In Computer & Graphics, 26(6):887-897, December 2002

Defining Behaviors for Autonomous Agents Based on Local Perception and Smart Objects ⋆

Luiz Gonçalves [a,*] , Marcelo Kallmann [b] , Daniel Thalmann [b]

[a] *Departamento de Computação e Automacão*
Universidade Federal do Rio Grande do Norte
DCA-Centro de Tecnologia, 59.072-970, Natal, RN, Brasil

[b] *Computer Graphics Lab (LIG)*
Swiss Federal Institute of Technology (EPFL)
CH-1015 - Lausanne, Switzerland

Abstract

An integrated framework is proposed 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 virtual agents to perform autonomous tasks. In our model, virtual "smart objects" encapsulate information about possible interactions with agents, including sub-tasks defined by scripts that the agent can perform. We then use information provided by low-level sensing mechanisms (based on a simulated retina) to construct a set of local, perceptual features, with which to categorize at run-time possible target objects. Once an object is identified, the associated smart object representation can be retrieved and a predefined interaction might be selected if this is required by the current agent mission defined in a global plan script. A challenging problem solved here is the construction (abstraction) of a mechanism to link individual perceptions to actions, that can exhibit some human like behavior due to the used simulated retina as perception. As a practical result virtual agents are capable of acting with more autonomy, enhancing their performance.

Key words: Autonomous Behavior, Smart Objects, Local Perception.

1 Introduction

We introduce techniques that can be used by a virtual human agent to perform tasks in a more autonomous way, based on possible interactions with the objects in its environment. A rich description for interactive virtual objects is used including information about possible interactions with the human agent. Such information contains, for example, a set of simple scripts that the agent can perform according to the object type and interaction capabilities. We embed in this model low-level mechanisms used for local perception and close manipulation, approximating our agent to realistic models such as robotic platforms or humans. The perception mechanism identifies meaning features in order to categorize perceived objects. The categorization associates an index identifying an object class. This index is then used to retrieve the associated virtual model of the object, and 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 informa-

⋆ Based on "Programming Behaviors With Local Perception and Smart Objects: An Approach to Solve Autonomous Agents Tasks" by Luiz Gonçalves, Marcelo Kallmann, and Daniel Thalmann which appeared in Proceedings of SIBGRAPI 2001 XIV Brazilian Symposium on Computer Graphics and Image Processing, 2001 IEEE.

* Corresponding author. DCA-CT, Universidade Federal do Rio Grande do Norte, CEP 59.072-970, Natal, RN, Brasil, Phone: +55 84 215 3771, Fax: +55 84 215 3738

Email addresses: `lmarcos@dca.ufrn.br` (Luiz Gonçalves), `marcelo.kallmann@epfl.ch` (Marcelo Kallmann), `daniel.thalmann@epfl.ch` (Daniel Thalmann).

URLs: `http://www.dca.ufrn.br/~lmarcos` (Luiz Gonçalves), `http://ligwww.epfl.ch/kallmann.html` (Marcelo Kallmann), `http://ligwww.epfl.ch/thalmann.html` (Daniel Thalmann).

tion about the several tasks nor about the objects or the environment itself. The concepts of mission planning, plan decomposition in tasks, and task achievement (see [4]) are substantially simplified. The knowledge about interactions with the scene objects are retrieved from the object representations 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 [15]. 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. The Agent Common Environment (ACE) [14] 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. In the current work we have created and tested grasping interactions with primitive objects in ACE, coordinated by the 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 of scenes 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. The mechanisms developed here provide real-time feedback to different stimuli type. For these reasons, this system architecture is not only relevant to robotic systems, but also for virtual, computer animated agents.

As a main 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 identify (using the perception module) and grasp simple objects. The low level interaction information (as the used grasp posture) 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

In the context of this work, an artificially animated agent could be either a robotic platform or a computer animated agent (as an avatar or other computer simulated device). 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 motion planning for moving an arm toward a given target (e.g. for reaching and/or grasping). Higher levels of the agent hierarchy 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 here to use "smart objects", in which some perception and high-level skills can be treated together as predefined 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 factor analysis, training, education, prototyping, and simulation-based design. A good overview of such areas is presented by Badler [1]. As an application example, an interesting system is proposed by Johnson [12], where a virtual human agent teaches users how to correctly operate industrial machines.

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 AI models as 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, in which case information regarding the object functionality must also be provided.

Agent-object interaction techniques were first

specifically addressed in a simulator based on natural language instructions using an object specific reasoning module [16]. 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, a classification of hand configurations for grasping is proposed by Cutkosky in [5]. Such classifications can then be used by knowledge-based grasping methods [21] [10] for the automatic selection of hand shapes and for the animation of grasping actions.

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 [11,24,18]. In previous contribution [8] 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 for depth representation. We have developed a pure reactive system, treating low-level manipulation and control of robot resources based on local perceptions of the environment.

The idea of using vision-based sensors for virtual human agents simulations is not new [19] [23] [3]. The first work addressing this problem for virtual human agents [19] uses a rendered image buffer of the virtual scene, with colors coding objects ids, thus simplifying the recognition phase. In our low-level perception module we use simulated stereo vision, that is, the scene objects are projected into simulated retinas, one for each eye, being thus much more realistic and general. By using simulated stereo retinas we are able to really recognize objects, knowing how close they are to each known object model by performing any pattern matching algorithm. In this work we currently use a simple approach based on the Back-Propagation Neural Net (BPNN) [20,22,25] for pattern matching. Moreover, as we operate on the shape of the object, we are able to detect parts of a large object matching other known objects. However, we do not address in this work more complex problems that may occur, as for instance, due to object occlusion and ambiguities.

We make in this work the simplification of using an unidimensional retina. This is enough for the simulated examples presented here for detecting geometric primitives to grasp. Once the perception identifies the correct object to be manipulated (in order to follow a given global mission), the used low-level manipulation information is retrieved from the smart object representation and the human agent can perform the action proposed by the virtual object. We note that for more complex recognition tasks a bidimensional retina can be used, however demanding more complex and time consuming algorithms to deal with the retina images. We are currently working in the development of such a 2D model, using the OpenGL library.

3 Smart objects

Consider the simple example of opening a door: the rotation movement of the door must be provided a priory. 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 is called "SOMOD".

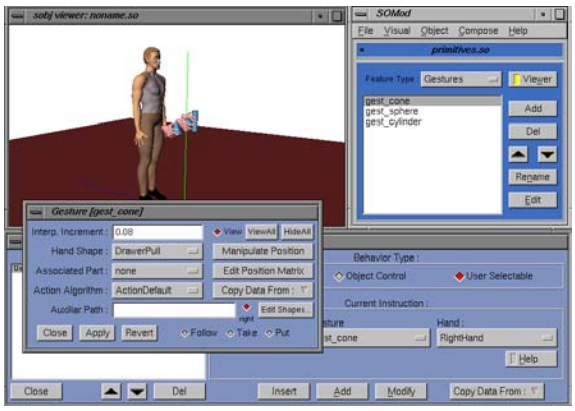

Fig. 1. Interactive graphical interface for modeling objects and their interactions.

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 may have reactive behaviors, but more than that, it is also able to provide the expected behaviors for its "users". In the case of our robot agent, the agent 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.

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 human agent model, derived from [9]. Briefly, the human agent uses perceived (simulated) sensory information (calculated by using the geometric models of some of the objects present in its world model) plus its pose and functional state to define a set of perceptual features. This feature set forms the input to a classifier producing an effective categorization (an index) for the object in focus. As stated earlier, we are using here a BPNN approach for the categorization step, which has produced good results. We have also implemented other two types of classifiers in previous works operating in two (simulated and real) robotic platforms [9,6,7]: a) the multi-layer perceptron trained with a backpropagation algorithm; and b) a Self Organizing Map (SOM) [6].

We emphasize that any classifier model could be used here, so that in practice the output of the classifier must be an index for the smart object model. This index allows our agent to retrieve the set of associated interaction information in the corresponding smart object. Then, 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 work-space* (manipulation space) of our human agent. Once again, new sensory information is acquired (simulated) and the process can be re-started (feature extraction), that means, other object may be activated depending on the agent movement and so on.

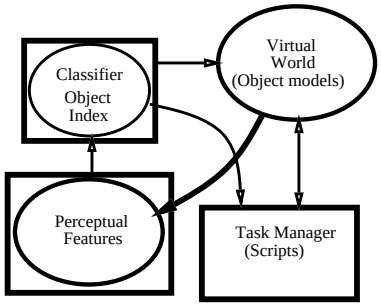

Fig. 2. Control schema.

We remark that this low-level operating loop follows a control theory approach, what guarantees stability and global convergence of the agent resources (if we think in these as being 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 accomplishment than the one being manipulated, reaches the *dexterous work-space* 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 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. The main advantage of using such approach is its immediate application to robotics platforms, without strong adaptations.

5 Modeling interactive object features

Feature modeling is an expanding topic in the engineering field [2]. 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 an interaction 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 is 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 behaviors 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) provided by SOMOD for performing interactions. 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. However, the same architecture can be used for more complex interactions being suitable for the manipulation of complex machines.

6 Mapping object characteristics into local perception

For the purposes of understanding the integrated framework, we describe how a robot simulator [9] operates, that is, we show how local perception and close manipulation can be combined in our human agent system. The simulated platform shown in Figure 3 has several integrated controllers (small programs or scripts that control each agent part) including control of "pan" and "vergence" movements (for its eyes) and control of joints for each of its two arms.

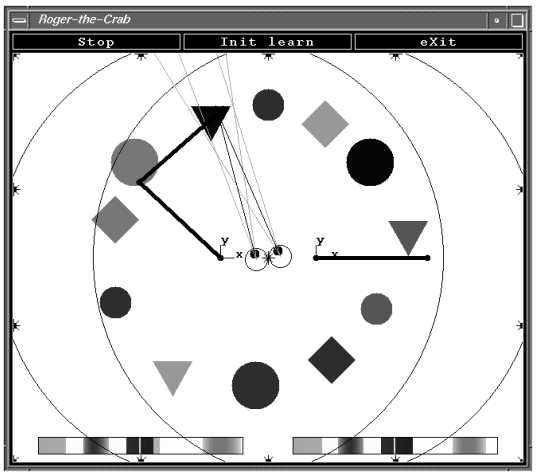

Fig. 3. The simulated robot and its world. The whole window represents a top view of the environment with the robot positioned in its center. The triangles, circles, and squares are (prismatic) objects positioned in the environment. In the example, the robot has its attention window, represented by the two conic regions leaving the eyes, in the triangle and it is manipulating this triangle (grasping/touching it) with one of its arms (left). The other arm (right) is retired in the initial positioning. The retinas, enlarged in Figure 4, can be seen at the bottom in small size. Besides the central triangle, the viewing angle also includes the huge circle, to the left of the triangle, and the small circle and part of the square, to the right. Note the images are inverted.

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 in humans. Also, 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 (color) of a given virtual object model can be used to calculate simulated visual information for our agent. In the old simulator version, we used X-lib tools (Motif) and a hand-coded module to perform all calculations necessary for this simulation. This produces two unidimensional retinas as seen in Figure 4 that are enough for the purpose of the current work.

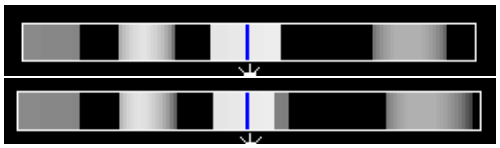

Fig. 4. Examples of 1D simulated retinas. Top figure represents left eye retina and bottom is for right eye. The eyes are verged in the triangle, as seen in the environment represented in Figure 3. The projection of the objects to construct each retina is done by using a simple Phong illumination model.

In another related work, we are currently improving this visual data simulation so we can have 2D retinas. Each retina is represented by an image captured (calculated) from the environment representation by using a Z-buffer (OpenGL). This new simulator interface is shown in Figure 5 and 2D examples of its retinas are seen in Figure 6, just before and after a saccadic motion. In this new version, we consider using hardware acceleration to speed up the construction of the images forming the sensory input. As already mentioned, in the current work we use 1D retinas mainly for the sake of speed and easy of computation. The extensions to 2d are still being evaluated. The visual and haptics servoing is done in the same way as the close manipulation of the resources, that is, as a closed loop. So, at each time step, the virtual agent calculates the above simulation and puts the result in the agent retinal images (visual buffer). As a result, the virtual agent can use the perceptual information provided in its retinas (it might also be in a haptics storage area) in real time, for example, to help disambiguate objects. Note that here we start to build a more realistic virtual human agent, with built-in perception capabilities. Moreover, this sensory information can be easily mapped into the working configuration space of this simulator because it regards a topological relation with the object positions.

As a result, we get an agent that can work in a continuous and more complex world, even if 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 a straightforward task. Finally, obstacle avoidance algorithms can make use of the perception module, since the agent can infer from stereo vision information the position of objects in its path.

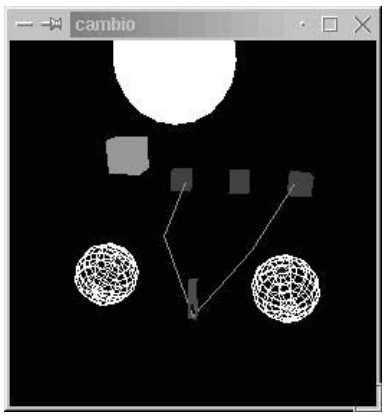

Fig. 5. Visual and haptics perception simulator for a 3D environment being constructed. The balls represent the eyes (or the cameras in a robot). The arms are represented by the two polylines leaving the central point in between the eyes. They will be further positioned far of each other, of course.

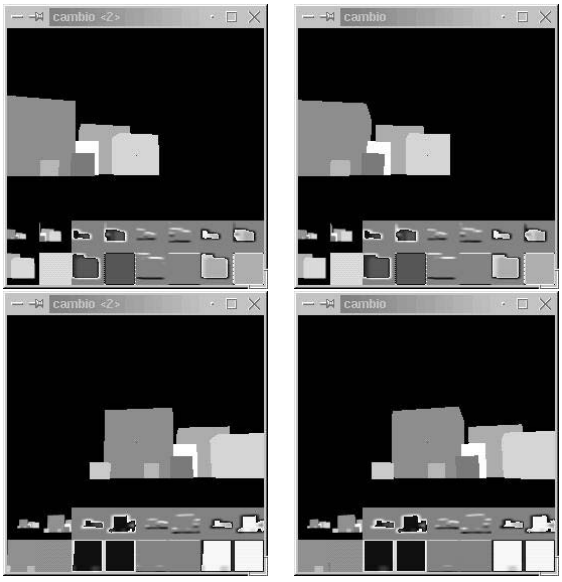

Fig. 6. Examples of 2D simulated retinas. Top figure shows the 2D simulator with both (left and right) retinas verged on the same object (the box at image centers). Bottom images show the retinas after a saccadic motion performed from one box to another.

6.1 Constructing perception

Figure 3 shows a situation of local perception and possible close manipulation of objects in a room. Coincidentally, the virtual model of the environment used to produce that scene is represented exactly in the same way as in the SOMOD modeler, that is, as a list of objects, ordered by its distance to the virtual agent position (center of its eyes). This sorting is very fast because it works in a reasonable (small) set of objects. That is, at each time instant, only a small set of objects is close enough to the virtual agent position in order to result in interactivity (of course for simply represented worlds). So the amount of computations necessary for updating this list is small. We encapsulate the module for local perception described in the previous section in the SOMOD architecture in a straightforward way. This encapsulation basically includes script lines (Python) to run some "C" modules (the perception modules). Basically, this integration is as depicted below.

(1) the "Simulate-Sensors" module runs to calculate the simulated sensory information (using the list of objects and agent pose/positioning) and return (filling in the memory data structures) the simulated retina/haptics information;
(2) the classifier module (BPNN memory) runs, receiving as parameter the visual, haptics, and odometry (human agent pose/positioning) data, stored in a shared memory; the return of this module is an index for an object identified by this module;
(3) the agent animation control module can start at any time an object script which will effectively move the agent resources, after querying the perception module about an object index;
(4) a module runs for updating the perceived world (updating and/or resorting object list) based on the current agent positioning and movements performed in the previous step.

So, as a first step, the system uses the list of objects provided by SOMOD as input to produce simulated (1D) retinas. That is, the visual sensing is directly simulated from the simplified environment definition provided by SOMOD. An intensity value is calculated for each one of the pixels in the human agent's retinas, in function of the radiance of the environment (object) patch corresponding to it, by using a simple Phong illumination model. A Gaussian noise process simulates acquiring errors, asserting a more natural retinal image. Haptics (including proprioceptive and tactile) simulation is done based on the arbitrary value attributed to each object mass. Also, based on local movements, the current positioning of the human agent (odometry) can be updated and, consequently, the objects list updated/re-sorted.

This visual, haptics, and positioning simulation basically provides local sensed information to be used by the simulated human agent to construct its perception. We yet use simple image processing operations to reduce the amount of input data and to abstract features. The result of data reduction (first phase) is a *multi-resolution* (MR) data representation. A set of *multi-features* is calculated over the MR data to provide feature abstraction. This set is used as input for categorization (the classi-

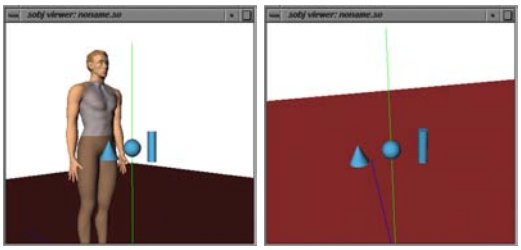

Fig. 7. Defining a grasping interaction for different objects.

fier). The result is an object index, with which all other information relative to an identified object (mainly, interaction features) can be retrieved.

We remember that each object has also the history of interaction with the human agent. For example, a sphere would have a script that gives a different configuration for the agent to perform a grasping than the cube. As the output of the perception module is just the object index, that will be used to start the grasping action which is stored in the virtual model of each object. And, as stated above, we assume that, in a given instant, only a small subset of objects present in the virtual environment are close enough to the human 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).

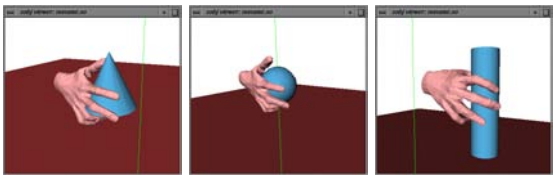

Fig. 8. Modeling objects interaction features (grasping).

6.2 Practical implementation issues

In practice, the perception module was integrated to the agent animation control module by defining an interface based on shared memory and tcp/ip messages. In this way the animation control can run in a dedicated computer querying another computer dedicated to the perception processing, if necessary. Note that both modules can run apparently in parallel. Currently the computation time required by the pattern matching of the perception processing is not so costly to justify a separation into more than one computer mainly because the perception is currently 1D based, what greatly simplifies the computation. The choice for this distributed processing interface was taken due to two main reasons: first to simplify the connection of the modules, which were originally developed targeting different platforms (SGIs vs. SUNs/PCs), and second to ensure the scalability of the architecture, for a future efficient 2D perception processing.

The process communication is based on two main steps: in a first initialization step, all declared smart objects which were defined as possible to be identified by the perception are passed to the perception module. In this phase, the geometry of the primitives are processed and their 1D retinal pattern are generated and classified (for some different points of view), and an id number is returned to the simulation computer. These are considered to be the perceptible primitives of the environment, so the BP memory (classifier) is previously trained with a current set of known objects. This initial phase can be done off-line since the set of smart objects and their geometry is previously known. For this reason, this phase is not shown in the control loop depicted in Figure 2. Note that other geometries composing the environment might be declared also to the perception module so that we might be able to detect all shapes in the environment which are similar to the declared primitives. This is the main advantage of our method: to detect the shape of declared primitives anywhere inside the environment, even if the primitive is just a part of a shape, e.g., a cylinder primitive could be detected as part of a door handle model.

Later in run time, the animation module sends the current position of the agent, its viewing direction and field of view to receive back from the perception module the ids of the primitive objects being perceived and other two information: at which position in the environment, and if it is a full object or part of an object. In our current case study, the returned object id corresponds to an smart object that will control the animation of the agent grasping an object with a given shape.

7 Experiments and demonstrations

In a practical experiment, we defined scripts for the best grasping to be executed by our human agent interactively, according to the object types seen in Figure 7: a cone, a cylinder and a sphere. In the interaction feature modeling phase, proper hand posture is chosen by an operator as seen in Figure 9. This set of objects (with its interaction features) is stored in our virtual environment. The system classifier (BPNN memory) is trained off-line with the set of object. Then, at running time, by using the simulated perception, the human 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 10.

We remark that the information generated by

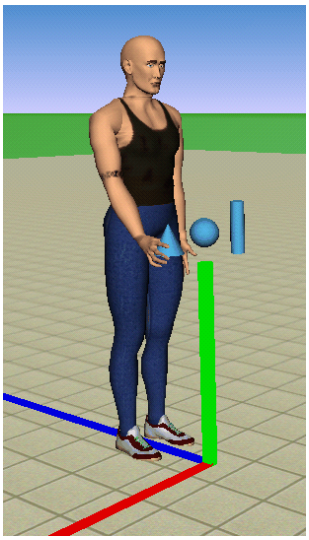

Fig. 9. Object grasping.

this sensory (perception) simulation is a realistic, retina-like image very close to a natural image of the object. Also, the tools used for classification (BPNN) were tested by a real robot platform [8] which was applied to attentional tasks involving pattern categorization. So, we conjecture that the model proposed here (including the smart objects) can also be used in a real platform. In this case one would take out the controller of simulation provided by the SOMOD architecture, and provide visual (and other sensory) information from the robot hardware. Also, the virtual object models with the history of interactions (primitives) have to be previously constructed (or learned) from real object models and stored in the agent memory in some way. An off-line training phase is necessary here for that. Then, at execution time, a real object can be identified and its virtual model with the action to be executed by the robot can be retrieved and the action performed. In this way, a robot does not have to know what is the exact positioning of an object in the environment. The main idea is that a virtual version of an identified object has the above-mentioned interaction information and can pass it to the robot in order to execute scripts, saving memory and time.

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 11), in which our simulator reach/grasp an object (a chair) close to a lit. This task sequence can be 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.

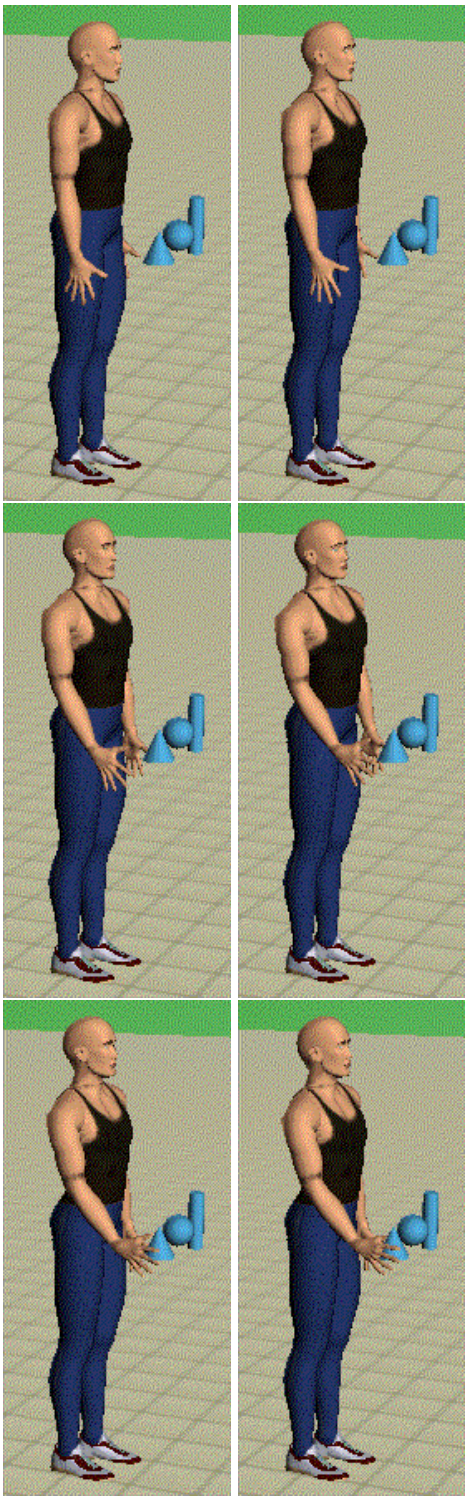

Fig. 10. Virtual Human Agent performing a grasping task.

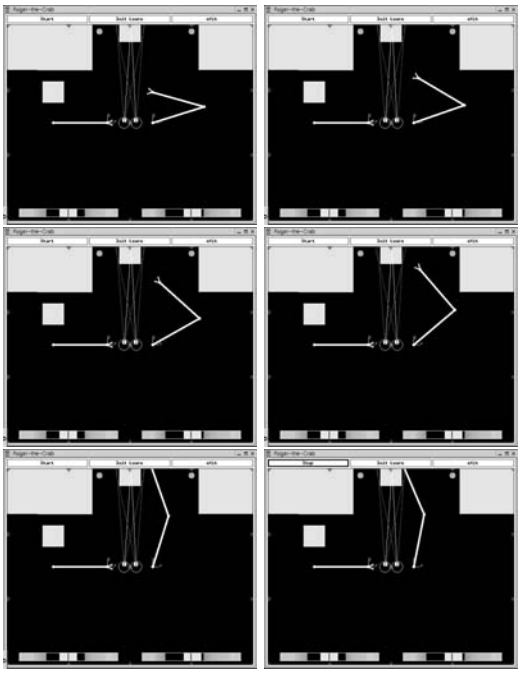

Fig. 11. Simulated robot performing a manipulation task (reaching).

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 needed 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 global scripts of other agents, external changes in the environment detected by perception, and internal predicted changes produced by a close resource manipulation.

One of the main reasons 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 feature models (smart objects) that represent virtual objects, with interactive features empirically defined. 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, as in the artificial life approach.

We plan to improve this implementation by increasing the set of low-level actions that the agent/simulated robot is able to perform (interactions with objects). Another direction is to test learning approaches to decide the possible interactions with the objects. In such context, the agent would start with no knowledge about the environment, and the system memory would have no perceptual pattern representations. The goal then is to incrementally construct a shared map of the environment, learning the existing perceptual patterns. After such a world representation is constructed, agents can perform other more specific tasks.

Finally, we also intend to extend our current unidimensional retina to a fully bidimensional one in order to test the recognition of more complex shapes. We intend as a final goal to have our software tools successfully running in both real robots and virtual characters, so that they can be able to perform complex tasks autonomously.

References

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

[2] 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.

[3] C. Bordeux, R. Boulic, and D.Thalmann. An Efficient and Flexible Perception Pipeline for Autonomous Agents. Proceedings of Eurographics '99, Milano, Italy, 23-30, 1999.

[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] 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.

[6] L. M. G. Gonçalves, 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.

[7] L. M. G. Gonçalves, C. Distante, A. Oliveira, D. Wheeler, and R. A. Grupen. Neural mechanisms for learning of attention control and pattern categorization as basis for robot cognition. In *In Proc. of International Intelligent Robots and Systems Conference (IROS 2000)*. IEEE Computer Society, October 30 - November 5 2000.

[8] L. M. G. Gonçalves, 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.

[9] L. M. G. Gonçalves, F. W. Silva, R. C. Farias, and R. A. Grupen. Towards an architecture for artificial animated creatures. In *Proc. of VI CGS/IEEE Computer Animation Conference*, pages 170–175, Los Alamitos, CA, May, 3-5 2000. IEEE Press. Conference held at Philadelphia, PA.

[10] 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.

[11] 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.

[12] 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.

[13] M. Kallmann and D. Thalmann. Modeling Behaviors of Interactive Objects for Real-Time Virtual Environments. *Journal of Visual Languages and Computing*, 2002, to appear.

[14] M. Kallmann, A. Monzani, C. A., 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.

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

[16] 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.

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

[18] 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.

[19] 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.

[20] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Proc. of the International Conference on Neural Networks (ICNN'93)*, pages 123–134. IEEE Computer Society Press, 1993.

[21] H. Rijpkema, and M. Girard, Computer Animation of Knowledge-Based Human Grasping. *Proceewdings of Siggraph'91*, pages 339-348, 1991.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, Parallel Distributed Processing: Explorations in the microstructure of cognition*, volume 1: Foundations. The MIT Press, Cambridge, Massachusetts, 1986.

[23] X. Tu, and D. Terzopoulos. Artificial Fishes: Physics, Locomotion, Perception, Behaviour. Proceedings of Computer Graphics, 43-50, 1994.

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

[25] P. Werbos. Backpropagation: Past and future. *IEEE International Conference on Neural Networks*, pages 343–353, 1988.