Exploiting physical dynamics for concurrent control of a mobile robot

Brian P. Gerkey Maja J Matarić
Gaurav S Sukhatme

http://robotics.usc.edu

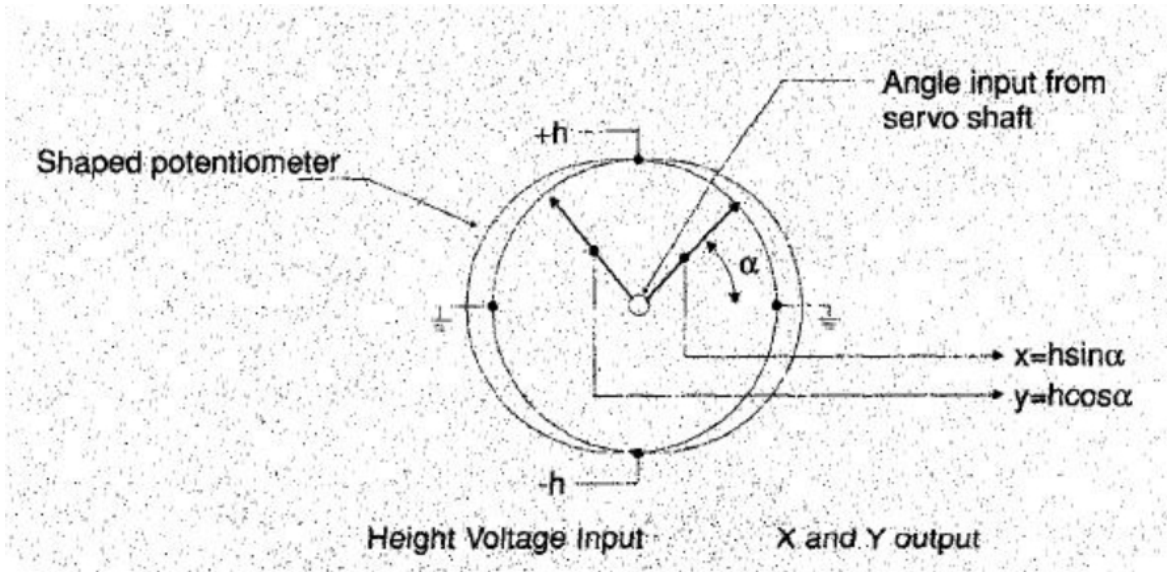
ICRA 2002, Washington, DC

Inspiration: Anti-aircraft fire control in World War II

- The problem: aim, configure, and fire an anti-aircraft gun in order to hit fast-moving enemy airplanes.
- Given the position (i.e., range and bearing) to the target, predict its future position and compute the relevant fire control parameters (e.g., gun position, fuse timing).
- This task requires **fast** evaluation of mathematical functions, including: addition, subtraction, multiplication, division, and integration.

D. B. Parkinson's Solution (Bell Labs)

- Attach servomechanisms to specially shaped potentiometers.
- By varying reference voltage and servo angle, mathematical functions are computed electromechanically.



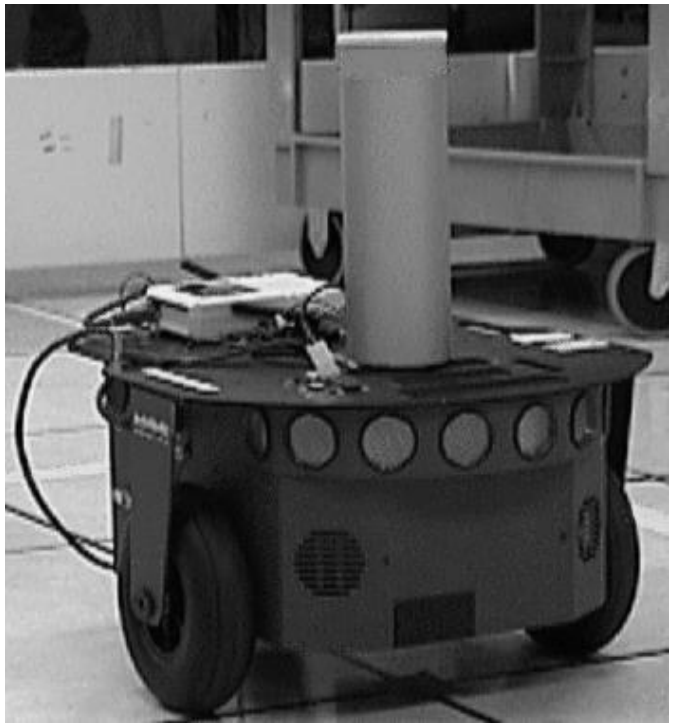
Concurrent mobile robot control

- **The task**: control the velocities of the two DC motors that drive the wheels in a differentially-steered robot such that the robot follows a desired trajectory.
- **Our approach**: Combine multiple control signals, but without conventional action selection; instead, simply send all commands to the motors.
- **Our hypothesis**: each motor will temporally average its inputs, computing the following function:

$$\Omega = \frac{\sum_{t=0}^{n} \omega_t}{n}$$

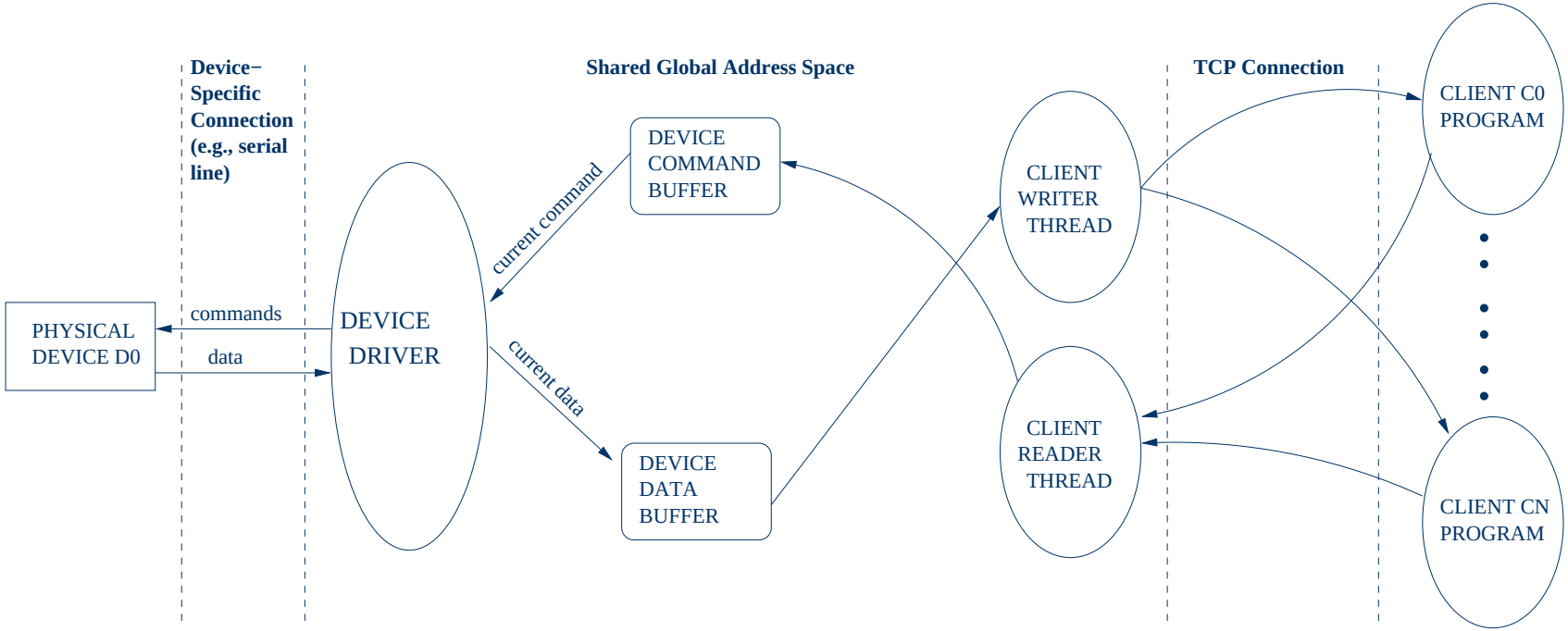
Validation: Hardware

- We used an ActivMedia Pioneer 2-DX, equipped with shaft-encoders that provide odometry.
- The robot houses an embedded computer, equipped with 802.11 wireless Ethernet, and running Linux.
- For ground-truth information, we employed an external metrology system that uses a laser range-finder accurate to 2cm.



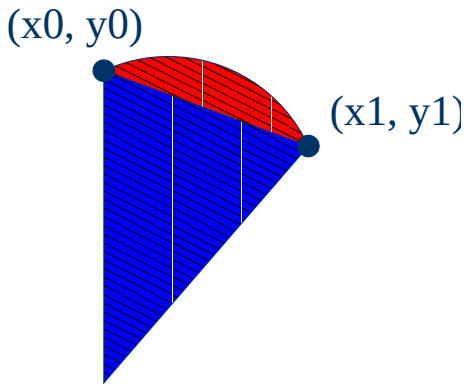
Validation: Software

- Interface to the robot's sensors and actuators is provided by the networked device server Player, which allows multiple clients to concurrently access hardware.



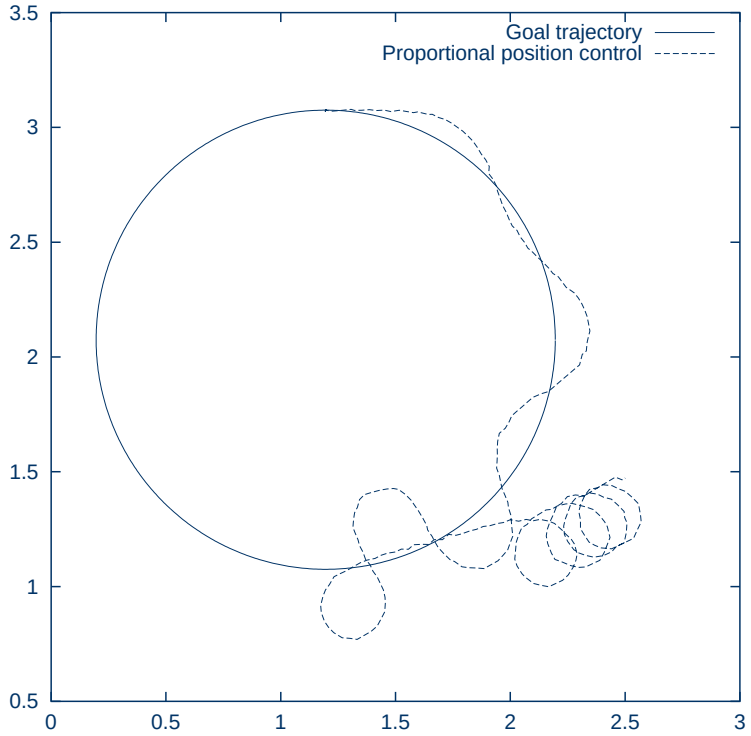
Validation: Task

- **The task**: follow a circular trajectory with radius 1 meter, based on odometry.
- **Control input**: (x, y, θ): the robot's current pose.
- **Control output**: (ω): desired angular velocity (forward velocity v is fixed at a small positive value).
- **Evaluation criteria**: accumulated triangular error:



Experiment I: Position-based P control

Rule: if the robot's current position (x, y) is outside the target circle, turn (proportionally to the error) toward the circle's center; otherwise turn away.

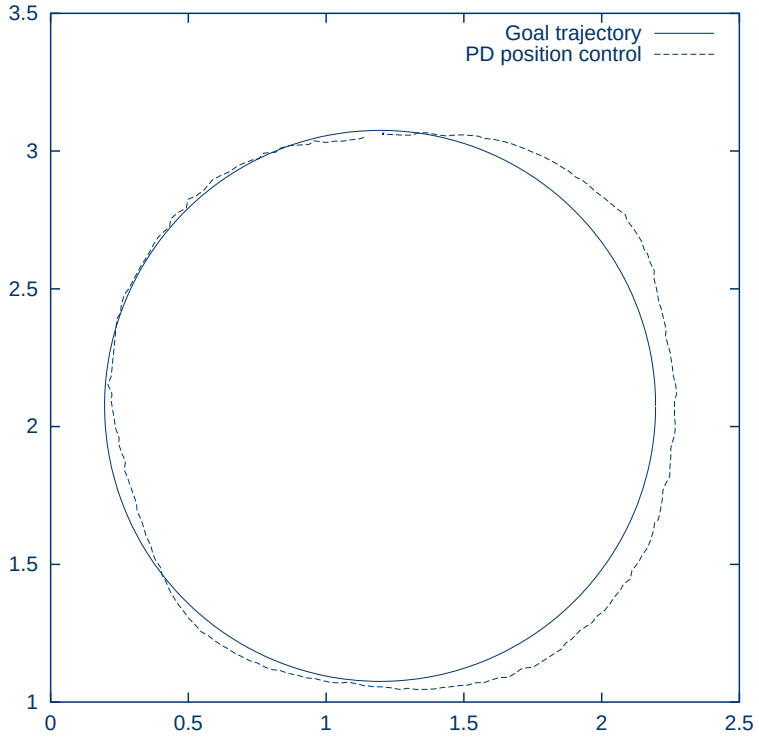


Example trial (units are meters)

Experiment II: Adding derivative control

- Our pure proportional controller suffers lag-induced overshoot.
- Control theory tells us to correct overshoot with a derivative term.
- We wrote a new controller that computes the derivative of the robot's positional error.
- We executed one each of the proportional and derivative controllers in parallel.

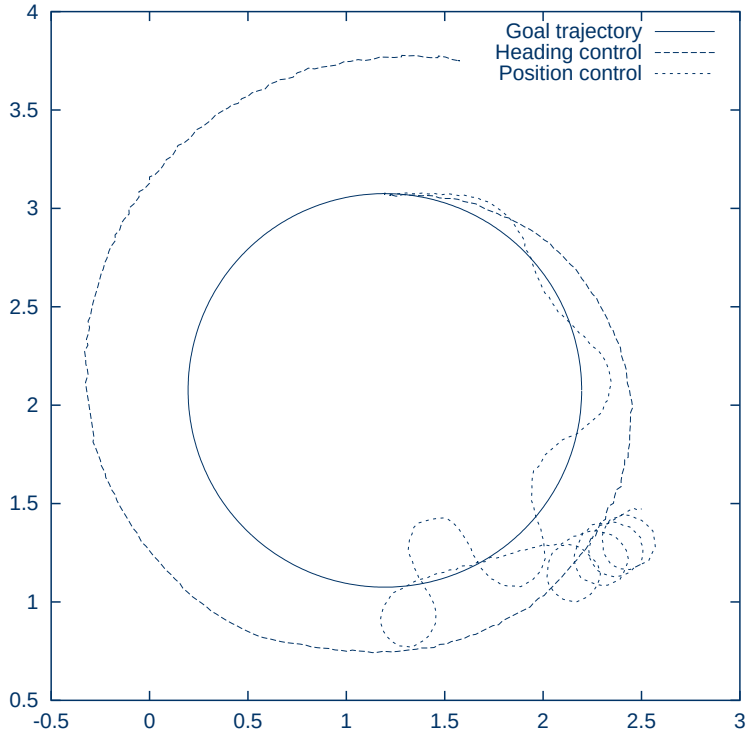
Experiment II (continued): Adding derivative control



Example trial of proportional and derivative controllers in parallel. The performance for this trial was 88.54%.

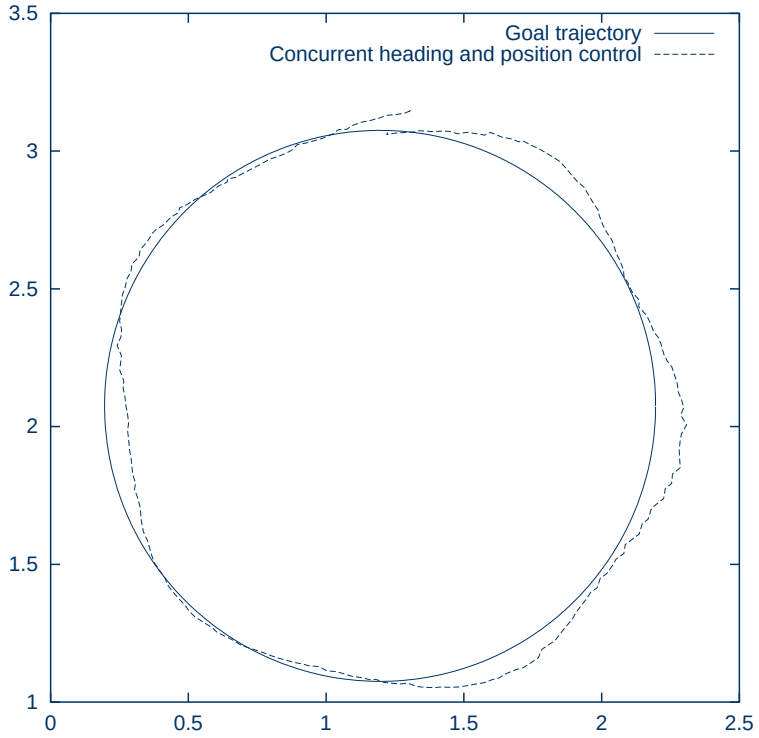
Experiment III: Heading-based P control

Rule: assume that the robot is on the boundary of the target circle, calculate the local tangent, and (proportionally) correct the robot's angular velocity.



Example trial of separate heading and position controllers.

Experiment III (continued): Heading & position P control

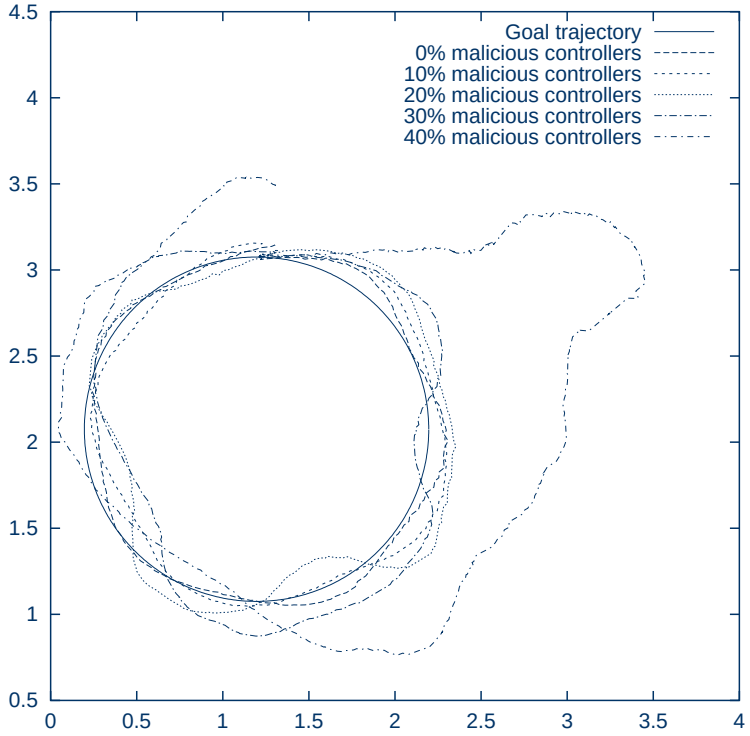


Example trial of heading and position controllers in parallel. The performance for this trial was 91.09%.

Experiment IV: Adding malicious controllers

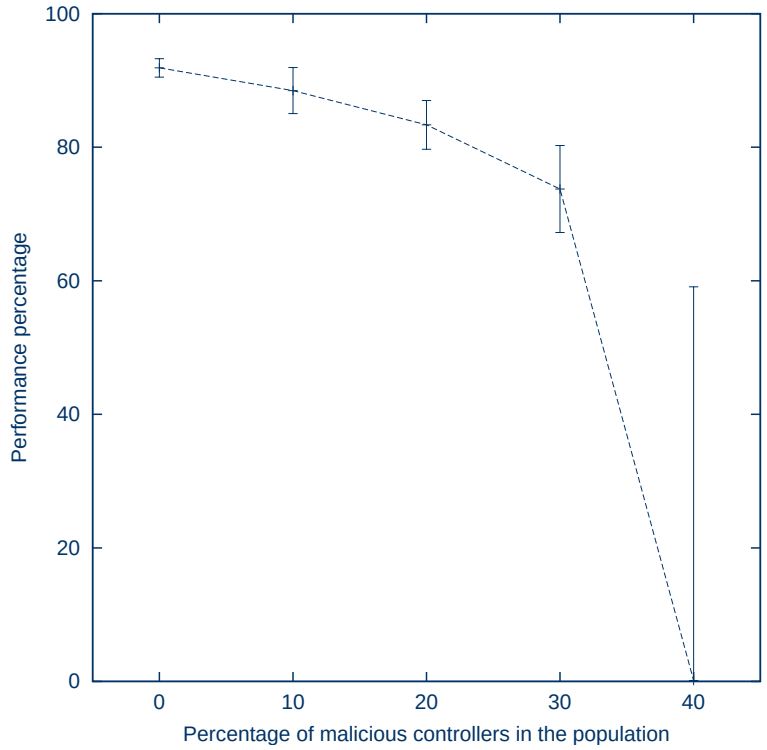
- Our goal was to test the fault-tolerance of our approach.
- We controlled the robot with a population of 50 proportional controllers, half position-based, half heading-based.
- We made a portion of the controllers “malicious”: they compute the same angular velocity as the others, but output the additive inverse.
- We ran 10 trials each of 5 different configurations, created by varying the proportion of malicious controllers from 0% to 40%.

Experiment IV (continued): Adding malicious controllers



Examples trials with various proportions of malicious controllers.

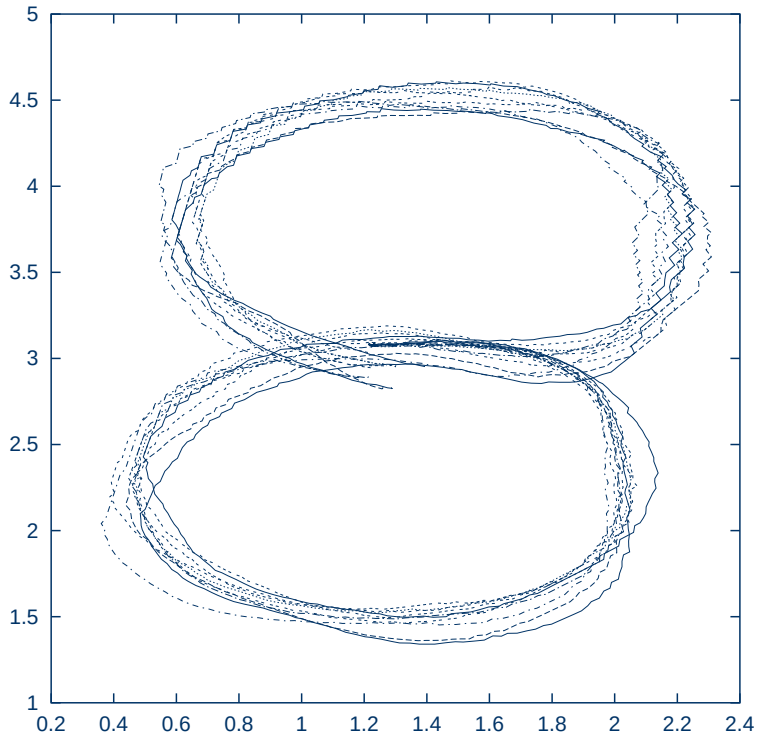
Experiment IV (continued): Adding malicious controllers



Performance on the circle following task as the proportion of malicious controllers increases.

Experiment V: Different trajectories

- The goal is to follow a figure-8 trajectory composed of two 0.75-meter circles.
- We ran one each of the proportional position and heading controllers (with slight modifications).



Overlaid tracks from 10 trials of the figure-8 task.

Conclusions

- As an alternative to conventional action selection, we propose *concurrent control*:
  - Multiple non-communicative independent controllers exert concurrent control of a robot, letting the motors mechanically "calculate" the macroscopic system behavior.
- With a physical robot, we validated our approach, showing it to be fault-tolerant, somewhat general, and inherently distributible and scalable.
- Of course, low-level concurrent control is not always viable, nor is it necessarily advisable; however, we find it interesting.

(possible) Future work

- Add controllers that operate on different sensor information (e.g., stereo vision, echolocation)
- Tune system performance by giving more or less weight to one kind of controller; this adjustment can be made by simply adding or removing controllers.

gerkey@robotics.usc.edu

http://robotics.usc.edu/~gerkey

http://playerstage.sourceforge.net