Abstract
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. Instead, this paper proposes 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.
Counter intuitively, 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.

Related works

Bacteria using a very simple strategy can reach food sources despite noise and world uncertainties [1]

Robots with similar behavior proposed [2]
- Robust to noise
- Do not stuck in local minima
- Robots distribute among multiple sources

Switching between two behaviors
- Rotate
- Go straight

The robot may not reach the goal due to hardware faults

For robot fault detection and recovery very advanced techniques available[3][4]

Apply biased random walk directly in the command space
- appropriate behaviors that exploit the working hardware are found

Control algorithm

If the conditions improved (e.g. got closer to the goal) keep the previous motor command
else choose a random motor command

\[ u_{t+1}^i = \begin{cases} 
  u_t^i + \eta_t^i R & \text{if } \Delta A_t \geq 0 \\
  \text{random selection} & \text{otherwise} 
\end{cases} \]

Performance improvement by noise

Stochastic resonance
Surprisingly, adding random perturbations of an opportune magnitude to the control input improves the performance

\[ u_{t+1}^i = \begin{cases} 
  u_t^i + \eta_t^i R & \text{if } \Delta A_t \geq 0 \\
  \text{random selection} & \text{otherwise} 
\end{cases} \]

Goal reaching example

Random perturbations help to detect commands that improve the robot condition (proximity to the goal) just slightly

Automatic selection of the perturbation amplitude

- Appropriate perturbation amplitudes give good control inputs that can be used for long time
- The control input variance decreases when the perturbation is appropriate
- The control input variance can be used to automatically select the appropriate perturbation amplitude

Perturbation magnitude adaptation algorithm

- \( \delta_t^i = 1.1 \)
- \( \sigma_{t+1}^i = \frac{(\sigma_t^i)^2}{2} \)
- \( \delta_{t+1}^i = \frac{1}{2} \) if \( t \text{ odd} \) and \( \sigma_t^i \geq \sigma_{t-1}^i \)
- \( \delta_{t+1}^i = \frac{2}{\eta} \delta_t^i \) otherwise
- \( \delta_{t+1}^i = \frac{\eta \delta^i_{t+1}}{\eta} \) if \( t \text{ odd} \)
- \( \delta_{t+1}^i \) otherwise
### Performance measurements

**Nonlinearities of the system**

Control algorithm

\[ a_{t+1} = \begin{cases} a_t + a_t R & \text{if } \Delta A_t \geq 0 \\ \text{random selection} & \text{otherwise} \end{cases} \]

Condition evaluation

\[ \Delta A_t = ||x_{t-1}|| - ||x_t|| \]

Distribution of the distance toward the goal traveled by \(10^4\) particles in \(N = 10^4\) steps of size \(s = 10^{-3}\)

**Dimensions of the search space**

\[ x_0 = [-1, 0, \ldots, 0]^T \in \mathbb{R}^p \]

\[ x_{t+1} = x_t + 8 \cdot u_t \]

**Low-pass filtering effects in the system dynamics**

Low-pass system dynamics

\[ v_t = (1 - 10^{-3}) v_{t-1} + 10^{-3} u_t \]

\[ x_{t+1} = x_t + v_t \]

**Dead time in the system dynamics**

Dead time system dynamics

\[ x_{t+1} = x_t + u_{t-d} \]

### Experiment

**Setup**

**Robot:** simulated mobile robot equipped with two independent wheels and an omnidirectional camera

**Task:** reach a red hemisphere

**Sensory information:** number of red pixels in the camera image

**Simulated four different damages**

- Variation of the size
- Change of the rotation axis
- Uncontrollability of a wheel

**Obscuration of 20% of the camera information**

### Results

- The algorithm could lead the robot to the target without any prior information on the robot structure (axis orientation, wheel position, etc.)
- The robot was able to reach the goal for each of the tested damages

### References