Programming Behaviors With Local Perception and Smart Objects: An Approach to Solve Autonomous Agents Tasks

LUIZ M. G. GONÇALVES[1], MARCELO KALLMANN[2] AND DANIEL THALMANN[2]

[1]Instituto de Computação
Universidade Estadual de Campinas
CP 6172, 13083-970, Campinas, SP, Brasil
lmarcos@ic.unicamp.br
[2]Computer Graphics Lab (LIG)
Swiss Federal Institute of Technology (EPFL)
CH-1015 - Lausanne, Switzerland
{marcelo.kallmann,daniel.thalmann}@epfl.ch

Abstract. We propose an integrated framework in which local perception and close manipulation skills are used in conjunction with a high-level behavioral interface based on a "smart object" paradigm as support for a virtual agent to perform autonomous tasks. In our model, virtual "smart objects" encapsulate information about possible interactions between the agent and its environment, including sub-tasks defined by scripts that the agent can perform. We use the information provided by low-level sensing mechanisms to construct a set of local, perceptual features, with which to categorize at run-time possible target objects. Once objects are activated, based on their interactivity information and on the current task script, the agent can reparametrize its behavior, according to its mission goal defined in a global plan script. A challenging problem solved here is the construction (abstraction) of the mechanism to link individual perceptions to actions. As a practical result virtual agents are capable of acting with more autonomy, enhancing their performance.

1 Introduction

We introduce techniques that can be used by a virtual agent to perform tasks in a more autonomous way, based on possible interactions with the objects in its environment. We provide an interface to derive virtual models for each object type, which contains, besides its geometry, information about possible interactions with the agent. Such information contains, for example, a set of previously defined simple scripts that the agent can perform according to the object type and interaction capabilities (this has been called the smart object approach [1]). We embed in this model low-level mechanisms used for local perception and close manipulation, approximating our agent to realistic models (as for example, robotic platforms or humans). The perception mechanism is mainly used to provide perceptual data. Meaning features are abstracted, which can be used to define an object category. This index is used to retrieve the virtual model of the object, consequently its interactivity information, allowing the agent to perform actions that contribute to the task goal accomplishment. The agent is driven by a global behavior, also defined in a script-like language that tells the mission plan to the agent. In this approach, the agent system does not need to keep information about the several tasks nor about the objects or the environment itself. The concepts of mission planning, plan decomposition in tasks, and tasks achievement (see [4]) are substantially simplified. The knowledge about interactions with the scene objects are retrieved from the objects representation, once the agent approaches them in order to perform a given task.

Our complete definition and representation of interactive objects is based on the description of interaction features: parts, movements, graspable sites, functionality, etc. In particular, interaction plans for each possible agent-object interaction are defined, detailing all primitive actions that need to be taken by both the object and the agent, in a synchronized way. Objects defined with such interaction information have been called smart objects [1]. We present some experiments using a human agent based simulation environment and also a simulated robot, both with the built-in capability to simulate agent-object interactions, providing an automatic agent control for interactions with the smart objects. The agent common environment (ACE) [2] has been used to validate modeled interactions on a simulated virtual environment. ACE incorporates many new solutions regarding the control of interactive virtual environments, and has been used as a system platform for research in behavioral animation. We have created and tested grasping interactions with primitive objects in ACE, coordinated by a local perception mechanism reused from a simulated robot environment.

Concerning our robot simulations, we combine pure reactive plans with a script like plan, choosing actions mainly based on current perceptions of the world at different positional and temporal scales rather than by planning over previously given geometric models as in traditional simulation techniques. Also, as we use reduced and abstracted information obtained from a simplified world representation, our system performs fewer computations and substantially improves its performance at the cost of less adaptability when facing non predicted situations. As a result, the mechanisms developed here provide real-time feedback to different stimuli type. For these reasons, this system architecture is particularly relevant to agent systems as robots, but also for virtual agents.

As a result, we present in this work an integration of a local perception module derived from a simulated robotics system with a graphical simulation environment where a virtual human is able to grasp simple objects. In our case, low level interaction information is defined and encapsulated per object within their description. We show examples of the object interactions introduced in this work in different grasping applications, and the results obtained are shown and discussed.

2 Background and related work

To implement an artificial agent system one can start from the development of individual and basic skills including perception and basic (low-level) manipulation of the agent resources, like the path planning and moving of an arm toward a given target (e.g. reaching and/or grasping). Higher levels can use these basic tools as support in order to achieve tasks, taking right decisions as answers to environmental stimuli. If we treat these issues separately, the set of low-level skills or perceptual abilities will eventually not agree with the necessities of the high-level operating processes. Or, on the opposite, it may be necessary strong adaptation for the implementation of the high-level mission/tasks control system. We propose in this work to use the "smart object" approach, in which some perception and high-level skills can be treated together as agent-object interactions, in order to minimize the above implementation problems.

Different applications on the computer animation and simulation field face the problem of animating agent-object interactions. Such applications encompass several domains, as for example: virtual autonomous agents in virtual environments, human factors analysis, training, education, prototyping, and simulation-based design. A good overview of such areas is presented by Badler [5]. As an application example, an interesting system is proposed by Johnson [6], where a virtual human agent teaches users how to correctly operate machines in many situations in an interactive application.

Commonly, simulation systems approach agent-object interactions by programming them specifically for each case. Such approach is simple and direct, but does not solve the problem for a wide range of cases. Another approach is to use recognition, planning, reasoning and learning techniques in order to decide and determine the many manipulation variables during an agent-object interaction. The agent's knowledge is then used to solve all possible interactions with an object. Moreover, such approach should also address the problem of interaction with more complex machines with some internal functionality, in which case information regarding the object functionality must be provided.

Agent-object interaction techniques were first specifically addressed in a simulator based on natural language instructions using an object specific reasoning module [7]. Our smart object description is more complex, encapsulating interaction plans, allowing to synchronize movements of object parts with the agent's hand, and to model the functionality of objects as state machines.

Not enough attention has been addressed to solve general agent-object interaction issues, including robotic agents. Most of the concerns are related to sub-problems, as for the specific problem of grasping. For instance, from the robotics area, a classification of hand configurations for grasping is proposed by Cutkosky in [9]. Also, Huang [8] proposes an algorithm for the automatic selection of hand shapes for grasping.

Concerning the model used for perception in real robots, we are inspired by several new approaches that have been suggested using multi-feature extraction as basis for cognitive processes [10, 11, 12]. In previous contribution [13] we provided a *working* model for low-level perception and close manipulation control using a real stereo head robot. Our model uses a practical set of features extracted from real-time sequences of stereo images, including static (spatial) and temporal properties, and also stereo disparity features. We have developed a pure reactive system, treating low-level manipulation and control of the robot resources based on local perceptions of the environment.

The idea of using vision-based sensors for virtual human agents simulations is not new. The first work to address this problem [18] uses a rendered image buffer of the virtual scene but with colors coding objects ids, thus simplifying the recognition phase. In our low-level perception module we use stereo vision results projected into a simulated retina, being thus much more realistic and general. However, in this work we use the simplification of working with an unidimensional retina. Once the perception selects the correct object to be manipulated (in order to achieve a given high level task), the used low-level manipulation information is retrieved from the smart object representation.

3 Smart objects

Consider the simple example of opening a door: the rotation movement of the door must be provided a priori. Following a top-down AI approach, all other actions should be planned by the agent's knowledge: walking to reach the door, searching for the knob, deciding which hand to use, moving body limbs to reach the knob, deciding which hand posture to use, grasping, turning the knob, and finally opening the door. This simple example illustrates how complex it can be to perform a simple agent-object interaction task. To overcome such difficulties, we use a bottom-up approach, that is, we include within the object description more useful information than only intrinsic object properties. By using feature modeling concepts, we identify all types of interaction features in a given object and include them as part of the object description. Our graphical interface program (Figure 1) allows the user to interactively specify all different features in the object, defining its functionality, its available interactions, etc. This smart object modeler application is called "SOMOD".

The adjective smart has been already used in different contexts. For instance, for interactive spaces instrumented with cameras and microphones to perform audio-visual interpretation of human users [17]. This ability of interpretation made them smart spaces. In the scope of this work, an object is called smart when it has the ability to describe in details its functionality and its possible interactions, by describing all needed low-level manipulation actions. A smart object does have reactive behaviors, but more than that, it is also able to provide the expected behaviors for its "users". In the case of our agents, reaction to the environment stimuli (perception) is programmed in order to correctly select which object to interact with, then to perform the required low level interaction, local interaction information stored within the smart object is used.

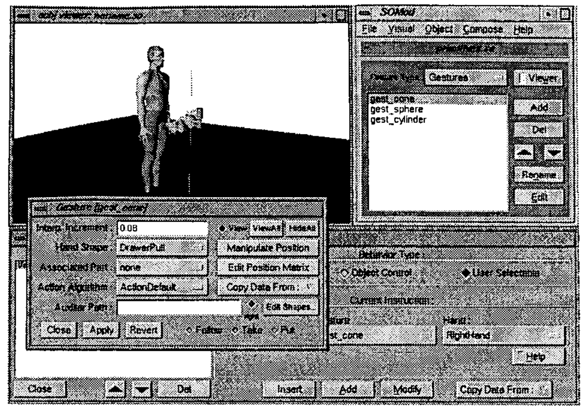

Figure 1: Interactive graphical interface for modeling objects and their interactions.

Different applications can retrieve useful information from a smart object to accomplish desired interaction tasks. The main idea is to provide smart objects with a maximum of information to attend different possible applications for the object. A parallel with the object oriented programming paradigm can be made, in the sense that each object encapsulates data and provides methods for data access. Applications using smart objects will have their own specific smart object reasoning module, in order to retrieve only the applicable object features for their specific needs.

4 Our Control Schema

Figure 2 shows the main aspects of the control schema that manages the low-level simulation step in our robot model, derived from [14]. Briefly, the agent uses sensory information perceived (calculated) plus its pose and functional state to define a set of perceptual features. This feature set will input to a classifier producing an effective categorization (a virtual index) for the object (or region) in focus. We have currently implemented two types of classifiers: a) a multi-layer perceptron trained with a backpropagation algorithm; b) a self organizing map [15]. This index allows our agent to retrieve the set of associated interaction information in the the correspondent smart object. So, the possible interactive actions can be retrieved and the agent effectively moved to perform the physical action. Also, note that if the current script goal is reached, other tasks can be chosen. A motion effectively brings the agent to a new pose and eventually puts a set of new information about other smart objects in the *dexterous workspace* (manipulation space). Once the new sensory information is acquired (simulated), the process can be re-started (feature extraction).

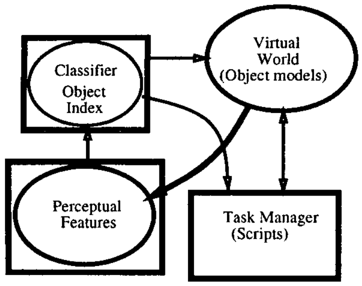

Figure 2: Control schema.

We remark that this low-level operating loop follows a control theory approach, what guarantees stability and global convergence of the agent resources (controllers) in the achievement of an action. At each time step, a small movement is performed followed by an update in the perceptual information of the agent. For example, if another smart object, with interaction attributes that are more important to the task goal acomplishment than the one being manipulated reaches the *dexterous workspace* of the agent, the first object can be posted somewhere (its scripts are disregarded) and other interaction scripts with the second object can start, considering the current situation (positioning) of the agent. This produces smooth and differentiable motion and also allows the agent to eventually reparametrize its task goal on-line during the execution of an action, taking into account the changes in perception. This approach is somewhat reactive, choosing actions based on perceptions of the world at rather than by using a geometric model as in traditional planning techniques. Also, as time is a critical parameter for real-time agent applications, by using this approach we guarantee that all computations necessary to perform a given step of motion are computed during the time interval given by the clock rate.

5 Modeling interactive object features

Feature modeling is an expanding topic in the engineering field [16]. The word feature may have several meanings, and a general definition is simply "a region of interest on the surface of a part". The main difficulty here is that, in trying to be general enough to cover all reasonable possibilities for a feature, such a definition fails to clarify things sufficiently to give a good mental picture.

5.1 Interaction features

In the smart object description, a new class of features for simulation purposes is used: interaction features. In such context, a more precise idea of a feature can be given as follows: all parts, movements and descriptions of an object that have some important role when interacting with an animated agent. For example, not only buttons, drawers and doors are considered as interaction features in an object, but also their movements, purposes, manipulation details, etc. Interaction features can be grouped in four different classes.

Intrinsic object properties are properties that are part of the object design: the movement description of its moving parts, physical properties such as weight and center of mass, and also a text description for identifying general objects purpose and the design intent.

Interaction information are useful to aid an agent to perform each possible interaction with the object: the identification of interaction parts (like a knob or a button), specific manipulation information (hand shape, approach direction), suitable agent positioning, description of object movements that affect the agent's position (as for a lift), etc.

Object behavior are used to describe the reaction of the object for each performed interaction. An object can have various different behaviors, which may or may not be available, depending on its state. For example, a printer object will have the "print" behavior available only if its internal state variable "power on" is true. Describing object's behaviors is the same as defining the overall object functionality.

Expected agent behavior is associated with each object behavior. It is useful to have a description of some expected agent behaviors in order to accomplish the interaction. For example, before opening a drawer, the agent is expected to be in a suitable position so that the drawer will not collide with the agent when opening. Such suitable position is then proposed to the agent during the interaction.

This classification covers the needed interaction features to provide common agent-object interactions. Still, many design choices appear when trying to specify in details each needed interaction feature. The most difficult features to specify are those relative to behaviors. Behavioral features are herein specified using pre-defined plans composed with primitive behavioral instructions (scripts). This model has shown to be the most straightforward approach because then, to perform an interaction, the agent will only need to "know" how to interpret such interaction plans in a straightforward way.

In the smart object description, a total of 8 interaction features were identified, with the intention to make the most simple classification possible. These interaction features are described in Table 1.

Feature Class	Data Contained
Descriptions	Object property
Parts	Object Property
Actions	Object Property
Commands	Interaction Information
Positions	Interaction Information
Gestures	Interaction Information
Variables	Object behavior
Behaviors	Object/agent (scripts)

Table 1: Types of interaction features.

5.2 Interpreting Interaction Features

Once a smart object is modeled, the agent system will be able to load it and to animate it with physical actions. To do that, the agent system will need to implement a smart object reasoning module, that will correctly interpret the behavioral plans (scripts) to perform interactions. For a complete description of such concepts see [3]. In the scope of this work, in order to demonstrate the integration of the local perception module, we have used mainly the interaction features related to define grasping actions with some primitive models.

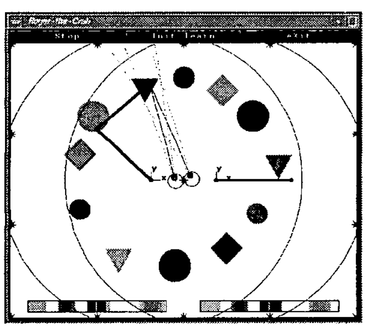

Figure 3: Simulated robot.

Figure 4: Examples of 1D simulated retinas.

6 Mapping objects characteristics into local perception

For the purposes of understanding the integrated framework, we describe how a robot simulator operates [14], that is, we show how local perception and close manipulation can be combined in our agent system. The simulated platform shown in Figure 3 has several integrated controllers (small programs or scripts) including control of pan (like a neck horizontal rotation) and vergence (eyes convergence/divergence) movements for its head and control of joints for each of its two arms. By developing an agent like the one shown in Figure 3, our main goal is to provide a virtual device with which to develop computational models for robots, also to study the relationship between vision and touch sensory systems. The construction of such being would allow the definition of new approaches to robotics based on simulation, thus decreasing operational costs and also assuring a safe management of the resources. The geometrical information (shape) and texture of a given virtual object model can be directly mapped into visual information for our agent. Note that here we start to build a more realistic virtual agent, with built-in perception capabilities. A visually perceived scene can be seen in Figure 4. The visual and haptics servoing is done in the same way as close manipulation, that is, in a closed loop. So, this agent can use perception provided in its retinas (it might also be in a haptics storage area) in real time, for example, to help disambiguating objects. Perception information can be easily mapped into the working configuration space of this simulator because it regards a topological relation with the object positions. In this way, we get an agent that can work in a continuous and more complex world, besides the virtual environment from which the sensory information was simulated is discrete and simple. We conjecture that a virtual agent can have more realistic behaviors by being constructed in this way. Another motivation that can be used is that its application to real robot systems is straightforward task. Finally, one of the main advantages is that problems as agent entering inside an object, obstacle avoidance, can be avoided here, since the agent can infer from visual information the position of objects in its path.

6.1 Constructing perception

Figure 3 shows a situation of local perception and possible close manipulation of objects in a room. Coincidentally, the virtual environment used here is represented exactly in the same way as in the SOMOD application, that is, as a list of objects, ordered by its distance to the agent position. This sorting is very fast because it works in a reasonable set of objects. Each object has also the history of interaction with the agent. For example, a sphere would have a script that gives a different configuration for the agent to perform a grasping than a cube. We embedded this model in the resulting architecture in a straight forward way. Visual sensing is directly simulated from the objects contained in the simplified environment definition. For example, for each one of the pixels in the agent's retinas, an intensity value can be calculated in function of the radiance of the environment (object) patch corresponding to it by using a simple Phong illumination model. A Gaussian noise process can simulate acquiring errors, asserting a more natural retinal image. Haptics (including proprioceptive and tactile) simulation can be done based on the arbitrary value attributed to each object mass. Also, based on local movements, the current positioning of the agent (odometry) can be updated and the objects list re-sorted. We assume that, in a given instant, only a small sub-set of all objects present in the virtual environment are close enough to the agent to result in interactions. In this case, the simulation process does not need to traverse the whole list of objects (remember they are sorted by distance).

This visual, haptics, and positioning simulation basically provides local sensed information to be used by the simulated agent to construct its perception. We yet use simple image processing operations to reduce and abstract this input data, resulting in a *multi-resolution* (MR) retina representation for data reduction. The classifier uses a set of *multi-features* over the MR images to provide feature abstraction for categorization. The result is an object index, with which all other information relative to an identified object (mainly, interaction features) can be retrieved.

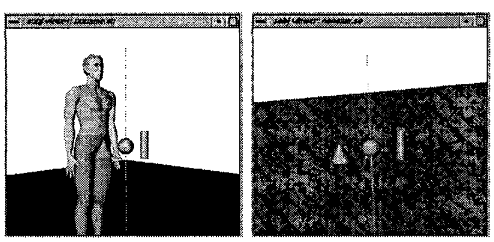

Figure 5: Defining a grasping interaction with different object types.

7 Experiments and demonstrations

In a practical experiment, we defined scripts for the best grasping to be executed by our agent interactively, according to the object types seen in Figure 5: a cone, a cylinder and a sphere.

Figure 6: Modeling objects interaction features (grasping).

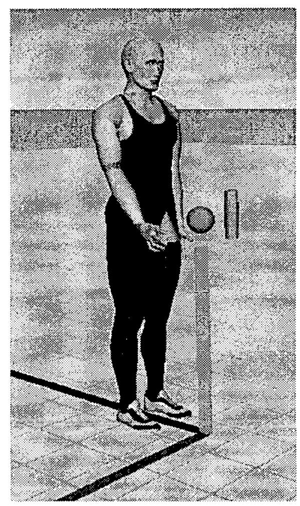

Figure 7: Object grasping.

In the interaction feature modeling phase, proper hand posture is chosen by an operator as seen in Figure 7. This set of objects (with its interaction features) is stored in our virtual environment. Then, by using the simulated perception, the agent is able to detect the object index and to retrieve this information to be used by the subsequent grasping, shown in the sequence of Figure 8.

We remark that the information generated by this perception simulation is a realistic retina like image, very close to a natural image of the object, and that the tools used for categorization were tested by a real robot platform [13] also applied to pattern categorization.

To illustrate how perception and the low-level resource manipulation skills operate in real-time in the simulated robot environment, we include a sequence (Figure 9), in which our simulator reach/grasp an object (a chair) close to a lit. This task sequence is performed by means of using a low-level script triggered by the object (a chair), which was out of place (interaction feature). So, after the positive identification provided by the system classifier, the interaction information encoded in the virtual version of the smart object chair can be used to effectively reach and grasp it.

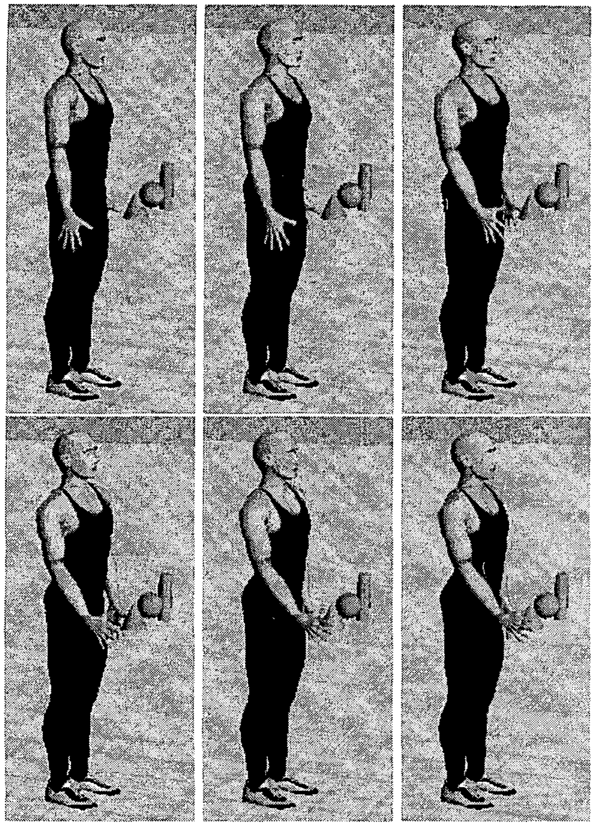

Figure 8: Virtual Human Agent performing a grasping task.

8 Conclusion and future work

We have proposed a task achievement approach using a high-level behavioral architecture based on smart objects and local perceptions to define which object to choose, consequently driving the agent actions. This approach allows agents to produce on-line its low-level actions or tasks necessary to accomplish a given mission. This avoids the drawback of encoding a complex plan (tasks) or other information about the environment/objects that are typical in the "mission planning" schemes. This capability is interesting and we conjecture that it can be adapted in a multi-agent context, taking into account the other agents global scripts, external changes in the environment detected by using perception, and internal predicted changes

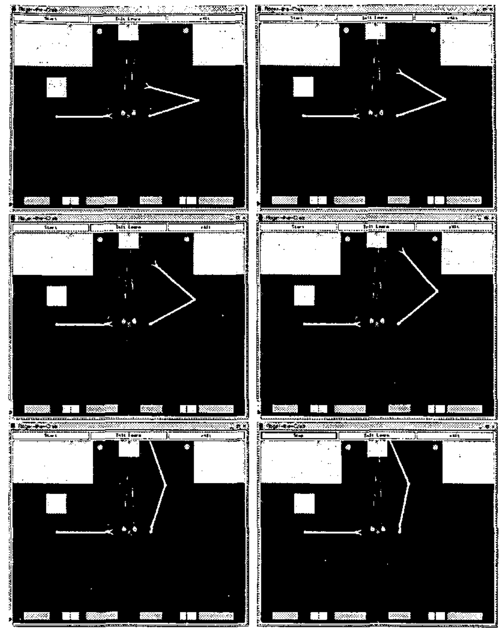

Figure 9: Simulated robot performing a close manipulation (reaching).

produced by a close resource manipulation. Note that one can also use interactions between several objects to re-edit the agents global scripts.

The main reason of using such approach is its possible application in a real robot system, without strong adaptations. The extraction of meaning perceptual features and the definition of world states from them is a challenging problem, partially solved in this paper. We have abstracted this by using a previously acquired combination of featured models that represent each virtual object, whose interactive features are defined empirically in the smart objects approach. In a more autonomous paradigm, an agent (or robot) would define these interactive features and discriminatory capabilities by using a learning approach, interacting directly with its environment.

We plan to improve this implementation by increasing the set of low-level actions that the agent/simulated robot are able to perform (interactions with objects). This work can be yet improved by using a learning approach to derive the possible interactions with the objects. As an example of an interesting task planned in this context, at starting the agent system has no knowledge about the environment (its world map is empty) nor the system knows about object patterns (system memory has no feature models or pattern representations). The goal is to incrementally construct a shared map of the environment, learning perceptual patterns, and geometry/topology of the objects. After such a world representation is constructed, agents can perform other more specific tasks. A final possibility of future work is to put the software tools running in a real robot and to evaluate its interaction capabilities with real objects.

References

[1] M. Kallmann and D. Thalmann. A behavioral interface to simulate agent-object interactions in real-time. In *IEEE Computer Animation 99*, pages 138–146. IEEE Press, 1999.

[2] M. Kallmann, A. Monzani, C. Aangela, and D. Thalmann. Ace: A platform for the real time simulation of virtual human agents. In *EGCAS'00 - 11th Eurographics Workshop on Animation and Simulation*, Interlaken, Switzerland, 2000.

[3] M. Kallmann. Object Interaction in Real-Time Virtual Environments. *DSc Thesis*, Swiss Federal Institute of Technology, Lausanne, Switzerland, 2001.

[4] S. Botelho and R. Alami. Robots that cooperatively enhance their plans. In *Proc. of 5th International Symposium on Distributed Autonomous Robotic Systems (DARS 2000). Lecture Notes in Computer Science.* Springer Verlag, 2000.

[5] N. N. Badler. Virtual humans for animation, ergonomics, and simulation. In *IEEE Workshop on Non-Rigid and Articulated Motion*, Puerto Rico, June, 1997.

[6] W. Johnson and J. Rickel. Steve: An animated pedagogical agent for procedural training in virtual environments. In *SIGART Bulletin, ACM Press, 8(1-4), 16-21*, 1997.

[7] L. Levinson and N. Blader. How animated agents perform tasks: Connecting planning and manipulation throught object-specific reasoning. In *Working Notes from the Workshop on "Towards Physical Interaction and Manipulation", AAAI Spring Symposium*, 1994.

[8] Z. Huang, R. Boulic, N. Magnenat-Thalmann, and D. Thalmann. Virtual humans for animation, ergonomics, and simulation. In *Proceedings of Computer Graphics International, Leeds, UK*, UK, June, 1995.

[9] M. Cutkosky. On grasp choice, grasp models, and the design of hands for manufacturing tasks. In *IEEE Transactions on Robotics and Automation, 5(3), 269-279*, 1989.

[10] L. Itti, J. Braun, D. K. Lee, and C. Koch. A model of early visual processing. In *Advances in Neural Information Processing Systems*, pages 173–179, Cambridge, MA, 1998. The MIT Press.

[11] P. A. Viola. Complex feature recognition: A bayesian approach for learning to recognize objects. AI Memo 1591, Massachusetts Institute of Technology, November 1996.

[12] R. P. N. Rao and D. Ballard. An active vision architecture based on iconic representations. *Artificial Intelligence Magazine*, 78(1-2):461–505, October 1995.

[13] L. M. Garcia, R. A. Grupen, A. A. Oliveira, D. Wheeler, and A. Fagg. Tracing patterns and attention: Humanoid robot cognition. *The Intelligent Systems and their Applications*, 15(4):70–77, July/August 2000.

[14] L. M. G. Gonçalves, F. W. d. Silva, R. C. Farias, and R. A. Grupen. Towards an architecture for artificial animated creatures. In *Proc. of VI CGS/IEEE Computer Animation Conference*, Philadelphia, PA, May, 3-5 2000. IEEE Press.

[15] L. M. Garcia, C. Distante, and A. Anglani. Self-growing neural mechanisms for pattern categorization in robotics. In *In Proc. of International ICSC Congress on Intelligent Systems and Applications (ISA 2000)*, December, 12-15 2000.

[16] S. Parry-Barwick and A. Bowyer. Is the features interface ready? In *Directions in Geometric Computing, Ed. Martin R., Information Geometers Ltd, Cap. 4, 129-160*, UK, 1993.

[17] A. Pentland. Machine understanding of human action. In *7th International Forum on Frontier of Telecom Technology*, Tokyo, Japan, 1995.

[18] O. Renault, N. Magnenat-Thalmann, and D. Thalmann. A Vision Based Approach for Behavioral Animation. *The Journal of Visualization and Computer Animation*, 1(1), 18-21, 1990.