Evolving the Morphology of a Compound Eye on a Robot

Lukas Lichtensteiger and Peter Eggenberger
AILab and Software Engineering
Department of Computer Science
University of Zürich
Winterthurerstrasse 190, 8057 Zürich, Switzerland
FAX: +41-1-635 68 09; Email: llicht@ifi.unizh.ch,eggen@ifi.unizh.ch

In: Proceedings of the Third European Workshop on Advanced Mobile Robots (Eurobot '99), 6-8 September 1999

Abstract

This paper reports on an experiment in evolving the morphology of an artificial compound eye with 16 light sensors on a robot. A special robot was designed and constructed that is able to autonomously modify the angular positions of the individual light sensors within the compound eye. The task of the robot was to employ motion parallax to estimate a critical distance to obstacles. This task was achieved by adapting the morphology of the compound eye by an evolutionary algorithm while using a fixed neural network to control the robot.

1 Introduction

In the usual engineering approach, a robot is hand-designed to perform a specific task. The design choices for many of the system's parameters depend on models of the robot-environment interaction. These models often cannot describe actual real-world conditions very accurately. It is therefore desirable to build robots that can adapt to their environment by automatically re-adjusting some of the system's parameters to match the actual conditions found in the real world. One method for automatic fine-tuning of design parameters is by the use of artificial evolution. The field of artificial evolution dates back to the sixties where people like Rechenberg and Schwefel [15] developed computational models of evolution. In the field of robots and autonomous agents these techniques have been applied during the last ten years. A typical application would be the following: for a given robot system an evolutionary algorithm has to evolve the neural control structure for a certain task. In many cases the neural network structure was given and the neural weights were evolved, or learning rules had to be evolved or the neural architectures had to be constructed [8, 2, 7, 12, 17, 18, 1]. But in all these approaches the morphology, in other words, the agent's shape, the positioning of the sensors, the way of locomotion etc., is assumed to be given and only the neural control is evolved. This restriction is not surprising: Most of the robots built today are quite flexible where the choice of control structure implemented is concerned. When it comes to the mechanical configuration, they are much more restricted by their designers. However, we believe that it is very important to be able to also adapt the robot's morphology to the desired task and environment. There is increasing evidence that in many cases a 'good' morphology can simplify the control structure and make it more stable with respect to environmental changes [13, 10]. It has been suggested that there exists a kind of ecological balance between morphology, neural control, task and environment [14]. In this paper we present a robot that is able to automatically modify its sensor morphology (to a certain degree) in order to improve its performance on the task of estimating a critical distance to some obstacle.

2 Method

In our experiments we use a robot with adaptive morphology (see section 2.2) where the process of changing the morphology is automatically controlled by an evolutionary process (see section 2.4). This means that in contrast to the conventional approach the physical characteristics of the robot are not completely predetermined at design time but can be modified to some extent by the robot itself to improve its performance on a given task. Modification of the morphology is controlled by an evolutionary process obtaining its feedback from the robot's behavior. We present an example of a robot that is able to modify the positions of the facets within a compound eye using an evolution strategy (ES, see section 2.3) in order to better estimate a critical distance to obstacles using motion parallax. The phenomenon of motion parallax [19] (see section 2.1) has the advantage that it is very well understood and favors non-trivial sensor morphology. Motion parallax is used by insects such

as the house fly to avoid obstacles. Biologists know that in this context the morphology of the compound eye of the fly, i.e. the distribution of the facets (ommatidia), is of importance [16, 9, 4]. To keep the focus on adaptivity of the morphology we used a hard-wired control structure (a primitive neural network) that was fixed during the whole experiment, i.e. in contrast to the usual approach of adapting the control structure, our robot 'learned' its task by changing its morphology.

2.1 Motion parallax

Figure 1 shows how an observer R moving at constant velocity v with respect to an obstacle X sees the obstacle under different angles α at different times t, t', t''. From the observer's point of view the obstacle moves with the same speed v but in opposite direction. Let v_t denote the component of v that is perpendicular to the vector r from R to X. Since $v_t = v \sin\alpha$ the angular velocity with which the image of X moves through the visual field of the observer R is

$$\omega = \frac{v_t}{r} = \frac{v \sin\alpha}{r}. \tag{1}$$

If the obstacle X is fixed and the observer can measure its own speed v as well as the angle α and the angular velocity ω it can calculate from equation (1) its distance r to the obstacle at any time [19]. In our case we would like to estimate the distance of closest approach d to determine whether the current track would keep us far enough from the obstacle at all times. Since $d = r \sin\alpha$ we have

$$d = \frac{v}{\omega} \sin^2 \alpha. \tag{2}$$

Let us assume that the observer uses some sensors each of which can detect the obstacle if it is seen at a particular angle α. We define

$$u = \frac{dn}{dt} \tag{3}$$

where u is the number of sensors that are activated per unit time (i.e., the number of neighboring sensors that detect the obstacle during a unity time interval. u is the speed of the image of the obstacle through the visual field measured in sensor units). If we define

$$\rho(\alpha) = \frac{dn}{d\alpha} \tag{4}$$

to be the (angular) density of these sensors (i.e., the number of sensors per radians) then

$$u = \frac{d\alpha}{dt}\frac{dn}{d\alpha} = \omega\rho \tag{5}$$

Let us now consider a very special density distribution for the angular sensors,

$$\rho = k\frac{1}{\sin^2\alpha} \tag{6}$$

where k is an arbitrary constant. Then equation (2) becomes with (5)

$$\begin{aligned} d &= v\frac{\rho}{u}\sin^2\alpha \\ &= \frac{kv}{u} \end{aligned} \tag{7}$$

In contrast to (2) equation (7) no longer depends on the angle α under which the obstacle is seen but only on the number of sensors u activated per unit time and the observer's speed v.

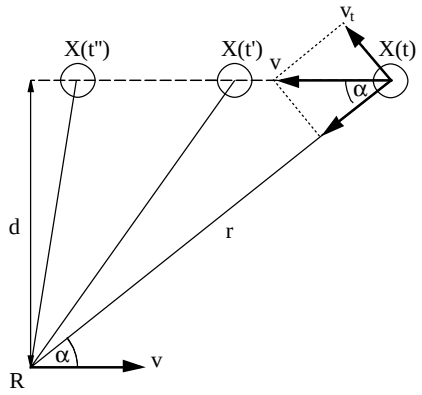

Figure 1. Illustration of motion parallax. An observer moving at position R with a velocity v that is constant relative to an obstacle X can estimate the distance r at different times t, t', t'' by using sensors able to detect the angular velocity $\omega = d\alpha/dt$ of the apparent movement of the observed object. d is the distance of closest approach.

2.2 A robot with adaptive sensor morphology

We designed and built a special robot that is able to automatically adapt its sensor morphology within a certain range (see figure 2 for a picture of the robot. Figure 3 shows a Block Diagram of the robot and a schematic representation of one sensory unit). The robot employs sixteen light sensors. In front of each sensor a thin intransparent tube is mounted reducing the aperture of the sensor to about two degrees. These tubes are the primitive equivalent to the biological ommatidia [16, 9, 4]. Each tube is mounted at the sensor-side on a cog-wheel. All cog-wheels share a common vertical axis but their direction can be controlled individually using sixteen small electrical motors and feedback signals from potentiometers. Each tube can thus be rotated around the common vertical axis within a range of about 200 degrees. The resolution of the angular position of the

tube depends on the angular position: however, for the angles used in our experiment it was usually around 2 degrees. The angular positions of the tubes are the system's parameters that are directly controlled by an evolution strategy [15] and changed according to the given fitness (see section 2.4). The sensory system is essentially one-dimensional: Tubes are arranged on top of each other only to allow for free rotation. Since the system is symmetric along the vertical axis and since we are using only obstacles with the same symmetry (like vertical bars) the situation is the same as if all tubes were mounted at the same height. This corresponds to a one-dimensional circular array of sensors whose angular positions can be freely varied.

To steer the robot a simple, homogeneous neural network was implemented which was able to detect motion (see figure 4). Basically the neural net consists of elementary motion detectors which are activated if an object moves at a speed higher than some critical value (in principle there is also an upper limit to the speed: when the obstacle moves too fast the sensors will no longer be able to detect it since they are not sampled frequently enough by the processor). The network consists of two layers: In each layer all neurons are identical. Neurons of type A receive their input directly from the sensors. They are activated when the sensor reading rises above a certain threshold (In our case the neurons were activated at a "dark to bright" transition that was significant enough. This can be seen as some kind of edge detection mechanism). Then the neurons remain active during a decay time τ that is equal for all neurons of type A. Neurons of type B become active if two neighboring neurons of type A are active at the same time and become inactive as soon as this condition is no longer valid. Note that all weights in the network are equal. During the whole evolutionary process the network remained the same, no weights or connections were modified.

Consider now a strong enough stimulus moving along the array of sensors. If the time it takes to move from one A-neuron to its neighbor is smaller than τ this means that the first A-neuron will still be active when the second A-neuron becomes active and thus the corresponding B-neuron is triggered. On the other hand, if the (apparent) movement of the stimulus is slow enough then the first A-neuron has already become inactive again and the B-neuron is not triggered. Therefore the parameter τ determines a critical speed between two neighboring sensors (τ is equal for all sensors). Note that the network does not discriminate the direction of movement and would therefore work equally well if the robot was moving backward. (However, we require that the time between successive stimulations of a given A-neuron is always bigger than τ. This can be achieved by introducing a corresponding "latency period" for the A-neurons during which the neuron cannot be triggered again. But even then there can be situations where the system is fooled: Consider for example two obstacles at different lateral distances from the robot's track. Then at some point the stimulus from the closer obstacle will "overtake" the stimulus from the farther obstacle on the A-neuron layer which may result in an erroneous activation of the corresponding B-neuron. For simplicity we only used one obstacle in our experiments.) Whenever one of the B-neurons becomes active a signal is sent that triggers the necessary avoidance action. In our experiment the robot did not actually turn away from the obstacle but the avoidance signal was used do calculate the fitness function for a particular individual (see section 2.4).

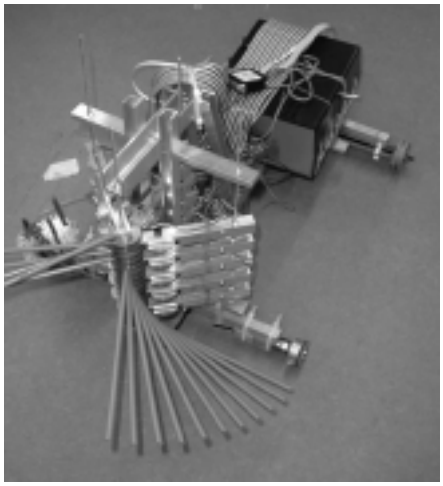

Figure 2. A robot that is able to position its sensors autonomously using electrical motors. Each of the 16 long tubes contains a light sensor which can detect light within an angle of about 2 degrees. The tubes can be rotated about a common vertical axis. An evolution strategy was used to determine the angular position of each sensor.

2.3 Evolution Strategy

Evolutionary algorithms (EAs) is a term for a class of stochastic optimization and adaptation techniques which allow to solve optimization problems. EAs provide a framework consisting of genetic algorithms (GAs) [6], evolutionary programming (EP) [3], and evolution strategies (ESs) [15]. These techniques have been successfully applied in diverse areas, such as machine learning, combinatorial problems, engineering problems such as the design of propellers or flux optimization through a tube, VLSI design, or numerical optimization. Each of these EAs is designed along different methodologies. Despite their differences, all EAs use random variation and selection from a population of in-

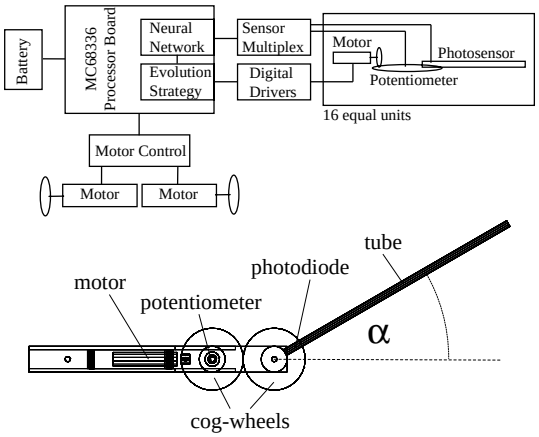

Figure 3. Top: Block Diagram of the robot. Bottom: Schematic representation of one of the 16 sensor units.

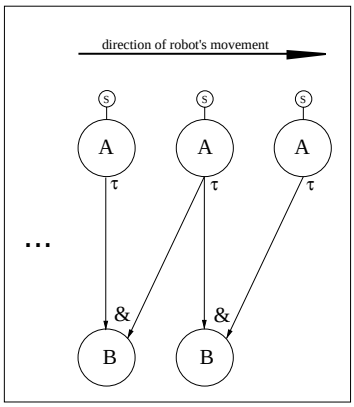

Figure 4. A two-layered artificial neural network is used to control the robot. This network is never changed throughout the evolutionary process. Neurons A remain active for a certain decay time τ after being triggered by a stimulus at the corresponding sensor S. Therefore neurons B are activated if two neighboring sensors receive stimuli within a time interval smaller than τ.

dividuals. Typically, such population-based search procedures generate offspring in each generation. A fitness value (defined by a fitness or objective function) is assigned to each offspring. Depending on their fitness, each population member is given a specific survival probability. A generic form of most evolutionary algorithms can be described as follows: (1) Initialize and evaluate the fitness of each individual of a population, (2) Select the parents according to a selection scheme, (3) Recombine and mutate selected parents with a specific operator, (4) Repeat the cycle. For the present experiment an evolution strategy (ES) was used to vary and select genomes.

2.4 Evolving the morphology of the robot's compound eye

The task of the robot would be to avoid an obstacle (a light source) if the point of closest approach of its track to the obstacle (the lateral distance) was closer than a critical distance d_0 and *not to avoid* it if this distance was greater than d_0 (*Not avoiding* is also important: A robot that avoids whatever it sees would be quite useless, consider for example the task of moving through a gap between two obstacles). The robot would be moving straight ahead at constant speed v_0 while the obstacle was immobile. The problem with this experimental setup is that it requires too much supervision of the robot during the experiment: Using constant speed the robot has to be run both at lateral distances slightly greater than d_0 (when it should not avoid) as well as at distances slightly smaller than d_0 (when it should avoid) to be able to determine the critical distance d_0. This is very impractical in an evolutionary procedure because for each individual during evolution the robot has to be placed at different distance settings. Since the evolutionary process typically needed a total of a few hundred individuals to be tested over a total time of several hours the overhead of positioning the robot manually was prohibitive.

Therefore we decided to use a somewhat different but functionally equivalent setting: Instead of varying the distance around d_0 we varied the speed of the robot around a critical speed v_0 (one setting slightly above v_0 and one setting slightly below v_0). The robot then had to avoid the light if it was running faster than v_0 and not to avoid it if it was running slower than v_0. Since in (2) and subsequent equations ω only depends on the ratio of v and d this is a completely equivalent task (for the robot the apparent motion of the obstacle is the same) under the condition that the distance to the obstacle remains fixed at d_0 (There may be subtle differences between the two tasks since light intensities are different at different distances but for our purposes these effects are neglectable). In order to guarantee that the distance d_0 is always constant we put the robot on a rail track. In this way the robot could move back and forth without changing its lateral position and the experiment could be left unattended during the evolutionary process.

One run of the experiment was performed in the following way: All sensors were connected to their corresponding A-neuron in the neural network at all times and the neural network was never changed. At the beginning of the experiment all sensors were placed at positions on the right side of the robot so they would never get a stimulus from

the light tube which could only be seen on the left side of the robot. (We used vertical fluorescent tubes as light sources to account for the vertical arrangement of sensors (see section 2.2). Ideally all sensors would lie in the same plane and then any type of light source could be used.) The first sensor (corresponding to an A-neuron at one end of the neural network, see figure 4) was then moved to a position about 90 degrees from the front on the left side where it remained fixed for the rest of the experiment. The second sensor (connected to the A-neuron that was the neighbor of the A-neuron for the first sensor) was moved to a position at some random angular distance from the first sensor. The distance between the two sensors was then evolved by an evolution strategy (ES) using 5 individuals per generation (see section 2.3). As soon as the fitness of an individual was high enough the second sensor was also fixed and would not move anymore for the remainder of the experiment. Then the third sensor (with respect to the A-neurons) was positioned randomly and its angular distance to the second sensor was evolved by the ES and so on until all sensors were positioned on the left side of the robot.

The fitness of each individual was determined as follows: The robot would move forward on its rail track past the light tube at a speed slightly greater than v_0. For every B-neuron in the neural network that was activated (and would have caused the robot to turn away) the fitness was increased by one (the robot did not actually turn away but stayed on its track in order to be able to continue the evaluation for the other sensors at the same lateral distance d_0 from the light tube). When the light tube was past the backmost sensor the robot would move back to its starting point (without evaluating the fitness while driving backwards). Then the robot started to move forward again but at a speed slightly slower than v_0. This time the fitness was increased by one for every sensor that detected the light tube *only if the corresponding B-neuron was not activated*. After the backmost sensor was past the light the robot again returned to its starting position without measuring. If all sensors that had been positioned so far reacted correctly (i.e., 'avoiding' when moving forward at fast speed and 'not avoiding' when moving forward at slow speed) and thus the maximum fitness possible for the current number of sensors was obtained then the current sensor was fixed and evolution moved to the next sensor. If not all sensors reacted correctly the ES modified the position of the current sensor (relative to the position of its predecessor) according to the fitness received and another run for evaluating the fitness would recommence for the same sensor.

It is important to note that one reason for using motion parallax for obstacle avoidance is that the robot can decide if its track will lead it too close to an object regardless of the angle under which this object is seen. Thus if the robot moves on a track that passes too close by an obstacle first the frontmost sensors will signal the potential collision and then one after the other until finally the sensors at the side are reached. This has the big advantage that the robot can determine potentially dangerous obstacles anywhere in its visual field and also at an early point in time. In our experiment for every sensor that made a correct decision (avoid/no avoid) the fitness was increased by one. This means that the more sensors are positioned correctly the higher the total fitness achieved (remember that the neural network remains fixed throughout the experiment and all sensors are connected to their corresponding neuron at all times).

Figure 5 shows the robot moving on its track past the light tube at different time points during the experiment. The top image shows the situation right at the beginning when only the first sensor is fixed and the second sensor is evolved. All other sensors point to the right of the robot and thus will never detect the light tube and cannot contribute to the fitness. The bottom image shows a situation when most of the sensors have been positioned (the sensor pointing approximately towards the light is the one whose position is currently being evolved). The image in between shows an intermediary state.

Figure 6 shows the fitness of all individuals in one run of the experiment. It reflects the fact that sensor positions were evolved one after the other and so the fitness increases whenever a new sensor is positioned correctly but never decreases, because the other sensors remain in their correct positions. Note that the fitness function we employed does not bear any information in which way to change the value of a particular sensor position to improve the fitness (the fitness function is completely flat away from the solution). The ES therefore does essentially a random walk until it finds the right solution for the given sensor.

3 Results

Figure 7 shows sensor distributions that were evolved in the experimental setup. Each of the three runs used a different random seed which resulted in slightly different solutions being found by the ES. Each run took 4 to 5 hours to complete. The inhomogeneous distribution of sensors is clearly visible although the exact positions of the sensors differ considerably between runs. This is mostly due to the relatively big difference between the two velocities used for the avoid and the non-avoid task. In other words, there is a relatively broad range of velocities where no fitness feedback was given and both behaviors were accepted. Consequently there is a relatively big range of angular positions that all represent solutions to the task. This follows from equation (7): We have constant $d = d_0$ with good precision since the robot moves on rails. Also the critical time between two sensors (when the behavior changes) is just τ and therefore $u = 1/\tau$ which is also constant with high

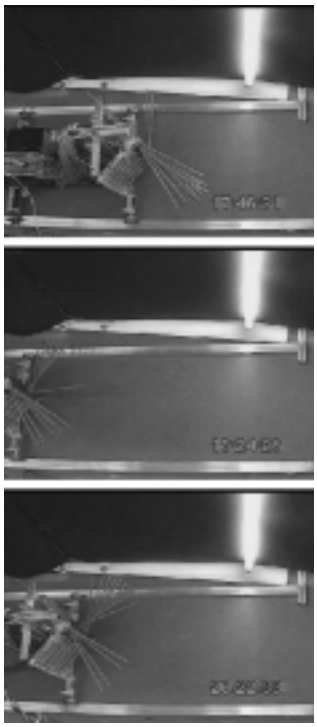

Figure 5. The robot on its track during evolution moving beside an obstacle (the vertical light tube). Three different states of sensor positions are shown during evolution. The top figure shows a typical configuration at the beginning, the middle one an intermediate state and the bottom one a late state (the randomly directed sensors on the right are not yet positioned). Note that the sensors are not uniformly distributed but become more densely spaced toward the front (see equation (6)).

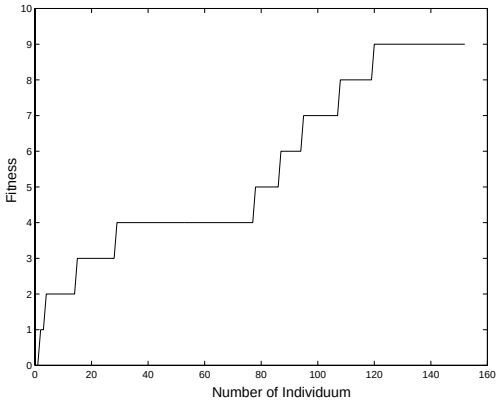

Figure 6. A typical run of the evolution of the sensor positions. The fitness increases whenever a new sensor is positioned correctly but never decreases, because the other sensors remain in their correct positions.

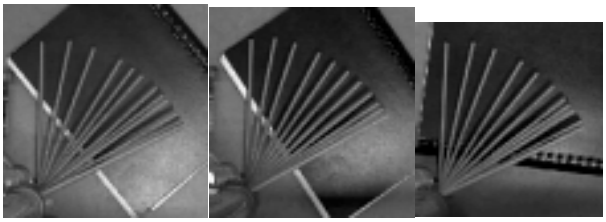

Figure 7. Sensor distributions that were evolved using different random seeds. Each run took 4 to 5 hours to complete. The inhomogeneous distribution of sensors is clearly visible although the exact positions of the sensors differ considerably between runs. The robot is running from left to right.

precision. Rewriting (7) yields

$$kv = \frac{d_0}{\tau} = \text{constant} \tag{8}$$

Therefore the larger the difference between the two values of v that demand different behavior the broader the range in accepted values for k becomes. Therefore according to (6) more different sensor distributions represent solutions of the robot's task. Ideally, both velocities should be as close as possible to the critical velocity v_0, one slightly above and one slightly below. However, then the interval of sensor positions where a solution exists becomes also very small and the ES has to find the exact positions according to (6) which takes an arbitrarily long time due to the shortcomings of our fitness function mentioned in section 2.4, namely that the fitness only changes when the solution is found. The choice of the two velocities was a compromise between still obtaining an inhomogeneous spacing for the sensors but not having to wait too long for evolution to complete its task. A method to speed up the evolutionary process would be to use a simulator to evolve a coarse morphology and only do the final tuning of critical parameters on the real robot.

To verify that the solutions for the sensor morphology found by the evolutionary process produced indeed different behavior on the robot than for example a homogeneous distribution of sensors we compared the two cases. The systems for both cases were completely identical except that in one case the sensor morphology of the robot was set by hand to a homogeneous distributions (all equal interommatidal spacings) and in the other case it was the result of a run of

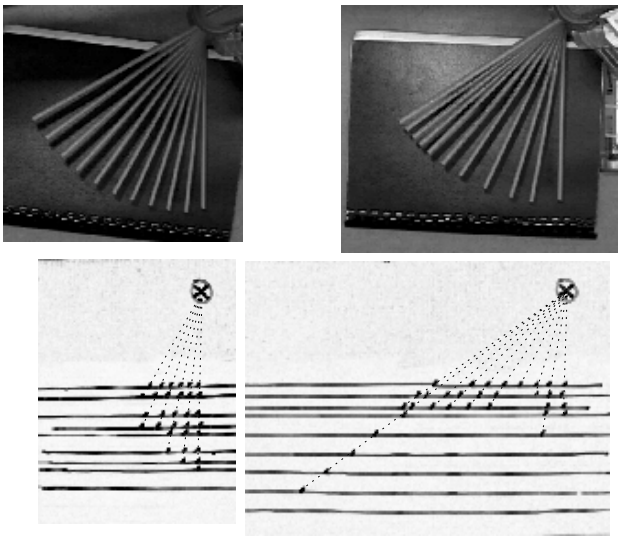

Figure 8. Comparison of the behavior of two robots that were completely identical except that the sensor morphology of one was set by hand to a homogeneous distribution (all equal interommatidal spacings) and the sensor morphology of the other was the result of a run of the evolutionary process. There are clear differences which can be explained by the different sensor positions (the figure is explained in the text).

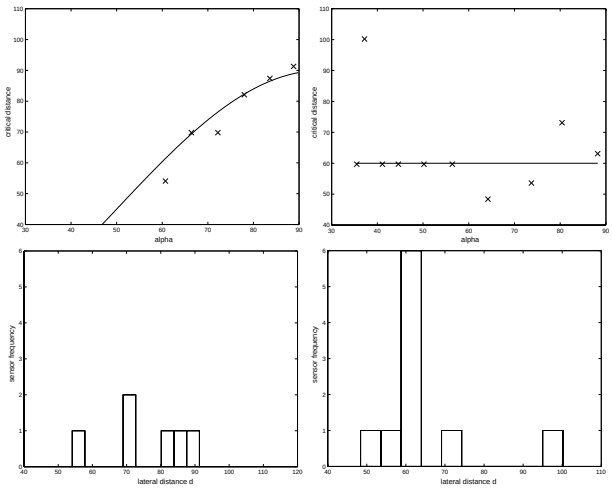

Figure 9. Critical distance determined from the behavior of the same robots as in figure 8. The two images on the top show plots of the critical distance versus the angular position of the respective sensor. The solid lines represent theoretical predictions and are shown for qualitative comparison (the free parameters were chosen 'by hand'). The two histograms at the bottom show the number of sensors that measure a certain critical distance. The images on the left are for the homogeneous sensor distribution whereas the ones on the right are for the evolved morphology. It can be seen that for the evolved morphology most sensors measure the same critical distance (around 60) whereas for the homogeneous sensor distribution the critical distance is different for most of the sensors (because it depends on their angular positions).

the evolutionary process (Note that this is not really a comparison between an approach using an adaptive morphology and another approach using an adaptive neural network because in the present comparison both networks were fixed so the robot with the homogeneous sensor distribution had no way of adjusting the parameters of the neural network correctly). For the comparison the distance of the rail tracks to the obstacle was varied and for each distance the robot was run along its track. Whenever it would have avoided the obstacle the robot made a mark on the ground. The results are shown in the two images at the bottom of figure 8. The image on the left shows the case for the homogeneous sensor distribution and on the right the results for the evolved distribution are shown. The mark near the top right corner of each image shows the position of the light tube (the obstacle). The robots moved from left to right and the horizontal lines represent the tracks at different distances from the light. The dotted rays emanating from the light and passing through the marks on the tracks represent different angles under which the obstacle was seen when the robot would have avoided. Therefore each of these angles corresponds to the angular position of a particular sensor. The two images at the top of figure 8 show the corresponding sensor morphologies.

The two most prominent differences between the behaviors for the two morphologies are the following:

- In figure 8 in the bottom left image the marks extend much less to the left than in the bottom right image. This is because for the homogeneous sensor distribution the time the image of the obstacle needs to move from one sensor to its neighbour becomes larger than τ for small angles and therefore the neural network does no longer avoid.

- The dependency of the critical distance on the angle under which the obstacle is seen is different for each case. The critical distance is approximately the lateral

distance from the obstacle (measured vertically in figure 8) of the mark that is furthest away from the obstacle for a given angle (i.e., the most remote mark on a certain ray emanating from the obstacle). For smaller lateral distances than the critical distance the robot will avoid while for greater distances it will not. The top of figure 9 shows the dependency of the critical distance on the angle under which the obstacle is seen for each case. The solid lines represent theoretical predictions (the free parameters were chosen 'by hand') while the crosses are the experimental results from figure 8. The image on the left again shows the case of the homogeneous sensor distribution whereas the image on the right shows the results for the evolved sensor distribution. Note that in the later case according to equation (7) theoretically the critical distance should be constant (independent of the angle α). By contrast the homogeneous sensor morphology in the top left picture theoretically results in a different critical distance for every angle (the biggest critical distance is obtained for the most lateral sensor, see (2)). The bottom of figure 9 shows how many sensors produced a certain value for the critical distance. For the evolved sensor distribution (right image) most sensors reported a critical distance at around 60 (in arbitrary units).

Note that in figure 8 all tracks are more or less parallel to each other. This is not really necessary since all that matters is the lateral distance to the obstacle (the point of closest approach to the obstacle), regardless of the direction of the track.

4 Conclusions

In this paper we considered a robot with adaptive sensor morphology. By autonomously adapting the morphology of its compound eye using artificial evolution the robot was able to achieve a relatively good performance on the task of estimating a critical lateral distance to an obstacle (see figure 9, right side). The morphologies that were evolved (see figure 7) qualitatively resemble the theoretical predictions (equation (6)). We showed that for the present task it was possible to improve the behavior of a robot by adapting its morphology while keeping its control structure fixed. This is in contrast to the usual approach in robotics where the morphology of the robot is fixed and only its control structure is changed. The advantages of using a morphology that is adapted to a certain task will be discussed elsewhere [10].

5 Acknowledgments

We thank Rolf Pfeifer for continuing support. This project was supported by SNF Grant No. 2000-053915.98.

References

[1] P. Eggenberger. Cell interactions as a control tool of developmental processes for evolutionary robotics. In [11], pages 440–448, 1996.

[2] D. Floreano and F. Mondada. Evolution of plastic neurocontrollers for situated agents. In [11], pages 402–4101, 1996.

[3] D. B. Fogel. Evolutionary computation: Toward a new philosophy of machine learning intelligence. *Biological Cybernetics*, 63:487–493, 1990.

[4] N. Franceschinie, J. Pichon, and C. Blanes. From insect vision to robot vision. *Phil. Trans. R. Soc. Lond. B*, 337:283–294, 1992.

[5] W. Gerstner, A. Germond, M. Hasler, and J.-D. N. (Eds.), editors. *Seventh International Conference of Artificial Neural Networks (ICANN'97)*, Cambridge, MA, 1997. Sprinter.

[6] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addision-Wesley, 1989.

[7] F. Gruau and D. Whitley. The cellular developmental of neural networks: the interaction of learning and evolution. Technical Report 93-04, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, France, 1993.

[8] I. Harvey. Species adaptation genetic algorithms: The basis for a continuing saga. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 346–354. MIT Press/Bradford Books, Cambridge, MA, 1992.

[9] G. A. Horridge. Insects which turn and look. *Endeavour*, I:7–17, 1978.

[10] L. Lichtensteiger and P. Eggenberger. in preparation.

[11] P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, editors. *From animals to animats 4: Proceedings of the fourth international conference on simulation of adaptive behavior*. MIT Press, 1996.

[12] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to evolve autonomous robots: Different approaches in evolutionary robotics. In P. M. R. A. Brooks, editor, *Artificial Life IV*. Cambridge, MA: MIT Press, 1994.

[13] F. Panerai and G. Sandini. Oculo-motor stabilization reflexes: integration of inertial and visual information. in press.

[14] R. Pfeifer. Building fungus eaters : Design principles of autonomous agents. In [11], pages 3–12, 1996.

[15] I. Rechenberg. *Evolutionsstrategie' 94*. frommann-holzboog, 1994.

[16] T.S.Collett. Peering - a locust behavior pattern for obtaining motion parallax information. *Journal of experimental Biology*, 76:237–241, 1978.

[17] J. Vaario. Modelling adaptive self-organization. In *Proceedings of the Fourth International Workshop on the Synthesis of Living Systems*, 1994.

[18] J. Vaario and K. Shimohara. Synthesis of developmental and evolutionary modeling of adaptive autonomous agents. In [5], pages 721–726, 1997.

[19] T. C. D. Whiteside and G. D. Samuel. Blur zone. *Nature*, 225:94–95, 19970.