Robot control inspired by Escherichia Coli chemotaxis

Fabio DallaLibera*, Shuhei Ikemoto**, Takashi Minato***,
Hiroshi Ishiguro** ***, Emanuele Menegatti*

*University of Padova, Italy
**Osaka University
***JST ERATO

Living beings like bacteria search food using extremely simple strategies that reveal to be very robust. From this observation we derive an algorithm for robot control. Most literature on the topic defines basic robot behaviors that are used to mimic bacteria movement in the physical space. This paper proposes instead to work directly in the motor command space, allowing the robot to determine itself the control signals to use. The strategy underlying the algorithm consists in simply repeating control signals that lead to condition improvements. Counterintuitively, adding random perturbations to the control signal being repeated improves the performances. A model of the phenomenon is provided and an algorithm for automatic choosing the perturbation magnitude is proposed. Results of experiments on the robustness and practical applicability of the approach are reported. In particular we show that the algorithm is able to perform robot navigation even when no information on the robot structure is available and the robot undergoes hardware damages.

Key Words Search, Motion Generation, Random Walk

1. Introduction

Living beings like bacteria often employ very simple but robust strategies to locate food sources. For instance, Escherichia Coli performs a random walk biased toward increasing concentrations of nutrients [1]. In particular this bacterium has only two way of moving, rotating clockwise or counter-clockwise. When it rotates counter-clockwise the rotation aligns its flagella into a single rotating bundle and it swims in a straight line. Conversely clockwise rotations break the flagella bundle apart and the bacterium tumbles in place. The bacterium keeps alternating clockwise and counterclockwise rotations. In absence of chemical gradients the length of the straight line paths, i.e. the duration of counter-clockwise rotations, is independent of the direction. The bacterium therefore essentially performs a random walk. In case of a positive gradient of attractants, like food, Escherichia Coli instead reduces its tumbling frequency. In other terms, when the conditions are improving the bacterium proceeds in the same direction for a longer time. This simple strategy allows to bias the overall movement toward increasing concentrations of the attractant, despite the simplicity of the mechanism and the difficulties in precisely sensing the gradient. This methodology has been applied to robot navigation in [2]. It was shown that while gradient descent is faster for tracking a single source, biased random walks perform better in the presence of multiple and dissipative sources and noisy sensors and actuators. Furthermore the stochastic nature of the algorithm prevents it from getting stuck in local minima.

In [2] two basic movements, proceed straight and change direction randomly, are defined beforehand and the control algorithm switches between these two behaviors, prolonging the straight runs when the conditions improve (e.g. when the robot is getting closer to the goal). This approach strongly limits the robustness over unexpected hardware failures. In fact, if due to hardware damages one of these basic movements does not operate as expected in many cases it will not be possible to accomplish the tasks.

A vast literature on recovering from hardware damages is available, ranging from failure detection [3] to very advanced works on self-modeling [4]. However, simply applying the random walk directly in the motor command space can reveal sufficient to obtain robustness to hardware failures in many cases. Concretely, operating random walk in the motor command space introduces the possibility of exploring new motor commands that utilize the available hardware configuration.

This kind of approach is employed by our algorithm, reported in Section 2. Roughly speaking, the algorithm simply selects a random control signals and keeps repeating it as long as the robot's conditions improve. We then show the very unintuitive fact that adding random perturbations to the selected action can improve the performances. For instance, adding random perturbations of appropriate magnitude can decrease the reaching time in a navigation task. We provide an explanation of the phenomenon and propose an algorithm for automatically adjusting the perturbation magnitude.

Section 3 reports experimental results. We show that the algorithm is able to control systems that include high dimensionality, strong nonlinearities, low-pass filtering effects or dead time. Simulation results with a mobile robot indicate that the algorithm can be applied to practical problems such as driving a mobile robot toward a goal without knowing the robot structure. Furthermore, the flexibility of the algorithm allows it to bring the robot to the goal even if substantial hardware damages occur.

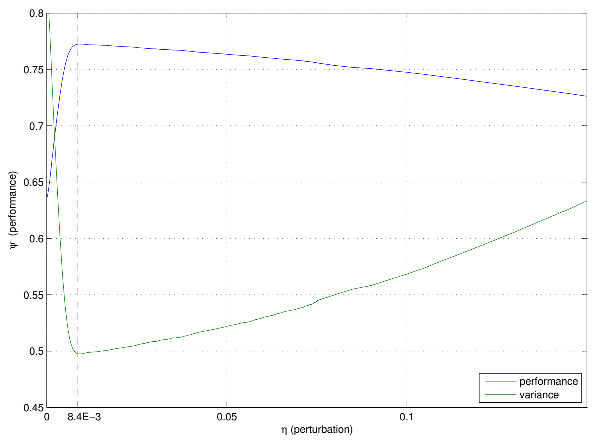

Figure 1: Average performance obtained for different values of η. The graphs were obtained placing the particle in $x_0 = [-1, 0]^T$ with $s = 10^{-6}$, $N = 10^4$ and repeating the test 10^5 times.

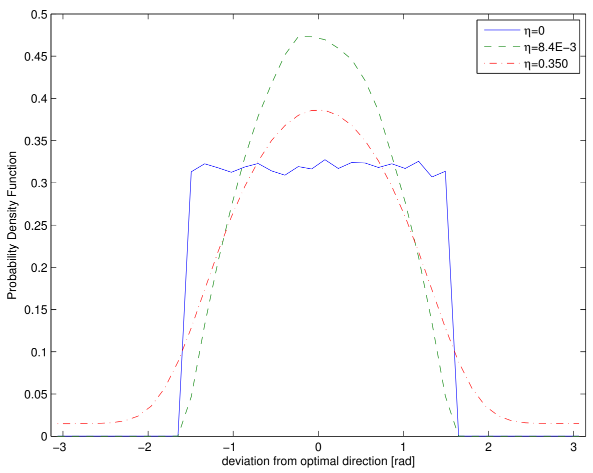

Figure 2: Distribution of the angles followed by the particle with respect to the optimal direction obtained for $\eta = \{0, 0.0084, 0.33\}$.

Finally, Section 4 concludes the paper summarizing the results.

2. Control Algorithm

As reported in the introduction, Escherichia Coli proceeds by movements in random directions, but when moving toward increasing concentrations of nutrients the movement in that direction is prolonged. Imagine now to have a robot. Denote by $u_t \in \mathbb{R}^p$ the motor control signal and by ΔA_t the change in the quality of the robot conditions obtained by applying u_t. For instance, ΔA_t could represent how much the robot got closer to a goal. Then a behavior similar to the Escherichia Coli's one could be obtained by taking $u_{t+1} = u_t$ if $\Delta A_t \geq 0$ and selecting u_{t+1} randomly[1] if $\Delta A_t < 0$.

However, if instead of $u_{t+1} = u_t$ we add a perturbation $u^i_{t+1} = u^i_t + \eta^i R$, $R \sim \mathcal{N}(0, 1)$ for each of the components of the input ($1 \leq i \leq p$), then for opportune values of η^i we can obtain a performances increase. As a simple example, suppose to have a particle in a two dimensional space that must approach a goal. Suppose the particle to move by steps of fixed length s, along the angle indicated by u_t. Formally let $x \in \mathbb{R}^2$ be the particle position, $u_t \in \mathbb{R}$, $x_{t+1} = x_t + s \cdot \begin{bmatrix} \cos(u_t) \\ \sin(u_t) \end{bmatrix}$. Assume for simplicity to have the goal located in $[0, 0]^T$, then the variation of the robot conditions ΔA_t is given by $\|x_{t-1}\| - \|x_t\|$. Finally, assume to express the performance as the average decrease in the distance to the goal for a single step, $\psi = \frac{\|x_0\| - \|x_{N-1}\|}{N}$.

Fig. 1 reports the average performance ψ for different values of η. For $0 < \eta < 0.33$ the performance is better than for $\eta = 0$.

Figure 2 provides an intuitive explanation. Suppose to repeat the test a high number of times and observe the distribution of the directions taken by the particle with respect to the optimal direction (i.e. heading straight to the goal). For $\eta = 0$ the distribution is approximately uniform in the range $[-\pi/2, \pi/2]$. For opportune values of η the distribution becomes more peaky. In fact inefficient paths along directions that are far from the optimal are modified by the perturbations into directions that lead to a negative ΔA_t and are therefore discarded in a short time. For too high values of η the perturbations start to increase the probability of every orientation, including the ones highly far from the optimal and, consequently, the probability of choosing directions very close to the optimal decreases.

Observing Fig. 1 we can see that the variance of u_t decreases when the performance increases. Intuitively, if the system selects good inputs these can be used for a long time, and the variance decreases. This can be used as a criterion to dynamically adapt η^i_t: we estimate the variance of u^i_t by picking some samples, slightly increase(/decrease) η^i_t, and estimate the variance of u^i_t again. If the variance decreased then we increase(/decrease) η^i_t once more, otherwise we decrease (/increase) it.

Assuming to estimate the variance using just two samples[2] the algorithm becomes

$$u^i_{t+1} = \begin{cases} u^i_t + \eta^i_t R & if\ \Delta A_t \geq 0 \\ random\ selection & otherwise \end{cases}$$

$$\delta^i_0 = 1.1$$

$$\sigma^i_t = \frac{(u^i_t - u^i_{t-1})^2}{2}$$

$$\delta^i_{t+1} = \begin{cases} 1/\delta^i_t & if\ t\ odd \wedge \sigma^i_t \geq \sigma^i_{t-2} \\ \delta^i_t & otherwise \end{cases}$$

[1] In the following we assume to select u_t using a uniform distribution over the whole motor command space, but the results remain essentially the same using different distributions.

[2] A higher number of samples provides a better estimate of the variance and therefore of the variance change, but slows down the adaptation. Note that, however, two samples are always sufficient to guarantee a right estimation of the whether the variance increased or not with a probability higher then 0.5.

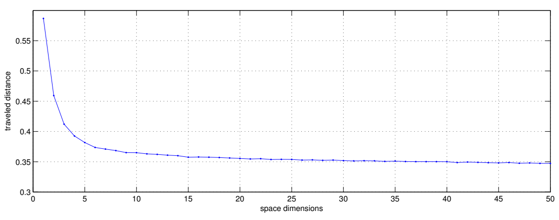

Figure 3: Average movement toward the goal by the particles for different settings of the space dimension p.

$$\eta_{t+1}^i = \begin{cases} \eta_t^i \delta_{t+1}^i & if\ t\ odd \\ \eta_t^i & otherwise \end{cases}$$

3. Experimental results

In order to evaluate the robustness and generality of the presented algorithm we conducted a series of experiments. At first we tested the performances of the algorithm as the dimensions of the space increase. The experiment was conducted placing 10^4 particles in a p dimensional space at $x_0 = [-1, 0, \ldots, 0]^T \in \mathbb{R}^p$ and assuming their position to be modified by $x_{t+1} = x_t + s \cdot u_t$ where $s = 10^{-4}$. Fig. 3 reports how much the particles got closer to the goal in $N = 10^4$ steps as p varies. We notice that the performance is reasonably high even for high values of p.

We then fixed $p = 2$, set the initial position of the particles to $x_0 = [10, 10]^T$ and imposed $s = 10^{-3}$. We firstly performed a test to check whether the adaptation of η_t^i works correctly. In particular, we identified the best performance achievable by a constant value of η, expressly for $\eta^1 = \eta^2 = \eta^* = 0.0034$ the particles got 8.234 units closer to the goal in $N = 10^4$ steps. With the adaptive algorithm the average distance traveled by the particles toward the goal was 8.129, i.e. the 98.7% of the optimal value.

To study the effect of noise in the measurement of ΔA_t we conducted an experiment where we switched the sign of ΔA_t with probability ω. It resulted that the distance traveled toward the goal decreases linearly increasing ω, reaching 0 for $\omega = 0.5$.

Next we checked whether the algorithm can cope with nonlinearities. In this experiment, the movement of the particles was set to $x_{t+1}^i = x_t^i + s \cdot f^i(u_t)$ where $f^i(x) = \frac{1}{\pi} \arctan\left(\frac{(sin(2\pi x+\xi^i))^T Q^i sin(2\pi x+\xi^i)}{(sin(2\pi x+\zeta^i))^T P^i sin(2\pi x+\zeta^i)}\right)$. In this expression the sin function is applied element-wise and $Q^i, P^i \in \mathbb{R}^{2\times 2}$ and $\xi^i, \zeta^i \in \mathbb{R}^2$ were randomly initialized. Figure 4 shows that the performance is very high for most of the particles.

In order to verify whether the algorithm works with systems that include low-pass filter effects we then simulated the case $x_{t+1} = x_t + v_t$ and $v_t = (1 - 10^{-\rho})v_{t-1} + 10^{-\rho}u_t$. Figure 5 shows the average performance as ρ varies. We notice that for some settings of ρ the performance is higher than with $\rho = 0$. This is due to a smoothing effect introduced by the low pass filter that makes the trajectories more straightly headed to the goal.

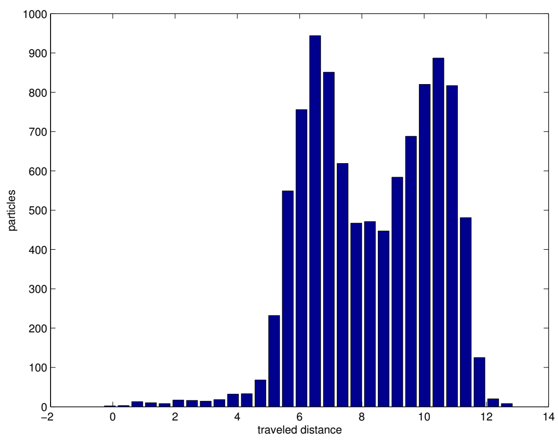

Figure 4: Distribution of the distance toward the goal traveled by 10^4 particles in $N = 10^4$ steps of size $s = 10^3$ using the nonlinear functions $f^i(x)$.

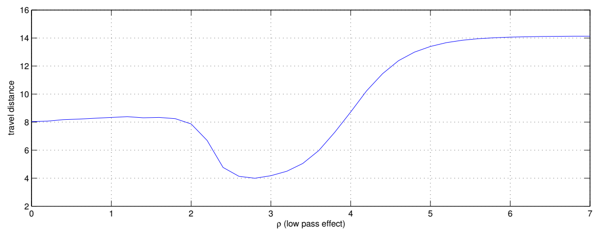

Figure 5: Average movement toward the goal for different settings of the low pass filter.

Figure 6 reports the performance for a system that includes a dead time (delay). Expressly, we simulated the case $x_{t+1} = x_t + u_{t-d}$. Performance degrades as the dead time increases, but does not reach 0, i.e. the algorithm is still able to drive the particles to the goal whatever the dead time is.

Using ODE[3], we simulated a mobile robot equipped with three spherical wheels having diameter of 15 cm. The two front wheels are directly actuated by two independent motors whose maximum velocity is 0.5 rad/s while the rear wheel is free to rotate in any direction. The task is to reach a red hemisphere of radius 10 m placed at

[3]Open Dynamics Engine, a free library for simulating rigid body dynamics. For details see `http://www.ode.org`.

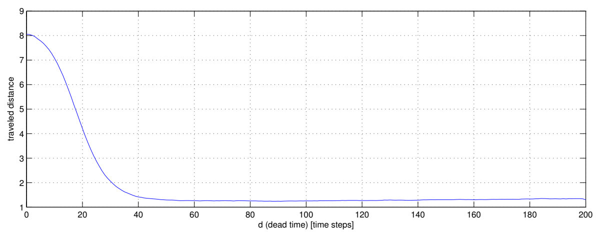

Figure 6: Average movement toward the goal for different dead time values.

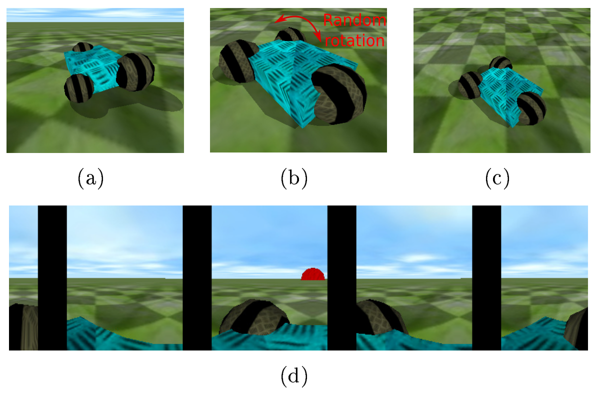

Figure 7: The four damages simulated. (a) Reduced size of one wheel. (b) Uncontrollability of one wheel (c) Rotation of the rotation axis of one wheel (note the direction of the stripes) (d) Obscuration of part of the acquired image.

a distance of 30m. The robot is equipped with an omnidirectional camera and the value of ΔA_t is determined observing the change in the number of red pixels in the image. In particular, if the R component of a pixel is more than double the maximum of the G and B components, then the pixel is considered red.

We simulated the robot in five different conditions, in the normal condition and with four types of damages (see Figure 7):

1. one wheel size is reduced to two thirds of its normal size
2. one wheel becomes uncontrollable, i.e. its movement is completely random
3. one wheel rotation axis direction is turned 90 degrees along the Z axis and becomes parallel to the longitudinal axis, i.e. the rotation of the wheel instead pushing the robot forward and backward pushes the robot to the left or to the right
4. 20% of the camera image becomes obscurated

In all the cases the algorithm was able to drive the robot to the goal. Fig. 8 reports the distribution of the number of time steps required to reach the target.

In summary, the experiments conducted show that the algorithm reveals to be very robust. It is able to work in highly dimensional spaces and cope with sensory noise, system nonlinearities, low-pass filtering effects and dead time. Furthermore simulation experiments with a simulated mobile robot suggest applicability to practical problems. In detail the algorithm performed well in driving the robot to a goal despite a number of substantial hardware damages.

4. Conclusions

This paper presents a control algorithm inspired by Escherichia Coli's chemotaxis. All relies on the simple assumption that if a control signal improved the state of the system, then applying the same signal for a while will often give good results. The algorithm requires as

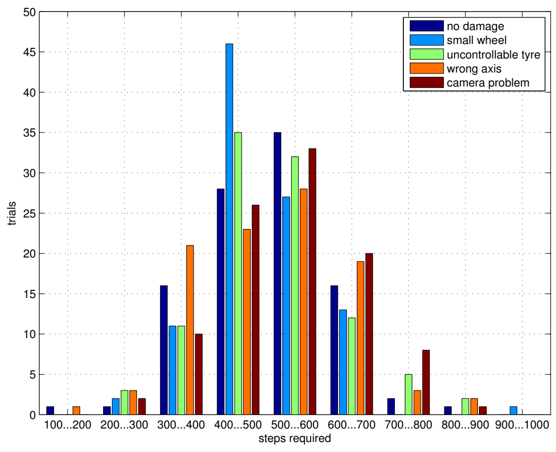

Figure 8: Distribution of the time steps required to reach the target. The time step was set to 5s, and the experiment was repeated 100 times for each robot condition.

input only a single binary value that indicates whether the conditions improved or not during the previous time step. This makes the algorithm applicable to a big class of problems, as long as it is true that repeating commands that improved the robot conditions have a good probability to give further improvements in the successive steps.

We showed that, unexpectedly, if the control signal is modified by random perturbations of opportune magnitude then the performances of this simple algorithm can be improved. A two dimensional case is provided as an example and used to derive an algorithm for the automatic adaptation of the perturbation size. Results of several tests that suggest the generality and robustness of the algorithm were presented. In detail we showed that robust navigation of a mobile robot toward a goal can be easily achieved without any parameter tuning.

Future works will involve introducing learning in the system. In detail, we can imagine to use the presented algorithm as a way to collect statistics on the relationships between the sensory information and the performance of different motor commands. This could in turn be used to extract information on the topology of the sensory state and action space before starting to learn the optimal policy, i.e. the mapping that gives the correct action to take given a certain perceived state.

References

[1] Adler, J. "The sensing of chemicals by bacteria". Scientific American, vol. 234, 40–47, 1976.

[2] Dhariwal, A., Sukhatme, G.S. and Requicha, A.A.G. "Bacterium-inspired robots for environmental monitoring". pp. 1436–1443.

[3] Scheutz, M. and Kramer, J. "Reflection and reasoning mechanisms for failure detection and recovery in a distributed robotic architecture for complex robots". pp. 3699–3704. 2007.

[4] Bongard, J., Zykov, V. and Lipson, H. "Resilient machines through continuous self-modeling". Science, vol. 314(5802), 1118–1121, 2006. doi:10.1126/science.1133687.