The Effect of the Environment in the Synthesis of Robotic Controllers: A Case Study in Multi-Robot Obstacle Avoidance using Distributed Particle Swarm Optimization

Ezequiel Di Mario, Iñaki Navarro and Alcherio Martinoli

Distributed Intelligent Systems and Algorithms Laboratory,
School of Architecture, Civil and Environmental Engineering,
École Polytechnique Fédérale de Lausanne
{ezequiel.dimario, inaki.navarro, alcherio.martinoli}@epfl.ch

Abstract

The ability to move in complex environments is a fundamental requirement for robots to be a part of our daily lives. While in simple environments it is usually straightforward for human designers to foresee the different conditions a robot will be exposed to, for more complex environments the human design of high-performing controllers becomes a challenging task, especially when the on-board resources of the robots are limited. In this article, we use a distributed implementation of Particle Swarm Optimization to design robotic controllers that are able to navigate around obstacles of different shape and size. We analyze how the behavior and performance of the controllers differ based on the environment where learning takes place, showing that different arenas lead to different avoidance behaviors. We also test the best controllers in environments not encountered during learning, both in simulation and with real robots, and show that no single learning environment is able to generate a behavior general and robust enough to succeed in all testing environments.

Introduction

In simple environments, it is usually straightforward for human designers to anticipate the different conditions a robot will be exposed to. Thus, robotic controllers can be designed manually by simplifying the number of parameters or inputs used. However, for more complex environments, the human design of high-performing controllers becomes a challenging task. This is especially true if the on-board resources of the robot are limited, as humans may not be aware of how to exploit limited sensing capabilities.

Machine-learning techniques are an alternative to human design that can automatically synthesize robotic controllers in large search spaces, coping with discontinuities and nonlinearities, and find innovative solutions not foreseen by human designers. In particular, evaluative, on-board techniques can develop specific behaviors adapted to the environment where the robots are deployed.

The purpose of this paper is twofold. First, to verify whether different behaviors arise as a function of the learning environment in the adaptation of multi-robot obstacle avoidance. Secondly, to test how the learned behaviors perform in environments not encountered during learning, that is, to evaluate how general are the solutions found in the learning process. The adaptation technique used is Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995), which allows a distributed implementation in each robot, speeding up the adaptation process and adding robustness to failure of individual robots.

The remainder of this article is organized as follows. Section Background introduces some related work on PSO, and on the influence of the environment in robotic adaptation. In the Hypotheses and Methods section we propose two hypotheses that motivate our research and describe the experimental methodology used to test them. Section Results and Discussion presents the experimental results obtained and discusses the validity of the proposed hypotheses. Finally, we conclude the paper with a summary of our findings and an outlook for our future work.

Background

The background for this article is divided into two subsections, one briefly introducing PSO and related work on distributed implementations and robustness in the presence of noise, and the second one dealing with environmental complexity and its role in the adaptation of robotic controllers.

Particle Swarm Optimization

PSO is a relatively new metaheuristic originally introduced by Kennedy and Eberhart (1995), which was inspired by the movement of flocks of birds and schools of fish. Because of its simplicity and versatility, PSO has been used in a wide range of applications such as antenna design, communication networks, finance, power systems, and scheduling. Within the robotics domain, popular topics are robotic search, path planning, and odor source localization (Poli, 2008).

PSO is well suited for distributed/decentralized implementation due to its distinct individual and social components and its use of the neighborhood concept. Most of the work on distributed implementation has been focused on benchmark functions running on computational clusters (Akat and Gazi, 2008; Rada-Vilela et al., 2011). Implementations with mobile robots are mostly applied to odor source localization (Turduev and Atas, 2010; Marques et al., 2006), and robotic search (Hereford and Siebold, 2007), where the particles' position is usually directly matched to the robots' position in the arena.

Most of the research on optimization in noisy environments has focused on evolutionary algorithms (Jin and Branke, 2005). The performance of PSO under noise has not been studied so extensively. Parsopoulos and Vrahatis (2001) showed that standard PSO was able to cope with noisy and continuously changing environments, and even suggested that noise may help to avoid local minima. Pan et al. (2006) proposed a hybrid PSO-Optimal Computing Budget Allocation (OCBA) technique for function optimization in noisy environments. Pugh and Martinoli (2009) showed that PSO could outperform Genetic Algorithms on benchmark functions and for certain scenarios of limited-time learning in the presence of noise.

In our previous work (Di Mario and Martinoli, 2012), we analyzed in simulation how different algorithmic parameters in a distributed implementation of PSO affect the total evaluation time and the resulting fitness. We proposed guidelines aiming at reducing the total evaluation time so that it is feasible to implement the adaptation process within the limits of the robots' energy autonomy.

Role of the Environment

Regarding complexity, Al-Kazemi and Habib (2006) analyzed the internal behavior of PSO when the dimension of the problem is increased. They used different metrics to conclude that the PSO particles behave in a similar way independently of the complexity of the problem. Auerbach and Bongard (2012) studied the relationship between environmental and morphological complexity in evolved robots, showing that many complex environments lead to the evolution of more complex body forms than those of robots evolved in simple environments.

Nolfi (2005) proposed that the behavior of a robot (and of any other agent) depends on the interaction between its controller, its body, and the external environment (that can also consist of other robots). These interactions are non-linear and affect the behaviors as well as the learning process.

Nolfi and Parisi (1996) evolved neural network controllers for robotic exploration, switching between two different environments during the evolution process. They evolved two different neural networks: with and without the capability to learn how to behave in the environment where the robot is placed. Different behaviors resulted from evolution depending on whether learning was allowed and on the environment where the robots were tested.

Islam and Murase (2005) evolved a robotic controller for obstacle avoidance and used tools from chaos theory (return maps and Lyapunov exponents) to measure the complexity of the resulting behaviors in the learning environment and other testing environments.

Nelson et al. (2003) evolved robotic controllers while increasing the complexity of the environments during evolution. They compared the resulting fitness and evolution process with evolution performed only in the most complex world.

Berlanga et al. (2002) studied a coevolutive method for robot navigation where the initial positions of the robots used for evolving the controllers are also evolved. They evolved solutions for several environments (in most cases of similar complexity), and tested their fitness in the arena where each controller was evolved as well as in the remaining arenas. They did not find significant performance differences between the controllers, probably due to the similar complexity of the arenas used for learning.

Hypotheses and Methods

This article discusses how the environment affects the adaptation of controllers for multi-robot obstacle avoidance using a distributed implementation of PSO. Robots navigate autonomously in the presence of other robots in square arenas with obstacles of different size and shape. We look at the different environments where learning takes place, analyze the resulting behaviors, and test how the controllers perform in the environments where they did not previously learn.

Hypotheses

The experiments conducted in this paper are motivated by the following hypotheses regarding the influence of the environment in the adaptation of robotic controllers:

Hypothesis 1 *Different environments lead to different behaviors of the adapted controllers. This might be specially significant for considerably different environments (e.g., empty arena vs. very narrow corridor).*

Hypothesis 2 *Some learning environments may generate more robust controllers that perform better in situations not encountered during learning. This leads to the problem of choosing the correct environment (or set of environments) for the adaptation process in order to make the resulting controller robust to variations in the environment.*

Fitness Function

We use a metric of performance based on the work of Floreano and Mondada (1996), which is present in several studies on learning obstacle avoidance (e.g., Lund and Miglino (1996), Pugh and Martinoli (2009), Palacios-Leyva et al. (2013), and our own previous work Di Mario and Martinoli (2012)). The fitness function consists of three factors, all normalized to the interval [0, 1]:

$$
\begin{aligned}
f &= f_v \cdot (1-\sqrt{f_t}) \cdot (1-f_i) && (1)\\
f_v &= \frac{1}{N_{eval}} \sum_{k=1}^{N_{eval}} \frac{\max\{v_{l,k}+v_{r,k},0\}}{2} && (2)\\
f_t &= \frac{1}{N_{eval}} \sum_{k=1}^{N_{eval}} \frac{|v_{l,k}-v_{r,k}|}{2} && (3)\\
f_i &= \frac{1}{N_{eval}} \sum_{k=1}^{N_{eval}} i_{max,k} && (4)
\end{aligned}
$$

where $\{v_{l,k}, v_{r,k}\}$ are the normalized speeds of the left and right wheels at time step k, $i_{max,k}$ is the normalized proximity sensor activation value of the most active sensor at time step k, and N_{eval} is the number of time steps in the evaluation period. This function rewards robots that move forwards quickly (f_v), turn as little as possible (f_t), and stay away from obstacles (f_i).

Experimental Platform

Our experimental platform is the Khepera III, a differential wheeled robot with a diameter of 12 cm. It is equipped with nine infra-red sensors for short range obstacle detection, which in our case are the only external inputs for the controllers, and two wheel encoders, which are used to measure the wheel speeds for the fitness calculations.

Since the response of the Khepera III proximity sensors is not a linear function of the distance to the obstacles, the proximity values are inverted and normalized using measurements of the real robot sensor's response as a function of distance. This inversion and normalization results in a proximity value of 1 when touching an obstacle, and a value of 0 when the distance to the obstacle is equal to or larger than 10 cm.

Simulations are performed in Webots (Michel, 2004), a realistic physics-based submicroscopic simulator that models dynamical effects such as friction and inertia. In this context, by submicroscopic we mean that it provides a higher level of detail than usual microscopic models, faithfully reproducing intra-robot modules (e.g., individual sensors and actuators).

Controller Architecture

The controller architecture used is a recurrent artificial neural network of two units with sigmoidal activation functions $s(\cdot)$. The outputs of the units determine the wheel speeds $\{v_{l,t}, v_{r,t}\}$, as shown in Equation 5. Each neuron has 12 input connections: the 9 normalized infrared sensors values $\{i_1, \cdots, i_9\}$, a connection to a constant bias speed, a recurrent connection from its own output, and a lateral connection from the other neuron's output, resulting in 24 weight parameters in total $\{w_0, \cdots, w_{23}\}$.

$$
\begin{aligned}
v_{l,t} &= s(w_0 + \sum_{k=1}^{9} i_k \cdot w_k + w_{10} \cdot v_{l,t-1} + w_{11} \cdot v_{r,t-1})\\
v_{r,t} &= s(w_{12} + \sum_{k=1}^{9} i_k \cdot w_{k+12} + w_{22} \cdot v_{l,t-1} + w_{23} \cdot v_{r,t-1})
\end{aligned}
\qquad (5)
$$

Environments

We conduct experiments in four different environments, shown in Figure 1. The first one is an empty square arena of 2 m x 2 m, where the walls and the other robots are the only obstacles. The second and third environments are based on the same bounded arena, where cylindrical obstacles of two sizes are added in different numbers. The second environment has 20 medium-sized obstacles (diameter 10 cm), while the third has 40 small-sized obstacles (diameter 2 cm). The fourth environment is the same size as the empty arena with an inner wall of 1.5 m creating a continuous corridor of 25 cm width.

In simulation, the cylindrical obstacles are randomly repositioned before each fitness evaluation, meaning that the second and third environments are dynamic. In real-robot experiments, the obstacles are kept in fixed positions, the variation between runs is provided by the randomized initial pose of the robots. The third environment was not tested with real robots given the difficulty of keeping such thin cylinders vertical during collisions, but it should be noted that this kind of obstacles can occur in real environments, for example in the case of a chair or table with very thin legs.

All experiments are conducted with 4 robots. The method for initializing the robots' pose for each fitness evaluation is different between simulation and experiments with real robots. In simulation, the initial positions are set randomly with a uniform probability distribution, verifying that they do not overlap with obstacles or other robots. For the experiments with real robots, in the empty arena a random speed is applied to each wheel for three seconds to randomize the robots' pose. In the two arenas with obstacles and in the corridor one, the robots are manually repositioned to avoid disturbing the location of the obstacles, and then the robots turn in place with a random speed for two seconds to randomize their orientation.

Adaptation Algorithm

The optimization problem to be solved by the adaptation algorithm is to choose the set of weights $\{w_0, \cdots, w_{23}\}$ of the artificial neural network controller such that the fitness function f as defined in Equation 1 is maximized. The chosen algorithm is the distributed, noise-resistant variation of PSO introduced by Pugh and Martinoli (2009), which operates by re-evaluating personal best positions and aggregating them with the previous evaluations (in our case a regular

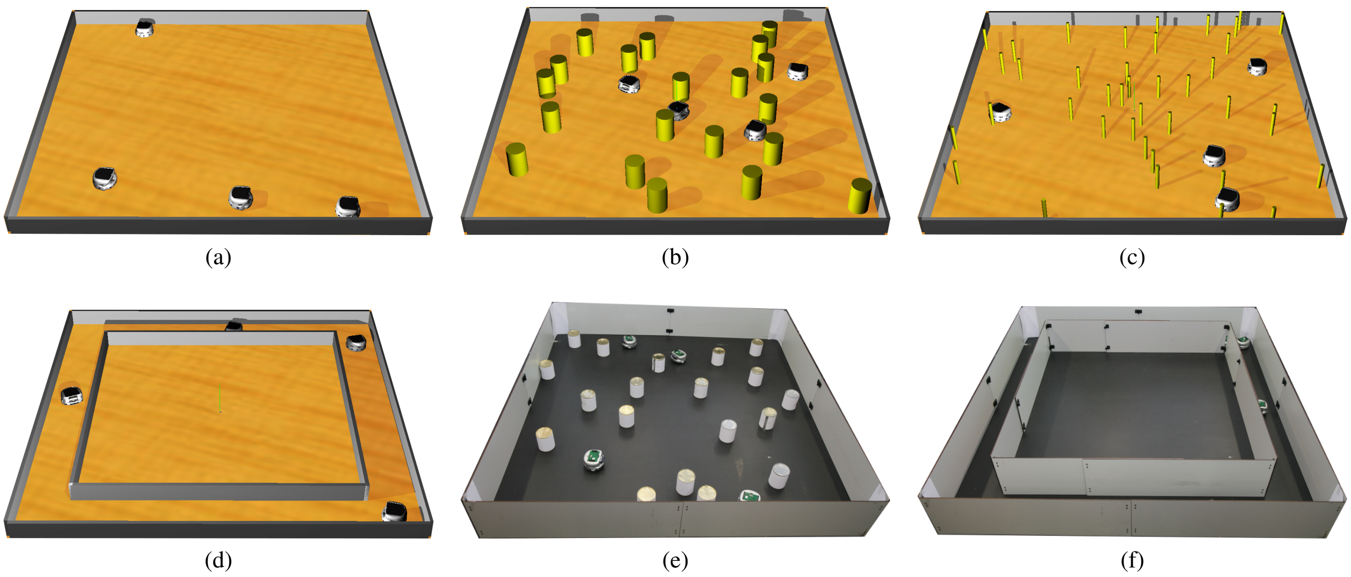


Figure 1: Different environments used in the adaptation and evaluation of the controllers. (a) Empty arena in simulation. (b) Medium-sized obstacles arena in simulation. (c) Small-sized obstacles arena in simulation. (d) Corridor arena in simulation. (e) Real medium-sized obstacles arena. (f) Real corridor arena.

```
1:  Intialize particles
2:  for $N_i$ iterations do
3:      for $\lceil N_p/N_{rob} \rceil$ particles do
4:          Update particle position
5:          Evaluate particle
6:          Re-evaluate personal best
7:          Aggregate with previous best
8:          Share personal best
9:      end for
10: end for
```

Figure 2: Noise-resistant PSO algorithm

average performed at each iteration of the algorithm). The pseudocode for the algorithm is shown in Figure 2.

The position of each particle is a 24-dimensional real-valued vector that represents the weights of the artificial neural network. The velocity of particle i in dimension j (shown in Equation 6) depends on three components: the velocity at the previous step weighted by an inertia coefficient w_I, a randomized attraction to its personal best $x^*_{i,j}$ weighted by w_P, and a randomized attraction to the neighborhood's best $x^*_{i',j}$ weighted by w_N. $rand()$ is a random number drawn from a uniform distribution between 0 and 1. The position of each particle is updated according to Equation 7.

$$\begin{aligned} v_{i,j} &:= w_I \cdot v_{i,j} + w_P \cdot rand() \cdot (x^*_{i,j} - x_{i,j}) \\ &\quad + w_N \cdot rand() \cdot (x^*_{i',j} - x_{i,j}) \qquad (6) \\ x_{i,j} &:= x_{i,j} + v_{i,j} \qquad (7) \end{aligned}$$

The algorithm is implemented in a distributed fashion, which reduces the total evaluation time required by a factor equal to the number of robots. Even if the learning in this paper is performed only in simulation, the algorithm can easily be executed completely on-board with very low requirements in terms of computation and communication.

Each robot evaluates in parallel a possible candidate solution and shares the solution with its neighbors in order to create the next pool of candidate solutions. The neighborhood presents a ring topology with one neighbor on each side. Particles' positions and velocities are initialized randomly with a uniform distribution in the $[-20, 20]$ interval, and their maximum velocity is also limited to that interval.

The PSO algorithmic parameters are set following the guidelines for limited-time adaptation we presented in our previous work (Di Mario and Martinoli, 2012) and are shown in Table 1.

Results and Discussion

The results of this article are presented as follows. First, we perform the learning in simulation in the four environments previously mentioned. Then, the best controller from each learning environment is tested in every environment in simulation. Finally, the four controllers from each learning

Table 1: PSO parameter values

Parameter	Value
Number of robots N_{rob}	4
Population size N_p	24
Iterations N_i	200
Evaluation span t_e	40 s
Re-evaluations N_{re}	1
Personal weight w_P	2.0
Neighborhood weight w_N	2.0
Dimension D	24
Inertia w_I	0.8
V_{max}	20

environment are also tested with real robots in three of the four environments.

Learning in Simulation with PSO

Since PSO is a stochastic optimization method and the performance measurements are noisy, each PSO optimization run may converge to a different solution. Therefore, for statistical significance, we performed in simulation 100 PSO adaptation runs for each learning environment. Figure 3 shows the progress of the PSO learning at each iteration for the four environments. Vertical bars show the standard deviation among the 100 PSO runs.

The highest performance corresponds to the empty arena since it is the easiest environment with just the bounding walls and the other robots acting as obstacles. The fitness in both environments with cylindrical obstacles is very similar for the whole learning process. The slowest learning rate occurs for the narrow corridor, indicating that this environment is more challenging for the learning algorithm. By the end of the adaptation process the performance is slightly lower than in the arenas with cylindrical obstacles.

It should be noted that the learning environment has a significant impact in the variation between runs, as the standard deviation is lowest in the empty arena and it increases markedly for the more complex environments.

Trajectories can be a useful tool to identify the behavior of the robots, as we have seen in our previous work (Di Mario et al., 2011). Figure 4 shows the resulting trajectories of the best learned behaviors in simulation for each environment where adaptation took place. It can be seen how in the empty arena and in the medium-sized obstacles arena the robot trajectories are straight until they find an obstacle (wall, cylindrical obstacle, or other robot), performing then a sharp turn and continuing straight afterwards.

The trajectory learned in the arena with small-sized obstacles is curvilinear when there are no obstacles within range. When the robot detects an obstacle, it makes a sharp turn to later continue its curvilinear movement. The small obstacles are thinner than the distance between two contiguous infrared sensors, so sometimes the robots are not able to detect

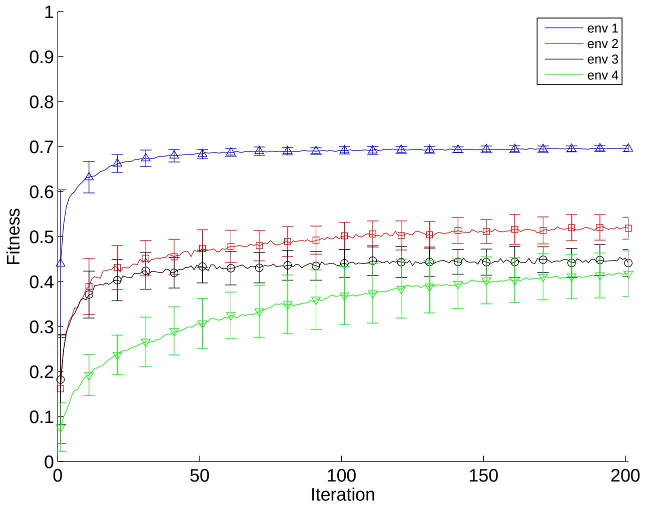


Figure 3: Best fitness found at each iteration for 100 PSO optimization runs. Bars represent the standard deviation across runs. Fitness in empty arena in blue (env 1). Fitness in arena with 20 medium cylindrical obstacles in red (env 2). Fitness in arena with 40 small cylindrical obstacles in black (env 3). Fitness in corridor arena in green (env 4).

them. Curvilinear movements may help in avoiding getting stuck in front of the small obstacles, and thus the behavior learned with PSO does not involve moving in straight lines as in the other cases.

In the corridor arena, the robot moves along the corridor, turning 90 degrees to head into the following sub-corridor, and thus exploring the whole arena.

As we conjectured in Hypothesis 1, the different environments cause the robots to learn different behaviors. In the next section we will show how the learned controllers behave in the other environments that were not encountered during learning.

Testing in Simulation

In the previous section, we obtained four different controllers corresponding to each environment where learning took place. In this section, we test the controllers in all environments to see how they perform in situations not encountered while learning, i.e., to see how general and robust are the obtained behaviors.

Figure 5a shows the boxplot of the fitness of 20 evaluation runs performed in simulation for each controller and testing environment. Since all experiments are conducted with 4 robots, this results in 80 fitness measurements per controller and environment. For the sake of brevity, we use T to denote testing environment, L for learning environment, and we number the environments from one to four in the following order: empty arena, arena with 20 medium cylindrical obstacles, arena with 40 small cylindrical obstacles and

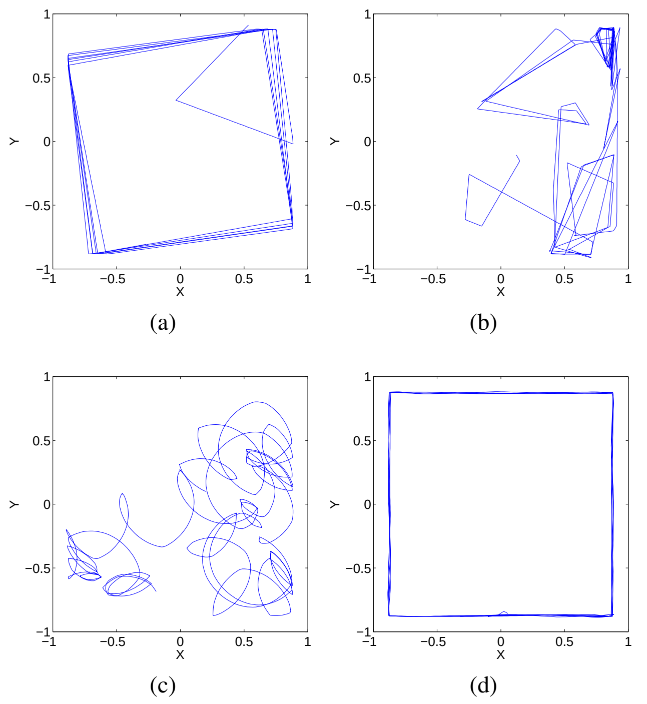


Figure 4: Trajectories of one of the four robots during a single experiment in simulation for the controllers learned in the four environments under study. (a) Empty arena. (b) Medium sized obstacles arena. (c) Small sized obstacles arena. (d) Corridor arena.

corridor arena. Thus, $T1L4$ for instance should be read as: test performed in the empty environment with the controller learned in the corridor environment.

As expected, for each environment, the controller learned in the testing environment has the highest performance. However, for the simplest environment ($T1$), there is no significant difference between the performance of controllers $L1$, $L2$, and $L4$. Regarding Hypothesis 2 concerning the generality of the learned behaviors, controller $L4$ seems to be the most robust, as it significantly outperforms all other controllers in the corridor and still performs almost as good as $L1$ in $T1$ and reasonable well in $T2$, although it performs poorly in $T3$.

Further insight on the performances can be obtained by analyzing the trajectories described by the robots in the different environments. Out of the 16 evaluation conditions, we show the ones we consider most interesting in Figure 6.

The behavior of controller $L1$ is similar to that of controller $L2$ in all testing environments (for example, compare the trajectories from Figure 6a and Figure 4b), since they employ similar avoidance strategies: moving in straight lines and making sharp turns near obstacles. This result becomes evident when considering that the medium-sized cylindrical obstacles are very similar in shape and size to

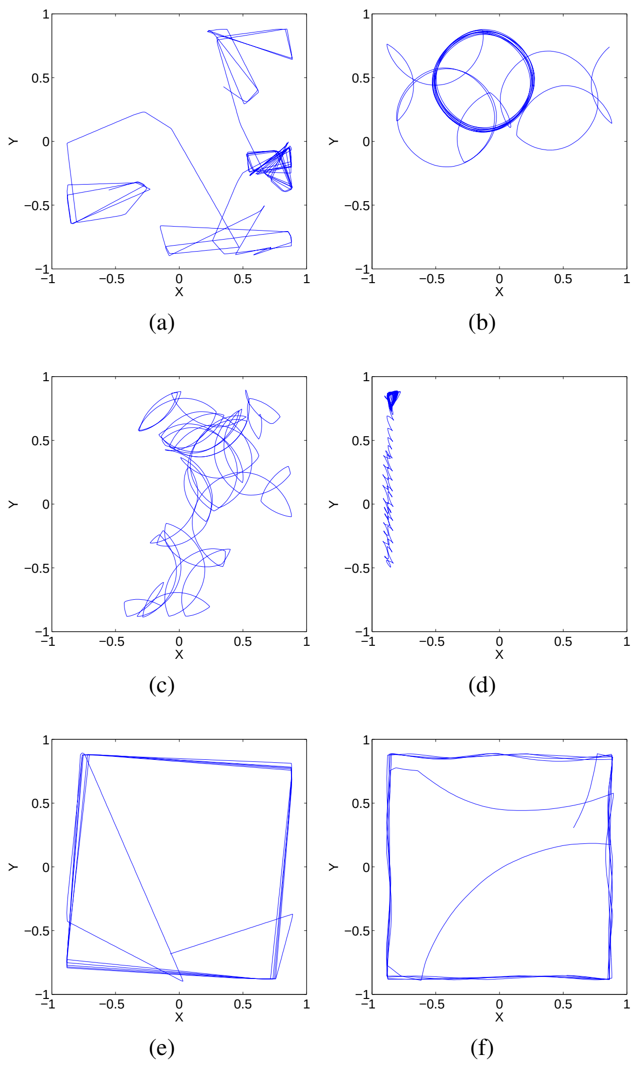
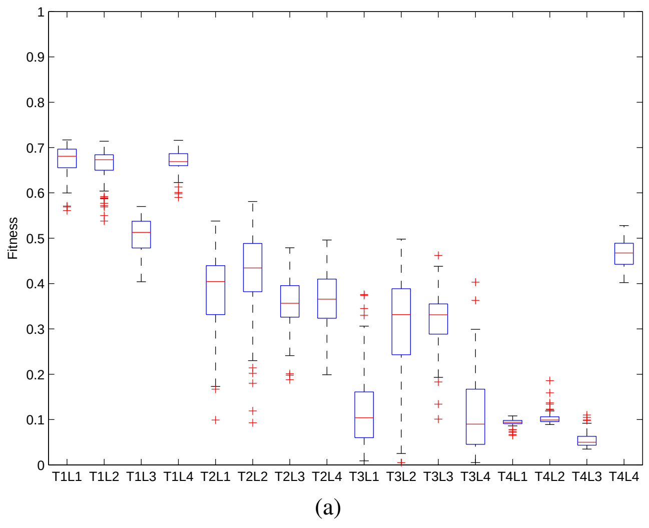


Figure 6: Trajectories of one of the four robots during a single experiment in simulation for different learned controllers (LX) and testing environments (TX). (a) T2L1. (b) T1L3. (c) T2L3. (d) T4L3. (e) T1L4. (f) T1L4*.

the Khepera III robot. However, maybe due to the higher obstacle density of Environment 2, controller $L2$ is more robust in the sense that it performs better in environments 3 and 4.

The curvilinear behavior of controller $L3$, which enables it to avoid very thin obstacles, is also observed with the larger obstacles of Environment 2 (Figure 6c), and results in fully circular trajectories in the empty environment (Figure 6b). However, this controller as well as controllers $L1$ and $L2$ were not able to move along the corridor, doing instead short straight movements alternated with sharp turns (Figure 6d).

Controller $L4$ was the only one able to move smoothly along the corridor in Environment 4, performing well in all environments except $T3$. The behavior learned can be ob-



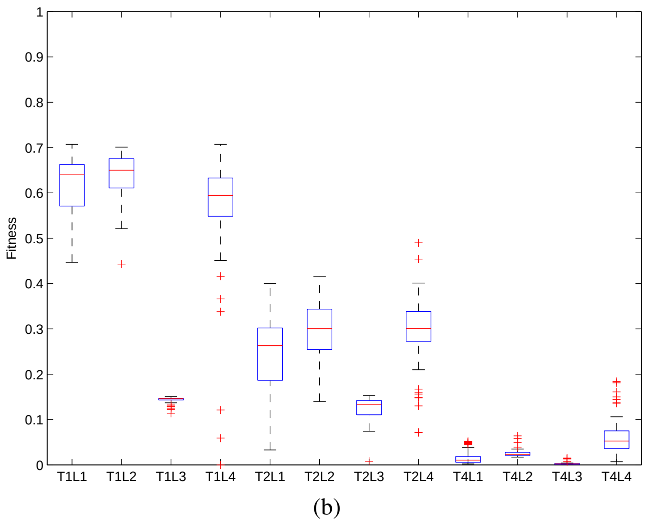


Figure 5: Boxplot showing the fitness of the four learned controllers (L1-L4). (a) Evaluated in the four testing environments (T1-T4) in simulation. (b) Evaluated in three testing environments (T1, T2 and T4) with real robots. The box represents the upper and lower quartiles, the line across the middle marks the median, and the crosses show outliers.

served when tested in the empty environment ($T1L4$) in Figure 6e. The robot moved straight performing a 90 degree sharp turn when finding an obstacle. This exact 90 degree turn was learned in the corridor environment to perform the transition from one sub-corridor to another.

As mentioned previously, we run 100 PSO runs for each environment, and controller $L4$ is the best-performing one from the 100 runs in the corridor environment, but we noticed that not all the resulting controllers have the same behavior. A different controller resulting from the corridor environment is shown for the empty arena ($T1L4*$) in Figure 6f. The robot learned a wall-following behavior, performing a curvilinear movement in the absence of obstacles.

However, when testing this controller in the corridor ($T4L4*$) the trajectory looks exactly the same as the one from $T4L4$. Thus, it is interesting to notice that this behavior could only be observed when testing in other environments than the learning one, which shows the importance of using varied environments to observe the whole range of behaviors of a given controller.

Testing with Real Robots

In order to validate the results obtained in simulation, we tested the same controllers with real robots in environments 1, 2, and 4. We did 20 evaluation runs with 4 robots, leading to 80 fitness measurements per case. The resulting fitness is shown in Figure 5b.

As in simulation, the performance of controllers $L1$ and $L2$ was similar. Again, controller $L4$ seemed to be the most robust, outperforming all other controllers in the corridor and performing similarly to the best controllers in the other two environments.

Controller $L3$ suffered a noticeable performance drop when going from simulation to reality due to an unmodeled effect: the Khepera III motors' were not able to work smoothly at low speeds, and thus the inner wheel in the circular movements in open spaces was practically stopped, resulting in circles with a very small radius.

Finally, controller $L4$ was also able to move along the corridor as in simulation, although the behavior was not as smooth and turns midway through the corridor were more frequent than in simulation (probably due to inaccuracies in the sensor model and the increased noise in real environments). Thus, the real-world performance was much lower.

Conclusion

In this paper, we studied the effect of the environment on the multi-robot learning of an obstacle avoidance behavior. We showed that the same controller architecture, fitness function, and learning algorithm implemented in different environments lead to different avoidance behaviors, such as moving in straight lines with sharp turns, curvilinear movements, and wall-following around obstacles. We then tested the learned controllers in environments not encountered during learning, both in simulation and with real robots, which allowed us to see the full range of behaviors of each controller. Finally, we saw that no single learning environment was able to generate a behavior general enough to succeed in all testing environments.

As future work, we intend to study the interplay between architectural complexity and capability of generalization. In other words, we would like to know how to design a learning environment, or maybe a set of environments if required, that lead to general and robust avoidance behaviors while maintaining the architecture complexity low. It would also be interesting to study the interplay between a certain fitness function and the required architecture complexity. This work is part of our ongoing effort to develop distributed, noise-resistant adaptation techniques that can optimize high-performing robotic controllers quickly and robustly.

Acknowledgement

This research was supported by the Swiss National Science Foundation through the National Center of Competence in Research Robotics.

References

Akat, S. B. and Gazi, V. (2008). Decentralized asynchronous particle swarm optimization. In *IEEE Swarm Intelligence Symposium*.

Al-Kazemi, B. and Habib, S. (2006). Complexity analysis of problem-dimension using PSO. In *WSEAS International Conference on Evolutionary Computing*, pages 45–52.

Auerbach, J. E. and Bongard, J. C. (2012). On the relationship between environmental and morphological complexity in evolved robots. In *Genetic and Evolutionary Computation Conference*, pages 521–528. ACM Press.

Berlanga, A., Sanchis, A., Isasi, P., and Molina, J. M. (2002). Neural network controller against environment: A coevolutive approach to generalize robot navigation behavior. *Journal of Intelligent and Robotics Systems*, 33(2):139–166.

Di Mario, E. and Martinoli, A. (2012). Distributed Particle Swarm Optimization for Limited Time Adaptation in Autonomous Robots. In *International Symposium on Distributed Autonomous Robotic Systems 2012, Springer Tracts in Advanced Robotics 2014 (to appear)*. Available at http://infoscience.epfl.ch/record/182403.

Di Mario, E., Mermoud, G., Mastrangeli, M., and Martinoli, A. (2011). A trajectory-based calibration method for stochastic motion models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4341–4347.

Floreano, D. and Mondada, F. (1996). Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26(3):396–407.

Hereford, J. and Siebold, M. (2007). Using the particle swarm optimization algorithm for robotic search applications. In *IEEE Swarm Intelligence Symposium*, pages 53–59.

Islam, M. M. and Murase, K. (2005). Chaotic dynamics of a behavior-based miniature mobile robot: effects of environment and control structure. *Neural Networks*, 18(2):123 – 144.

Jin, Y. and Branke, J. (2005). Evolutionary Optimization in Uncertain Environments - A Survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks*, pages 1942 – 1948 vol.4.

Lund, H. and Miglino, O. (1996). From simulated to real robots. In *IEEE International Conference on Evolutionary Computation*, pages 362–365.

Marques, L., Nunes, U., and Almeida, A. T. (2006). Particle swarm-based olfactory guided search. *Autonomous Robots*, 20(3):277–287.

Michel, O. (2004). Webots: Professional Mobile Robot Simulation. *Advanced Robotic Systems*, 1(1):39–42.

Nelson, A., Grant, E., Barlow, G., and White, M. (2003). Evolution of complex autonomous robot behaviors using competitive fitness. In *International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pages 145–150.

Nolfi, S. (2005). Behaviour as a complex adaptive system: On the role of self-organization in the development of individual and collective behaviour. *ComPlexUs*, 2(3-4):195–203.

Nolfi, S. and Parisi, D. (1996). Learning to adapt to changing environments in evolving neural networks. *Adaptive Behavior*, 5:75–98.

Palacios-Leyva, R. E., Cruz-Alvarez, R., Montes-Gonzalez, F., and Rascon-Perez, L. (2013). Combination of reinforcement learning with evolution for automatically obtaining robot neural controllers. In *IEEE International Conference on Evolutionary Computation*, pages 119–126.

Pan, H., Wang, L., and Liu, B. (2006). Particle swarm optimization for function optimization in noisy environment. *Applied Mathematics and Computation*, 181(2):908–919.

Parsopoulos, K. E. and Vrahatis, M. N. (2001). Particle Swarm Optimizer in Noisy and Continuously Changing Environments. In Hamza, M. H., editor, *Artificial Intelligence and Soft Computing*, pages 289–294. IASTED/ACTA Press.

Poli, R. (2008). Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, 2008(2):1–10.

Pugh, J. and Martinoli, A. (2009). Distributed scalable multi-robot learning using particle swarm optimization. *Swarm Intelligence*, 3(3):203–222.

Rada-Vilela, J., Zhang, M., and Seah, W. (2011). Random Asynchronous PSO. *The 5th International Conference on Automation, Robotics and Applications*, pages 220–225.

Turduev, M. and Atas, Y. (2010). Cooperative Chemical Concentration Map Building Using Decentralized Asynchronous Particle Swarm Optimization Based Search by Mobile Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4175–4180.