A MULTI-AGENT APPROACH TO SELF-ORGANIZING VISION SYSTEMS

THORSTEN GRAF AND ALOIS KNOLL
University of Bielefeld, Faculty of Technology
P.O.Box 10 01 31, D-33501 Bielefeld, Germany
E-mail: {graf, knoll}@techfak.uni-bielefeld.de

We present a new multi-agent system architecture for modelling self-organizing computer vision systems, that provide a great degree of flexibility: Firstly, the vision systems can be easily incorporated into complex applications; secondly, the vision systems are able to adapt dynamically to different (possibly changing) tasks and environmental conditions; thirdly, the architecture provides the ability to handle different competitive information; and lastly, the addition of new processing modules can be done without rebuilding the complete system. We describe in detail the basic concepts of the proposed multi-agent system including the agent architectures, the control strategies as well as the communication language. The multi-agent system architecture is demonstrated on the basis of a distributed object recognition system modelled as a society of autonomous agents.

Keywords: multi-agent architecture, computer vision, object recognition

1 Introduction

The generation of computer vision systems applicable to complex industrial and robotics environments is an important and difficult task. Generally, these applications impose several conditions to the system architecture: Firstly, it must be simple to incorporate the vision system into complex applications; secondly, the vision system must be able to adapt dynamically to different (possibly changing) tasks and environmental conditions; thirdly, the architecture must provide the ability to handle different competitive information; and lastly, the addition of new processing modules must be able without rebuilding the complete system.

Recent research has indicated that modelling vision systems as societies of autonomous agents is a promising and powerful approach to tackle these objectives. Boissier and Demazeau[1] have proposed the MAVI system, a multi-agent system for visual integration, that is based on the ASIC[2] multi-agent control architecture. This architecture is subdivided into different processing layers, which is is too complex for most vision applications and makes it difficult to handle the system.

Following the purposive vision paradigms Bianchi and Rillo[3] have inspired a multi-agent vision system employing a behavior based decomposition in specific tasks, while Yanai and Deguchi[4] have developed an object recognition system for integrating different vision strategies. Contrary to the MAVI system, these approaches share a more rigid architecture which reduce the applicabilty to different environmental conditions and requirements.

In this paper we propose a new multi-agent system architecture, where we have tried to combine the advantages of the agent architectures mentioned above and to give some new impetus.

The paper is arranged as follows. Section 2 covers the basic concepts of the proposed multi-agent system architecture. This includes a description of the agent architectures (Sect. 2.1), of the control strategies (Sect. 2.2), and of the communication language (Sect. 2.3). In Sect. 3 we demonstrate an implementation of the architecture in a distributed object recognition system. Finally, in Sect. 4 we conclude the paper and give some future research directions.

2 The proposed architecture

2.1 Agent architecture

In contrast to other multi-agent systems, that are used for modelling computer vision systems, our architecture consists of two different classes of agents: master and slave agents. The former perform all of the high-level communication, planning, and most of the processing tasks while the latter are responsible for assistance only. Generally, the slave agents are related to a particular master agent and perform time-consuming tasks in teams.

Master agent

The architecture shared by all master agents is composed of five different modules:

1. *Communication module:*
   The communication module is responsible for connecting to other agents using a contract net protocol. It contains methods for sending and receiving messages as well as mechanisms for performing the negotiation. Additionally, the communication module includes agent specific functions for wrapping different data types.

2. *General knowledge:*
   This data base is used for storing general knowledge, like basic planning strategies and the grammar of the communication language.

3. *Individual knowledge:*
   The individual knowledge base contains all knowledge concerning the particular agent, like task-specific planning strategies, processing functions and the provided vocabulary. Note, that it is not required that all agents share the same vocabulary.

4. *Inference engine:*
   Based on general and individual knowledge the inference engine accomplishes all planning and interpretation tasks of the agent. The most extensive work that must be performed by the inference engine is the generation of programs appropriate to solve requests of other agents.

5. *Working memory:*
   The working memory is used by the inference engine for storing various information, like subresults and knowledge about the dynamic environment.

Slave agent

Since the purpose of a slave agent is simply to assist a master agent in performing time consuming tasks, the slave agent can communicate with its corresponding masters only. Slave agents are completely controlled by master agents, i.d. the master decides how many slave agents he wants to use as well as which particular processing function must be performed. Therefore slave agents need only a simple architecture containing the communication module, processing functions and rudimentary mechanisms for interpreting messages.

2.2 Control strategies

In our multi-agent system architecture the control mechanisms of computer vision systems are completely decentralized, i.d. each agent accomplishes requested tasks according to its own knowledge and goals. This provides the capability for modelling flexible computer vision systems.

In order to perform a particular vision task, the corresponding agent proceeds as follows:

1. The agent analyzes the given task, particularly the instruction, the destination specifications as well as additional constraints. Note, that it is generally not required that the agent must understand all of the source specifications.

2. According to the task specifications the agent automatically generates a complex *CLIPS*[5]-style program script, that is composed of all required processing functions as well as further requests to other agents.

3. The agent executes the generated program and performs exception handling for unexpected events.

An example of a simple program script, which is automatically generated by a feature extraction agent, is shown in Tab. 1. This program script is subdivided into three different sections: Firstly, in lines 01–04 the agent requests the extraction of an edge image from the agent society. Secondly, in lines 05–10 all feature extraction functions are performed and lastly, in lines 11–15 an answer is generated.

Table 1. Automatically generated program script

```
01 : (bind ?var-gen7 (make-instance instance-gen9 of GMessage
      (type REQUEST)))
02 : (send ?var-gen7 put-text "(extract ?dest ?src)(is-a ?dest image)
      (has-type ?dest edge)(is-a ?src image)(has-source ?src message)
      (has-index ?src 0)")
03 : (send ?var-gen7 put-data (send [input-message] get-data))
04 : (send-message (instance-name-to-symbol ?var-gen7))
05 : (bind ?var-gen10 (send ?var-gen7 get-data 0))
06 : (bind ?var-gen3 (extract-edge-points ?var-gen10 10))
07 : (delete-data ?var-gen10)
08 : (bind ?var-gen5 (extract-conics ?var-gen3 ellipse))
09 : (bind ?var-gen2 (extract-lines ?var-gen3))
10 : (delete-data ?var-gen3)
11 : (send [output-message] put-type ANSWER)
12 : (send [output-message] add-text "(extract lines UNKNOWN-2)")
13 : (send [output-message] add-data ?var-gen2)
14 : (send [output-message] add-text "(extract ellipses UNKNOWN-2)")
15 : (send [output-message] add-data ?var-gen5)
```

2.3 Communication Language

The interaction among the different agents is performed using a flexible human readable and writeable communication language. Similar to other systems that are based on the speech act theory we are making the message type explicit:

```
<message> ::= <type> <content>
```

Nevertheless, the allowed message types differ in some important respects: The message type *request* is used to request the assistance of other agents. Generally, this type indicates that an agent is not able to perform a particular vision task on its own. An *answer* message is a reply to a request or script message and the message type *inform* is used for passing additional information to other agents. Furthermore, we provide the message type *script*, that can be used for gaining direct access to the capabilities of an agent avoiding the interpretation mechanisms. Since the agents generally generate programs in order to perform requested tasks dynamically, programs can be passed directly.

The content of a message is composed of a message text and additional message data. An excerpt of the formal grammar used for specifying a message text is shown in Tab. 2. For reasons of efficiency this grammar allows only one goal for each message, where a goal is not a specific entity but rather some kind of global plan, which can be restricted by additional conditions. These conditions can be complex logical expressions containing variables as well.

Table 2. Formal grammar of messages

```
<message-text>   ::= (<goal> <variable>*) <goal-condition>*

<goal>           ::= convert | extract | introduce | ...

<goal-condition> ::= <attr-condition> | <not-condition> |
                     <and-condition> | <or-condition>
<attr-condition> ::= <is-a-attr> | <has-name-attr> | ...
<not-condition>  ::= (not <goal-condition>)
<and-condition>  ::= (and <goal-condition>+)
<or-condition>   ::= (or <goal-condition>+)
<is-a-attr>      ::= (is-a <variable> <object-class>)
<object-class>   ::= feature | image | ...
```

Suppose, we want an object recognition system to extract all ledges as well as all known red objects shown in an image taken from a camera, whose server is called 'pythia', we can simply write the following message text:

```
(extract    ?dest ?src)

(is-a       ?dest object)
(or (has-name  ?dest ledge)
    (has-color ?dest red))

(is-a       ?src  image)
(has-source ?src  camera)
(has-server ?src  pythia)
```

There are two important points to note here: Firstly, the interpretation of such messages can be realized in a very efficient manner using the pattern-matching facilities of expert system tools; and secondly, entities can be identified not only by their unambiguous names but also by their features and attributes. Although such querying requests are very useful and important to vision applications, they are generally not supported by other agent-based vision systems.

3 Experimental results

As a testbed for the proposed agent architectures and communication language we have transformed our object recognition system[6] into a society of autonomous agents. The agents have been implemented in C++ using the multi-agent generation tool *MagiC*[7]. This tool provides classes for building different types of agents and mechanisms for encapsulating all of the negotiation protocols and communications. The knowledge as well as planning strategies of the agents have been modelled using the expert system tool textitClips 6.10[5].

The society of autonomous agents consists of five different agent types, three master agents and two slave agents, each one corresponding to a particular vision task:

1. *Master/slave image processing agents:*
   Since most of the image processing algorithms, like convolution and edge detection, are very time consuming, we build both classes of agents, a master image processing agent as well as a slave image processing agent. The master agent performs all of the high-level communication with the society and incorporates strategies for splitting up particular image processing tasks in order to be accomplished by a dynamic team of slave agents.

2. *Feature extraction agent:*
   The feature extraction agent is responsible for feature specific tasks, including extraction of edge points from images and fitting of geometric primitives like lines and ellipses.

3. *Master/slave object recognition agents:*
   The object recognition agents perform a recognition process based on the fuzzy invariant indexing technique[6]. This process consists of grouping of geometric primitives, invariant calculation, hypothesis generation and verification. Similar to the image processing agents the master agent establishes all of the communication and planning strategies while the slaves are responsible for the execution of particular recognition tasks.

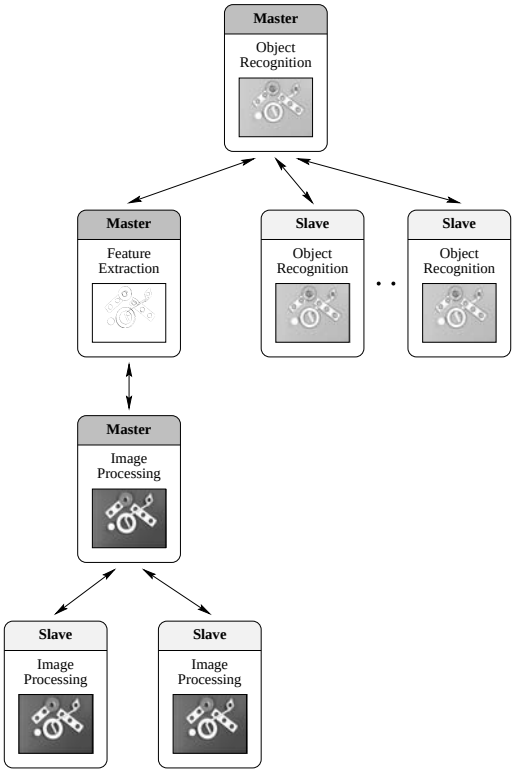
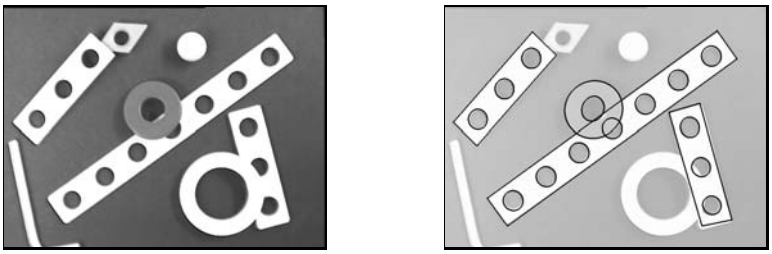


Figure 1. Self-organized system structure

Although this multi-agent system is able to recognize the object domain of (quasi-)planar wooden toy objects only, this is no principle limitation for the proposed multi-agent system architecture. In order to improve the recognition capability of the system new agents can be added without rebuilding the complete agent society.

We have run a number of agents of each type on different plattforms including Linux-PCs and Sun-Solaris-Workstations. Though we have not explicitely imposed any hierachical structure on the system, the agent society is capable of solving complex vision tasks. For doing so, the system organizes itself.

For example, if we request the task given in Sect. 2.3, the system takes on a transient system structure as sketched in Fig. 1.



(a) Original image (b) Result

Figure 2. Recognition result for a test scene

The result of this task is shown in Fig. 2. As can be seen, the system takes an image from the specified camera (Fig. 2a) and extracts all of the objects (Fig. 2b), that match the object specifications, namely two ledges and one red rim. All other objects shown in the image are ignored.

The addition or deletion of agents at run time causes no problems concerning the stability of the system (assuming that all agents necessary to accomplish a task are available). These behaviours are an important factor for both the autonomy as well as the openness of the system.

4 Conclusion

In this paper we have presented a new multi-agent system architecture dedicated for modelling distributed computer vision systems. It is constructed as

an assembly of autonomous agents which organize themselves in order to solve a particular vision task. As shown, this architecture has several distinguishing features, among them it gives flexibility, modularity, autonomy and openness to the system.

Future research covers the integration of different competitive recognition methods and the integration of the system into a complex robotics scenario.

Acknowledgments

T. Graf's contribution to this work was in part funded by the Deutsche Forschungsgemeinschaft within the postgraduate research unit "Aufgabenorientierte Kommunikation" (task-oriented communication).

References

1. O. Boissier and Y. Demazeau. Mavi: a multi-agent system for visual integration. In *Proc. IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems, Las Vegas, Nevada, USA*, pages 731–738, 1994.
2. O. Boissier and Y. Demazeau. Asic: An architecture for social and individual control and its application to computer vision. In *Proc. European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, pages 107–118, 1994.
3. R. Bianchi and A. Rillo. A purposive computer vision system: a multiagent approach. In *Workshop on Cybernetic Vision 1996, Proc. IEEE Computer Society*, pages 225–230, 1997.
4. K. Yanai and K. Deguchi. An architecture of object recognition systems for various images based on multi-agent. In *Proc. International conference on Pattern Recognition, Brisbane, Australia*, pages 278–281, 1998.
5. Artificial Intelligence Section. *CLIPS Reference Guide Volume I Basic Programming Guide*. Lyndon B. Johnson Space Center, 1998.
6. T. Graf, A. Knoll, and A. Wolfram. Recognition of partially occluded objects through fuzzy invariant indexing. In *Proc. IEEE International Conference on Fuzzy Systems, Anchorage, Alaska, USA*, pages 1566–1571, 1998.
7. C. Scheering and A. Knoll. Framework for implementing self-organized task-oriented multisensor guidance. In *Proc. International Symposium on Intelligence Systems and Advanced Manufacturing, Boston, USA*, 1998.