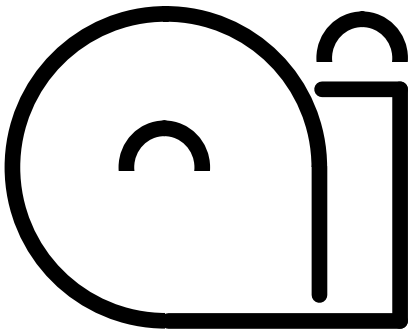


Österreichisches Forschungsinstitut für / Austrian Research Institute for / Artificial Intelligence

TR–2005–03

Bernhard Jung, Paolo Petta

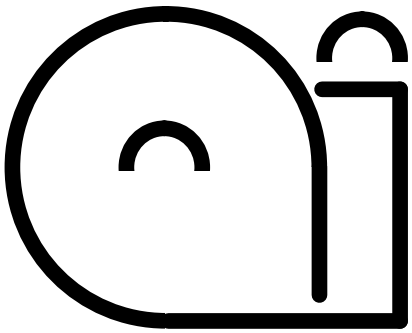
Towards Multi-Agent Coordination in Cognitive Vision

• Freyung 6/6 • A-1010 Vienna • Austria •

• Phone: +43-1-5336112 •

• mailto:sec@oefai.at •

• http://www.oefai.at/oefai/ •



Österreichisches Forschungsinstitut für / Austrian Research Institute for / Artificial Intelligence

TR–2005–03

Bernhard Jung, Paolo Petta

Towards Multi-Agent Coordination in Cognitive Vision

The Austrian Research Institute for Artificial Intelligence is supported by the Federal Ministry of Education, Science and Culture.

Citation: Jung B., Petta P.: Towards Multi-Agent Coordination in Cognitive Vision. In Zillich M., Vincze M. (eds.): 1st Austrian Cognitive Vision Workshop, OCG (Austrian Computer Society), Vienna, Austria, EU, books@ocg.at Band 186, 2005, pp.9-18.

Towards Multi-Agent Coordination in Cognitive Vision

Bernhard Jung, Paolo Petta

Austrian Research Institute for AI (OFAI)
A-1010 Vienna, Freyung 6/6
{*bernhard.jung,paolo*}*@oefai.at*

Abstract:
This paper casts a simple cognitive vision design into a multi-agent framework and therein addresses the questions how and to what extent explicit consideration of coordination may affect the design and performance of such systems. We give a description of the evolution of our implementation, show how in it tracking and reasoning emerge out of the distributed interaction between component agents, describe individual agent capabilities and the conversation policies employed, and compare its performance to the one of the original design.

1 Introduction

To frame the domain of our work, we first give a short description of the scenario used and a brief introduction to coordination in multi-agent systems. We use the Hide&Seek M6 setting (aka "shell game") of the Joint Research Program (JRP) Cognitive Vision[1)]. It consists of a stationary camera, two black cups and a red or orange ball on a table, and a single human hand moving and lifting the cups in turn to hide and unhide the ball. The questions to be answered by the system include: "Where is the ball?", "What is hiding the ball?", and "What was the trajectory of a cup, the ball, or the hand?". Beyond detection and tracking of objects, these all also require some capability to reason about occlusion.

[Wooldridge 2002] defines *multi-agent systems* (MAS) as "…composed of multiple interacting computing elements, known as *agents*. Agents are computer systems with two important capabilities. First, they are … capable of *autonomous action*—of deciding *for themselves* what they need to do in order to satisfy their design objectives. Second, they are capable of interacting with other agents—not simply by exchanging data, but by engaging in analogues of the kind of social activity that we all engage in every day of our lives: cooperation, coordination, and the like." We view agents mainly as a software engineering paradigm to cope with *interaction*, which may be the most important single characteristic of complex distributed software.

Coordination becomes necessary when there are interdependencies among agent activities [Lesser 1998]. These can occur e.g. when agents intend to work on the same or overlapping sub-

[1)] To be available at <URL:http://www.acin.tuwien.ac.at/groups/robtec/fsp/fsp.htm>.

problems, have alternative methods or data to generate a solution, or when the results of one subproblem contribute to the solution of another. Coordination typically requires the *detection* of interdependencies, a *decision* which coordination action to apply, and some *coordination mechanism* (for an overview of available techniques, see e.g. [Omicini & Ossowski 2003]) that shapes the way the agents perform these tasks.

2 Towards an agent-based scenario solution

We begin this section by describing an existing solution to the shell game scenario described in the previous section. We then detail how this design was first recast into a multi-agent system architecture subsequently used as a reference benchmark.

2.1 The original design

An implementation of a more traditional approach to derive trajectories and occlusion relations of objects in the scene was provided by the JRP partner EMT[2)]. Its design is based on three object detectors that perform colour-based segmentation with fixed thresholds and calculate different properties, for the contiguous regions derived, including area; centre of gravity; bounding box; and solidity (defined as the ratio of total and convex area). The decision of *object detection* is then based upon these properties. These detectors are also used to *track* objects by considering two objects in subsequent frames to be the same if the distance between their centres of gravity remains below a given threshold. The tracker can thus determine *new* objects and the *loss* of previously detected and tracked ones. This information is used by a reasoner to maintain an *occlusion tree* with information about the occlusion relations between objects (which object is hidden by which other one).

For each lost object, a hiding object is looked for: if successful, the lost object is assigned to its hider and hereafter assumed to move together with it. New objects are first assumed to have been *unhidden* and a match is attempted to hidden object entries in the occlusion tree. If a match is found, the hitherto hidden object is unlinked from its hider and updated with the information from the newly detected object. The new object is thus *merged* with the old one. In this manner, stable reasoning about occlusion of the ball and temporary occlusions by the hand hiding the cups is achieved.

2.2 Agentifying the solution

To cast the existing EMT solution into a multi-agent system, we first modularised the monolithic MatLab-based architecture into specialised detectors for balls, cups, and skin; trackers; and an occlusion tree reasoner. For the next steps, we had to extend the ZWORK framework[3)]

[2)] Andreas Opelt, Gerald Schweighofer, Axel Pinz, and Raffaelo Tomasi from the Institute of Electrical Measurement and Measurement Signal Processing, Graz University of Technology.

[3)] ZWORK is the basis for the common agent-oriented framework in the Cognitive Vision JRP.

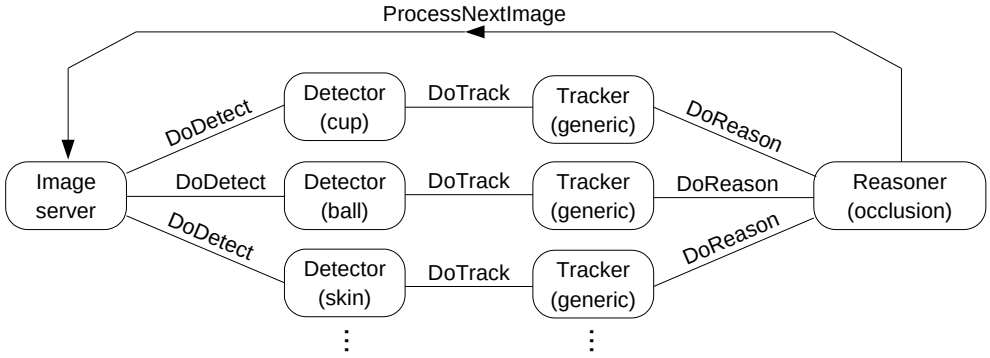


Figure 1: A simple multi-agent architecture with detector, tracker, and reasoner agents.

[Ponweiser et al. 2003], which allows to run components as autonomous service-oriented agents. These extensions included:

- Dynamic creation and shutdown of single agents, allowing to start new agents at run-time; to *register* them *by name* (a prerequisite for direct communication); and to unregister and terminate them when no longer needed (thus freeing up resources).

- A direct, datagram-like message passing service that allows addressing by *name* in addition to the service oriented, connection-based, request-response communication patterns previously available. This extension obviates the need to define service definitions explicitly, simplifying the programming effort substantially.

This design, comprising single instances of detector, tracker, and reasoner agents was subsequently used as benchmark wrt. performance, correctness and architectural design.

Further splitting of the original detector into three different specialised instances—one for each type: ball, cup, and skin—leads to a slightly different design that is *extensible* in terms of detectors for new object types (see figure 1) without having to change the overall architecture or the tracking and reasoning agents. Due to this parallelisation of detection and tracking, the reasoner agent now has to coordinate the sub-results the tracker agents deliver at different times. The easiest solution is to wait for the arrival of all sub-results for each frame before performing the reasoning step. While this modularised design may be useful when it comes to integrate new or different types of detectors (or trackers), it basically follows the overall approach of the original EMT architecture. In contrast, the alternative design discussed next puts objects into the spotlight.

3 Tracking and reasoning by agent coordination

In our final, fully agentified architecture, each object is represented by an individual agent. These *object agents* are additionally supported by *detector agents*, that provide an anchoring to the objects detected in the image. In this design, tracking and reasoning functionalities *emerge* out of the interactions among object agents and between object and detector agents (see figure 2). Although the

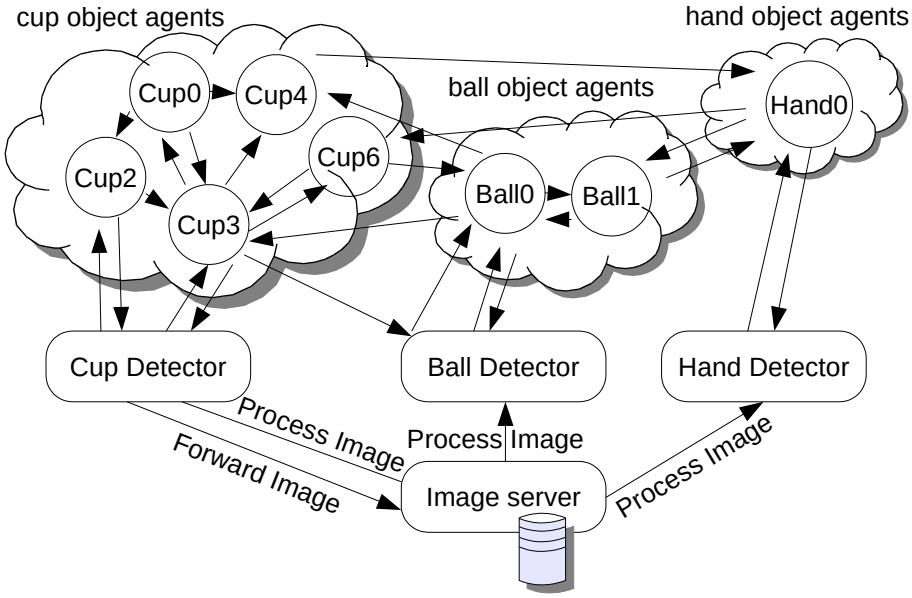


Figure 2: Architecture consisting of detector agents interacting with object agents.

design of the conversation policy [Greaves et al. 1999] is guided by the ideas of tracking and (occlusion tree) reasoning and include specific "tracking" and "reasoning" phases (see below), the overall tracking and reasoning capabilities expressed by the system cannot be pinned down to particular component agents, but comes about in a distributed fashion in the system. In the following, we first describe the responsibilities of detector and object agents and then discuss the conversation policy governing agent interaction in more detail.

3.1 Detector agents

A *detector agent* is responsible for the processing of new images provided by the image server agent and the detection of blobs of objects of a given type. It further distributes the information about blobs to other interested object agents of the same type. The interest of an object agent is expressed in terms of multiple regions of interest and is specified on demand[4)]. A detector agent is further responsible for mediating among multiple object agents claiming a blob. In MAS terminology, a detector agent holds an *auction* and sells the blob to the object agent bidding with the highest confidence in the blob being the object it represents (as determined by measures over centre of gravity and bounding boxes). A detector agent will spawn *new object agents* for unmatched blobs (i.e., not claimed by the existing object agent population), and serves as a *matchmaker agent* [Wong & Sycara 2000] distributing requests to object agents to identify a hider object for a disappeared object.

Furthermore, it is the joint responsibility of the set of all detector agent instances to coordinate the tracking and reasoning phases in the conversation policy (see 3.3).

[4)] The specification of more than one area is required to handle reappearance after occlusion correctly, as explained in the following.

3.2 Object agents

An *object agent* represents an object in the scene and is associated to the detector agent that spawned it. Object agents are responsible for matching of blobs offered by their detector agents to the data maintained locally about the last known blob and thus for coherent tracking of objects. They further handle disappearance and reappearance events in the scene by linking and unlinking themselves to hider objects. Once linked, a *contract* is established [Jennings 1996] between the agents representing the hider and the hidden objects, and the hider agent subsequently propagates position changes to the hidden object agents. Object agents also send updates of their areas of interest to their detector agent. These areas are usually extended bounding boxes around current object positions (and in the case of hidden objects also the locations of their disappearance).

By this means, detector agents automatically offer blobs to object agents representing hidden objects; reappearance can be detected, and no "object merging" as in the original EMT approach is necessary. If an object agent's reference object goes undetected for some time without identification of an appropriate hider, the object agent will eventually assume its object disappeared from the scene and die.

3.3 Conversation policy

Coordination among agent instances is governed by an encompassing conversation policy. A simplified diagram of the policy is shown in figure 3. The policy is started by the image server that sends `ProcessImage` messages to the (three) available detector agents. In the case a detector agent fails to identify any blobs of its kind in the current image within the regions of interest of its object agents, it sends them `NoBlobFound` messages, to be confirmed by a `BlobConfirmed` return message. Otherwise, the detector agents try to assign each of the blobs detected to one of their existing object agents, according to the location of the blob and the areas of interest of the object agents being managed: a blob may be offered to multiple object agents, and an object agent may have multiple blobs offered by its detector agent. To this end, the detector agents send `DetectedBlobs` messages to their object agents. The object agents then select the blob that is most likely to be the object it represents, and return a `BlobSelection` message containing the index of the preferred blob and a confidence measure[5)]. The detector agent waits for all `BlobSelection` messages to be returned and resolves ambiguous selections by sending a `ConfirmBlob` message to the object agent that expressed the highest confidence. In subsequent iterations, blobs remaining are offered to object agents not yet awarded a blob, and the sub-policy ends with a (possibly empty) remainder set of not assignable (new) blobs for which the detector agents spawns new object agents. The *tracking phase* thus ends with receipt of the messages confirming creation of new agents (if any). Before starting the reasoning phase, *all* detector agents must have finished the tracking phase.

[5)] Calculated over the centre of gravity of the blob offered and the last known object position or location of disappearance.

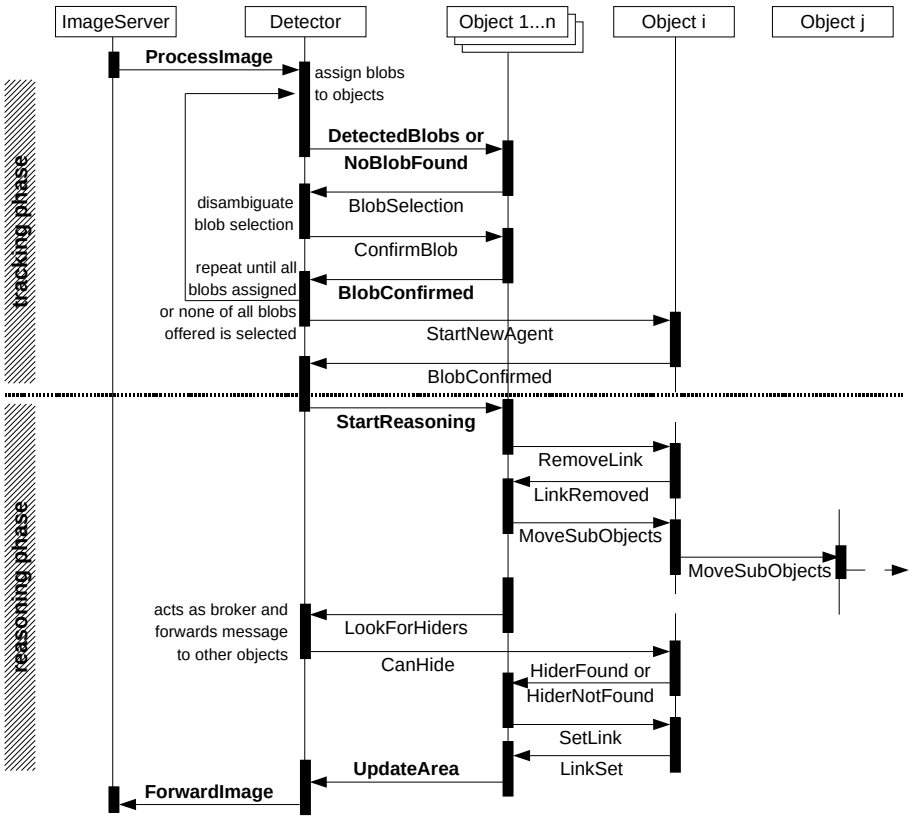


Figure 3: Schematic sequence diagram for the conversation policy. Mandatory messages are set in bold font. Parallel messages, conditions for messages, and synchronisation between detector agents, are omitted for clarity. See main text for details.

In the *reasoning phase*, detector agents take on the role of matchmakers. We adopted this as approach of choice, given that detector agents know about all object agents they have spawned, and a message broadcast is on the one hand currently not available in ZWORK, and on the other hand not necessarily desirable, as detector agents have information about the area an object is located in, and can thus perform a kind of targeted multi-casting for `CanHide` messages (see below). This phase is initiated by the detector agents sending `StartReasoning` messages to all known object agents. These determine whether they have reappeared, in which case they send a `RemoveLink` request to their hider object agents, who remove the link and confirm removal (`LinkRemoved`). Next, every hider object agent sends a `MoveSubObjects` message to the hidden object agents linked to it, thus propagating the relative movement of its own object. This message is forwarded recursively to further hidden objects.

The next policy step is to identify hider objects for all disappeared objects. Disappeared object agents broadcast `LookForHiders` messages to *all* detector agents; these forward these requests as `CanHide` messages to all objects they know to be in the neighbourhood of the disappeared objects. Potential hiders determine confidence values for them being the cause of specific occlusions and accordingly return `HiderFound` or `HiderNotFound` messages to the disappeared object agent. In the case of multiple possible hiders, the disappeared object agent selects the one with the highest

confidence and sends a `SetLink` request to it. This request is confirmed by a `LinkSet` message from the hider object agent. Object agents end their participation in the policy by sending an `UpdateArea` message to their detector agent, thus adjusting their area of interest. Upon receipt of all `UpdateArea` messages from their object agents by the different detector agents the reasoning phase ends, and the image server is asked by means of a `ForwardImage` message to proceed to the next image.

4 Results

Performance measurements were done for the three component (section 2.2) and the fully agentified (section 3) solutions on a Pentium M 1.6 GHz notebook under Gentoo Linux[6)]. The ZWORK framework was started with the appropriate agents, processed 100 frames of the scenario (with all objects visible and an occlusion of the ball occurring), and terminated. The time of the whole process was measured. To get an idea of the relative processing time of the three sub-components in the simple solution, we added one component after the other. Afterwards, we executed the agentified solution. The following table summarises the results:

components	time (secs)
none (just image server)	6.0
detector	37.5
detector+tracker	41.7
all three components	46.6
agentified solution	40.5
approximate tracking & reasoning times for:	
three components solution	9.1
agentified solution	3.0

This shows clearly that the detector algorithms are the major bottleneck.[7)] Because the same detector algorithms are used in all configurations, the difference of about six seconds between the simple and the agentified solution can be attributed to the way tracking and reasoning are designed and implemented. MatLab-generated code seems to be inherently slower than hand-written C/C++ code, so that these results cannot be interpreted as the agentified solution being three times faster at tracking and reasoning than the three component solution. Nevertheless, it does show that the agentified solution is no worse.

The test for correctness of both approaches was done only by observation of the overlayed information (bounding boxes and names of tracked objects with links to bounding boxes and names of

[6)] Working memory requirements appear to be secondary, on the order of 40MB resident and 160MB virtual size of a multi-threaded process.

[7)] Optimising the MatLab-generated code for colour conversion from RGB to HSV in the skin detector reduced the time from the originally required $\sim$77 to around 38 seconds.

hidden objects) on the whole image sequence. The approaches showed slight differences in the times of object reappearance during phases of occlusion, but all objects are tracked and labelled correctly at the end of the occlusion episode.

5 Related work

The modularisation of the EMT solution was guided by the structure of the available MatLab code resulting in detector, tracker, and reasoner agents. Our final solution used only detector and object agents. The agents of neither solution correspond directly to what [Boissier & Demazeau 1992] termed *basic agents*. These are the cells of a two-dimensional matrix of a focus level axis (contours, highlights, range data, stereo-vision, ...) and a representation level axis (image, image features, scene features, ...). They follow a traditional decomposition due to [Marr 1982], whereas the agents discussed follow more or less a functional decomposition (EMT, section 2.2) or object-centred one (section 3). [Bianchi & Rillo 1996] follow up on the work of Boissier and Demazeau and present a distributed control architecture applied to purposive computer vision tasks. The system is specified in terms of a set of behaviours which are decomposed into tasks and delegated by autonomous agents (AAs) to basic agents (BAs).

[Graf & Knoll 2000] propose a multi-agent system architecture which provides a system with a greater degree of flexibility than Boissier & Demazeau and Bianchi & Rillo. They design agents that are responsible for specific vision tasks and accomplish them by a goal-driven communication process. They distinguish master agents with complex planning and interpretation task capabilities, and slave agents encapsulating image processing algorithms.

Recently, a special issue of the pattern recognition journal on agent-based computer vision was published [Rosin & Rana 2004]. The papers of this issue provide a snapshot of the state of the art in the area. They demonstrate how multi-agent systems can be applied to a variety of vision tasks, but also that the full potential of the agent and multi-agent paradigm is still to be uncovered.

6 Discussion

The conversation policy of our current design has two major synchronisation points[8)], one after the tracking phase, and the other after the reasoning phase. Both are necessary to perform a consistent tracking and reasoning and to keep in sync with the image data. Whether this explicit synchronisation can be removed—e.g., by another conversation policy or intelligent scheduling—is subject of future work.

Implementation of conversation policies can be facilitated by framework support for hierarchical

[8)] There are also some minor synchronisation points, e.g. after link removal for a disappeared object, right before the `MoveSubObject` messages.

finite state machines. As this support is currently not available in ZWORK, the current implementation is not well structured, and the conversation policy was assembled from existing code and design fragments that guided the implementation.

The autonomy of the agents in our system is not yet an absolute necessity for the kind of coordination we investigate, as various timing issues have been neglected for the time being. In particular, all agent behaviour is synchronised with the video frame rate and not real-time. Nevertheless, the agent paradigm already helps in the design of the overall system and of the interactions to be coordinated.

As an alternative to the agentified solution, one could come up with two functions corresponding to the tracking and reasoning phases and performing the same kind of tracking and reasoning our system does. Arguably, these could be easier to implement, not requiring an agent framework and autonomous agents, nor explicit synchronisation with the image stream. We propose, however, that a main feature of our approach lies in the change of viewpoint on the problem of tracking and reasoning afforded by it. Although the design of appropriate conversation policies requires more (or at least different) skills and leads to a complex system, it does simplify the integration of heterogeneous detectors and coverage of specific objects (e.g., objects not capable of hiding or being hidden). Even so, we grant that thinking and designing locally from an agent's viewpoint, as well as locally in space (in terms of regions of interest), requires non-local dependencies to be captured explicitly (e.g., the impact of changes in lighting or cast shadows) and may lead to unpredictable (for better or worse) system behaviour (a typical problem of complex systems). But while these properties of such a MAS may be seen as disadvantages when building industrial systems, they may in fact be an important advantage when it comes to grasping the complexity of vision.

As the current results show, our fully agentified solution does not differ significantly from the original design in terms of correctness or speed. This may be seen to support the criticism of agent-based systems that *"they are just an elaborate and unnecessary metaphor, and often do not actually provide better results than traditional techniques"* [Rosin & Rana 2004]. However, agent-based systems in general, and our approach with object agents in the spotlight in particular, provide an alternative approach to computer vision and could be just as well seen as a step towards considering *"the full potential of the "agent" and multi-agent paradigm"* (ibid.)

7 Acknowledgments

The authors would like to acknowledge the fruitful and patient discussions and interactions within the Cognitive Vision JRP, as well as the incentives, in particular by the coordinator, that all formed valuable contributions to the authoring of the present paper.

The original EMT design has been provided by Axel Pinz (FWF S9103-N04), the Matlab Code is the result of a student's project work by Raffaelo Tomasi.

The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry for Education, Science and Culture and by the Austrian Ministry for Transport, Innovation and Technology. This research was carried out in the context of the research project S9106-N04 of the FWF Austrian Science Fund.

References

[Bianchi & Rillo 1996] Bianchi R.A.C., Rillo A.H.R.C.: A Distributed Control Architecture for a Purposive Computer Vision System, *Proc. of IEEE Intl. Joint Symposia on Intelligence and Systems (IJSIS'96).*, pp. 288–294, 1996.

[Boissier & Demazeau 1992] Boissier O., Demazeau Y.: A Distributed Artifi cial Intelligence View on General Purpose Vision Systems, in Werner E., Demazeau Y., Decentralized AI 3: Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Kaiserslautern, Germany, August 5–7, North-Holland, Amsterdam/New York, pp. 311–330, 1992.

[Graf & Knoll 2000] Graf T., Knoll A.: A Multi-Agent System Architecture for Distributed Computer Vision, *International Journal of Artificial Intelligence Tools*, 9(2):305-319, 2000.

[Greaves et al. 1999] Greaves M., Holback H., Bradshaw J.: What Is a Conversation Policy?, in Bradshaw J. et al. (eds.), Workshop Notes, "Specifying and Implementing Conversation Policies", Third International Conference on Autonomous Agents (Agents '99), Seattle, WA, USA, 1999.

[Jennings 1996] Jennings N.R.: Coordination Techniques for Distributed Artifi cial Intelligence, in O'Hare G.M.P., Jennings N.R. (eds.), Foundations of Distributed Artifi cial Intelligence, Wiley, Chichester/London/New York, 187-210, 1996.

[Lesser 1998] V. Lesser: Reflections on the Nature of Multi-Agent Coordination and Its Implications for an Agent Architecture. *Autonomous Agents and Multi-Agent System.* **1(1)**:89–111, Kluwer Academic Publishers, 1998.

[Marr 1982] Marr D.: Vision, Freeman and Company, New York, 1982.

[Ponweiser et al. 2003] Ponweiser W., Umgeher G., Vincze M.: A Reusable Dynamic Framework for Cognitive Vision Systems, Proc. of the Workshop on Computer Vision System Control Architectures (VSCA 2003), 2003.

[Omicini & Ossowski 2003] A. Omicini, S. Ossowski: Objective versus Subjective Coordination in the Engineering of Agent Systems. in Klusch M. et al.(eds.): *Intelligent Information Agents: The AgentLink Perspective*. LNAI 2586 (State-of-the-Art Survey). Springer-Verlag Berlin Heidelberg, pp. 179–202, March 2003.

[Rosin & Rana 2004] Rosin P.L., Rana O.F. (eds.): *Pattern Recognition*, Special Issue on Agent Based Computer Vision, 37(4):627–855, April 2004.

[Wong & Sycara 2000] Wong H., Sycara K.: A Taxonomy of Middle-Agents for the Internet, in Durfee E. et al. (eds.), Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS-2000), July 10-12, 2000, Boston, MA, Institute of Electrical and Electronics Engineers, Inc., New York, NY, pp.465-466, 2000.

[Wooldridge 2002] Wooldridge M.: An Introduction to Multiagent Systems. John Wiley & Sons, Chichester, England, 2002.