Towards an Architecture for Artificial Animated Creatures *

Luiz-Marcos Garcia[1] **Fernando W. da Silva**[23] **Ricardo C. Farias**[4]
Roderic A. Grupen [5]

[1] Laboratoire d'Analyse et d'Architecture des Systemes
7, Avenue du Colonel Roche, 31077 Toulouse France
lmgarcia@laas.fr

[2] Universidade Federal do Rio de Janeiro
CP 68511, 21945-970, Rio de Janeiro, RJ, Brasil
nando@lcg.ufrj.br

[3] Instituto de Matemática Pura e Aplicada
Estrada Dona Castorina, 110, 22460 Rio de Janeiro, RJ, Brazil
nando@visgraf.impa.br

[4] State University of New York at Stony Brook
Stony Brook, NY - 11794-3600
rfarias@ams.sunysb.edu

[5] University of Massachusetts
140 Govenor's Drive, Comp Sci Building, Amherst MA 01003 USA
grupen@cs.umass.edu

Abstract

This work presents a software methodology used to implement artificial animated creatures. The proposed architecture deals with dynamics and kinematics issues and provides a well structured way to control the creature resources. Three artificial creatures with different sensors/actuators/dynamics characteristics show the versatility of the architecture. "*Roger-the-Crab*" simulates a *crab*-like creature, with two eyes (cameras) and two arms. The simulator "*The-Worm*" is a multi-link creature similar to an *earth-worm*. The last platform is the LPR-UMASS articulated "*Stereo-Head*" robot. As a practical result, by using the same control structure, the above artificial animated creatures are able to perform different tasks, involving the construction of attentional maps of their environments from sensory information and progression on a rough terrain.

1. Introduction

In this work we present a new concept, a general purpose architecture used to support artificial animated creatures. The model supports sensing capabilities, deals with physics constraints as dynamics and kinematics issues, and uses a robotics control theory approach to design the manipulators control laws. Furthermore, in the adopted philosophy, the creature's processes do not deal with the construction of a geometric model to reason and act. Sensory information (the "sensory buffer") is provided to the creature processes to compute its "perceptual buffer". This is joined to other type of information like the creature's pose and functional state to define the current "perceptual state". The creature can make decisions and act mainly based on the information contained in this perceptual state. By looking the results of the experiments performed, we can see that this new paradigm contributes with some improvements. First, it is easier for the creature to react, choosing actions, based on its own perception of the world rather than by using directly a geometric model or other complete world representation. Second, the system deals with less data, what substantially improves its performance. Third, by using a control theory approach, and considering physics constraints as dynamics and kinematics issues, we approximate simulation and reality. Our simulated agents behave more naturally and the same architecture can be used by the real robot platform. Finally, the resulting architecture allows to deal with dynamic environments in efficient way, providing real-time feedback to a eventual stimuli change. So, the main advantage of using such architecture is to allow the agent to act autonomously in order to perform its animated behavior.

*This work was supported by FAPERJ and CNPQ/Brazil, and by NSF under IRI-9503687, IRI-9704530, and CDA-9703217

In our context, artificial animated creatures can be simply understood as devices that can imitate alive creatures. The main characteristic of such entity is its real-time sensing, planning and acting capability. It is a dynamic being that learns in some automated way, and reacts to stimuli performing the right actions as feedback. In general, a body model for animated agents consists of a skeleton formed by links and joints organized hierarchically. In our creatures, the links are rigid and have physical specifications like mass, length, and they are connected by using only revolute joints. We have used three different artificial creatures in the experiments and demonstrations to validate our architecture. The first platform "*Roger-the-Crab*" simulates a crab-like robot with two eyes and two arms. The main task devoted to *Roger* is to learn attentional control as a feedback to environmental stimuli. The second platform "*The-Worm*" simulates a multi-link, an earth-worm like creature that uses its bidirectional end-effectors (or extremities) to progress. The third platform is a real robot, a *Stereo-Head* with four degrees of freedom: pan, tilt, left vergence, and right vergence. In this robot platform the main experiments involve using attentional control and categorization behaviors to construct and keep an attentional map of the environment.

2. The Control Architecture

Figure 1 shows the main aspects of our control architecture. Briefly, information from the sensory buffers is abstracted to construct a perceptual buffer (feature maps). The creature uses this set of features plus its pose and functional state to define its perceptual state. Then, the task controller can evaluate the current task goal achievement. Based on the results of this evaluation it eventually selects a new task if the current task goal is achieved or re-parameterizes the current task. This allows the attentional mechanism to choose the right action as a feedback to the environmental stimuli, based on a task dependent policy. If this action involves changing attention, the attentional mechanism changes the current attention window (topic of interest) and calculates the movements to be applied to the degrees of freedom of the creature resources to attend the new position. These parameters are sent trough the servo controllers. This effectively brings the creature to a new pose and also puts a new set of information in the sensory buffers, restarting the cycle (feature extraction).

2.1 Constructing the Perceptual State

The result of moving a resource is a new configuration of its links and joints. This information defines the creature's pose. Changes in the sensors positioning carry out updates in the sensory buffer, which contains all kind of sensory information. In our robot platform, the sensory buffer is represented by the Datacube IP memories, and it contains mainly visual information obtained from the stereo cameras. In simulation, a high-level application computes (simulates) sensory information from a world representation by using computer graphics and other standard techniques. The perceptual buffer is a general structure that contains several feature maps abstracted from all sensory buffers. The existence of this structure can be computationally justified by following Marr [5] and Nishihara's [6] approaches if we further assume that some small set of perceptual features are necessary and sufficient to build an effective and computationally efficient system. The functional state of the creature is established after an action selection. It contains information regarding the set of high-level actions that have been performed and also the current one. It also contains information regarding the state of all controllers. All the above information will be useful for the determination of the creature's perceptual state. This can be represented as a vector, which contains all types of information that would be useful for determining the current execution status of a given set of tasks.

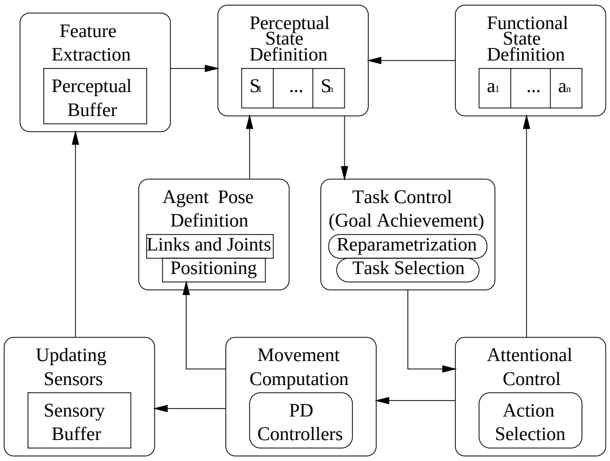

Figure 1: System Control Architecture

2.2 Task Manipulation and Attention Control

Based on the information provided by the perceptual state vector the system can evaluate the current task. If the immediate task goal is achieved, a new task is chosen and the parameters of this task are retrieved. Otherwise, the system re-parameterizes the current task. Then, based on a task dependent policy, the attention control mechanism defines the action which must be performed next in order to pursue the task goal. The execution of an action eventually involves the execution of a series of other low-level intermediate actions. For instance, when the action involves a physical movement, low-level mechanisms (controllers) operate until a convergence criterion is attended (target is reached). Eventually a chosen action involves shifting the attentional focus. We solve this attentional problem by computing salience maps and simply taking the most activated region in these maps as being the target. The activation value of each position is calculated by a (task dependent) weight transfer function, which transfers activation from attentional feature maps (perceptual cues) extracted from the perceptual buffer and also from other attentional feature maps empirically determined.

2.3 Defining Target and Computing Motion

When a selected action involves mechanical movements in the DOFs, this implies in the definition of the target position. As each resource has its own configuration space (C-space), each position of the environment can be represented relatively to a resource by using a configuration map (C-map). Given a target specified by the attentional mechanism in C-space coordinates of a particular resource, the creature computes the best instantaneous path by using a standard planning technique that solves a harmonic function [2]. Once the instantaneous displacements are computed in C-space coordinates, the movement parameters can be retrieved, by using motion equations. This involves formulating dynamic and kinematic equations to define the creature's pose and to compute the velocity and torque/acceleration for each PD controller. Then, the real or simulated effectors and actuators can run using these motion parameters to move the creature resources.

As the creatures' 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. 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 as:

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

Here the term F represents any miscellaneous external force (contact loads, friction, etc). A simple Euler integrator can be used to update positions and velocities of the creature's manipulator. As the type of all joints 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, a forward-model, a highly non-linear and coupled system shown in Figure 2 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 3. The gains K_p and K_d introduced in the controllers are determined iteratively.

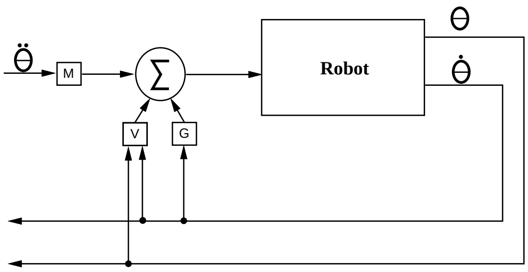

Figure 2: Non-linear and coupled forward-model.

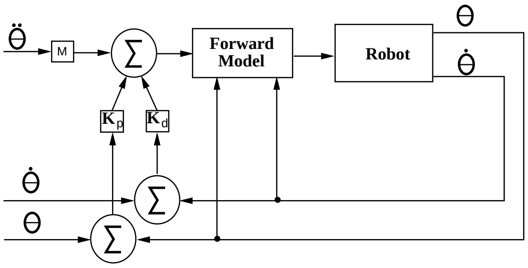

Figure 3: Feed-forward linear and decoupled system.

Finally, to manage the agent pose or to allow simulation of physical motion, we compute the update transforms that describe position of the agent's manipulators (the state of each joint and link). For each link i of an agent resource, we define a homogeneous transform (rotation + translation) matrix T_i that describes the 3D transformation to be applied, in order to get coordinates of its extremity in world coordinate frame. Forward kinematics for a given link (joint) i of a resource can be simply calculated by multiplying the previous links matrices recursively, from frame 0 until frame i, as $T_{i-1}T_i$. Inverse kinematics Equations of an agent resource (given the coordinates (x, y) of the end-effector, return joint angle solutions), is also re-parameterized after a forward-kinematics iteration. Inverse kinematics can be used in conjunction to the current pose of the agent to determine whether a goal is reachable or to determine a resource configuration necessary to reach a target.

3. Experimental Environments

Three different artificial animated creatures and their environments will be resumed in this section. They have used the same architecture described above to simulate/perform their animated behaviors.

3.1 The Simulator Roger-the-Crab

The simulator "*Roger-the-Crab*" presented in figure 4 has concurrent controllers for its neck (pan), for each eyes (vergence) and for each arm (shoulder and elbow). *Roger*'s environment (a small 2D room) can be specified and on-line changed. Currently, we can include (or remove) several types of objects like circles, squares and triangles. Also, the walls can have different (uniform) colors. Visual sensory data is simulated from this simplified world representation by using a simple computer graphics technique. For each pixel of *Roger*'s unidimensional retinas (in the bottom of Figure 4), an intensity value is calculated by using a Phong illumination model. Also, a Gaussian noise process simulates acquiring errors, asserting a more natural retinal image. Haptics simulation is done based on an arbitrary value attributed to each object mass. In case of object grasping, the object mass and the proprioceptive information relative to the arm configuration are mapped to the torque and velocity necessary to lift the object.

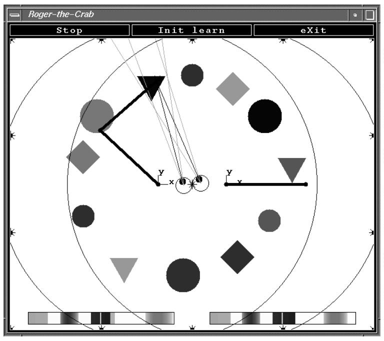

Figure 4: *Roger-the-Crab* interface.

3.1.1 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 complicated because of the two DOFs. A last step to simulate motion is an update in the transform matrices T_i defined above, that describe position of *Roger*'s manipulators (the state of each joint and link). For the eyes, a location in the horopter (point in which the eye axes cross) is determined by 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.

3.1.2 Features for Attention and Categorization.

To extract visual features, filters defined as Gaussian partial derivatives (of order 0, 1 and 2) operating at three levels of resolution (or with different diameters σ), are first applied to *Roger*'s retinas. The result is a multi-resolution-multi-feature (MRMF) representation with three levels of 32 pixels each (nine filter response vectors overall). From this representation we compute *intensity* and *texture* features as local average and variance, respectively, in the vicinity of the results of each Gaussian response. Stereo disparity features are obtained from the correlation between pairs of pixels in the second order filter response of each retina. Motion is simply obtained as the difference between consecutive image frames also in a multi-resolution. The size of an object (or region of interest) is extracted from stereo measurements and using an edge detection process. The weight (only haptics feature used) is calculated as described above, in case an end-effector (a hand) enters in contact with an object.

The above reduced and abstracted features are used for both attention and categorization behaviors. For attention, *Roger* computes a set of attentional feature maps, 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*. 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. The features described above will be extracted from the region of interest containing the object in focus (a triangle) 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 [3, 4]. One is a simple straight-forward strategy, a hand-coded solution, and the other one was derived by using Q-learning [7]. In the monitoring task devoted to Roger, as a result of the attentional policies, all regions of interest in the environment are visited (looked at or touched by Roger) and categorized.

3.2 The Simulator *The-Worm*

In this example, we have implemented an autonomous, virtual creature called *The-Worm*. It looks like an earth-worm and has a structure based on a hierarchy of rigid segments connected by revolute joints, as shown in Figure 5. The initial hierarchy arrangement of *The-Worm* is defined by the user, and may include closed-loops, multiple branches and end-effectors (see part *b* of Figure 5).

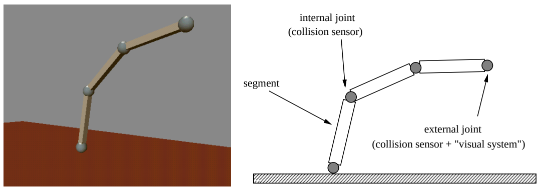

Figure 5: Close view at *The-Worm* and its internal structure.

3.2.1 Functional Control and Sensing.

Figure 6 presents a particular schematic view of the architecture, showing the control cycle that manages, for each step of simulation, the perception, reaction, and motion of *The-Worm*. According to a predetermined task, which may vary from a simple "crawl over 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. A tactile sensor, implemented at each joint of *The-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, which will then be used for feature extraction and the perceptual state definition.

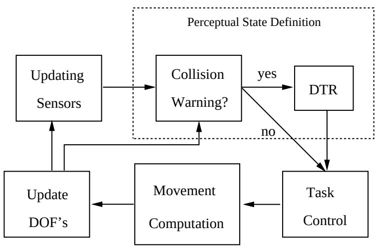

Figure 6: Particular view (functional control scheme).

3.2.2 Dynamic Graph Rearranging.

The-Worm employs a DGR algorithm that modifies its internal hierarchical structure in real-time according to its perceptual state. Each time an end-effector touches the ground, the whole hierarchy is transformed, rearranging, from the root to the end-effectors (e.g. swap the root and end-effector positions in the hierarchy description). This makes more natural and, consequently, efficient the computation of forward and inverse kinematics over the creature's structure. Moreover, this allows a precise localization of end-effectors in the structure any time during the simulation. The use of DGR algorithm is particularly interesting in cases where the structure assumes closed-loop configurations. Figure 7 exemplifies a closed-loop configuration for a multi-branched worm that occurs when two or more end-effectors touch the ground at the same time (left of Figure 7). By evaluating the current positioning of the structure, which is part of the perceptual state, the Task Control module can trigger a set of commands that will drive the creature in a more natural and efficient way (right of Figure 7). The DGR algorithm can also be useful in the presence of obstacles which may cause a singularity as shown in the part *a* of Figure 8. The DGR algorithm inserts an invisible joint at the segment, allowing the movement of the whole hierarchy over the new pivot until the end-effector touches the ground or the obstacle (part *b* of Figure 8).

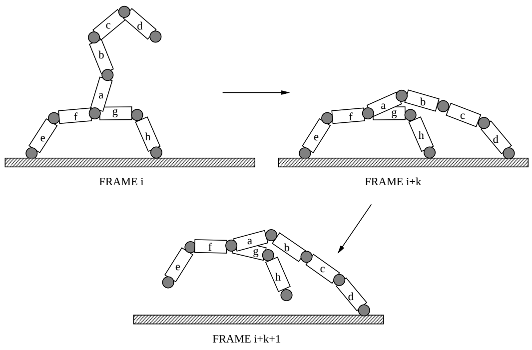

Figure 7: A DGR algorithm is used to drive the creature.

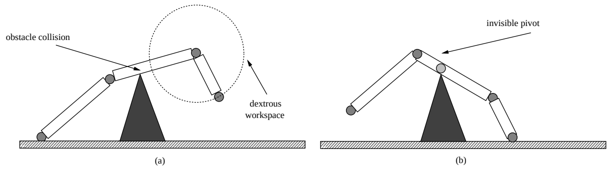

Figure 8: Avoiding singularity.

3.2.3 Motion Computation and Control.

Each node of the hierarchy has a data structure associated to it, which contains 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-the-Crab*, after motion parameters computation, the movement is simulated by propagating transformation matrices available at each node, using a standard forward kinematics procedure. In some cases, however, there is a need for direct end-effector positioning (the inverse kinematics problem). In this case, we adopted the real-time algorithm "Cyclic-Coordinate Descent" (CCD) [8]. To control the motion behavior of *The-Worm*, the *task control* module decomposes the goal into a set of small subprograms as in [9]. Each subprogram consists on a specific motion that *The-Worm* can perform (crawl, full extension and so on). These motions are generated by a series of local motor programs (LMP), each one responsible for changing a specific rotation parameter in a joint or in a set of joints. This process is constantly supervised by the Task Control module, which also evaluates the performance and needs of change in the motion parameters, based on the creature's perceptual state. All movements of *The-Worm* are parameterized such that each end-effector acts like a leg, propelling the creature throughout the environment.

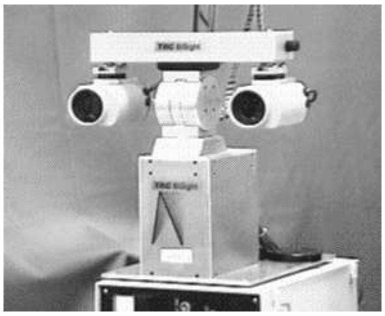

Figure 9: LPR/UMASS Stereo Head platform.

3.3 The Stereo Head Robot Platform

Our stereo head robot, shown in Figure 9, consists of two video cameras mounted on a TRC Bisight head providing four mechanical degrees of freedom: pan, tilt, and left and right vergences. Motion is controlled via a PMAC/Delta TAU interface. Images from each camera are input to a dedicated Datacube pipeline array (image) processor, which is mainly used to reduce and abstract data. As in *Roger-the-Crab*, a *multi-resolution* (MR) representation in four levels (column images in Figure 10) accomplishes data reduction. A *multi-feature* (MF) representation provides abstraction to support attention and identification behaviors. It is obtained by applying Gaussian partial derivative kernels, in two orthogonal directions each, to intensity images and to consecutive frames difference ($I_{t+1} - I_t$) representing motion. Pixel sampling is also performed, reducing the original 512×480 pixel images to 16×15 pixels. The resulting perceptual buffer is a MRMF representation as shown in Figure 10. The Datacube device can perform this data reduction and abstraction at 15 frames per second, thus allowing to achieve real-time performance necessary in our active vision framework.

3.3.1 Attention and Categorization Behaviors

The desired attentional behavior for the Stereo Head involves computing a salience map for each eye, taking the

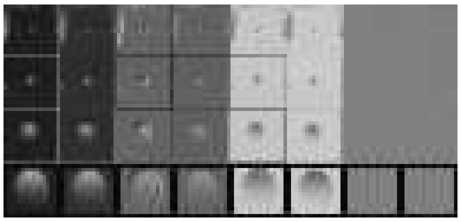

Figure 10: MRMF matrices.

winner region, and generating eye movements to foveate on that region. The salience maps are calculated from attentional feature maps generated from the MRMF representation (perceptual cues) and also by using other attentional feature maps which are empirically determined. A simple weighted summation of all attentional features gives the activation value of each position in the salience maps. Once a goal is determined the system computes a "coarse saccade" or the displacements for both eyes to focus on the winning position. Then, the parameters for the DOFs of the Stereo-Head are computed and sent through the PMAC interface which effectively produces the physical movements. After a coarse saccade, due to several sources of errors, the target may not be in the eye centers (fovea). Fine movements are performed iteratively for both eyes to bring the goal to the fovea.

Since attention is in focus, a set of features is computed from the Datacube output (MRMF) for categorization behavior. Our current approach uses local statistical features as *average* and *variance* plus stereo disparity and motion patterns, in the vicinity of sampled positions. Identification is done using an associative memory implemented by a multi-layer perceptron with a back-propagation training algorithm [1] (BPNN). The BPNN maps the features to an address in the long term memory (LTM) which stores various other information. If the representation is new, supervised learning is automatically invoked, inserting the new feature set into LTM and updating/retraining the BPNN. Once a representation is classified as new or identified, the current position in the attentional maps is updated and attentional features sufficient to detect on-line changes are stored. This update also enables a shift of attention to another region.

4. Experiments and Results

Experiments and demonstrations involving attention, identification, progressing, and combined tasks were performed for all three creatures to test their animated behavior. For the *Stereo-Head* and *Roger-the-Crab*, basically, many instances and types of objects are placed in their restricted environment. We expect that both creatures, initially without any knowledge, focus attention on all objects, learn their characteristics inserting a pattern representation for each one in memory, and incrementally construct/update their attentional maps. At

Appr	Att Shift	Eye/Arm Improv	Posit Ident	New Obj	Map Update
(1)	72	76	29	27	56
(2)	70	82	20	32	52

Table 1: Data obtained after 204 control-loops for Q-learning(1) and simple(2) approaches.

the end a monitoring behavior is adopted, searching for changes in the environment. For *The-Worm*, the goal is mainly to take decisions on where or how to move, based on its perceptual state.

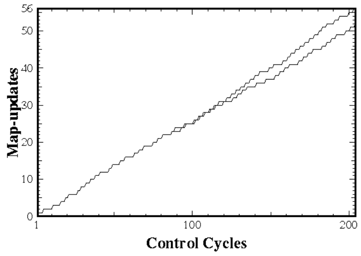

Figure 11: Partial evaluation.

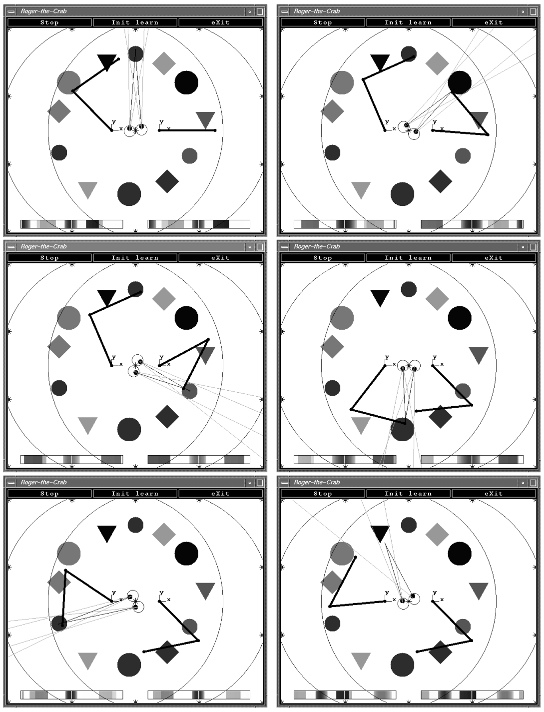

Figure 12: *Roger* constructing attentional maps.

4.1 Experiments for Roger-the-Crab.

Both approaches, Q-learning and the straightforward strategy, briefly discussed in subsection 3.1.2 have been used and the results have been acceptable. Figure 11 shows a partial evaluation of both strategies. Some analyses can also be made using table 1. Similar results were obtained from other experiments in which the system runs until no more un-mapped representations were detected. In these, all regions of interest in the environment were visited (looked at or touched by *Roger*).

Figure 12 shows some pictures selected from a sequence in which *Roger* visits (looks at and/or grasp or touches) all regions/objects in its environment. By the results we could see that our creature performed its animated behavior by using both methods. The architecture worked well in the attentional task, that is, the construction and maintenance of attentional maps of the environment.

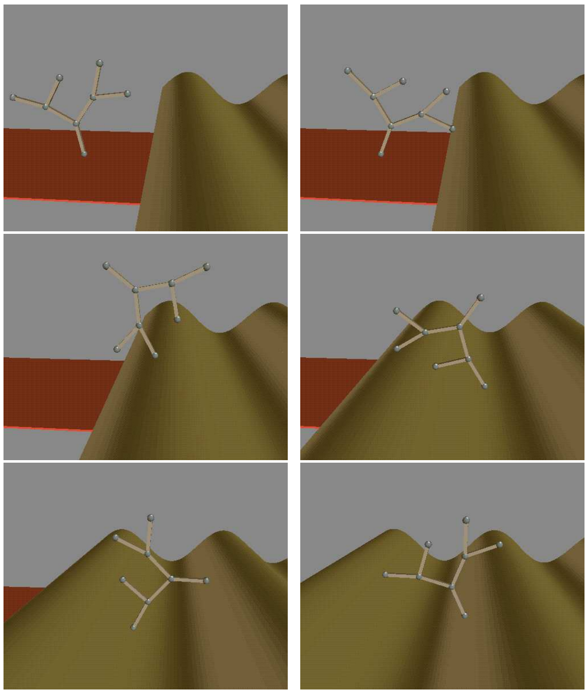

Figure 13: *The-Worm* progressing on irregular terrain.

4.2 Experiments for The-Worm.

Our simulated worm was tested in three different situations involving uneven terrains and obstacles. The goal of such testings was to evaluate the creature's skill to take reasonable decisions based on its perception of the surrounding environment.

In the example shown in Figure 13, a multi-branched Worm progress over an irregular terrain, with 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 its oscilattors phase to optimize its path).

Figure 14 shows the use of a DGR algorithm that acts based on the creature's perceptual state in order to avoid an obstacle.

Figure 15 shows *The-Worm* climbing a stair. In this case, the Task Control module has used features ex-

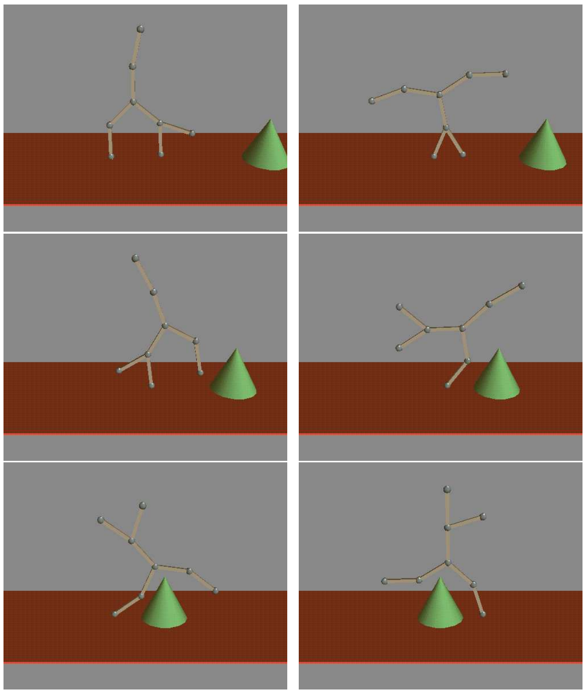

Figure 14: Obstacle avoidance using DGR and sensory information.

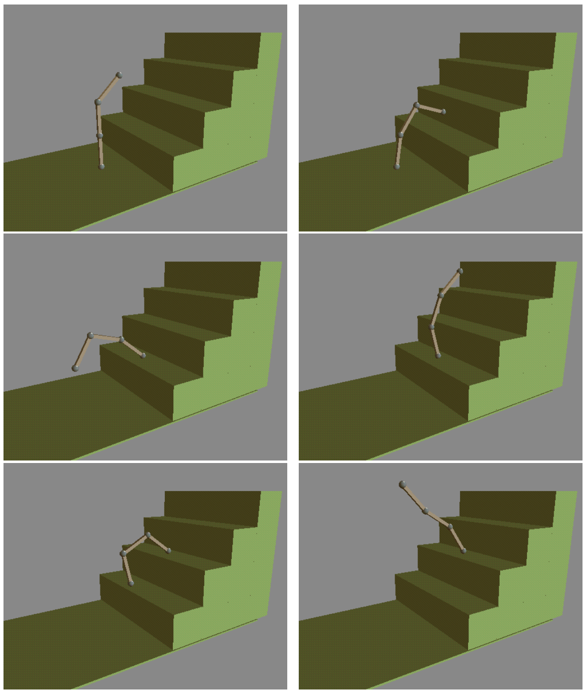

Figure 15: *The-Worm* climbing a stair.

tracted from sensory information to calculate the correct phase components for the oscillators which drive the creature. This allows a smooth stair climbing.

4.3 Experiments for the Stereo-Head.

Three types of demonstrations were conducted involving attentional behavior. In the first type, we indicate the objects to the robot by touching or pointing to them in a sequence. Figure 16 shows a sequence illustrating one of the experiments of this type. The stereo head goes after the motion cues, detecting all objects on the table. In the second experiment type, there is no motion cues, so the robot relies solely on intensity cues. Figure 17 illustrates one of the experiments. The attentional mechanism makes the robot visit all regions using the other perceptual cues. In the third experiment type, after all objects are inserted in the attentional maps, we either move or remove an object. The robot updates the attentional maps using mainly motion and intensity cues for changes (or movements) inside the field of view. If a change occurs outside field of view, the robot eventually returns its attentional focus to that region and detects the changes by comparing the current attentional features with stored ones. Figure 18 shows a sequence where the robot follows/track an object by virtue of differences observed in the working attentional maps. The attentional maps are rectified in each instant.

Figure 16: Robot following motion appeals.

5. Conclusion and Future Work

By using the architecture presented in this work, the three totally different creatures presented could be implemented without any significative modification of their structures. By using features abstracted from sensory information, we decrease the amount of computation necessary for on-line applications. Also, another important result was the development of an architecture which can also be applied to real platforms, decreasing the distance between simulation and reality. A new sensor can be easily attached to the system and the task set can be increased without needs of strong adaptation or substantial changes in the architecture.

We have chosen tasks like monitoring and progression on a rough terrain to test the platforms because the bottom-up aspects of attention can be fully explored in these tasks. Furthermore, a 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 dynamical attentional maps. As example of a more advanced and general task which involved feature extraction, attention, motion computation, and changes in the environment, the creature *The-Worm* used the basic architecture to navigate in its environment. Note that all it needs is a local perception of the environment provided by its simple sensors.

Figure 17: Attention without motion cues.

An immediate extension of this work is to increase the feature space and/or the set of tasks that the creatures can perform. We can also define other simple tasks based on the current feature space. Then, it would be possible to derive various policies, each one appropriate for a given task. In particular, we can apply reinforcement learning to determine those policies. Also, experiments involving insertion of a motion capture interface are planned. We hope that the motion captured from human actors combined with operations like motion warping would make our creatures perform in a more natural way, with smooth changes in its movement.

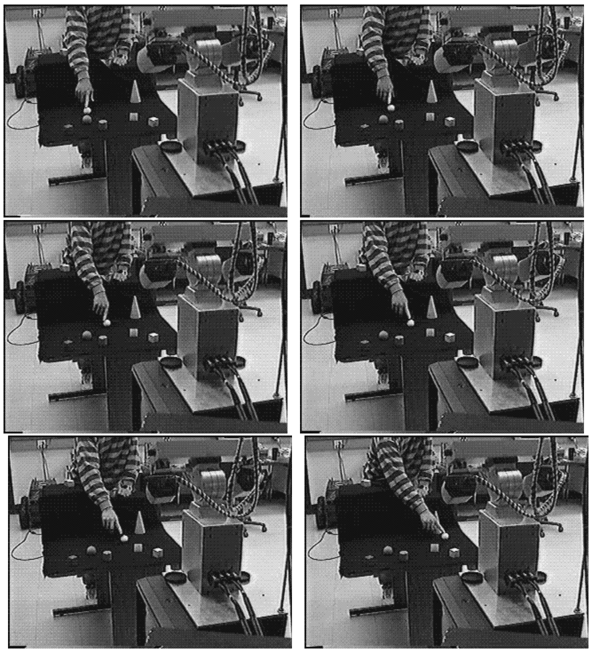

Figure 18: Tracking of a "golf ball".

References

[1] H. BRAUN and M. RIEDMILLER. Rprop: A fast and robust backpropagation learning strategy. *In Proc. of the International Conference on Neural Networks*, pages 123–134, 1993.

[2] C. I. CONNOLY and R. A. GRUPEN. On the applications of harmonic functions to robotics. *Journal of Robotic Systems*, 10(7):931–946, 1993.

[3] L. M. G. GONÇALVES, A. A. F. OLIVEIRA, and R. A. GRUPEN. Multi-modal stereognosis. *III International Conference on Autonomous Agents (Agents '99)*, May 1999.

[4] L. M. G. GONÇALVES, D. WHEELER, A. A. F. OLIVEIRA, and R. A. GRUPEN. Towards a framework for robot cognition. *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '99)*, November 1999.

[5] D. MARR. *Vision – A Computational Investigation into the Human Representation and Processing of Visual Information.* The MIT Press, Cambridge, MA, 1982.

[6] K. NISHIHARA. Minimal meaningful measurements tools. Technical report, Teleos Research, 1991.

[7] R. S. SUTTON and A. G. BARTO. *Reinforcement Learning: an Introduction.* The MIT Press, Cambridge, MA, 1998.

[8] L. WANG and C. C. CHEN. A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4):489–499, august 1991.

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