A Distributed Control Architecture for a Purposive Computer Vision System.

Reinaldo A. C. Bianchi[1], Anna H. R. C. Rillo[2].
[1/2]Divisão de Automação e Inteligência Artificial - Laboratório de Sistemas Integráveis
[2]Departamento de Engenharia de Computação e Sistemas Digitais

Escola Politécnica - Universidade de São Paulo - Brazil.
Av. Prof. Luciano Gualberto, travessa 3, 158.
05508-900 São Paulo, SP, Brazil.
E-mail: rbianchi@lsi.usp.br, arillo@lsi.usp.br

Abstract

In this paper we present a distributed control architecture engaged in purposive computer vision tasks. In this context, the vision system's purpose is translated into a set of behaviors, which are decomposed in specific tasks. A Multi-Agent approach is used to model purpose, behaviors, tasks and the relationship among them. Purpose is modeled by a society of autonomous agents, 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. As a testbed, the architecture of a specific system for visually guided assembly is presented, and a distributed implementation is described.

1. Introduction

Motivated by the present discussion between the Reconstructive and the Purposive paradigm of Computer Vision, and seeking a formal theory of behaviors, as proposed by Aloimonos [1], this paper presents a Distributed Control Architecture engaged in purposive computer vision tasks. Along with this Architecture, we define a Distributed AI based methodology to build a Purposive Computer Vision system.

In this context, the vision system's purpose is translated into a set of behaviors, which are decomposed in specific tasks. A Multi-Agent approach is used to model purpose, behaviors, tasks and the relationship among them. Purpose is modeled by a society of autonomous agents, 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.

The Architecture proposed has several distinguishing features. These features allow the cooperation between low and high processes, giving modularity, flexibility, and autonomy to the system, allowing the addition and deletion of agents responsible for behaviors, and facilitating the integration of system and environment.

This paper presents the purposive paradigm and its relation to the reconstructive one in section 2. In section 3 will be given an overview of Distributed Artificial Intelligence and Multi-Agent Systems. Works related to this are presented in section 4. An overview of the proposed architecture is given in section 5 and section 6 presents an implementation of the architecture in a real system. Finally section 7 concludes this paper.

2. Purposive computer vision

The reconstructive paradigm [11] is the dominant one and the first to appear. According to it, vision's goal is "to solve the problem of reconstructing an accurate representation of the 3-D scene and its proprieties from image cues such as shading, contours, motion, stereo, color, etc." [2]. The main concerns about this paradigm, as presented by Tarr and Black [11], are that "recovery algorithms are not robust in the presence of noise and they are irremediably inefficient".

Despite the fact that such assertion is questionable, the discontentment with the reconstructive paradigm became evident in the late 80's, giving rise to some new paradigms, like Active, Animate and Qualitative Vision. With the integration of the ideas proposed by these paradigms, and the addition of other new ones, the purposive paradigm arose.

The purposive paradigm [1] believes that vision must be considered within the set of tasks an agent must accomplish, and tries to find in the purpose of the agent the constrains to solve the ill posed problem of vision. Another fundamental characteristic of this paradigm is the large integration of visual modules with other AI ones, like planning, reasoning and learning modules, since vision is not considered as a self contained problem, which can be better treated if integrated with other AI modules.

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 main problem is how to organize solutions and define primitive tasks, focusing on architectures for integration of visual systems.

Aiming at this problem, Aloimonos [1] asks for a formal theory of behaviors, a kind of "behavioral calculus", and he suggests that some discrete formalisms are good candidates for formalizing behaviors. Seeking an answer to this question, we look at Distributed AI and Multi-Agent Systems theory as a field where this formalism can be found.

3. Distributed Artificial Intelligence and Multi-Agent Systems

Distributed Artificial Intelligence (DAI) is defined by Bond and Grasser [6] as: "the field of AI concerned with concurrency in AI computations, at many levels." It can be divided into two fields: Distributed Problem Solving (DPS), concerned on how to solve a particular problem with several cooperating modules, and Multi-Agent Systems (MAS), concerned with the coordination of the behaviors of several autonomous intelligent agents to solve one or more goals. DPS and MAS are not disjoint areas, as DPS can be viewed as a part of the solution of a problem in the MAS approach.

In the next section works that are related to this one, most of them based on DAI, will be presented.

4. Related works

Some works that based and inspired this work, and from where many ideas and criticism originated, are: Brooks [5] *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 [8] Distributed Control Architecture, where a system is divided into processing levels and where independent processes communicate through a blackboard; and Boissier and Demazeau [3],[4] ASIC Multi-Agents Control Architecture and the MAVI system, which integrate different visual modules using MAS theory.

One difference from our architecture to Brooks' Subsumption architecture, is that all behaviors of a system (represented by Autonomous Agents) can communicate with each other, forming a completely connected network. Other differences between both architectures are:

- while the decision capabilities of a Subsumption based system are hidden in the Finite State Machines and are not explicitly distributed, they are explicit and distributed in agents in our architecture, so we can know where a decision is being made;
- in our architecture the interface between agents is better defined than the interface between layers in the Subsumption Architecture. In this way, one does not need to have a precise knowledge of the topology of a layer to add a new behavior on the system, having only to add an agent to it. Therefore, adding behaviors in our architecture is easier than in the Subsumption architecture.
- finally, there is some symbolic knowledge (qualitative) in each agent, while there is no explicit representation in the Subsumption Architecture.

The main difference between our architecture and ASIC/MAVI [4] is that the latter has a more traditional approach to computer vision concerning the representation levels, based on the three levels proposed by Marr, while our architecture the modularization is purposive and behavioral, with less intermediate levels.

Another difference between both architectures is that in ASIC, the agent model is subdivided into layers based on the processes an agent has (a decision layer, an adaptation layer and a control layer, and only this last one interacts with the environment), while in our architecture an agent is simpler, relying on more basic agents to accomplish its tasks, and all agents interacts directly with the world.

Rivlin et al [10] and Aloimonos [1] also inspired this work. The first presents a framework for a purposive recognition system and the second discusses the purposive paradigm and asks for a theory of behaviors. Aloimonos [1] also presents a problem with Brooks architecture, specifically that in it should enable low and high level processes to cooperate directly, which it doesn't.

Trying to solve these problems, we propose the following architecture.

5. The proposed architecture

The previous sections introduce the context in which the control architecture proposed is embedded. In this section we present the architecture in details and the characteristics which enable it to show improvements over other known architectures.

In the proposed architecture, a system is modeled with autonomous agents (AAs), each one responsible for a specific behavior, organized in a society with rules of behavior where they have to communicate 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, which are responsible for specific tasks. This decomposition of the system in agents with their behaviors is similar to the task decomposition proposed by Brooks [5].

In this society, each autonomous agent is connected to all other agents, through a decentralized communication network, which topology is that of completely connected network. This topology is presented schematically in figure 1.

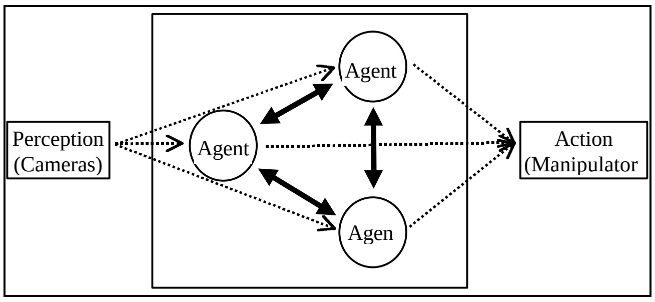

Figure 1 - Architecture Scheme.

The autonomous agents can be classified as the focus-agents described in Boissier and Demaseau [3], 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 autonomous agents 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, 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 the 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. But this camera could a resource if acting in an active vision system, where each agent could compete to control the process of data acquisition.

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

The definition of how the autonomous agents can allocate resources of the system is 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 behaviors by taking away resources. Finally, the authority structure has a logic nature, while the communication network between the autonomous agents has a physical nature.

This architecture can be observed from three different abstraction levels (based on a division proposed by RIVLIN et al [10]) concerning its behavioral aspects:

- in the first level, named intentional level, is the global purpose of the system. The system is represented here by a society of autonomous agents;
- in the intermediary level, named behavioral level, are the behaviors the system has to reach its intentions. It is represented by autonomous agents, each one showing a diverse behavior;
- in the third level, named task level, are the tasks in which the behaviors are decomposed. The system is represented here as a society of basic agents, with the autonomous agents acting as controllers for the basic agents.

This behavioral levels are given in table 1.

Behavioral Levels	**Description**
Intentions	Purpose of the system.
Behaviors	Behaviors in which the intention of the system is divided.
Tasks	Tasks in which the behaviors are divided.

Table 1 - Behavioral levels of the system.

Table 2 shows the behavioral levels, the Multi-Agent model used to implement each level, and their relation to the Brooks Subsumption Architecture.

Behavioral Level	**Multi-Agent Model**	**Brooks Subsumption Architecture**
Intentions	A society of autonomous agents.	The purpose of the system.
Behaviors	An autonomous agent to each behavior.	The layers of the architecture.
Tasks	Basic Agents in a society, composing the autonomous agents skills.	The Finite State Machines (FSM) that composes each layer.

Table 2 - Comparison between the behavioral levels, the MA model and the Subsumption Architecture.

We propose an empirical division of the system, based on Brooks [5] Task Decomposition and Rivlin [10] to split intentions in behaviors, and based on Boissier and Demazeau [3],[4] to split behaviors in tasks.

One of the main features of the proposed Architecture is the definition of an Autonomous Agent, which is viewed below.

5.1. 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 [4]. 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 language used by the AAs to communicate;

<*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. An 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.

5.2. AA's Communication Language

A Communication Language for the AAs is defined, based on the one described in Boissier and Demazeau [4], as:

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

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

request - used by an AA to request the control of a resource to another. 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;

inform - used to transfer the control of a resource from one AA to another. 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 (for example, when releasing a manipulator, if its grip is open or closed, with or without an object).

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

Messages which nature is **control** 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 with the insertion or deletion of an agent.

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

<nature>	<type>	<content>
control	**addNewAgent**	**<name of the new agent> <above which agents> <below which agents>**
	deleteAgent	**<name of the agent>**
	acknowledge	**<name of the agent>**
decision	**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 - The types of messages defined in the communication language of the autonomous agents.

5.3. Basic Agents (BAs)

In the proposed architecture, the 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, are located at this BAs, and is in them that the processing of this tasks occurs.

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, the BAs in the set an AA has connection can also interact among themselves and with other BAs in sets connected to other AAs. Figure 2 schematically draws an example of 2 AAs with their sets of BAs.

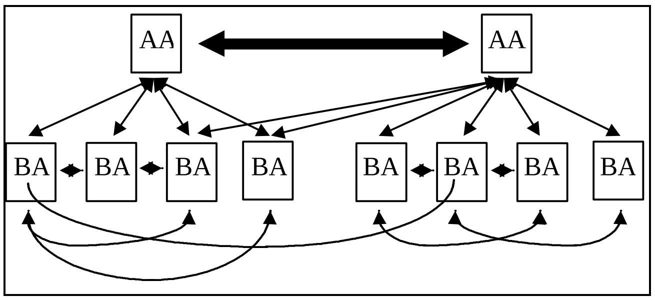

Figure 2 - Example of two autonomous agents and their basic agents.

The knowledge about the set of BAs an AA can interact is defined in a list containing the name of the BAs and the tasks each one can accomplish or the information they can provide. This list is located in the field <*basic agents*> of each autonomous agent.

In the same way, the BAs have knowledge about other BAs they can interact, and the respective task/information this other BAs can accomplish/provide. Therefore, it is clear that the basic agents also have decision capabilities, to be able to divide their tasks into sub-tasks other BAs can accomplish and rebuild the partial information provided by them to achieve one sub-goal.

The information (data and task requisitions) exchange among BAs and AAs is done through a message exchange, where a message is defined as:

<*message*> ::= <*type*><*to which*><*content*>

Where:

<*type*> - is the type of the message, if a request for task/data or a submission of information;

<*to which*> - is to which agent is the message. It can be sent to one or many agents, among AAs and ABs;

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

Here again, we must point out that this definition is based on Boissier and Demazeau [3]. Also, the basic agents proposed here are similar to the basic-agent used by them to analyse some computer vision systems. In that work, a basic-agent is defined as the building blocks of the systems, arising from the intersection between the vertical splitting (in tasks) and the horizontal splitting (in representation levels) of a system. Finally, they are also similar to the Finite State Machines of the Subsumption architecture.

6. Implementation of the architecture

This section briefly shows that the architecture is being implemented on a Flexible Assembly Cell [9] at the Escola Politécnica da Universidade de São Paulo, composed of several workstations, two robotic manipulators and cameras, in a system performing simple visually guided assembly tasks. 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 (figure 3), is being implemented on several workstations, where the agents are executed as independent and parallel distributed processes, communicating through the cell network.

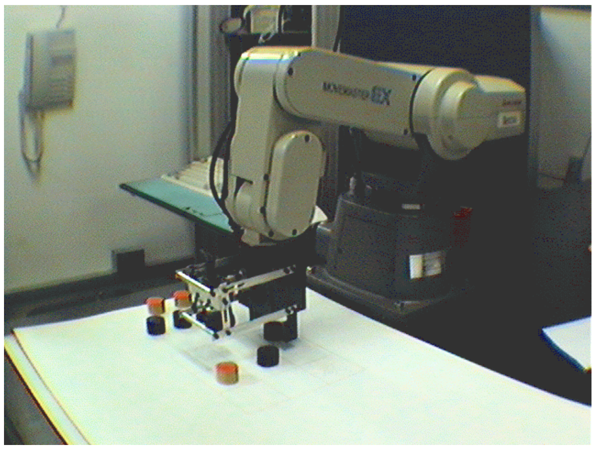

Figure 3: One of the manipulators at the Flexible Assembly Cell.

For this application was defined three different behaviors, each one corresponding to an autonomous agents, which are:

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. This planning can be simple or complex, reactive or temporal, depending on the complexity desired for the system. In the present state of this implementation, the agent does not have automatic planning, and the plan is therefore previously defined.

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

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

Recalling that resources are 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, we determined that the only resource the agents share in this application is the manipulator, as all them can use the cameras at the same time and, in this implementation, the cameras are not active sensors.

Also, the agents' society has three rules, which makes the three agents compete for the control of the manipulator (the only resource). 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 authority structure defined for the autonomous agents is: the Collision-Avoider is the one with higher authority, the Cleaning agent is the second in rank and the Assembler is the one with less authority. One can see that this structure has as main goal preserve the physical integrity of the system, and as a secondary goal, the global purpose of the system.

The example given below describes, in a LISP like manner, the messages that are exchanged when an object is placed on the workspace during an assembly. First, the Collision-Avoider agent requests the control of the manipulator to the Assembler, 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, the Collision-Avoider releases the control of the resource, which is requested by both the Cleaner and the Assembler agents; the Avoider agrees with the requisitions, and transfers the control to the agent with higher authority, the Cleaner, which assumes the control; when the Cleaner finishes to clean the workspace, it releases the control, which is requested by the Assembler; request accepted by the Cleaner, the Assembler reassumes the control and goes back 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)(request)(cleaner)
   (manipulator))
((decision)(agreed)(collisionAvoider)
   (manipulator))
((decision)(inform)((collisionAvoider)(
   give)(cleaner)(manipulator) (piece in
   grip)))
((decision)(inform)((cleaner)(take)
   (manipulator)(piece in grip)))
comment: the object is taken away.
((decision)(free)(cleaner)
   (manipulator))
((decision)(request)(assembler)
   (manipulator))
((decision)(agreed)(cleaner)
   (manipulator))
((decision)(inform)((cleaner)(give)
   (assembler)(manipulator)(piece in
   grip)))
((decision)(inform)((assembler) (take)
   (manipulator) (piece in grip)))
```

Example 1 - Message exchange during an assembly interruption.

7. Discussions and Conclusion

One feature which is worth noting is that in the presented architecture all behaviors of a system (represented by Autonomous Agents) can communicate with each other, resulting in a completely connected network. Another relevant feature is the modularity this architecture provides, as one does not need to have a precise knowledge of the topology of a layer to add a new behavior on the system, having only to add an agent to it.

Finally, we conclude that the Multi-Agents is a very fruitful approach to model an architecture for a purposive computer vision system, since it makes easier the mapping of the system purpose in its behaviors, and of the behaviors in tasks.

8. References

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

[2] BLACK, M. J.; ALOIMONOS, J.; HORSWILL, I.; SANDINI, G.; BROWN, C.M.; MALIK, J.; TARR, M.J. Action, representation, and purpose: re-evaluating the foundations of computational vision. In: INTERNATIONAL JOINT CONFERENCE ON AI, 1993. **Proceedings.** 1993. p.1661-6.

[3] 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.

[4] 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.

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

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

[7] DEMAZEAU, Y; MÜLLER, J. Decentralized Artificial Intelligence. In: DEMAZEAU, Y; MÜLLER, J. (eds.) **Decentralized AI 1.** Amsterdam, Elsevier, 1990. p.3-13.

[8] ELFES, A. A distributed control architecture for an autonomous mobile robot. **Artificial Intelligence**, Computational Mechanics Publications, v.1, n.2, 1986, p 135-44 .

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

[11] TARR, M. J.; BLACK, M. J. A computational and evolutionary perspective on the role of representation in vision. **CVGIP: Image Understanding**, v.60, n.1, p.65-73, July 1994.