network.

Three different behaviors were defined for this application, each one corresponding to an autonomous agent:

Assembler agent: to accomplish an assembly, picking up pieces on the workspace with the manipulator and putting them in a desired location. The goal of the assembly and the type of pieces involved in it can change, for example, from the assembly of a known object to the selection of pieces by shape or color. To be able to do this, this agent must be apt to plan the activities involved in the assembly. In the present state of this implementation, the agent does not have planning capacity, and the plan is previously defined.

Cleaner agent: to clean the workspace, which is a previously defined area, where the assembly is made. In this manner, unwanted objects put on this area must be taken away by this agent.

Collision Avoider agent: to avoid collisions of the manipulator with objects that move in the workspace (other manipulators, a hand), aiming the preservation of the system's physical integrity.

Three rules were defined for the Agents society, organizing the competition among them, in order to control the only resource shared in this application: the manipulator. The rules defined for the system are:

Rule # 1: Only one agent can control the manipulator in a given moment.

Rule # 2: Any agent can request the control of the manipulator to an agent with less authority than itself, at any moment.

Rule # 3: An agent can only request the control of the manipulator to an agent with more authority than itself if that agent is releasing the control.

The following authority structure as defined for the autonomous agents: the Collision-Avoider is the one with higher authority, the Cleaning agent is the second in the rank and the Assembler is the one with less authority. One can see that this structure has as main goal preserving the physical integrity of the system.

Concerning about the system implementation, a schematic representation of the autonomous and basic agents and their communication links is presented in figure 4.

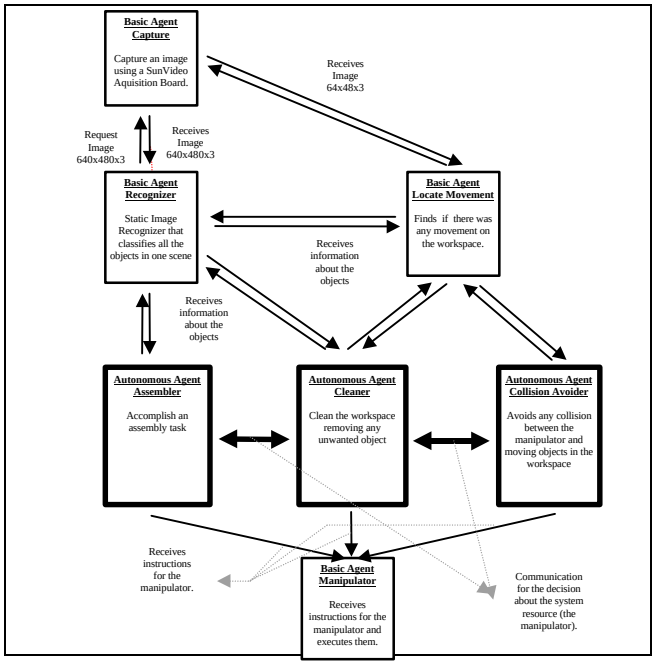
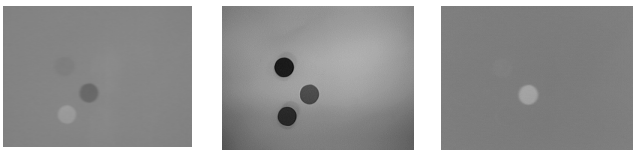


Figure 4: Scheme of the agents and its links for the system.

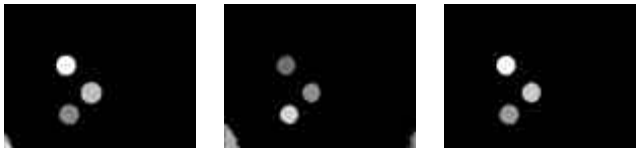
The basic agent Capture is responsible for the image acquisition, managing a SunVideo system designed for image acquisition and video compression in real time. The SunVideo consists of a Sun SBus board for SPARCstations with on-board video compression engine and the XIL Imaging Library, and is used for multimedia applications and video conferences. An image taken with the camera at the top of the cell is presented in figure 5.



Red Green Blue

Figure 5: Sample image from the camera.

The Static Image Recognizer receives an image and creates a list containing a description of the scene, with an "id" for each object, its type (one among a few known types), size in pixels and the position of the object in the scene. This agent is based on the blob coloring algorithm described in [10] to label regions of the image, adapting the algorithm to deal with color images. It works in 3 steps: first it makes a region labeling; then regions are grouped into objects; the centers of mass of the objects are calculated and finally unknown objects (like shadows in the corners of the image) are removed. The result is shown in figure 6.



Grouped Regions Objects in the scene Remaining Objects

Figure 6: Resulting objects in the image.

The basic agent Locate Movement discovers if there is any movement on the workspace. To simplify its implementation, only movements of red objects are detected. This agent examine consecutive 64x48 images received from the Capture agent. When it detects any movement on this small image, it uses the list generated by the Recognizer to accurately locate the movement.

The basic agent Manipulator is a simple program that receives instructions, as move to a position, close grip, open grip, and executes them. It is needed because the communication with the manipulator is made through a parallel port, and must run in a specific machine.

The Autonomous Agents are simple implementations of the behaviors already described, using the basic agents to perform their basic tasks. For example, the assembler agent request for the basic agent recognizer a list with the localization of all objects on the workspace. Then, the recognizer request an image for the basic agent capture, and creates a list for the assembler agent. After using the received list to plan an assembly sequence, the assembler agent sends instructions to the basic agent manipulator, executing the assembly plan.

The communication between the agents is implemented using the Parallel Virtual Machine (PVM) library. The PVM is a system for distributed processing implementations that offers tools for communication between tasks, like point to point messages and broadcasts, and tools to control the spawning of tasks. It was chosen due to the simplicity it allows in the implementation of message exchanges between the agents, and because it allows the definition of which workstation a process must run. Besides that, it is widely used in the academic community, which means better support for developers, having available an Internet newsgroup where assistance can be found.

Finally, we present an example that shows, in a LISP like manner, the messages that are exchanged when an object moves in the workspace during an assembly. First, the Collision-Avoider agent requests the control of the manipulator to avoid the possible collisions; the Assembler agent agrees with the requisition and transfers the control to the Collision-Avoider; when the danger of a possible collision ceases, Collision-Avoider releases the control of the resource, which is requested by Assembler; the request

is accepted by Collision-Avoider and Assembler gets the control and returns to its job.

```
comment: the control is with the Assembler.
((decision)(request)(collisionAvoider)(manipulator))
((decision)(agreed)(assembler) (manipulator))
((decision)(inform)((assembler)(give)(collisionAvoider)(manipulator)(piece in grip)))
((decision)(inform)((collisionAvoider)(take)(piece in grip)))
comment: the collision is avoided.
((decision)(free)(collisionAvoider) (manipulator))
((decision)(request)(assembler) (manipulator))
((decision)(agreed)(collisionAvoider) (assembler))
((decision)(inform)((collisionAvoider)(give)(assembler)(manipulator) (piece in grip)))
((decision)(inform)((assembler)(take) (manipulator)(piece in grip)))
```

Example 1 - Message exchange during an assembly interruption.

5. Discussions and conclusion

Multi-Agent approach proved to be a promising method to model architectures for purposive computer vision systems. There are several reasons for this conclusion:

1. it simplifies the mapping of system purpose in its behaviors, and of behaviors in tasks;
2. it makes explicit the interaction between behaviors;
3. all behaviors of a system can communicate with each other, resulting in a completely connected network;
4. the autonomous agents provide modularity: one does not need to have a precise knowledge of the internal structure of other agents in order to add a new behavior to the system.

However, the use of basic agents to avoid the repetition of processing efforts was not as good as expected. As can be seen in figure 4, the construction of the behavior of the autonomous agents based on basic ones is too complex. This makes harder implementation and maintenance of the system, as one must know which basic agents the system has in order to be able to implement a new autonomous agent. We think that the best solution would be to provide each autonomous agent with total independence from the others. It is known that this will add computational load to the system but this can be dealt with the addition of more processing machines to each new agent. In this way, the processing time will be the same even though the total load has been increased. In addition to that, fewer links between agents will decrease message exchange, unloading the network. This allows faster replies to requisitions between autonomous agents, decreasing the response time of the system.

6. References

[1] ALOIMONOS, Y. What I have learned. **CVGIP: Image Understanding**, v.60, n.1, p.74-85, July 1994.

[2] RILLO, A. H. R. C.; BIANCHI, R. A. C.; MOREIRA Jr, B.; FERRAZ, F. Integrando Visão e Comportamento: Uma aplicação de reconstrução propositiva. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, 11., São Paulo, 1996. **Anais.** São Paulo, Sociedade Brasileira de Automática, 1996. P. 573-578.

[3] BOND, A. H; GASSER, L **Readings in Distributed Artificial Intelligence**. Morgan Kaufmann, San Mateo, CA, 1988.

[4] BROOKS, R. A. Intelligence without representation. **Artificial Intelligence**, v.47, p.139-59, 1991.

[5] ELFES, A. A distributed control architecture for an autonomous mobile robot. **Artificial Intelligence**, Computational Mechanics Publications, v.1, n2,1986, p135-44.

[6] BOISSIER, O.; DEMAZEAU, Y. A distributed artificial intelligence view on general purpose vision systems. In: DEMAZEAU, Y; WERNER, E. (eds.) **Decentralized AI-3.** Amsterdam, Elsevier, 1992. p.311-330.

[7] BOISSIER, O; DEMAZEAU, Y. ASIC: An architecture for social and individual control and its application to Computer Vision. In: EUROPEAN WORKSHOP ON MODELING AUTONOMOUS AGENTS IN A MULTI-AGENT WORLD, 1994. **Proceedings.** 1994. p.107-18.

[8] RIVLIN, E.; ALOIMONOS, Y.; ROSENFELD, A. **Purposive Recognition: a framework.** CS-TR 2811, University of Maryland, College Park, 1991.

[9] RILLO, M.; RILLO, A.H.R.C.; COSTA, L.A.R. The LSI assembly cell. In: IFAC/IFIP/IFORS/IMACS/ISPE Symposium on information control problems in manufacturing technology, 7º, Toronto, 1992. **Proceedings.** IFAC, 1992. p. 361-5.

[10] BALLARD, D.; BROWN, C. **Computer Vision**. Englewood Cliffs, Prentice-Hall, 1982.