Closing the loop: Evolving a model-free visually guided robot arm

Thomas Buehrmann[1], Ezequiel Di Paolo[1]
[1]Centre for Computational Neuroscience and Robotics
Department of Informatics, University of Sussex
{t.buehrmann, ezequiel}@sussex.ac.uk

Abstract

Dynamic neuro-controllers are incrementally evolved for reaching and tracking movements by a physically simulated robot arm. An active vision system capable of controlling gaze direction and focus replaces the need for internal models of the robot. It is shown that closing the feedback loop allows for robot control being robust to changes in environment, sensors and robot-morphology.

Introduction

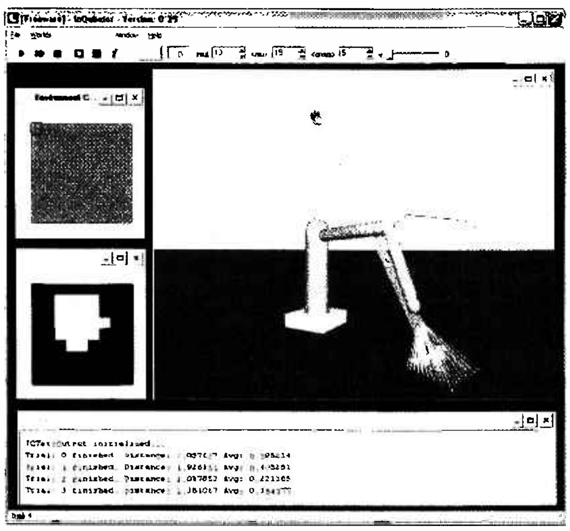
The task of controlling a robotic arm can be defined as the movement of the end-effector to visually identified positions or along particular trajectories in workspace. For visually guided movements, computational approaches traditionally consist of the following stages: i) a visual observation of the target is translated to desired position and orientation of the robot's end-effector by using visual pre-processing, feature extraction, inverse perspective projections or other computer vision based methods; ii) inverse kinematics is used to translate the desired end-position into a set of desired joint angles; iii) trajectory planning: a path, i.e. a series of intermediate points is calculated in joint space along which the robot moves from its current state to the desired state; iv) actuator commands (e.g. torques) are calculated which when applied to the arm move the end-effector along the desired intermediate points (dynamics problem). All of these transformations depend on using a correct model of the robot. For accurate control, these models must be constructed and calibrated so that they correspond with the parameters of the real robot. This introduces several problems. First, supervised learning schemes (Massone, 1998; Jordan and Wolpert, 1999) are difficult to apply if obstacle avoidance or other non-trivial behaviors are to be taken into account. Representative input-output samples describing the correct behavior are hard to produce in this case. Secondly, these controllers rely on static configurations of their environment as well as their sensorimotor interfaces. Because they use models calibrated to a particular environment and robot morphology, they can't adapt to changing environments or bodily reconfigurations. This rigidity leads to high costs of maintenance and calibration. Lack of flexibility can be solved either by re-calibrating the internal model during the robot's lifetime, or alternatively by foregoing the use of a model at all. Instead, the robot-environment feedback loop can be closed and a controller constructed which translates directly from sensory input to joint dynamics (e.g. (van der Smagt, 1995)).

This report deals with the second alternative. It shows how a minimalistic approach inspired by evolutionary robotics and the dynamical systems perspective allows for flexible and robust robot control without using prior knowledge about the robot and its environment in form of internal models. In summary, neural controllers are evolved to use low-level active vision for the guidance of a physically simulated robot arm. By using a genetic algorithm, the robot controller is evolved to directly translate from the visual domain to joint dynamics in order to reach for and track objects in its environment. Since the controller does not use any models of the robot or its sensors, there is no need for calibration. Instead of computationally expensive processes involving filtering, edge finding, feature extraction, flow field analyses or transformation of data into world-based frames-of-references, this system relies on active and egocentric low-level vision only.

Methods

Robot platform

The robot arm in this project is an articulated model that consists of three segments linked by 1-dimensional rotational joints which are controlled by angular motors. The motors apply torque to a joint in order to pivot it at a desired speed. In the experiments described, torque limits are set and neural networks control the arm by specifying the desired velocities for each joint. Also, the range of motion of two connected bodies is limited by setting stops on the joint (maximum and minimum angles). Calculation of the physics (including gravity and inertia) is provided by ODE (http://opende.sourceforge.net). Figure 1 shows the configuration of the robot arm. To avoid both complex image processing (as e.g. in triangulation from stereoscopic



```
ICTextOutput initialised...
Trial: 0 finished Distance: [illegible] Avg: 0.395414
Trial: 1 finished Distance: 1.926[illegible] Avg: 0.4[illegible]5281
Trial: 2 finished Distance: 1.017053 Avg: 0.221365
Trial: 3 finished Distance: 1.161017 Avg: 0.1[illegible]
```

Figure 1: Robot arm, laser range sensors, and camera images.

images) as well as incorporation of prior knowledge (as in heuristics on the relation between image- and actual size of objects) a rather atypical sensor system was used to provide 3-dimensional visual information: a minimal camera system consisting of a two-dimensional array of "laser range sensors" (figure 1). Each individual sensor is a light-ray whose output is proportional to the distance of its collision with an object in the world. A number of such rays is arranged in a rectangular grid. The rays all originate at the same focal point in space and the angle between them determines the camera's field-of-view. By changing this angle, a camera effectively has an adjustable focus and thus control over image resolution[1]. In fact, two such cameras were used in this project. The first one is a world-based camera mounted above the robot arm. Its position is fixed, but it is able to move in two degrees-of-freedom: pan around the vertical axis and tilt around its local x-axis. The second camera is mounted to the end-effector (eye-in-hand, or egocentric setup). The advantage of this camera is that the information it provides is relative to the end-effector and can be used in the final approach to target objects even when the arm occludes the first camera.

In addition to visual input, controllers also have access to proprioceptive information. Angles of the three robot joints as well as the orientation of the camera can be used by the modular neurocontrollers.

Neuro-Controller

Continuous-time recurrent neural networks are used for the neurocontrollers. The state of each node is described by

$$\tau_i \dot{y}_i = -y_i + \sum_{\forall j} w_{ji} \phi_j (y_j + \vartheta_j) + g\mathcal{I}_i(t) \tag{1}$$

[1]A similar active vision system was evolved in (Kato and Floreano, 2001) to perform shape discrimination

where y_i is the cell potential of that neuron, τ_i its time constant, w_{ji} the weights of incoming synapses, ϕ the sigmoidal function $\phi(x) = 1/(1 + e^{-x})$ calculating the firing rate, ϑ the threshold of the neuron and $g\mathcal{I}$ gain-scaled input respectively. The parameters for each neuron are obtained from appropriate scaling of elements in the genotype (distributed over the range [0,1]). Weights and biases were scaled to the interval $[-4, 4]$, time constants to $[0.1, 10]$ and input gains to $[0, 10]$. The Euler method with a time step of 0.2 was used for integrating the differential equations.

Genetic Algorithm

A genetic algorithm was used to evolve fixed network architectures. A linear ranking selection scheme and stochastic universal sampling were used for reproduction. Also, elitism was applied by always keeping the best individual of each generation. Recombination is realized through an ordinary two-point crossover operator. The particular form of mutation used is a variation of the *creep* operator determined by two parameters: one specifies the maximal amount of mutation for all components while the other one determines the probability of mutation for individual components. Mutated values are clipped to the interval $[0, 1]$.

An incremental approach to evolution was used in three different ways. First, the desired behavior was decomposed into several independent behavioral competencies. The overall system was then partitioned into sub-modules which are individually evolved to produce one of the more basic behaviors. Secondly, some of the modules were evolved to produce solutions to a series of increasingly complex evaluation tasks. This was done to avoid local minima of unsatisfying solutions when the initial search space was too big. Finally, evolutionary parameters (such as mutation probability) sometimes were interactively decreased throughout evolution to allow for the population to converge on and optimize the best solution it had found so far.

Fitness Evaluation

The performance measure to be maximized by the controllers consists of mainly two terms. First, those controllers receive higher fitness that reduce the Euclidean distance between the robot end-effector and the target from the beginning of a trial to its end. From the distance at time t (d_t) and the distance at the start of the trial (d_0) this fitness value is given by $f_d = \frac{1}{T}\sum_{t=0}^{T}(1 - \frac{d_t}{d_0})$ where T is the total number of time steps per trial (in the case of the camera trying to center the target, angular distance between direction of view and direction of target was the measure used). The second fitness term tries to minimize movement at the end of the trial so that the robot arm finally comes to a stop: $f_v = 1 - \frac{1}{\dot{\theta}_{max}}\sum_{i=1}^{3}(\dot{\theta}_i)$, where $\dot{\theta}_i$ are the absolute velocities of all joints and $\dot{\theta}_{max}$ the maximum velocity. Although breaks will be used in some experiments at the

end of movement, smooth deceleration towards targets is more desirable. Usually, this term is multiplied with the distance measure at the end of a trial, such that high fitness values can be achieved only if the robot arm simultaneously gets close to the target and has minimum velocity.
Each individual in the population is tested in several trials of fixed length. At the beginning of each trial the robot arm and the external camera are initialized to their resting positions, while the target is placed randomly in a cubic volume within the arm's workspace. The overall fitness of a controller is calculated after its last trial by averaging its individual trial fitnesses.

Experiments

Preliminary experiments without the visual system had shown that a single feedforward CTRNN can be evolved to move the robot arm in ways appropriate for reaching and tracking (using Cartesian coordinates of end-effector and target as well as joint angles as inputs). When cameras were included however, the system was broken down into four neuro-controllers which were evolved successively. The first one enables the external camera to find and centre on objects within its visual field. The second module controls the horizontal orientation of the robot arm and makes it align with the camera's direction of view. The third controller is responsible for finding and closing in on the target in the vertical plane, while keeping the orientation of the end-effector such that the target could actually be grasped. An optional fourth controller is connected to the eye-in-hand camera, and allows the arm to track an object even when it is occluded by the arm on the image of the external camera.

External Camera

The external camera has a global view on the scene, and if it's able to centre on the target, its angular position and the distance information of its sensors can be used to inform the arm about the location of the target. The neural network controlling this camera (figure 2) was designed taking into account the symmetries (vertical and horizontal) of the task as well as the sensor arrangement. Through this 'quadlaterally symmetric' architecture, incorporating knowledge about the task (not the robot or its environment), the control problem is simplified and evolutionary search made easier. Parameters of the 104 neurons in the controller were encoded by 42 real numbers (in addition to symmetrical connections, biases and time constants were shared extensively). The initial angle of the camera's focus and the random positions of the target were initialized such that the target was always located within the camera's field-of-view. However, since the space between sensory rays increases with distance from the focal point, objects sometimes lay in between the rays, thus not producing any input. An individual's fitness was equal to the averaged fraction of

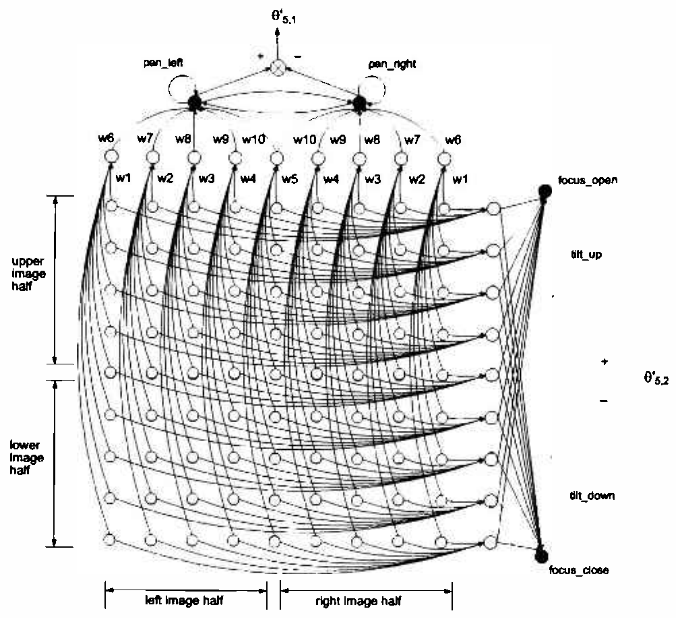


Figure 2: The network has one input neuron per distance sensor of the camera. All input neurons in a given row or column project to a hidden neuron using the same weight. Connections from the left half of the image are mirrored by connections from the right. Also, rows and columns share the same weights. The hidden neurons symmetrically project to three pairs of output neurons which control the camera's pan and tilt angle as well as its focus. To allow for richer dynamics the pairs of output neurons are interconnected fully recurrent.

angular distance covered from the beginning to the end of a trial (f_d). After only a few generations valid solutions were found for reaching for static targets. Near optimal fitness was achieved by generation 100. The best individuals from the last 100 generations on average received 98.2% of the maximum fitness, the population average 89.1%. For static targets without interference of the arm, the camera within a few steps reduces the angular distance to less than 0.1°. The focus is used by the controller to adjust the camera's resolution to the size of the target. While the camera has a tendency to close the focus, the outer sensory rays are used to interrupt this behavior. Consequently, the rays optimally cover the surface of the target object. This behavior is independent of the size or the shape of the target. The tendency to reduce the angle between individual rays has another advantage. If a target initially is positioned between some of the rays, the focusing behavior makes it likely that one of the rays will eventually intersect the object. The evolved focus mechanism turned out to be essential to achieve a high precision centering response, as well as a robust way of avoiding distraction through the robot.

Controlling Arm Orientation

Orienting the robot arm is easy if the active external camera is centred on the target. In this case, all the neurocontroller has to do is to use the angles of the robot and the camera to reduce the distance between both. The basic network evolved for this task is shown in figure 3. A power-switch

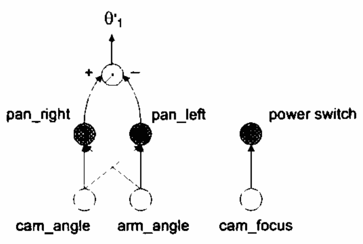


Figure 3: Module controlling rotation around vertical axis.

is used to allow movement (horizontal rotation) of the robot arm only if the output of the corresponding neuron exceeds a threshold of 0.5. The controller can thus decide when to start and stop moving depending on its input (e.g. the camera's focus). With this information available, controllers can evolve which wait until the camera has found the target before making the arm interfere with its sensors. Given a population size of 60, a valid solution was existent in the first randomly initialized population. Using the distance between arm and target as well as the arm's velocity at the end of a trial for the fitness function ($F = f_{d_a} * f_v$), the averaged performance of the last 10 generations was 99.94% of the maximum for the best individual and 96.93% for the population average. In 100 trials, the best controller of the last generation on average reduces the angular error to 0.56° (with a standard deviation of 0.14°). The velocity of the arm at the end of a trial is negligible. It can also be observed, that the arm only starts to move after the camera has focused on the target.

Reaching

In the next step, the eye-in-hand camera was included for approaching the target in the vertical plane. The task to be solved consists of two parts. First, because in the initial state targets will most likely not be located within the field-of-view of the eye-in-hand camera, the arm has to be moved to a position from which the object can be perceived in the first place. Such a behavior necessitates spontaneous internal dynamics of the controller, since no sensory information will be present at this stage (possible in CTRNNs through (self-)recurrent connections and non-zero biases). Once the object intersects the internal camera's sensors, the arm can be guided towards the target position. After experimenting with different neural architectures, it became apparent that the two joints can be assigned different roles. While the third joint (controlling the limb to which the camera is attached) mainly has to centre the target on the camera's field-of-view, the second joint can and must be used to approach the target. This task-decomposition inspired the neural architecture shown in figure 4. The

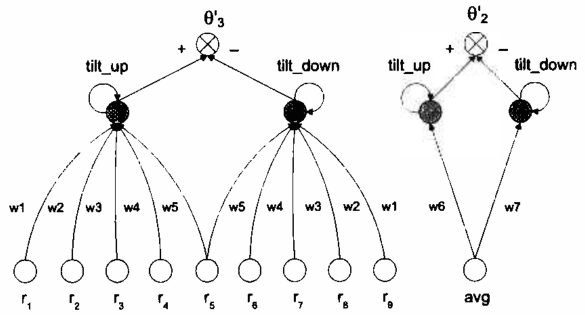


Figure 4: Neuronal modules controlling second and third joint.

idea is analogous to the controller for the external camera. Instead of having an array of input neurons however, a hidden neuron directly receives the average of all sensory activations from one row of the camera array. Also, because of the task decomposition employed, all information needed for controlling joint j_2 is the relative distance of the target. Directional information is only needed for centering the target(j_3). Hence, the module controlling θ_2 only receives the overall average of sensor activation.

The controller was evolved using a fitness function consisting of several different terms. First, throughout the trial an individual was rewarded for maximizing the average sensor activation so as to favour individuals which came as close to the target as possible. Secondly, controllers were rewarded for reducing the angular distance to the target. This term enforced the arm to approach the target full frontal rather than from an angle which would not allow to actually grasp the object. This was necessary because there were no additional degrees of freedom in the wrist which could be used to independently orient the hand relative to the target. Third, controllers were punished whenever their joint angles reached their limits. This way, individuals were filtered out which took over the population by producing stereotypic movements. Finally, at the end of a trial the product of the terms for reducing angular distance and minimizing velocity was added to the fitness. Individuals which did no move at all or got stuck on the robot base or the camera fixture were assigned a fitness of zero.

A first evolutionary step produced controllers which made the arm centre the target on the image of the camera but failed in actually approaching it. Also, it produced small oscillations around the direction of the target. To make the arm approach the target, in a second incremental step the best solution so far was further evolved using a fitness function having larger weights on the absolute distance covered and the final velocity. Additionally, another power-switch was included which depending on the averaged activation level of the camera could stop the second and third limb from moving.

From the trajectory in figure 5 and figure 6 it can be seen that the distance is now minimized in each of the three

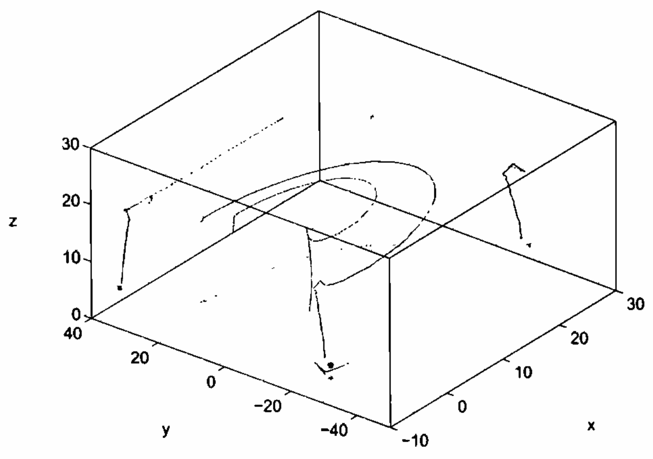


Figure 5: Positioning task: trajectories of end-effector and elbow joint and projection on three planes. Dots mark the position of the target. End-effector and target are initially positioned on opposite sides of the robot base along the y-Axis.

dimensions. In 100 trials, the average Euclidean distance at the end of a trial was 0.24 (which is roughly 1% of the length of the arm). The behavioral strategy is as follows: after the target is found ($t \approx 90$), the arm rotates and lifts its second limb while slowly lowering the third limb. As soon as the target object enters the eye-in-hand camera however ($t \approx 220$), the second limb is now lowered in order to approach the target, while the third limb centers the object (oscillating movements). The change in camera activation during approach leads to deceleration of the robot arm towards the target.

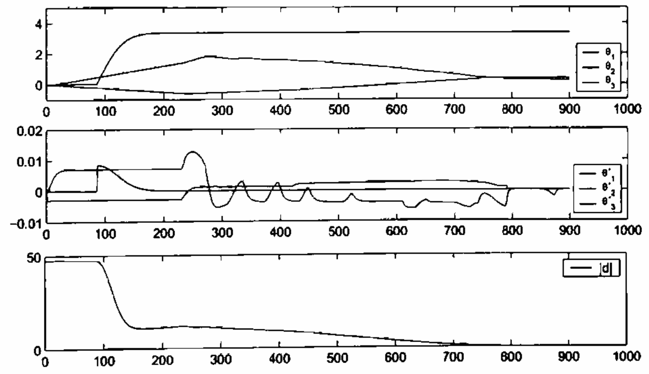


Figure 6: Joint angles(top), joint velocities(middle) and distance from the target for the same positioning task.

At time step 800 the controller uses the power-switch to stop the arm from moving.

Tracking

Since the arm is almost perfectly aligned with the external camera and thus occludes the target object whenever it is close to it, object tracking can not be guided by the external camera. Instead, the pan angle has to be determined by the eye-in-hand camera. Consequently, to evolve tracking behavior, another controller was introduced. Its connectivity is similar to the controller depicted in figure 4 on the left. It gets as inputs the averaged activations from individual columns in the sensory array. This controller however has feedforward connections only, and as a consequence of its symmetry does not produce any output if no input is available. This property in turn allows for a simple (hand-coded) override mechanism coordinating the two modules which influence the pan angle (j_1). As long as there is no output from the new controller (in absence of the target), the corresponding angle is controlled by the external camera. As soon as the eye-in-hand camera has found the target however, the resulting input drives the new controller and its output is used instead. The final evolved behavior of the robot arm trying to track moving objects is shown in figure 7. Clearly, the arm is now able to follow moving targets in all three dimensions.

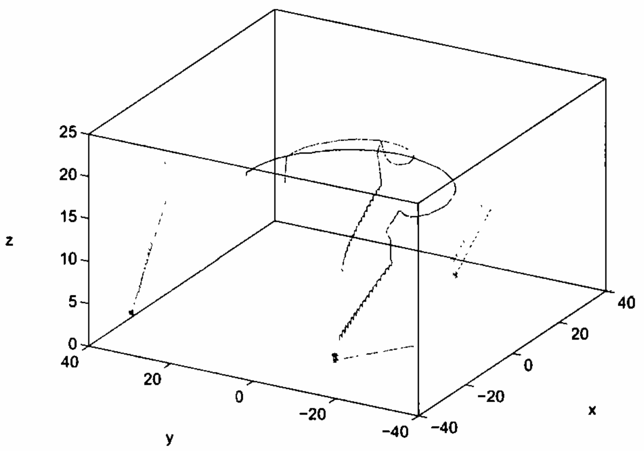


Figure 7: Trajectories and projections for a tracking task.

Robustness

The experiments described so far showed that modular neural networks can be evolved to implement robot controllers which are able to position the end-effector appropriately for grasping visually identified objects. The main advantage of such a controller is that it is not based on calibrated models of the robot, the sensors or the environment. Hence, it should be robust to all kinds of alterations in these factors without the need to re-evolve. In order to check for this property, the final controller was tested on a series of different conditions. Table 1 summarizes the average results from 13 tests of 100 trials each. In the first three tests the diameter of the spherical target object was varied. Obviously, the smaller the target, the higher the precision. Two reasons can be given for this result. First, a smaller target leads to a smaller angle of the external camera's focus, and hence to a better resolution and precision in its centering response. Second, if all sensors are equally activated (very close to

Condition	μ_{end}	σ_{end}	μ_{avg}	σ_{avg}
$\varnothing_t = 2.0$	0.35	0.69	13.76	2.62
$\varnothing_t = 1.5$	0.24	0.3	11.26	1.73
$\varnothing_t = 1.0$	0.22	0.16	11.50	2.34
no switch	0.43	1.12	11.81	2.48
$1/2 * \dot{\theta}_{max}$	0.86	1.61	20.12	3.02
$1/2 * \dot{\theta}_{max}$, g/9.0	0.2	0.15	22.52	4.34
$2 * \dot{\theta}_{max}$	6.17	5.51	12.32	5.51
IO-input	0.3	0.59	11.13	1.75
$\lvert l_3 \rvert + 3.0$	0.27	0.24	10.9	1.19
$\lvert l_3 \rvert + 6.0$	0.47	0.32	10.75	1.11
$\lvert l_2 \rvert + 10.0$	0.44	0.29	18.05	1.79
$\lvert l_2 \rvert, \lvert l_3 \rvert + 10.0$	0.57	0.40	16.66	0.88
$\lvert l_2 \rvert + 10.0, \lvert l_3 \rvert + 5.0$	0.50	0.36	16.96	1.36
g/9.0	0.31	0.27	12.8	3.12

Table 1: Performance measured over 100 trials. μ_{end} and σ_{end} denote the mean and standard deviation of the distance measured at the end of the trials. μ_{avg} and σ_{avg} are the corresponding values averaged over all time steps per trial.

the target), the arm exhibits no centering response anymore and is being lowered until some rays do not intersect the target anymore. Now, the bigger the object, the bigger the range of movement the arm can produce until the centering response is elicited again. Consequently, there is a bigger variance in positioning and hence a lower precision (if the target gets too small however, the external camera will have problems finding it). In another test, the functionality of the power-switch was deactivated. The result is an increase in the variance of the final position, because the arm starts producing small oscillations around the target again. Next, the output-gains determining the maximal velocity of the joint motors ($\dot{\theta}_{max}$) were changed[2]. A robot arm producing only half of the velocity usually produced during evolution, performs somewhat worse than under normal circumstances. The reason is identified by looking at the next test. Having the same gain but less gravity ($g/9.0$), restores performance to the expected level. Hence, it is likely that having a decreased maximum velocity, and thus less force because ODE applies an amount of force that is needed to achieve the desired velocity, it is harder for the arm to compensate for gravitational force. Doubling the velocity gain, in contrast, leads to movements so erratic that the arm can not reliably center on the target anymore and thus often misses the target completely. To test for independence from details of the sensors, in one test the inputs to the module controlling the centering response were changed to binary mode rather than continuous values encoding the distance of intersection (IO-input). Thus a sensor's response is 1 if intersection occurs and 0 otherwise. As can be seen from the table, there is no considerable decrease in performance. This setup could be useful when implementing a real robot system. A sampling mechanism combined with a threshold function applied to a traditional camera image could provide the same kind of visual feedback as used in the simulation. Another set of tests varied the length of the limbs ($|l_i|$). Since the usual lengths were 16 and 9 units for the second and third limb respectively, adding 10 units to each limb means a lengthening by 110% for limb 3 and 62.5% for limb 2. However, even in the most extreme cases the performance decreases only gradually. The accuracy would still be good enough to actually grasp the object. Finally, reducing gravity to a ninth of the usual value does not affect performance either.

[2]The length of the trial was also changed to allow the robot to move the same distance.

Conclusions

Robotic sensorimotor-coordination was re-formulated as the problem of designing embodied, situated and adaptive controllers which are dynamically coupled to an ever changing environment. This was seen in contrast to classical approaches in which a series of internal representations is generally constructed. It was shown that simple modular neural networks can be evolved as model-free controllers for visually-guided robot motion. Spatial coordination (alignment of gaze direction and arm orientation) and temporal coordination (delaying movements until target is identified) were achieved by coupling individual modules through proprioceptive feedback. In this approach, neither were explicit coordinate-transformations necessary, nor the learning of robot models or sensor calibration. The resulting system is able to reliably position and track objects in its environment and is robust to changes in sensory-, environmental- and morphological parameters. Its behavior is general enough to allow for the desired outcome even without the need for adaptation. For validation of these results, future experiments will aim at evolving a similar control scheme on a real robot in order to compare it with the simulation. Also, the particular form of end-effector trajectories can be improved by adding additional costs to the fitness function which are analogous to well-known trajectory optimization principles like minimum variance, minimum torque-change or minimum jerk (Jordan and Wolpert, 1999).

References

Jordan, M. and Wolpert, D. (1999). Computational motor control. In Gazzaniga, editor, *The Cognitive Neurosciences*. Cambridge, MA: MIT Press.

Kato, T. and Floreano, D. (2001). An evolutionary active-vision system. In *Proceedings of the Congress on Evolutionary Computation*. Piscataway, IEEE-Press.

Massone, L. (1998). Sensorimotor learning. In Arbib, M., editor, *Handbook of Brain Theory and Neural Networks*. MIT press.

van der Smagt, P. (1995). Visual robot arm guidance using neural networks. Ph.D. thesis, University of Amsterdam.