

Exploiting the Physical Agent/Environment Interactions to Evolve Neural Controllers for Autonomous Robots



Gianluca Massera (e-mail: gmassera@istc.cnr.it)

Institute of Cognitive Science and Technologies, National Research Council (CNR), Italy

Introduction

The behaviour of an embodied and situated agent always results from the dynamical interaction between the agents and the environment. When the physical aspects of the environment and of the agent are modelled accurately, properties emerging from

their physical interaction assume a prevailing role and can help the self-organisation of behaviour. In our experiments, the environment is not seen as a static element, but as a dynamic system with which the robot interacts with and modifies its parameters.

Evolved strategies are based on a continuous modification of the environment, which in turn produces dynamics from which the desired behaviour emerges. In this poster, I present some evolved strategies that

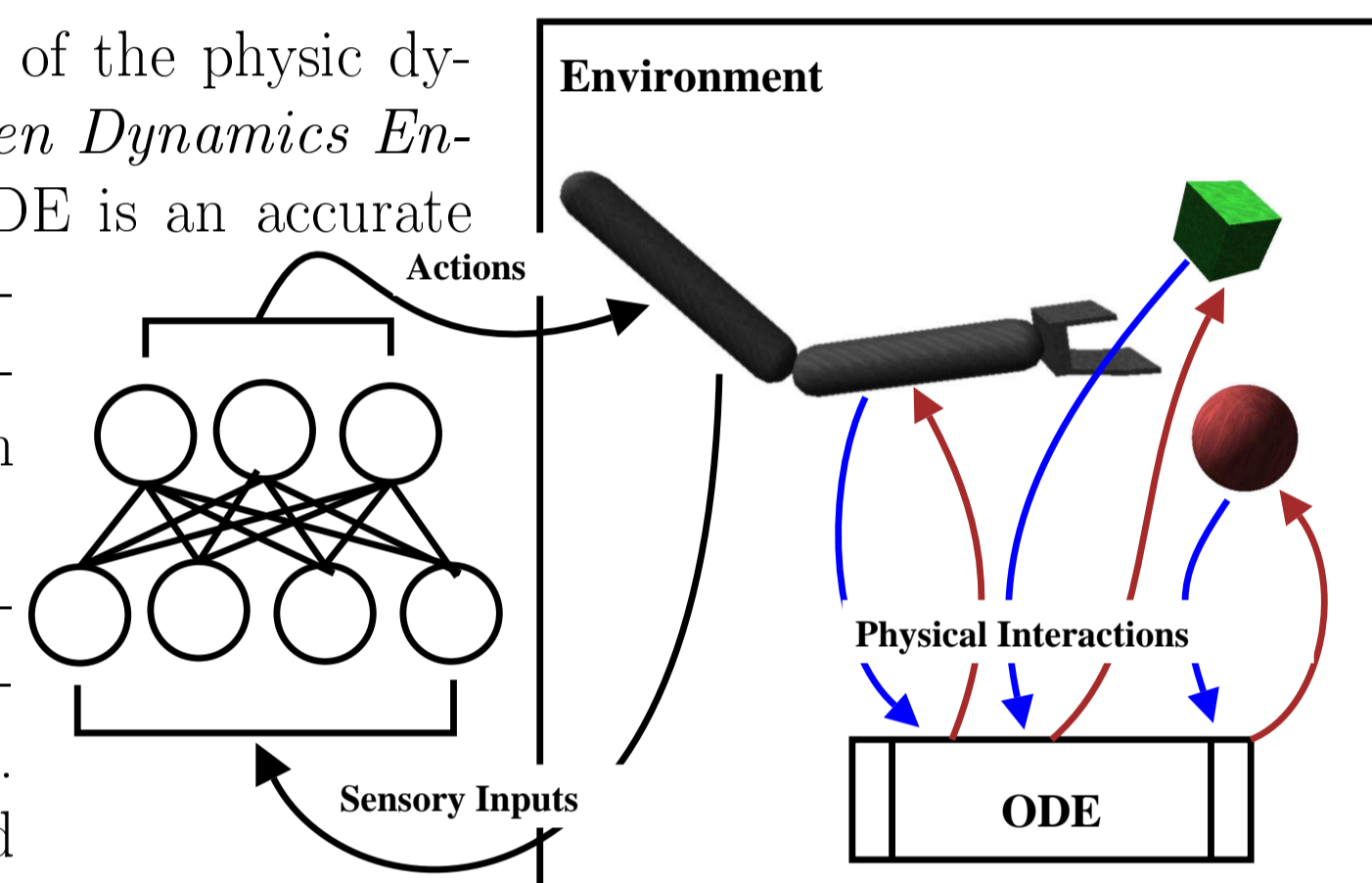
control an autonomous robotic arm able to grasp objects. In this task, the target object does not have a fixed position in the environment, but the robotic arm can move it. Therefore, the dynamics of collisions between the arm and the target object

play an important role in the accomplishment of the requested task. The evolved neural controllers exploit the physical arm/object interactions to avoid the collisions the cause the object move away, and to orient the target object for a stable grasping.

The Environment

The exploitation of the physical agent/environment interactions requires an accurate modelling of the physical dynamics. We used the ODE library (*Open Dynamics Engine*) to simulate the environment. ODE is an accurate library for simulating rigid body dynamics. It has advanced joint types, and integrated collision detection with friction and gravity forces.

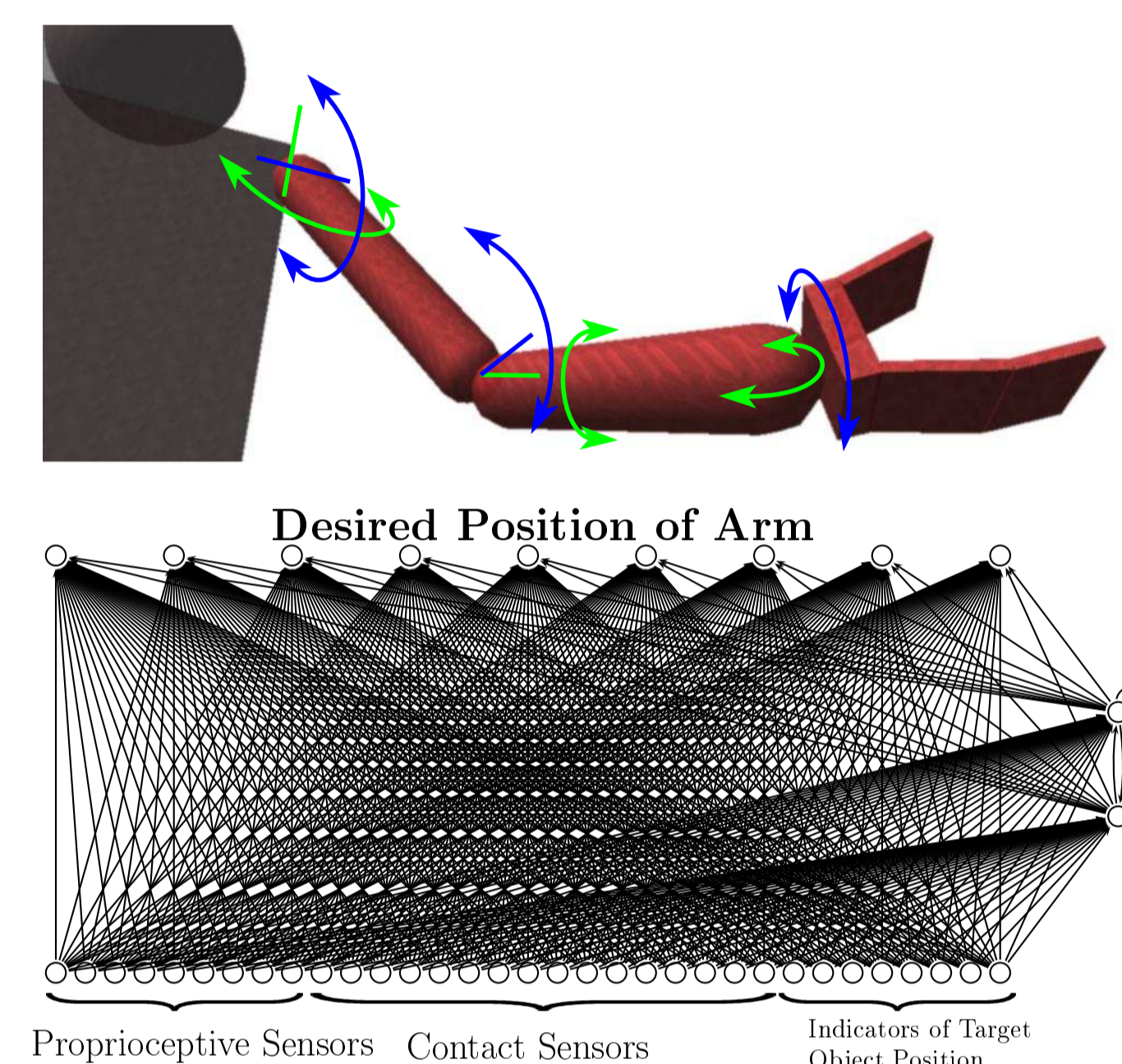
In the experiment setup, the neural network drives the motors of the arm and receives information by sensors on the arm. From the sensory inputs, the evolved neural controllers are able to extract useful information about physical interactions that occur. Consequently, they tune the behaviour on the basis of this information.



The Model

The Robotic Arm consists of various objects articulated by joints. Attached on the shoulder there is the first cylindrical section with 2 DOF joints. Next is the forearm, attached to the previous section with 2 DOFs. The wrist is joined with the forearm with 2 DOFs. The hand has two fingers: one thumb with 1 DOF and one index opposite to the thumb with two segments (2DOFs). In total, the arm has 9 DOFs, 6 joints and 6 components.

The Neural Controller is a feedforward neural net with two hidden recurrent neurons. The input layer is composed by 9 proprioceptive neurons coding the current position of the joints (one neuron for each DOF), 16 contact sensors are distributed on the fingers and forearm, and 8 indicators of the target object position. The output layer consists of 9 neurons coding the desired position of the joints (one neuron for each DOF).



The Task and Genetic Algorithm

The Task: "Grasp the Object!"

Each individual was tested for 10 trials, each consisting of the following phases:

- **Starting Position:** The arm is positioned as illustrated. However, the exact initial position randomly varies from one trial to the other (within a 2% range).
- **First Four Seconds:** This time is granted to the neural controller for reaching and grasping the target object.
- **Last Two Seconds:** The object-holding table is removed. The target object will fall down if the robotic arm does not grasp it well. This way, we can test the effectiveness of grasping, by measuring the time elapsed before the target object falls from the hand of the robot.

The Fitness is the average of the performance, evaluated during each of the ten trials. The performance of a neural controller in each trial is the sum of the following test results:

- **Contacts Test:** returns the number of contacts between hand and target object at the current time step.
- **Target Position Test:** returns a number proportional to the amount of the volume of the target object that is inside the hand at the current time step.
- **Grasp Test:** is executed during the last two seconds of simulation. It returns a high number (10000) if (i) the target object is inside the hand in contact with the fingers and (ii) the relative velocity between the object and the fingers is about zero for 25 consecutive time steps.

Selection and Reproduction: the twenty best individuals are selected according their fitness ranking. We then apply the mutation operator to create the next generation. The mutation is implemented flipping every bit of the genotype string with 1% probability.

Evolved Strategies

Grasp the Sphere. The orientation of the sphere is irrelevant for successful grasping, but the low friction with the table is critical because even a weak hit causes the sphere to move away. Therefore, the evolved strategies are characterised by movements calibrated in such a way that the sphere does not move away from hand. An important role is played by the inertia of the sphere. The neural controller must take into account the inertia for better calibrating the forces applied on the sphere by the arm.

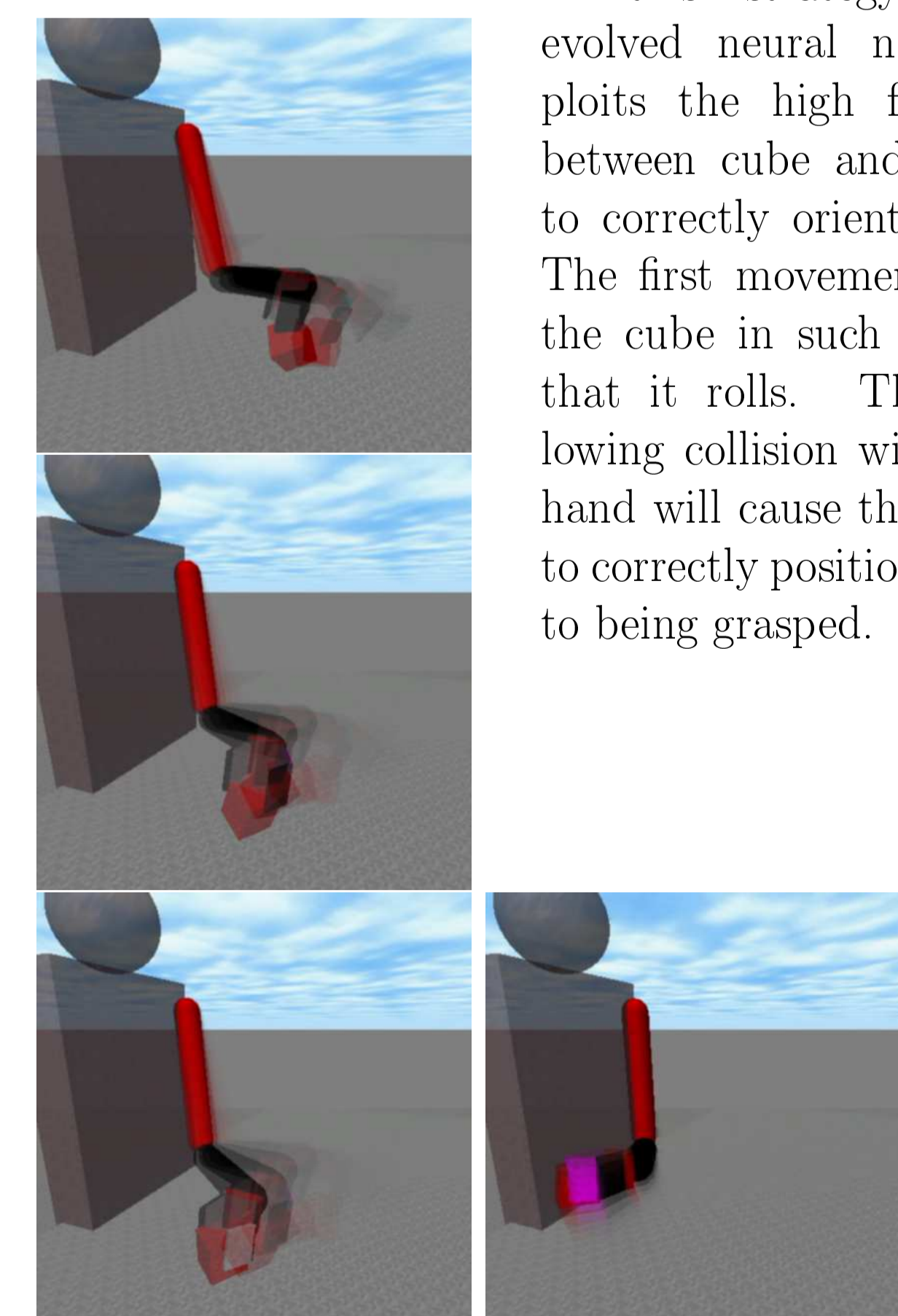
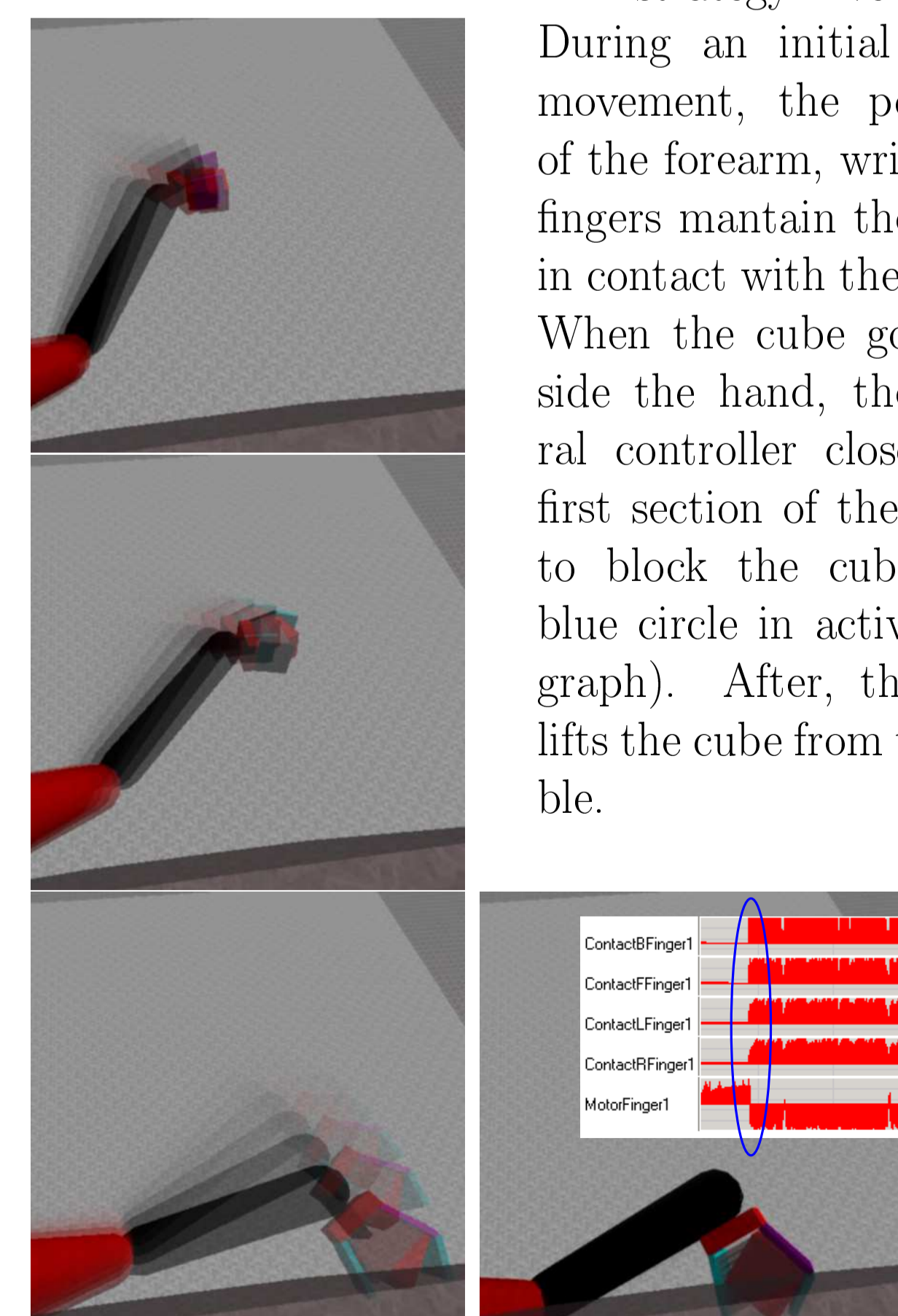
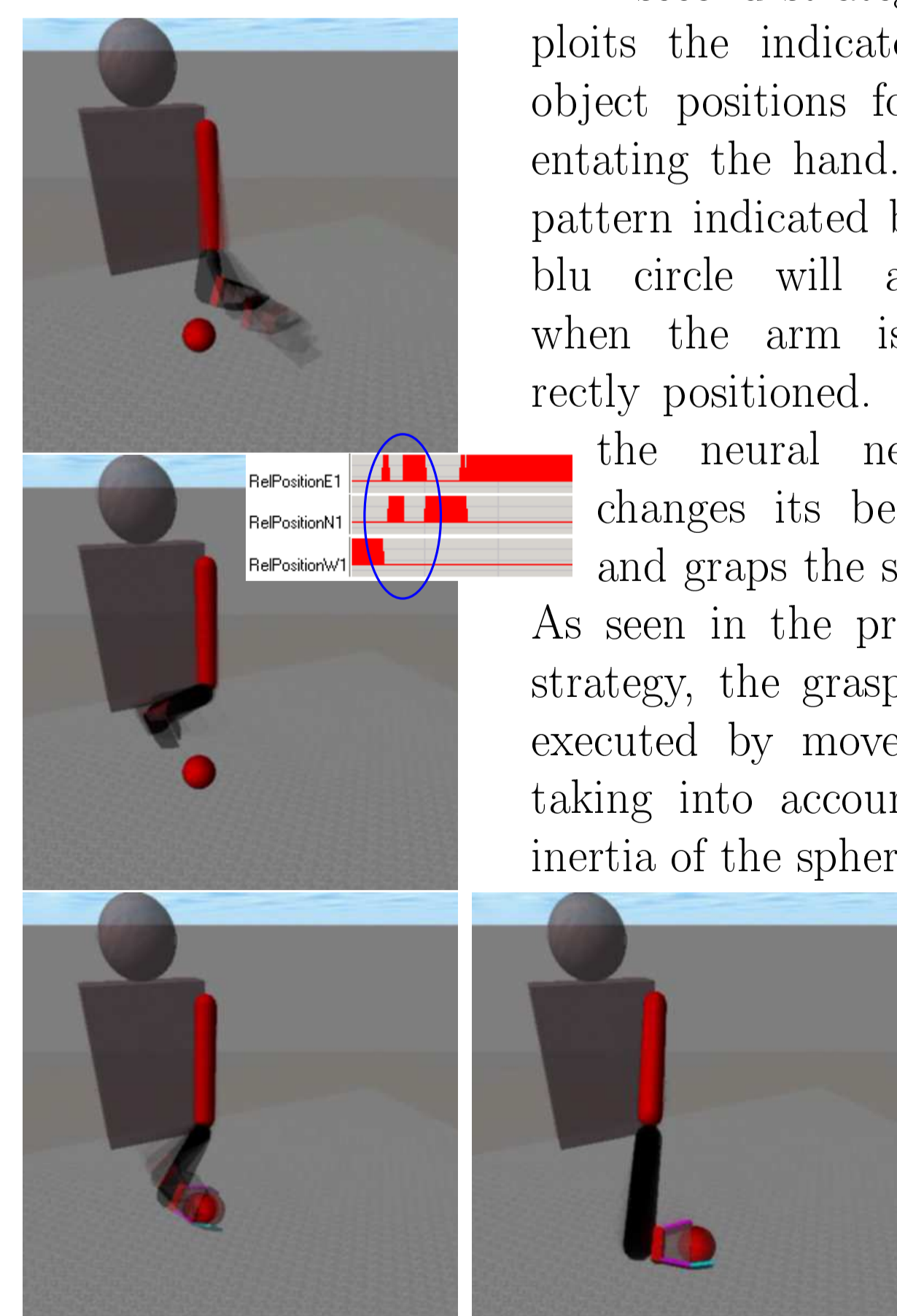
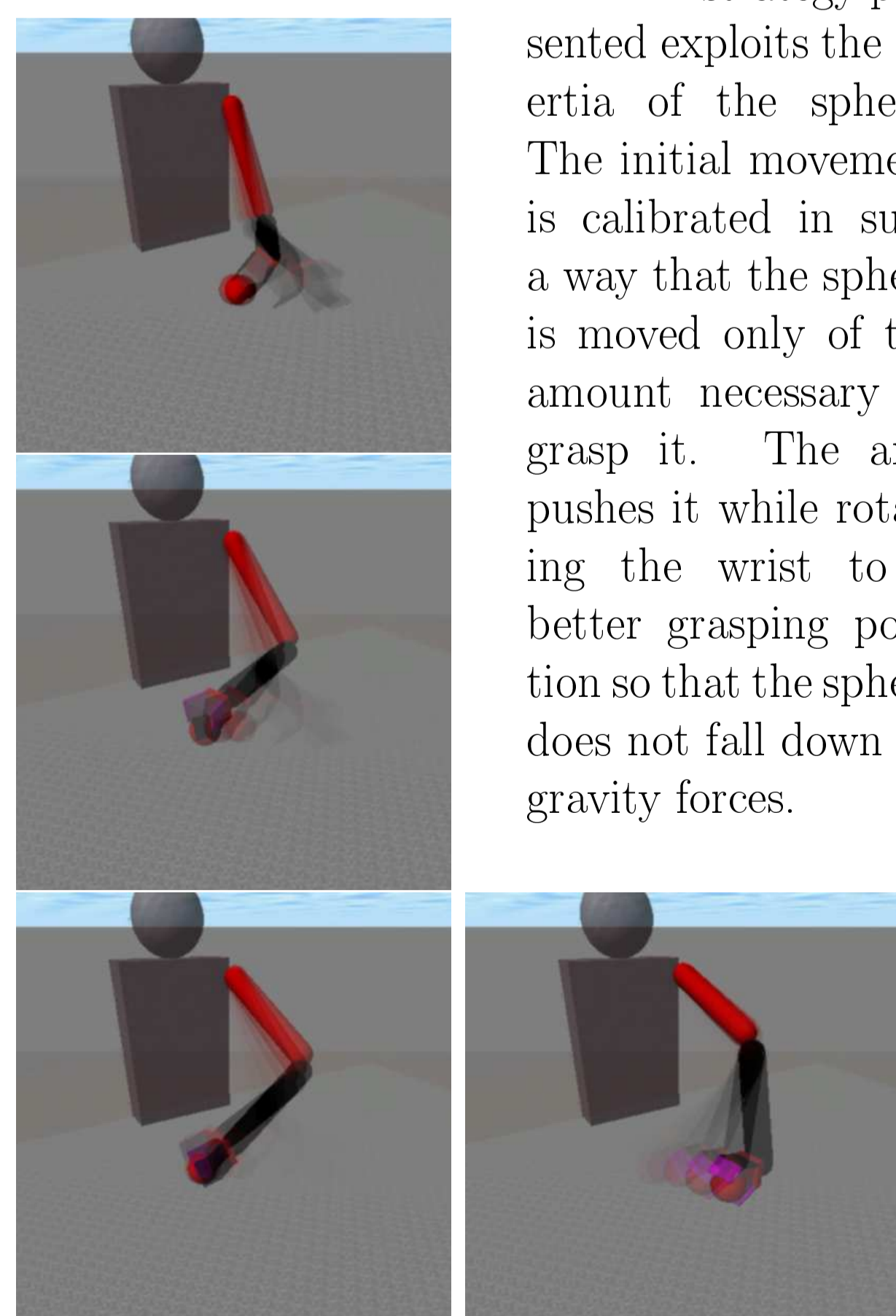
The first strategy presented exploits the inertia of the sphere. The initial movement is calibrated in such a way that the sphere is moved only of the amount necessary to grasp it. The arm pushes it while rotating the wrist to a better grasping position so that the sphere does not fall down by gravity forces.

This second strategy exploits the indicators of object positions for orientating the hand. The pattern indicated by the blue circle will appear when the arm is correctly positioned. Then the neural network changes its behavior and grasps the sphere. As seen in the previous strategy, the grasping is executed by movements taking into account the inertia of the sphere.

Grasp the Cube. Differently to grasping a sphere, the cube has to be oriented correctly before grasping it. Otherwise, the gravity pulls it away from the hand. In this case the friction can help the neural controller to orientate the cube correctly. When the cube is hit, it does not slide. Instead, it rolls away by changing its orientation. Therefore, the evolved neural controller exploits the high friction of the cube. This way, the arm hits it in the appropriate points so that it can correctly orient the cube and grasp it.

This strategy is very fast. During an initial rapid movement, the position of the forearm, wrist and fingers maintain the cube in contact with the hand. When the cube goes inside the hand, the neural controller closes the first section of the index to block the cube (see blue circle in activations graph). After, the arm lifts the cube from the table.

In this strategy, the evolved neural net exploits the high friction between cube and table to correctly orientate it. The first movements hit the cube in such a way that it rolls. The following collision with the hand will cause the cube to correctly position itself to be grasped.



Conclusion and Future Researches

- When the physical aspects of the environment and of the robot are modelled accurately, the physical properties of the objects and of the robot become critical parameters for the accomplishment of the required task. In our experiments, the evolved individuals are able to take account of the physical properties of the objects (e.g. inertia) without any dedicated neural input.
- In this case studied, we evolved autonomous robots able to grasp simple objects (sphere and cube). The task is simplified by fixing the initial position of the target object. These choices helped us to better tune the physical parameters of the environment (frictions, velocities, etc.).
- In future, we plan to evolve controllers able to grasp objects of different sizes and shapes, at different initial positions. We intend to add some additional sensory information to help the finding of the target object in the environment by neural controller. An example is to use visual information.