

Evolutionary Design of Controllers for Autonomous Line Tracking Mobile Robots Using Population Based Incremental Learning

Shahab Kalantar¹, Parisa Eslambolchi¹, MajidNili Ahmadabadi^{2*}
 Robotics Lab, Elec. & Comp. Eng. Dept., Center of Excellence for Control Eng.
 Faculty of Engineering, University of Tehran
 *mnili@ut.ac.ir

ABSTRACT

This paper concerns the design of sensor topology and reaction controllers for autonomous mobile robots following a line. Artificial evolution [1,2] is used as the design methodology. Here, it is shown PBIL [3] is powerful enough to evolve autonomous creatures exhibiting complex behavior. Designing optimal sensor topology, symmetric and asymmetric control mechanisms, reactive controllers with feedback, and suitable criteria for evaluating evolved robots are among the topics discussed in this paper. Intuitive insight into the nature of the problem proved to be a crucial determinant of the success of evolution. Defining some measure of stability turned out to be very useful. The idea can certainly be extended to other benchmark problems. PBIL is demonstrated to be much more effective than GA in exploiting the salient features in both fitness criteria and controller architectures. All the simulations were carried out by the EVO-ROB system, an open architecture, component based software framework especially designed for ER experiments.

Keywords: Autonomous mobile robot, population based incremental learning, line tracking

1. Introduction

Applying evolutionary methods is one of the promising approaches for creating autonomous creatures. The recent progress in A-Life, evolutionary algorithms and the science of animats indicates that Evolutionary Robotics (ER) will most probably prove itself as a vital area of research in the near future [4,5,6]. This paper is the second in a series of papers devoted to on-going research on ER in the robotics lab of ECE department of Tehran University [10,11]. The first part of this report [7] was aimed at a general discussion of applying autonomous robots in industrial environments, as well as appropriate control architectures and sensor topologies.

In [7], GA was used to evolve line following robots. Both symmetric and asymmetric architectures were considered. GA proved to be too weak to obtain optimal sensor topologies. A special non-evolutionary algorithm was designed for this purpose. There are at least three reasons why we have chosen line following (LF) behavior as our target: Firstly, LF is predictable, i.e. we can provide an exact definition for it; secondly, it is harder than obstacle avoidance (this will be dealt with in a subsequent paper), and finally, to our knowledge, no one has ever attempted to evolve it in the context of ER research. To cope with inability of GA to solve problems such as optimal sensor removal [7], we decided to search for more efficient methods. As will be shown in this paper, PBIL (combining genetic evolution with reinforcement learning) is much faster and more powerful than GA [3]. In the following sections we will introduce the line following behavior, environments used in experiments, stability, sensor topology, fitness evaluation, a review of PBIL and suitable control architectures. Finally, various experiments will be explained and the results discussed. Kinematics modeling and configuration are the same as in [7].

2. THE LINE FOLLOWING BEHAVIOR

A sensory-motor mapping $f : R^N \rightarrow R^M$ maps sensor values to motor signals. Identifying the behavior generated by the mapping, f , consists of finding a function \mathbf{y} such that for each

$x \in R^N$ and arbitrarily small \mathbf{e} , we have $\|\mathbf{y}(x) - f(x)\| < \mathbf{e}$. This is equivalent to minimizing the functional $E = \int_S \|\mathbf{y} - f\| dx$, where $S \subset R^N$ denotes the

sensory space. Here \mathbf{y} is the mathematical description of the computational process in an evolved control system. Harvey [8] points out some considerations concerning cognition and computation. In this paper, situations where the robot receives desired sensory patterns are called stable states, unstable states refer to transient situations between stable ones. As far as the line-tracking behavior is concerned, the sensory space is defined

by $S = \{0,1\}^4$. The actuating space is likewise defined as $V \subseteq R \otimes R$. The control mapping $\Delta : S \rightarrow V$ maps sensory perceptions to motor actions. The kinematics mapping $K : V \otimes C \rightarrow C$ relates an action (V_L, V_R) to a unique robot configuration where $C \equiv X \otimes Y \otimes \Theta$ and $X, Y, \Theta \subset R$. To simplify treatment, an auxiliary mapping

$w_{ENV} : C \rightarrow S$ is also defined which provides the sensory inputs given a certain position of the robot. The subscript ENV is used to emphasize the dependence on a particular environment. The robot life cycle is defined as the repeated application of mappings Δ , K and w_{ENV} . This induces the mapping

$\mathbf{a}_{ENV} \equiv w_{ENV} \cdot K \cdot \Delta : S \rightarrow S$. If we identify S with the state space of the dynamical system described by \mathbf{a}_{ENV} , then \mathbf{a}_{ENV} can function as the state transition function of a finite state automaton, with 16 state.

It should be noted that each state $s \in S$ corresponds to an infinite number of states in the configuration space of the robot, so that each s induces the equivalence class

$$[s] = \{(x, y, v) \in R \otimes R \otimes R \mid w_{ENV}(x, y, v) = s\}$$

As far as the observed behavior of the robot is concerned, it is unreasonable to assume that $[s]$ represents all of the elements of $[s]$, because it depends on dimensions of the robot and the line, the controller, and the position of the robot with respect to the line. If the robot and line dimensions are selected in such a way as to ensure the robustness of behavior for each $\mathbf{t} \in [s]$, then identifying $[s]$ with s will present no difficulties.

In this case, variation over $[s]$ will only result in a change of duration the robot spends in a particular state. Likewise, the sequence of states traversed by the robot naturally leads to the concept of sensory space trajectories. Robot performance

evaluation can be carried out with regard to the time it takes the robot to reach a stable state, starting from an unstable one, and the intermediate states during this transition.

In this research, four kinds of environments, each with two variations, were used to teach and consequently to evaluate the robot. Each one is defined by two parameters; one is the turning angle and the other is line width. We use the symbol ENV_N^w to denote an environment where $w \in \{<, >\}$ and $N \in \{I, II, III, IV\}$. $w=<$ denotes the facts that the line is narrower than the width of the robot, which results in the two lateral sensors of the robot to fall out of the line. When $w=>$, the line is wider than the width of the robot and all of the sensors are within the line. The latter case invariably leads to limit cycles. For this reason, we have limited our experiments to the $w=<$ case, thereby dropping it altogether and using the symbol ENV_N (which is equivalent to $ENV_N^<$). Figure 2-1 depicts this classification.

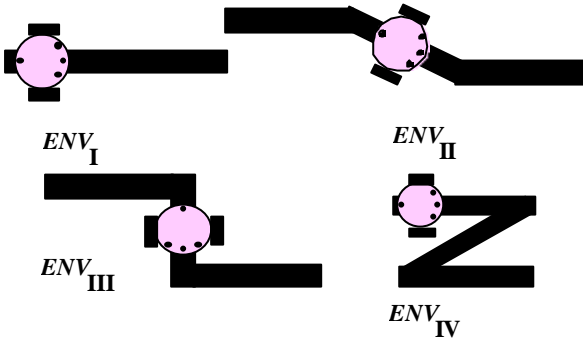


Figure 2-1

3. STABILITY ANALYSIS

Considering what was said about stable and unstable states, we could derive performance measures for evolved controllers based on the trajectories through the state space. Suppose $S_{10}=(0101)$ is the stable state for the robot in a $ENV_x^<$ environment. The stability of entering the line is defined as

$$h_x^1 : S \otimes \Theta \otimes D \rightarrow R \quad x \in \{<, >\}$$

$$(s, \mathbf{q}, d) \mapsto m$$

In the best case, we will have $d=0, \mathbf{q}=0$ and $S=S_{10}$ after letting the robot start from a particular pose. To simplify the problem, speed and time taken for the robot to reach stability is ignored and the line is chosen to be long enough to ensure that the robot has sufficient time to stabilize. In this case, the simplified measure $h_x^2 : S \otimes \Theta \otimes D \rightarrow \{t, f\}$ is sufficient, where

$h_x^2(s, \mathbf{q}, d)$ indicates the outcome of a particular run. If exact dynamical behavior is not important, This measure can be simplified further: $h_x^3 : S \rightarrow \{t, f\}$. With this level of precision, getting sufficiently close to $d=0$ and $\mathbf{q}=0$ will suffice. Note that

\mathbf{q} is the relative angle between the robot and the line and not an absolute value, so that the orientation of the line is not important. Figure 3-1 is a pictorial description of this measure. To measure the ability of the robot in turning we use the function

$$b_{x,y} : S_s \rightarrow \{t, f\} \text{ where}$$

$$y \in \{=90, >90, <90\} \quad , \quad x \in \{<, >\} .$$

Here, the robot starts from a stable state before turning and is supposed to reach a stable state after turning.

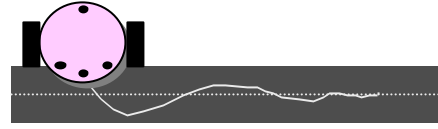


Figure 3-1

4. SENSOR TOPOLOGIES

The number and pattern of distribution of sensors, width of the line speed, have a major impact on the performance of the control system. In a previous paper [7], the relationships between these parameters have been investigated in some detail, including Perceptual aliasing, uncertainty, symmetry, and line width vs. robot diameter.

Figure 4-1 (a) shows a configuration, denoted SC_I , which has been used in most of the experiments.

The ultimate goal of evolutionary robotics is to undertake the design of every structural and behavioral aspect of an autonomous being, from control strategy to sensor configuration and physical shape.

In this paper, some of the experiments are devoted to evolution of the controller and the optimal sensor distribution at the same time. In these experiments, the underside of the robot is distributed with a pattern of sensors and the evolutionary algorithm is assigned the task of organizing an optimal configuration. This algorithm only removes redundant ones.

Figure 4-1 (b) shows the initial pattern (SC_{II}).



Figure 4-1

5. FITNESS EVALUATION

While designing a fitness function for the line following behavior, the following points should be considered:

1. The robot must move as near to the center of the line as possible.
2. The robot's axis should be aligned with the direction of the line.
3. The robot must move as fast as possible.
4. The evolution should produce symmetrical structures.
5. If minimizing the number of sensors is one of the goals, the maximum possible number of sensors must be deleted.
6. The robot must not change direction too often or, in other words, it should not have rotational vibrations.

In all of the experiments, fitness is computed incrementally during the lifetime of the robot. In each simulation step, a fitness value for the same step is computed and added to the accumulated fitness value. If t_0 denotes the start time of simulation and t_{max} the total number of simulation steps, then the fitness assigned to the robot at the end of its life would be

$$\Psi = \sum_{t=t_0}^{t_{max}} f(t) \quad , \quad \text{where } f(t) \text{ is the fitness value for each step. At}$$

the start of simulation we set $\mathbf{y} = f(t_0)$ and use

$\mathbf{y} \leftarrow \Psi + f(t)$ to update Ψ . $f(t)$ must be composed of various multiplicative terms to satisfy the foregoing requirements. We have used the following normalized variables to construct these terms:

$D_f = \|\mathbf{f}_L - \mathbf{f}_R\|$, the difference between the centers of the robot and the line, Robot's mean linear velocity, $V = |V_L - V_R|/2$, the difference in angles of the robot and the line, $D_d = \|\mathbf{o}_L - \mathbf{o}_R\|$, the difference between corresponding weights in symmetrical architectures, the total number of active sensors, the difference between the speeds of the left and right wheels, $D_v = \|\mathbf{n}_L - \mathbf{n}_R\|$, where \mathbf{o}_R denotes the robot center point, \mathbf{o}_L is a point on the center of the line which is nearest to \mathbf{o}_R , \mathbf{j}_R and \mathbf{j}_L are the rotation angles of the robot and the line, respectively. Figure 5.1 illustrates these parameters.

In a typical experiment, we would have

$$f(t) = V(1-D_f)(1-D_d)(1-D_v)$$

Other terms would be added depending on the experiment.

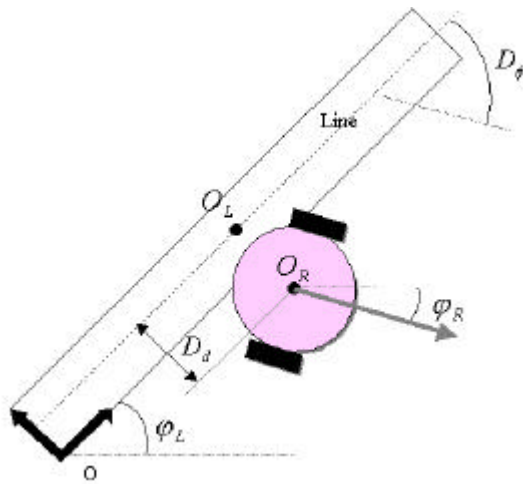


Figure 5-1

6. PBIL

Population Based Incremental Learning [6] is a combination of iterative and evolutionary optimization methods. This algorithm is based on GA mechanisms along with weight updating rules in supervised competitive learning. In PBIL, as in GA, discrete evaluations of potential solutions is the primary step: Here, we have used a version of PBIL which works on the binary alphabet. Weights should be encoded with specified precision. The number of bits assigned to each weight is pre-specified. Coding and decoding procedures are the same as in GA. The goal is to produce a probability of having '1' in each bit position. Sampling this vector produces the population in each generation. A 0.5 in a bit position indicates largest variability for this position. When the algorithm ends, the values of the probabilities in the vector should be such that sampling the vector would produce individuals with high fitnesses. Similar to learning in networks with competitive learning, the probability vector is gradually moved to one with high ranking bit values.

During each generation, sampling the vector generates a number of potential solutions. Every one of these solutions are then evaluated according to the fitness function.

The probability vector is pushed towards the solution vector with highest fitness; it is also pushed towards the complement of the solution with lowest (worst) fitness. The parameters of the algorithm are LENGTH (vector length), NUMBER-SAMPLES (population size), P[1..LENGTH] (probability vector), LR (learning rate), NLR (negative learning rate), MUT-PROBABILITY (mutation rate), and MUT-SHIFT (amount a mutation may change the value of a position).

After updating the probability vector, a new set of solution vectors is produced, and the cycle continues.

Furthermore, a special mutation operation is used when updating the vector: each position in the probability vector is shifted to a random direction with a small probability.

We have also included elitism in our implementation: To prevent the population from straying towards unfruitful regions of the search space, the best vector from the previous generation is preserved in the current generation and replaces the worst individual. Compared to GA, PBIL has less complexity but enjoys high speed and quality.

7. CONTROL MECHANISM FOR EVOLUTION

The robot control system is responsible for defining the velocities of the left and right wheels according to sensory information. In this paper as in [7], we have used the simple mechanism CS_I and CS_{II} . In these mechanisms, the velocities of the left and right wheels are computed as

$$V_L[t] = W_L V_L[t-1] + \sum_{i=1}^n W_{L,i} S_i[t] + W_0 \quad (i=l, r, b, f)$$

$$V_R[t] = W_R V_R[t-1] + \sum_{i=1}^n W_{R,i} S_i[t] + W_0 \quad (i=l, r, b, f)$$

This is a simple Braitenberg architecture where S_i is the activation level of sensors and W_0 is the idle activation in motors, which makes the robot still move even when no actuating signal is presented. The evolutionary algorithm updates these weights. In the symmetrical architecture we have:

$$W_R = W_L, \begin{cases} W_{L,r} = W_{R,l} & , & W_{L,f} = W_{R,f} \\ W_{L,b} = W_{R,b} & , & W_{L,l} = W_{R,r} \end{cases}$$

that defines $CS_{I,S}$ and $CS_{II,S}$. Using this limitation reduces the search space to half its size.

8. CLASS I EXPERIMENTS (SIMPLE ARCHITECTURES)

This group of experiments has been designed to evaluate the capability of PBIL in evolving optimal controllers of the CS_I kind in various environments. Figure 8-1 shows the fitness charts. The values chosen for parameters are as follows: LENGTH = 128, SAMPLE_SIZE = 200, MUT_RATE = 0.05, MUT_SHIFT = 0.02, LR = 0.1, NLR = 0.075. In experiment 1 (corresponding to ENV_I) and experiment 2 (corresponding to ENV_{II}), the algorithm exhibits similar behavior, but in experiment 2 higher fitnesses have been attained. In both of the experiments, after about 10 generations, the evolution has got very close to its final value. Compared to GA, this is very remarkable. It should, of course, be pointed out that this zero to final value is solely due to the stochastic nature of the algorithm albeit better exploited by PBIL.

Considering the fact that the best individuals of the previous generations are preserved in the next generations, the increase in the fitnesses of individuals, after nearing the final value, is not very considerable.

Irregular fluctuations of the worst fitness, and accordingly the average fitness, clearly shows high variance of individuals in the following generations.

The term $(1-D_v)$ has the least effect on fitness, but \bar{V} closely matches the average fitness. As expected, the pattern of evolution of weights is in complete accordance with maximum fitness.

Experiment 4, on the other hand, is a complete failure. Considering the environment (ENV_{IV}), this is not an unexpected result. Experiment 3 shows the best evolution in this class. The best individual has been born in generation 172. The fitness chart shows a steady rise in fitness. Also, the average and maximum fitness curves are very similar which shows low fitness variance among the individuals.

Figure 8-2 shows the result of testing the best individual in generation 200 of experiment 3 against the foregoing criteria (entering the line and turning). In each case, the initial position and rotation of the robot is indicated.

9. CLASS II EXPERIMENTS (SENSOR ELIMINATION)

In this group of experiments, we seek to determine how well PBIL can eliminate redundant sensors and obtain a minimal set of sensors appropriate for the specified task. The parameters are the same as the previous class. For each sensor, a single bit has been added to the chromosome that decides if the corresponding sensor is present in the final configuration or not.

Figure 9-1 shows the fitness charts for experiments 5 and 6 (corresponding to environments ENV_I and ENV_{II}). Experiment 5 is very similar to experiments 1 and 2, except the fact that here evolution has a much harder problem to tackle. As seen in Figure 9-1, evolution has not been successful in producing a symmetric structure (i.e. symmetry in distribution of sensors).

Experiment 6 seems to be the successful one in this class. The fitness chart indicates a smooth and steady rise in fitness, up to generation 42, after which, although no marked increase in fitness is observed, the algorithm continues to eliminate more and more sensors.

The best solution of experiment 6 was tested with respect to stability criteria. The results are depicted in figure 9-2 along with robot's position and orientation.

10. CLASS III EXPERIMENTS (PBIL PARAMETERS)

These experiments were designed to assess the effect of changing parameters on performance. All of these experiments are an exact copy of experiment 3 with the exception that in each of them, a different value has been used for a single parameter from those used in experiment 3.

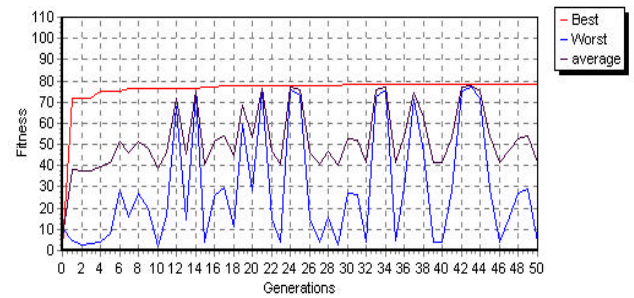
Some results are as follows reducing LR from 0.1 to 0.05, and increasing it to 0.8, faces the evolution with difficulty. Setting LR to 0.2 results in better fitness than the original value of 0.1. Although the maximum fitness is somewhat lower than in experiment 3, nevertheless a solution is reached in less time and exhibits more generalizability. Reducing the negative learning rate to 0.025 slows the evolutionary process, but increasing it to 0.2 results in much better individuals than experiment 3 (with $NLR = 0.075$).

Likewise, reducing the mutation rate to 0.008 led to unsatisfactory solutions, while increasing it to 0.1 both increased the maximum fitness and speeded up the algorithm (the best individual was attained in generation 23).

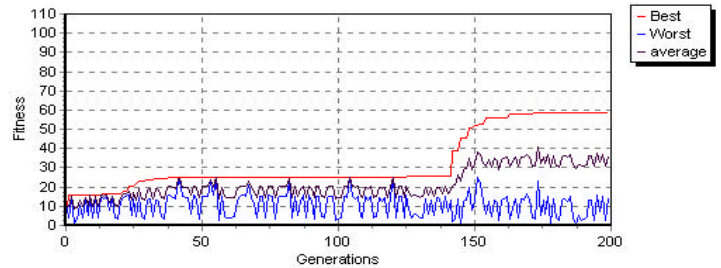
Similar results are obtained when reducing mutation shift to 0.01 and increasing it to 0.1.

11. CLASS IV EXPERIMENTS (SYMMETRICAL ARCHITECTURES)

In experiment 18 everything is similar to experiment 3, but the symmetry constraint has been applied. Figure 11-1 shows the result. Experiment 19 corresponds to experiment 7 (sensor elimination). Half of the weights are evolved and the final architecture is a symmetric one. Figure 11-2 shows the fitness chart. As seen, enforcing symmetry, in both cases, has resulted



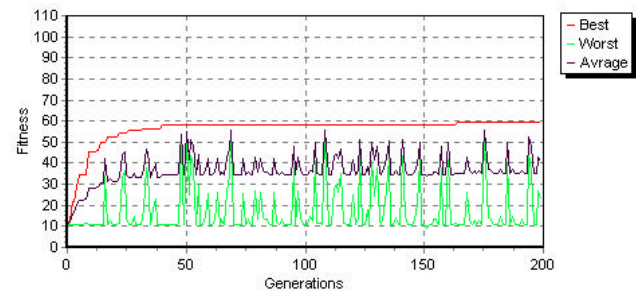
Experiment 2 corresponding to ENV II



Experiment 3 corresponding to ENV II
Figure 8-1



Figure 8-2



Experiment 6 corresponding to ENV II
Figure 9-1



Figure 9-2

in faster runs. Although the maximum fitness is lower relative to the nonsymmetrical cases, individuals have attained higher generalizability.

This conforms to the result obtained with GA[7].

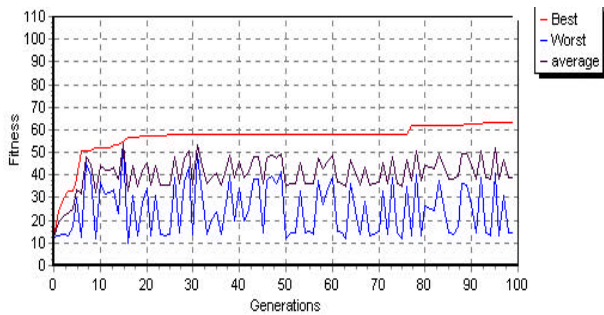


Figure 11-1

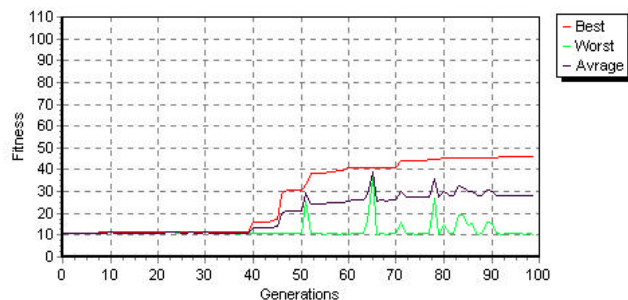


Figure 11-2

12. CLASS V EXPERIMENTS (FEEDBACK ARCHITECTURES)

Experiment 20 was designed to evolve type CS_{II} architectures in ENV_{III} . Parameters are the same as in experiment 3.

Figure 12-1 shows the fitness chart, while figure 12-2 shows the result of evaluating the best individual with the second criterion. This experiment shows that adding memory to the control architecture has speeded up the evolutionary process while, at the same time, resulting in higher fitnesses.

More detailed analysis of the behavior of this kind of architecture will be presented in a follow-up paper.

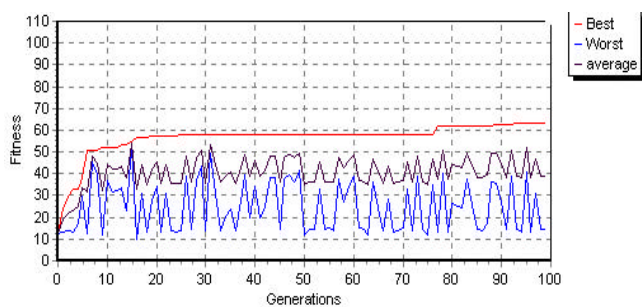


Figure 12-1



Figure 12-2

13. CONCLUSION AND FURTHER RESEARCH

It was shown that PBIL is a better candidate for evolutionary robotics experiments than GA. Making the algorithm faster will make possible online evolution and learning. Hardware implementation of PBIL on ASIC (one of the current research

areas in our LAB), links this line of activity with ongoing research in the evolvable hardware area [9].

Though the individuals generalize very well to environments other than the ones in which they were trained, but more complex environments will undoubtedly reduce this capability.

If we were sure of the speed of convergence, this would not pose a serious problem: as the environment changes, the robot will adapt itself to the new conditions.

Further research is aimed at improving the results of this paper in at least four aspects: The first one involves combining the line following behavior with other behaviors, e.g. obstacle avoidance and goal finding. The second aspect will delve further into detailed study of evolutionary mechanisms. The third aspect will consider more complex architectures, and the final one will lead to more fundamental results concerning the nature of behavior from a dynamical systems point of view.

14. REFERENCES

[1] Mondada, F. and Floreano, D., "Evolution and Mobile Autonomous Robotics", in Sanchez, E. and Tomassini, M. (eds.), Towards Evolvable Hardware, Berlin: Springer-Verlag, Lecture Notes in Computer Science, 1996.

[2] Meyer, J.-A., Husbands, P., and Harvey, I., "Evolutionary Robotics: A Survey of Applications and Problems", in P. Husbands and J.-A. Meyer, Proceedings of the First European Workshop on Evolutionary Robotics, Berlin: Springer-Verlag, Lecture Notes in Computer Science, 1998.

[3] Baluja, S., "Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning", Tech. Rep. CMU-CS-94-163, Dept. Comp.Sci., Carnegie-Mellon Univ., 1994.

[4] Floreano, D. and Mondada, F., "Evolution of Plastic Neurocontrollers for Situated Agents", in P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, and S. Wilson (eds.), From Animals to Animats IV, Cambridge, MA: MIT Press, 1996.

[5] Nolfi, S., Floreano, D., Miglino, O., and Mondada, F., "How to Evolve Autonomous Robots: Different Approaches in Evolutionary Robotics", in R. A. Brooks and P. Maes (eds.), Proceedings of the IV International Workshop on Artificial Life, Cambridge, MA: MIT Press, 1994.

[6] Braitenberg, V., "Vehicles: Experiments in Synthetic Psychology", Cambridge, MA, MIT Press, 1984.

[7] Nili, M., Mehdinezhad, S., and Kalantar, S., "Design of Optimum Sensor Configurations for Mobile Autonomous Line-Tracking Robots", Proceedings of the 8th Iranian Conference on Electrical Engineering, Tehran, 1999.

[8] Harvey, I., Husbands, P., Cliff, D., Thompson, A., and Jakobi, N., "Evolutionary Robotics: the Sussex Approach", Robotics and Autonomous Systems, 20, 1997.

[9] Sanchez, E. and Tomassini, M. (eds.), Towards Evolvable Hardware, Springer-Verlag, Lecture Notes in Computer Science, 1996.

[10] Kalantar, S., Eslambolchi, P., Nili, M., Technical report MI&R7912-1, Robotics Lab, TU, 1999 (in Persian).

[11] Kalantar, S., Eslambolchi, P., Nili, M., Using Population Based Incremental Learning in Evolution of Autonomous Mobile Robots with Line Tracking Behavior Proceedings of the 9th Iranian Conference on Electrical Engineering, Tehran, 2001 (in Persian).