Driving Vision Systems by Communication

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

Abstract

In this paper we propose our new multi-agent system architecture dedicated to build distributed computer vision systems. This architecture incorporates a communication language which provides a great degree of flexibility: firstly, the language permits the construction of flexible and self-organizing vision systems; secondly, it is capable of expressing complex facts and tasks; thirdly, it is simple, i.e. human readable and writable; and lastly, the messages provided by the communication language can be interpreted efficiently. We describe in detail the basic concepts of the multi-agent system architecture including the communication language as well as the agent architecture and the interaction strategy. As a testbed for the proposed architecture we have modelled an object recognition system as a society of autonomous agents which organize themselves according to a given recognition task by employing communication.

1. Introduction

The development of architectures for integrating image processing methods dedicated to solving particular vision tasks have received increasing research interest because the architecture of a vision system has a great impact on its applicability.

Conventional vision systems generally follow system architectures which make it difficult to comply with the flexibility requirements of modern complex applications, like real-world robotic tasks [1]. To incorporate a vision system into such applications its architecture must meet certain requirements: Firstly, it must be simple to use the vision system in complex setups; 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 possible without rebuilding the complete system.

Recent research has indicated that modelling vision systems as societies of autonomous agents is a promising approach to tackle these objectives. In [2] Boissier and Demazeau have proposed a multi-agent system for visual integration, called MAVI, which is based on the ASIC [3] multi-agent control architecture. This architecture is subdivided into different processing layers. Following the purposive vision paradigms Bianchi and Rillo [4] have inspired a multi-agent vision system employing a behaviour based decomposition in specific tasks, while Yanai and Deguchi [5] have developed an object recognition system for integrating different vision strategies. Contrary to the MAVI system, these approaches share a more rigid architecture which reduces the applicability to different environmental conditions and requirements.

Since the capability to communicate is an essential feature of autonomous agents, the structure of the communication language is significant for the flexibility and applicability of the whole system. We note that the syntax of communication languages used in the computer vision domain generally tends to be cryptic and too simple to express complex facts and tasks; this is true especially for that part of the communication language which is relevant for the application itself.

We therefore propose a new multi-agent system architecture incorporating a more sophisticated communication language. This architecture simplifies the generation of complex self-organizing vision systems. In Sect. 2 we explain in detail the basic concepts of the architecture: the structure of the autonomous agents (Sect. 2.1), the communication language (Sect. 2.2), and the interaction strategy (Sect. 2.3). In Sect. 3 we demonstrate an implementation of the architecture in a distributed object recognition system, which organizes itself according to given recognition tasks. Finally, in Sect. 4 our conclusions and directions of possible future research are presented.

2. Multi-agent system architecture

In the proposed multi-agent system architecture a computer vision system is modelled as a society of autonomous agents, each one responsible for a particular vision task. Since many of the computer vision algorithms are very time-consuming the architecture provides two different classes of agents: master and slave agents. The former perform all of the planning and most of the processing tasks while the latter are responsible for the time-consuming tasks only. Generally, the slave agents work in teams controlled by a corresponding master agent.

In the society the agents are connected to each other using a contract net protocol, whose topology is that of a completely connected network.

In the following we explain in detail the multi-agent system architecture: the structure of the autonomous agents, the communication language and the interaction strategy.

2.1. Agent architectures

Since the master and slave agents are responsible for different tasks within the vision system they differ in their architecture as well.

2.1.1. Master agent. The architecture shared by all master agents is sketched in Fig. 1. As shown, it is composed of

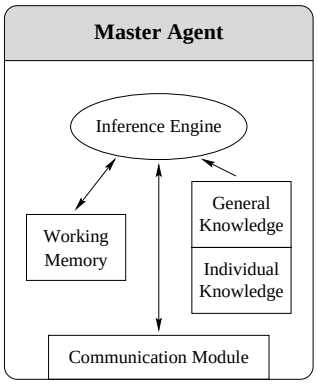


Figure 1. Architecture of a master agent

five different modules:

1. *Communication module:*
   The communication module is responsible for connecting to other agents. It contains methods for sending and receiving messages as well as 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. Most of the knowledge is stored in rules that are applicable in different situations and determines the general behaviour of master agents.

3. *Individual knowledge:*
   The individual knowledge base contains all knowledge concerning the particular agent, like specific planning strategies, processing functions and the provided vocabulary. Note that it is not required that all agents share the same vocabulary. Similar to the general knowledge base the individual knowledge is stored in rules and determines the individual behaviour of each agent.

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 to generate programs appropriate to solve requested tasks of other agents.

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

In order to perform a particular vision task, the responsible agent proceeds as follows: Firstly, the agent analyses the given task including the instruction, the destination specifications and additional constraints. Note that it is generally not required that the agent understand all of the source specifications. According to the task specifications the agent automatically generates a complex *Clips*[6]-style program script, that is composed of all required processing functions as well as further requests to other agents. Lastly, the agent executes the program script.

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

2.2. Communication language

The basic requirements for the communication language are as follows:

- It must permit the construction of flexible and self-organizing vision systems.
- It must be able to express complex facts and tasks.
- It must be simple to understand, i.e. human readable and writable.
- Efficient mechanisms for interpreting messages must be provided.

In order to meet these requirements we have developed a new communication language. We provide the use of message types making the intention of a message explicit:

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

The allowed message types are similar to the ones used in other communication languages [2, 4] in that they implement speech acts but differ in some important respects:

1. *request:*
   The message type *request* is used to request the assistance of other agents. Generally, this message type indicates that the agent can not perform a particular task on its own.

2. *answer:*
   An *answer* message is a reply to a request or script message, which can be both a result or an error message.

3. *inform:*
   The message type *inform* is used for passing additional information to other agents which is not necessarily needed for performing particular tasks. This type is also used for indicating the presence or absence of agents.

4. *script:*
   The message type *script* can be used for getting 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.

Furthermore the message content is subdivided into a message text and additional message data:

```
<content> ::= <text> <data>
```

```
<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-type-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 | ...
```

Table 1. Formal grammar of messages

where the `<text>`-slot contains a text string and the `<data>`-slot a list capable for storing different data types like images and edges. The text string itself can be both a message text or a *Clips*-style program script.

An excerpt of the formal grammar used for specifying a message text is shown in Tab. 1. 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.

Suppose, we want our 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 'penelope', 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  penelope)
```

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 fea-

tures 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.

2.3. Interaction strategy

In the proposed multi-agent system architecture the control mechanisms of computer vision systems are completely decentralised, i.e. each agent corresponds to particular vision tasks and accomplishes requests on its own knowledge and goals. No further control mechanisms among master agents, like hierarchical structuring, are imposed.

Therefore, the agents have to interact with each other in order to solve a requested vision task. The interaction is performed by a communication process that leads to a self-organization of the agent society. This self-organization process is goal-driven and proceeds as follows:

- If an agent requests a vision task, all master agents decide if they can accomplish the given task. As mentioned before, this is done by analysing the instruction, the destination specifications and the additional constraints. Generally, the source specifications of the task are neglected.

- All master agents that are responsible for the particular task make a bid. According to these bids the agent, that has requested the task, selects the masters that should award the contract.

- The selected masters generate appropriate program scripts according to the task specifications. If the source of the task is unknown the agents request a required sub-result determined from the unknown source.

There are some important points to note here: A drawback of this interaction strategy is that it can be established just at run-time if the vision system can accomplish a particular vision task. Furthermore, the interaction strategy produces some overhead because all master agents try to react to a request. Nevertheless, this overhead can be neglected since most of the vision algorithms are generally very time-consuming compared with the overhead.

The advantage of this approach is the flexibility of the resulting vision system: agents can be added and deleted at run-time without causing any problems. Furthermore, new functionality can be provided by simply adding appropriate agents without having precise knowledge of the internal structure of existing agents.

3. Experimental results

As a testbed for the proposed agent architectures and communication language we have transformed our object recognition system described in [7, 8] into a society of autonomous agents. The agents have been implemented in C++ using the multi-agent generation tool *MagiC* [9]. 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 *Clips 6.10* [6].

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 [7]. 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.

We have run a number of agents of each type on different platforms including Linux-PCs and Sun-Solaris-Workstations. Although we have not imposed any hierarchical structure on the system, the agent society is capable of solving complex vision tasks.

For example, if we request the task given in Sect. 2.2, the system takes on a transient system structure as sketched in Fig. 2. The corresponding trace of message passing is shown in Tab. 2. As indicated, the master object recognition agent is the only agent capable of recognizing objects

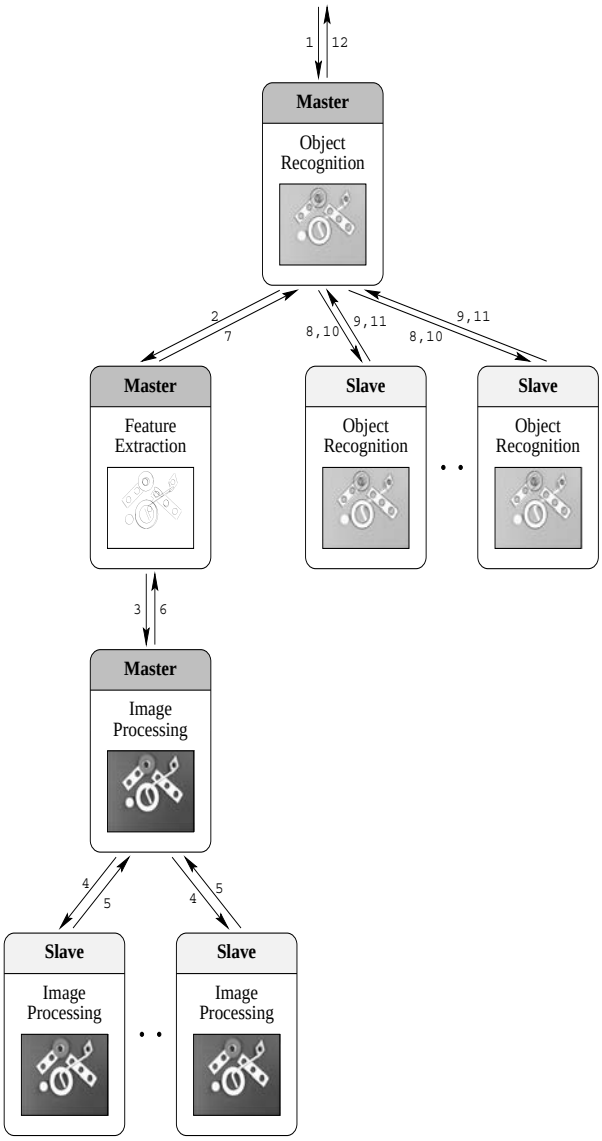


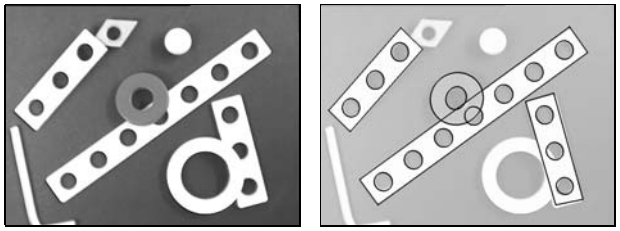
Figure 2. Self-organized system structure

Table 2. Trace of message passing

```
1: REQUEST:
   (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 penelope)
2: REQUEST:
   (extract ?dest ?src)
   (is-a ?dest feature)
   (or (has-type ?dest line)
       (has-type ?dest ellipse))
   (is-a ?src image)
   (has-source ?src camera)
   (has-server ?src penelope)
3: REQUEST:
   (extract ?dest ?src)
   (is-a ?dest image)
   (has-type ?dest edge)
   (is-a ?src image)
   (has-source ?src camera)
   (has-server ?src penelope)
4: REQUEST:
   (apply-canny)
5: ANSWER:
   (apply-canny)
6: ANSWER:
   (extract edge-image penelope-1)
7: ANSWER:
   (extract lines UNKNOWN-2)
   (extract ellipses UNKNOWN-2)
8: REQUEST:
   (generate-hypotheses)
9: ANSWER:
   (generate-hypotheses)
10: REQUEST:
    (verify-single-hypotheses)
11: ANSWER:
    (verify-single-hypotheses)
12: ANSWER:
    (extract rim UNKNOWN-2)
    (extract ledge-3 UNKNOWN-2)
    (extract ledge-7 UNKNOWN-2)
```

in images. Although the object recognition agent has no knowledge about accessing cameras, the agent is awarded the contract. In order to solve the recognition task the agent needs geometric primitives (especially lines and ellipses) extracted in an image. Since the source specification does not match this requirement, the agent requests to extract the geometric primitives from the unknown source specifications (2). Next, a feature extraction agent is awarded the contract. Again, this agent needs the assistance of the agent society to detect the required edge points from the unknown source (3). This sub-task is solved by the master image processing agent. The agent grabs an image from the specified camera (see Fig. 3a) and asks its slaves to apply an edge operator (4, 5). Note, that the communication between master and slave agents is very simple. Using the resulting edge image (6) the feature extraction agent extracts the particular geometric primitives and passes them to the master object recognition agent (7). Now, the master object recognition agent is able to recognize the specified objects. This is done with the assistance of the slave object recognition agents, which perform the hypotheses generation as well as the verification of the hypotheses (8-11). Finally, the recognition task is accomplished (12). The final result of this task is shown in Fig. 3b. As can be seen, the system recognizes all of the objects that match the object specifications, namely two ledges and one red rim. All other objects shown in the

image are ignored.



(a) Original image (b) Result

Figure 3. Recognition result for a test scene

The addition or deletion of agents at run-time causes no problems of system stability (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 flexibility of the system, i.e. to improve the recognition capability of the vision system new agents can be added.

4. Conclusions and future research

We have presented a new multi-agent system architecture dedicated for building self-organizing distributed computer vision systems that are only controlled by a given vision task. The architecture incorporates a communication language, which is capable of expressing complex facts and tasks without loosing its readability. As shown, this architecture has several distinguishing features, such as flexibility, modularity, autonomy and openness.

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

Acknowledgement

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] A. Knoll, B. Hildebrandt, and J. Zhand. Instructing cooperating assembly robots through situated dialogs in natural language. In *Proc. IEEE Conference on Robotics and Automation, Albuquerque, New Mexico*, 1997.

[2] 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.

[3] O. Boissier and Y. Demazeau. Asic: An architecture for social and individual control and its application to computer vision. In *Proc. European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 107–118, 1994.

[4] R. Bianchi and A. Rillo. A purposive computer vision system: a multi-agent approach. In *Workshop on Cybernetic Vision 1996, Proc. IEEE Computer Society*, pages 225–230, 1997.

[5] 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.

[6] Artificial Intelligence Section. *CLIPS Reference Guide Volume I Basic Programming Guide*. Lyndon B. Johnson Space Center, 1998.

[7] 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.

[8] T. Graf, A. Knoll, and A. Wolfram. Fuzzy invariant indexing: a general indexing scheme for occluded object recognition. In *Proc. International Conference on Signal Processing, Beijing, China*, pages 908–911, 1998.

[9] 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.