A Control Theory Approach for Real-time Animation of Artificial Agents *

FERNANDO WAGNER DA SILVA[1],LUIZ M. GARCIA[2], RICARDO C. FARIAS[3], ANTONIO A. F. OLIVEIRA[1],

[1] Universidade Federal do Rio de Janeiro (LCG/UFRJ)
CP 68511, 21945-970, Rio de Janeiro, RJ, Brasil
(nando,oliveira)@lcg.ufrj.br
[2] Laboratoire d'Analyse et d'Architecture des Systemes (LAAS/CNRS)
7, Avenue du Colonel Roche, 31077 Toulouse France
lmgarcia@laas.fr
[3] State University of New York at Stony Brook (AMS/SUNYSB)
Stony Brook, NY - 11794-3600
rfarias@ams.sunysb.edu

Abstract. We propose basic mechanisms in support to autonomous, artificial animated agents. We use an approach based on robotics control theory, dealing with physics constraints and dynamics and kinematics issues providing a well structured way to control the agent resources. We validate the mechanisms by presenting three computer animated platforms with different structures (sensors, actuators, and dynamics) which have used them. As a practical result, our animated agents are able to perform different tasks on the top of the same control structure.

1 Introduction

An aspect noted on existing architectures for simulation of artificial animated agents ([7, 1, 2] and others) is that the agent's pose and perceptual state are generally attached to a geometric model, consequently arbitrated. This turns the agents into simple task executors without any antonomy for decision or reaction capability. Also, the agent's movements are notedly disorganized and irregular and its behavior unrealistic. In this work we present a new concept, a general model for computer animation of artificial agents and the basic mechanisms in support to it. The model supports sensing capabilities and deals with physics constraints and dynamics and kinematics issues. In the adopted philosophy, the agent's processes do not deal with the construction of a geometric model, in order to represent the environment. From simulated sensory information (the "sensory buffer"), by reducing and abstracting data the agent system computes a "perceptual buffer", which is joined to other type of information like the agent's pose and functional state to define the current "perceptual state". So, the agent system can make decisions and act mainly based on the information contained in this perceptual state. A key aspect in computer animation is to produce movements that are smooth and natural over time. This can be guaranteed by using the control-theory approach introduced here and by considering physics constraints and the dynamics and kinematics aspects of motion.

In our context, artificial animated agents can be simply understood as devices that imitate living beings. The main characteristic of an animated agent is its real-time sensing, perhaps planning, and acting (or sometimes directly reacting) capability. Such being is a dynamic entity, which learns in some automated way and reacts to stimuli performing related actions as feedback. We tested the developed tools by using three different purpose artificial animated agents. The first agent (*Roger-the-Crab*) simulates a crab-like robot with two cameras and two arms. The main task devoted to *Roger* is to learn attention control and categorization behaviors as a feedback to environmental stimuli. The second agent simulates a multi-link, a snake-like agent that, instead of crawling on the ground, uses its two bidirectional end-effectors to progress. The third agent is a virtual human denominated affectively "*The-Human*". It uses information contained in its perceptual state and previously recorded, captured motion data to progress.

By looking the results of the experiments and demonstrations performed, we could see that this approach contributes with some improvements. First, it is easier for a virtual agent to react, choosing actions based on its own perception of the scene rather than by using directly a geometric model. Second, the system deals with less data, what substantially improves its performance. Third, by using a control theory approach, and considering physics constraints as gravity and inertia and dynamics and kinematics issues, we approximate simulation and reality. Our agents behave more naturally. Finally, the resulting architecture allows one to deal with dynamic environments in a more efficient way, providing real-time feedback to an eventual stimuli change. The agent maps the information contained in its perceptual state in actions, retrieving the motion parameters necessary to perform it. So, the main advantage of using such architecture is to allow the agent to act autonomously to perform its animated behavior (imitate living beings or robot agents).

*This work is supported by LAAS/CNRS France and by CNPq Brazil

2 Scene Representation and Sensing Simulation

In general, by following existent methodologies we can relate geometric models with simulation and real agent platforms in two opposite ways. While in real platforms roboticists have tried to make complete scene models from perception then to plan (or reason) and act based on that model, in simulation a geometric model of a scene is previously given to a virtual agent which tries to directly act based on the model information. In this work, we use a simple world representation, which is mapped to local sensory information when the system is operating. Besides our representation is not a standard geometric model as voxels, triangulations or meshes representations, we note that the last could also be used here indexed in a certain way. Sensing simulation from our world representation resulting in a local perceived set of information is the main reason of the success of our approach. This substantially improves the system performance, since it deals with less data. Simulated sensors like cameras, collision detectors, and other that allows to obtain haptics (including proprioceptive and tactile) information are used here. Note that this approach, besides allowing our agents to perform their behaviors in real-time, also introduces a more natural way to perform computer animation, perhaps closer to biological models. We use *Computer Graphics* techniques to calculate a local sensory buffer from the global environment representation. So, the result would be a simulated retina, which then can be accessed by the agent. Also, based on the agent pose (configuration of its links and joints) and on natural arbitrated parameters of the environment, haptics (including proprioceptive and tactile) information can be derived. The point is that, in general, local sensory information can be fast and easily simulated from a world representation. In this way, simulated platforms can approximate real platforms and local simulated sensory information rather than simplified, partial, or even complete world representations can be used by simulated agents to plan, reason, or to react in real-time. In this way, the system architecture proposed in this work deals with the definition and extraction of meaning features (the perceptual buffer) from the local sensory information derived from those representations. The philosophy introduced in this work is a first step in this direction. Note that in the case of a robot platform, cameras, sonars, infra-red range detectors, force/torque and other sensors would provide such sensory information.

3 Agent Modeling

To animate general articulated agents, a computational model must be employed to correctly represent its main structure, allowing a robust control of the animation parameters over time. Our agent model consists of a skeleton formed by links and joints organized hierarchically. Links are geometric segments of the structure that are allowed to move. Joints are the geometric constraints used to connect the links, allowing relative movements (also known as Degree of Freedom, DOF) between the segments of the structure. This structure may assume any level of realism depending on the application. This description is adequate to be used in most animation systems available today, since it reflects the structure of a general articulated figure.

As our agents' resources are dynamical systems, we must model the inertial parameters of their world before doing anything with them. This involves computing the influence of gravity and inertia on the system motion. Our agents live in a world with gravitational forces acting in the negative Z direction (down vertical). The equations of motion for a general n degrees of freedom open chain mechanism can be written as:

$$\tau = M(\theta)\theta'' + V(\theta, \theta') + G(\theta) \tag{1}$$

The term $M(\theta)$ is a positive definite and symmetric (so always invertible) matrix of inertia. The vector $V(\theta, \theta')$ incorporates all terms which depend on velocity (centrifugal and Coriolis forces). The vector $G(\theta)$ contains the gravity forces component. The angular acceleration vector in the instant t can be computed from Equation 1 as:

$$\theta''(t) = M^{-1}(\tau - V - G - F) \tag{2}$$

The term F represents any miscellaneous external force (contact loads, friction, etc). Note that real robots are physical devices which perform this computation in analog, but in general the above motion Equations (1,2) can be used to simulate the mechanism. Given suitable initial state variables, the state of the system can be predicted for all simulation steps in future time. Once the above parameters are computed, we can use a simple Euler integrator to update the positions and velocities of the creature's manipulator. As the type of joints of all agent resources is revolute, in the instant $t + \Delta t$ we have:

$$\begin{aligned} \theta'_{t+\Delta t} &= \theta'_t + \theta''_t \Delta t \\ \theta_{t+\Delta t} &= \theta_t + \frac{\theta''_t \Delta t^2}{2} + \theta'_t \Delta t \end{aligned} \tag{3}$$

By using the above formulation, the forward-model, a highly non-linear and coupled system, shown in Figure 1 is obtained. To put the agents working, we still need to linearize and de-couple. This involves considering the kinematics of the resources, the implementation of a feedback controller, and the empirical verification of the resulting second-order, closed-loop response. The incorporation of this feedback controller produces a dynamical system with feed-forward compensation, shown schematically in Figure 2. The inertial gains K_p and K_d introduced in the controllers are determined iteratively. Finally, to complete the motion simulation, we reparameterize the update transforms that describe position of the agent's manipulators or the state of each joint and link. We define a matrix T_i for each link i of an agent resource, which represents the homogeneous (3D) transform (rotation plus translation) to be applied in order to transport the world frame coordinates to

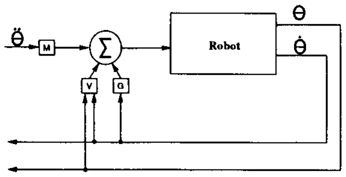

Figure 1: Non-linear and coupled forward-model.

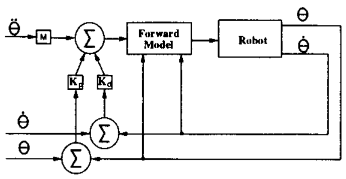

Figure 2: Feed-forward linear and decoupled system.

a link extremity. At each time that the controllers of a resource operate, these matrices are updated by using the current angular positioning (θ_i) of the links. Forward kinematics for a given link (joint) i can be simply calculated by multiplying the previous links matrices recursively as $T_{i-1}T_i$. This motion simulation also reparametrizes inverse kinematics Equations of an agent resource (given the coordinates (x, y) of the end-effector, return joint angle solutions). This is used conjunction with the current agent pose to determine whether a goal is reachable as well to determine which configuration would allow the resource to reach the goal. The above motion simulation works very fast to compute the motion parameters for each agent resource since it can be obtained by multiplications and sums of very simple matrices. Since the motion parameters are computed, the simple forward kinematics model can be used to update the agent resources, considering a simulated time interval (clock rate) and the current agent pose.

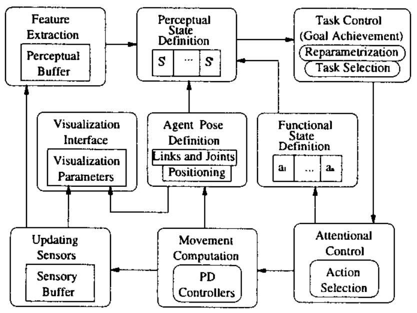

Figure 3: System Control Architecture

4 The Control Architecture

Figure 3 shows the main aspects of the control architecture used to perform our animated agents. Briefly, we use a simple model, a world description that is not an inherent part of the agent kernel system, and some computer graphics and other direct techniques as explained above to derive sensory information. Then, information from the sensory buffers are abstracted to construct a perceptual buffer (feature maps). The agent uses this set of features plus its pose and functional state to define its perceptual state. The agent's pose contains information regarding the positioning and configuration of its links and joints and the functional state contains the set of high-level actions performed or to be performed by the agent and also the state of all controllers. The perceptual state can be represented as a vector containing all type of information that would be useful for a given set of tasks. The task controller takes into account the information in the perceptual state to evaluate the current task goal achievement. Based on the results of this evaluation, it re-parameterizes the current task, allowing the agent to choose the right set of actions to perform, or eventually it selects a new task, if the immediate task goal is achieved. After a task selection or reparametrization, the attentional mechanism operates based on the current task parameters. Based on a task dependent policy, it selects a new action. A chosen action eventually involves shifting attentional focus and/or performing movements of the agent's resources. To determine the region where to put attention, we first compute salience maps from attentional feature maps extracted from the visual information. The attentional target is simply given by the most activated region in these maps. In case of physical movements, this implies in the definition of a target position for each agent resource, in coordinates of its (reachable) configuration space (C-space). To determine the best path we apply a gradient descent strategy to solve a harmonic function defined over the C-space, regarding restrictions (obstacles) of a potential map. Since differential displacements are determined in C-space coordinates, the differential motion parameters are retrieved by using the above Equations of motion (1, 2 and 3) and are sent trough the servo PD (positional derivative) controllers. This effectively brings the agent to a new pose and also puts a new set of information in the sensory buffers. The agent pose and sensory information can then be used by the visualization interface to display the agent's resources and its environment in a computer screen. This can be done by using a standard rendering/visualization procedure. Then, since the new sensory information is acquired (simulated), the process can be re-started (feature extraction).

We remark that this low-level operating loop follows a control theory approach, what guarantees stability and global convergence of the creature resources (controllers) in the achievement of an action. Note that the agent system performs actions involving motion by decomposing it in differential (instantaneous) sub-movements. At each time step, a small movement is performed, followed by an update in the agent sensory buffer consequently providing new information in the perceptual state. This produces smooth and differentiable motion and also allows the agent to eventually reparameterize its task goal on-line, during the execution of an action, taking into account the changes in the perceptual state. We note that this approach is inherently reactive, choosing actions based on perceptions of the world at different temporal scales rather than by using a geometric model as in traditional planning techniques. By using a robotics control theory approach, we introduce more realism in the motion animation of our agents. Also, as time is a critical parameter for computer animation, 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 simulated clock rate.

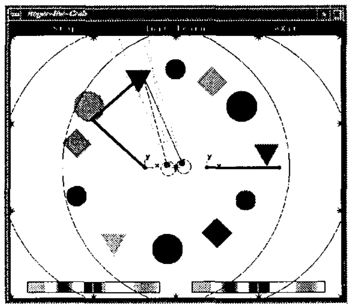
Figure 4: Roger the Crab.

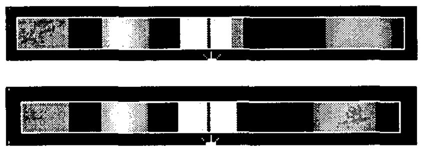
Figure 5: *Roger*'s left (top) and right (bottom) retinas.

5 The Simulator "Roger-the-Crab"

Our simulated robot *Roger-the-Crab* has seven controllers, one for the neck (pan), two for the eyes (vergence), and two for each arm (equivalent to a shoulder and elbow) all integrated in a single platform. *Roger*'s environment can be specified and dynamically changed. Currently, we can (on-line) include (or remove) in *Roger*'s room several types of objects like circles, squares, and triangles. The object size, intensity (color), and mass (for arm sensing simulation) can also be on-line specified and changed. Each wall (*Roger*'s environment is a rectangular room) can have a different (uniform) color, simulating an ideal background for the objects. Visual sensing is simulated from this world definition by using a Phong illumination model. For each of the 256 pixels in Roger's unidimensional retinas, seen in Figure 5, an intensity value is calculated in function of the radiance of the world patch corresponding to it. Following, a Gaussian noise process simulates acquiring errors, asserting a more natural retinal image. Haptics simulation is done based on the arbitrary value attributed to each object mass. In case Roger is grasping an object, the object mass and the proprioceptive information relative to the arm configuration is mapped to arm torque and velocity vectors necessary to lift the object.

5.1 Simulating *Roger*'s Dynamics.

Roger lives in a flat world, with gravitational forces acting in the negative Z direction (down vertical). Considering that the eyes are embedded in a 2D framework with 1 degree of freedom or one link (have you seen a crab eye?:-), very simple formulae can be derived from Equations 1, 2 and 3. The dynamics equations used by *Roger*'s arm controllers are a bit more complex because of the two DOFs. To complete the motion simulation, we update the transform matrices T_i seen above that describe position of *Roger*'s manipulators. In order to visualize *Roger*'s eye axes and their intersection and also in some visually guided tasks, we need to map eye goals in world coordinates (we refer to this as forward kinematics for the eyes). We determine a location in the horopter, the point in which the eye axes cross, by using a simple equation for the visual gaze along the gaze of the left eye. Inverse kinematics of an arm or eye (given the coordinates (x, y) of the end-effector or of the horopter, return joint angle solutions) is also formulated.

5.2 Data Reduction and Feature Abstraction

We promote data reduction and abstraction in order to support the construction of *Roger*'s perceptual state. To reduce data, we compute *multi-resolution* (MR) image representations from *Roger*'s retinas. MR intensity and motion images are composed of three resolution levels of 32 pixels each, calculated by applying mean filters in the original captured (simulated) images and on the difference between consecutive frames, representing motion. A *multi-feature* (MF) representation over the MR images provides feature abstraction. Gaussian MF are computed by convoluting the MR intensity images with Gaussian kernels, in three different kernel diameters (σ). Motion features are computed by applying a first Gaussian derivative to the previous MR motion images. We also compute stereo disparity features over the responses of the second order Gaussian MF. The result is a *multi-resolution-multi-feature* (MRMF) representation. To promote attention behavior, *Roger* computes a set of attentional feature maps from the MRMF and gathers information from these to construct salience maps. The most salient region in these maps will determine the attentional target. The controllers of the eyes/arms calculate the displacements to be applied to the degrees of freedom of *Roger* by using this target. Then, the motion parameters are retrieved using the above Equations of motion and *Roger*'s PD controllers effectively produces the movements to foveate (or to reach, in case of an arm movement) the target. In the situation displayed in Figure 4, the left arm and the controllers of the neck and the eyes of *Roger* have converged on a region containing a triangle. The above features will be extracted from that region of interest and joined to other type of information as *Roger*'s pose and functional state to compose *Roger*'s perceptual state. Based on the information contained in this perceptual state and on the current state of the supervisor (controllers), another action (generally a movement) will be performed as a feedback to the environmental stimuli. This carries out an attentional policy which can be understood as an strategy for choosing the best actions according the current perceptual state. Currently, we developed two policies for the purpose of performing a monitoring task [4].

6 The Simulator *Thc-Worm*

In this example, we have implemented an autonomous, virtual creature called *Thc-Worm*. It looks and acts like an earth-worm and has a structure based on a hierarchy of rigid segments connected by revolute joints, as shown in Figure 6. The initial hierarchy arrangement of *Thc-Worm* is defined by the user, and may include closed-loops, multiple branches and end-effectors (see part *b* of Figure 6).

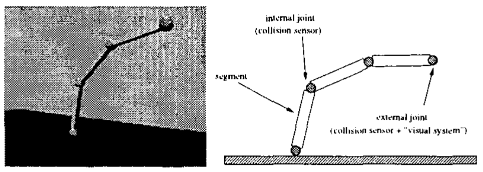

Figure 6: Close view at *Thc-Worm* and its internal structure.

6.1 Functional Control and Sensing.

The control cycle that manages, for each step of simulation, the perception, reaction, and motion of *Thc-Worm* is composed of a series of low-level programs. According to a predetermined task, which may vary from a simple "progress on this terrain" to a more complex situation such as "climb this stair", decisions are taken based on the current perceptual state. The goal achievement is constantly evaluated by the *task tontrol* module and, if necessary, the creature changes its internal hierarchical structure by using a Dynamic Graph Rearranging (DGR) algorithm. This algorithm modifies its internal hierarchical structure in real-time according to feedback provided by its sensors. Each time an end-effector touches the ground, the entire hierarchy is transformed, rearranging from the root to the end-effectors. This makes more natural and, consequently, efficient the computation of forward and inverse kinematics through the creature's structure. Moreover, this allows a precise localization of end-effectors in the structure. This is particularly important during the process of collision detection and motion planning/control. A tactile sensor, implemented at each joint of *Thc-Worm*, allows it to get the exact time that any segment touches an object or the terrain. Additionally, at each end-effector, an algorithm based on ray-casting simulates a simplified visual system. This allows the creature to generate its sensory buffer.

6.2 Motion Computation and Control.

Each node of the hierarchy has a data structure associated to it, which contains 3D information regarding the segment position, the relative orientation to the neighbor segment, and pointers to the attached joints. Kinematics and dynamics parameters of the segment are also stored. As in *Roger-thc-Crab*, after motion parameters computation, the movement is simulated by propagating transformation matrices available at each node, using a standard forward kinematics procedure. To control the motion behavior of *The-Worm*, we use an approach based on Zeltzer's work [8]. According to the goal defined by the attentional mechanism, the Task Control module decomposes it into a set of small subprograms. Each subprogram consists on a specific motion that *Thc-Worm* can perform (progress, full extension and so on) and are generated by a series of local motor programs (LMP). Each LMP is responsible for changing a specific rotation parameter in a joint (or set of joints). Therefore, to perform a progression motion with a fixed step length, a series of LMPs are called such as to modify the rotation parameters at specific joints in the hierarchy. This process is constantly supervised by the *task control* module, which also uses information collected by the sensors to evaluate the performance and necessity of changes in the motion parameters. If necessary, the *task control* module reparametrizes a current set of LMPs so that a motion is performed in a more efficient way to complete the task. Also, all movements of *The-Worm* are parametrized such that each end-effector acts like a leg, propelling the creature throughout the environment.

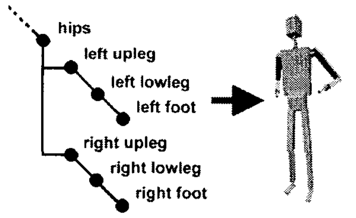

Figure 7: Agent topology and geometry.

7 The Virtual Human Agent (*Thc-Human*)

In this example (Figure 7), we have implemented an artificial animated agent similar to a human being. At the programming level, this agent is represented by using a modified version of Zeltzer's APJ structure [9], adapted to work with motion captured data. Such representation allows us to deal with motion processing techniques in a straightforward way. We use a motion capture approach as the basic component of locomotion for our human-like agent because this provides more realism, similarity with human motion, and makes possible the real-time visualization. On the other hand, captured motions act like scripts and are hard to modify in order to be adapted to different situations according to user interaction. However, such drawback can be avoided by using well known motion processing techniques. The information generated by a motion capture device is composed by a data set of samples, which represent the position and global orientation of a real object at uniformly spaced time intervals. In the case of human motion capture, the position and orientation of several joints of an actor are recorded, generating a set of 1D signals also known as motion curves. These curves are then processed and mapped onto a skeleton hierarchy, which will drive a virtual actor (that is, an agent) in the computer [10]. The simulation of sensors similar to *The-Worm* provides the input information used to construct the agent perceptual state, used to perform an autonomous navigation task. At every time-step, the agent selects a new set of motions or reparameterizes the current set based on the information contained in its perceptual state. This is task dependent and can be learned according to the task. In the experiments performed here, we determined what subset of motion is to be used at a given time empirically, according to the task goal and to the information currently in the perceptual state.

7.1 Motion Combination and Control

The signal-like nature of captured data suggests that it should be treated using the paradigms of signal processing theory [11]. Motion capture data processing has become an important field of research in recent years [10]. The crescent demand of powerful tools for motion editing has led to the development of several techniques such as warping [12], concatenation [13] and reparametrization [14]. We use these tools to combine small pieces of motion such as to generate new movements according to user interaction, in a process that will drive the agent in the virtual world. The set of motion processing tools used by our agent includes several well defined operations.

Motion filtering may be used to eliminate or attenuate specific frequencies of a motion signal. A low-pass filter can be applied to the motion curves of the agent in order reduce the noise introduced by the capturing process, thus making the motion more natural and smooth. Also, special filters can be used to create slow-motion and accelerated-time effects in a motion sequence [14]. **Motion concatenation** is a very simple operation which is widely used in many commercial applications involving avatars and user-driven agents, such as FIFA Soccer and Virtual Fighter. This technique consists on sequencing a number of motions, chosen from a memory-based motion library according to user interaction, performing a smooth transition between them. This can be done by using special motion interpolators such as spacetime [13] and physically based constraints. **Motion warping** functions can be applied to motion curves to provide local deformation. This causes changes in the orientation and position of specific sets of joints of the agent, changing the movement on-line. This can be used for example, for obstacle avoidance. **Motion time-warping** algorithms can be used to expand or contract the agent motions in time. The basic goal is to deform the motion without producing changes in the frequency contents of its curves. In the expansion case, this causes a cycling in the motion [15]. Similar techniques are used in soccer games in order to repeat a basic motion over time (e.g. a player pursuing the ball: cycling of a running motion).

8 Demonstrations and Experimental Result

We performed several demonstrations and experiments for our agents involving attentional monitoring tasks of the environment and progressing tasks.

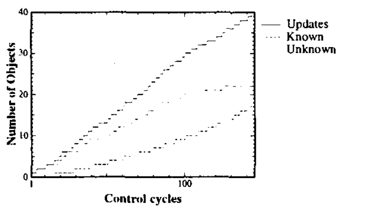

Figure 8: Overall evaluation. The lines show the total number of objects detected in the environment (upper solid line), number of new objects detected (middle line), and number of already known objects detected (lower line).

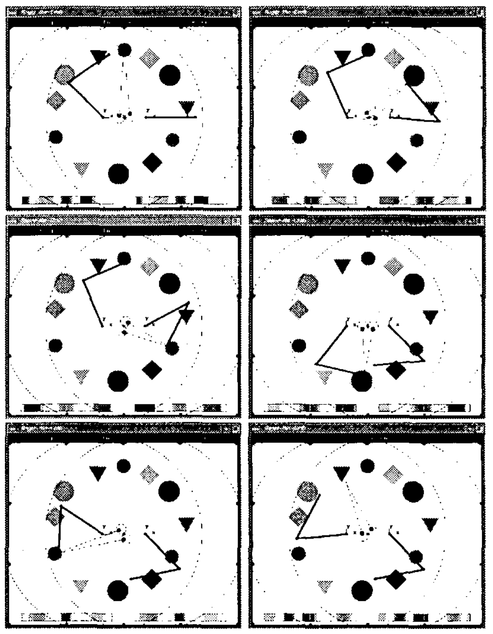

Figure 9: *Roger* constructing attentional maps.

8.1 Experiments and Demonstrations for *Roger*

The experiments for *Roger-the-Crab* included real-time monitoring tasks. Basically, many instances and types of objects are placed in its room. We expect that *Roger* focus attention on all objects, learn their characteristics, and incrementally construct/update its attentional maps. Figure 8 shows a global evaluation for one of the monitoring tasks in which *Roger* visits all regions of its environment. This result is a consequence of using a policy developed for attention control [4], implemented on the top of the architecture discussed in this paper. As a result of the attentional policy, all regions of interest in the environment are visited (looked at or touched by Roger). Figure 9 shows pictures selected from an animated sequence in which *Roger* visits all regions/objects in its environment. The results show that the mechanisms described in the paper allowed *Roger* to perform its animated behavior.

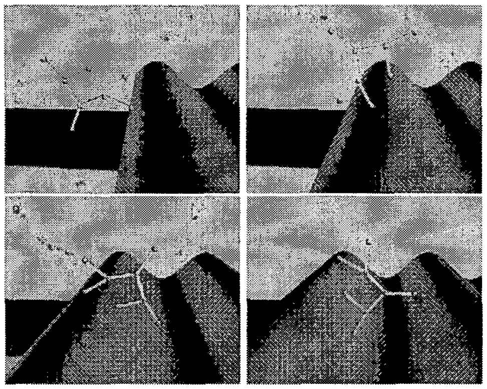
Figure 10: Progressing over irregular terrain.

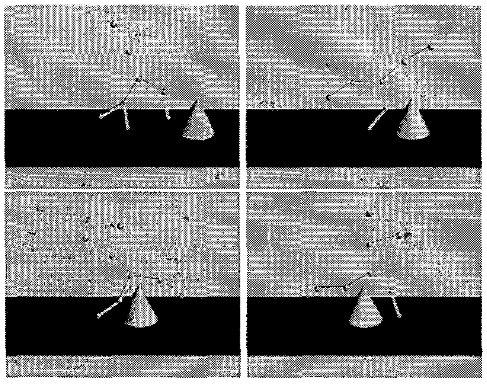
Figure 11: Obstacle avoidance.

8.2 Demonstrations for *The-Worm*

Our simulated worm was tested with three different situations involving uneven terrains and obstacles. The goal of such testings was to evaluate the creature's skill to take reasonable decisions, reacting autonomously to environmental stimuli based on its own perception of the surrounding environment. In the first demonstration (Figure 10), a multi-branched worm progress on irregular terrain, with a series of peaks and valleys. In this case, the main goal was to verify if all creature's sensors were working properly, detecting collisions and measuring distances from the end-effectors and several objects of the environment (thus allowing the creature to change the phase of its oscillattors in order to optimize its path). Figure 11 shows the use of a DGR algorithm together with sensory information collected by the creature in order to avoid an obstacle. Figure 12 shows *The-Worm* climbing a stair. In this case, the Task Control module has used the sensory information to calculate the correct phase components for the oscillators that drive the creature, thus allowing a smooth stair climbing.

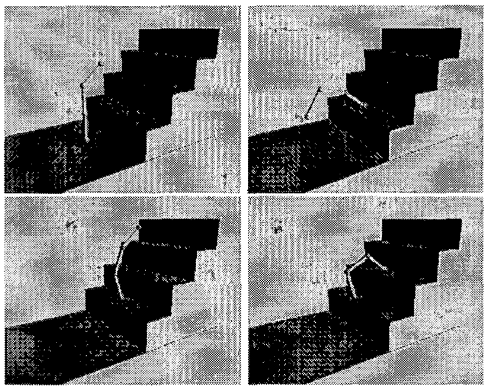
Figure 12: *The-Worm* climbing a stair.

8.3 Demonstrations for The-Human

We show an example in which our virtual human uses a motion warping operation to avoid an obstacle detected by using its perceptual state. In this case, the motion parameters obtained by using the original information in the perceptual state predicted a straight line walk. But, when the agent walks over a virtual scenario with an obstacle, a collision is expected to occur (Figure 13). Using this new perceptual information, the agent system detects the obstacle position and its internal motion engine can calculate the necessary warp factor that must be applied to a group of joints such as to avoid the collision with the obstacle (Figure 14).

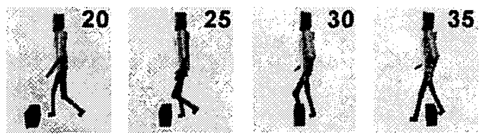

Figure 13: Original retrieved data.

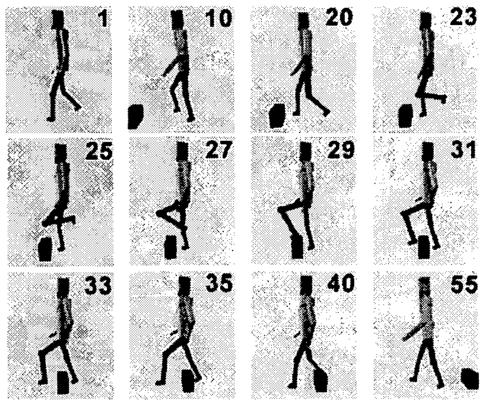

Figure 14: Agent performing motion warping.

9 Discussion, Conclusion and Future Work

We have introduced an useful animation approach and tools which were successfully used by different agents to perform real-time autonomous tasks. The three totally different agents described in the text could be implemented on the top of this architecture without any significative modification of their structure. The main reasons of the success were data reduction and the definition of meaning, abstracted features, decreasing the amount of on-line computations necessary. Decisions can be taken more rapidly using the reduced set of information provided in the perceptual state. Another important result was the development of an architecture which we believe can also be used in real platforms, decreasing the distance between simulation and reality. Once we use a control theory approach and consider physics aspects and the dynamics and kinematics of the agent resources, the simulated hardware can be substituted by the real hardware without any significative changes. Furthermore, by grounding our system in physical parameters and specifications, we conjecture that our agents acts with more realism, and would perform in a way that is much similar to real animated agents behavior such as robots, animals, or the human being. Finally, by using the tools and architecture described in this work, we have built as a result true computer animation laboratories (*Roger-the-Crab*, *The-Worm* and *The-Human*) that could be used for a variety of experimentations like multi-modal sensory integration ([5]), learning of attention control ([4]), development of self growing tools for pattern categorization behavior ([3]), and the embedding of motion capture techniques in a human like, virtual agent for achieving human like behavior ([10, 6]).

We have chosen the tasks of monitoring and progression onto rough terrains to test the agents because the functional aspects of the developed tools could be fully explored in these tasks. Furthermore, we conjecture that an animated agent most like a real creature has to learn the characteristics of its habitat and adapt to it in order to survive. In this way, questions relative to the current environment state can be addressed by a simple analysis of the agent perceptual state and of dynamical attentional maps. As an example of a more advanced general task which involved feature extraction, attention control, motion computation, and changes in the environment, the creature *The-Worm* and the virtual agent *The-Human* have used the basic architecture and tools to progress in their environments. Note that all they need is a local perception of the environment provided by their simple sensors.

An immediate extension of this work is to increase the feature space and/or the set of tasks that the agents can perform. We can also define other tasks based on the current feature space. Then, it would be possible to derive various policies, each one appropriate for a given task. Furthermore, we believe that tasks involving other behavioral aspects can also be done by using the same architecture, but deriving other policies. Thus, a final possibility for future works is to derive learning policies for general tasks considering these aspects. In this way, an agent can learn and perform tasks without strong interaction with an operator, augmenting its autonomy.

References

[1] M. Costa and B. Feijo. An architecture for concurrent reactive agents in real-time animation. *Proc. of SIBGRAPI'96*, pages 281–288, 1996.

[2] R. Farenc, N. Boulic and D. Thalmann. An informed environment dedicated to the simulation of virtual humans in urban context. *Proc. Eurographics'99*, pages 309–318, 1999.

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

[4] L. M. G. Gonçalves, G. A. Giraldi, A. A. F. Oliveira, and R. A. Grupen. Learning policies for attentional control. *IEEE Symposium on Computational Intelligence in Robotics and Automation (CIRA '99)*, November 1999.

[5] L. M. G. Gonçalves, A. A. F. Oliveira, and R. A. Grupen. Multi-modal stereognosis. In *Proc. of III International Conference on Autonomous Agents.*, pages 337–338, New York, USA, May 1999. ACM Press.

[6] L. M. Gonçalves, F. W. Silva, A. F. Oliveira, L. Velho, and J. Gomes. Embedding a motion-capture interface in a control structure for human–like agent behavior achievement. In *Agents 2000 Workshop on "Achieving Human-Like Behavior in Interactive Agents"*, Barcelona, Spain, June, 2000.

[7] K. Perlin and A. Goldberg. Improv: A system for scripting interactive actors in virtual worlds. In H. Rushmeier, ed., *SIGGRAPH 96*, Annual Conference Series, pp 205–216. ACM Press, Addison Wesley, aug 1996.

[8] D. Zeltzer. Motor control techniques for figure animation. *IEEE Computer Graphics and Applications*, 2(9):53–59, november 1982.

[9] D. Zeltzer and K. Sims. A figure editor and gait controller for task level animation. In *SIGGRAPH Course Notes no. 4*, pages 164–181, august 1988.

[10] Silva, F., Velho, L., Cavalcanti, P., Gomes, J., An Architecture for Motion Capture Based Animation. In *Proc. of X Brazilian Symposium of Computer Graphics and Image Processing* (October 1997), pp. 49–56, IEEE Press.

[11] Williams, L., Brudelin, A., Motion Signal Processing. In *Computer Graphics (SIGGRAPH'95 Proceedings)* (August 1995), pp. 97–104.

[12] Witkin, A., Popovic, Z., Motion Warping. In *Computer Graphics (SIGGRAPH'95 Proceedings)* (August 1995), pp. 105–108.

[13] Cohen, M., Rose, C., Guenter, B., Bodenheimer, B., Efficient Generation of Motion Transitions Using Spacetime Constraints. In *Computer Graphics (SIGGRAPH'96 Proceedings)* (August 1996), pp. 147–154.

[14] Silva, F., Velho, L., Gomes, J., Motion Reparametrization. In *EUROGRAPHICS Technical Note (short-papers proc.)* (September 1998), pp. 1.5.1–1.5.4, Springer-Verlag.

[15] Silva, F., Velho, L., Gomes, J. and Goldenstein, S., Motion Cyclification by Time x Frequency Warping. In *Proceedings of SIBGRAPI'99, XII Brazilian Symposium of Computer Graphics and Image Processing* (October 1999), pp. 49–58, IEEE Press.