Robots & Simulation

Anders Lyhne Christensen

IRIDIA's weekly robotics meeting

January 25th 2006

Why use simulation?

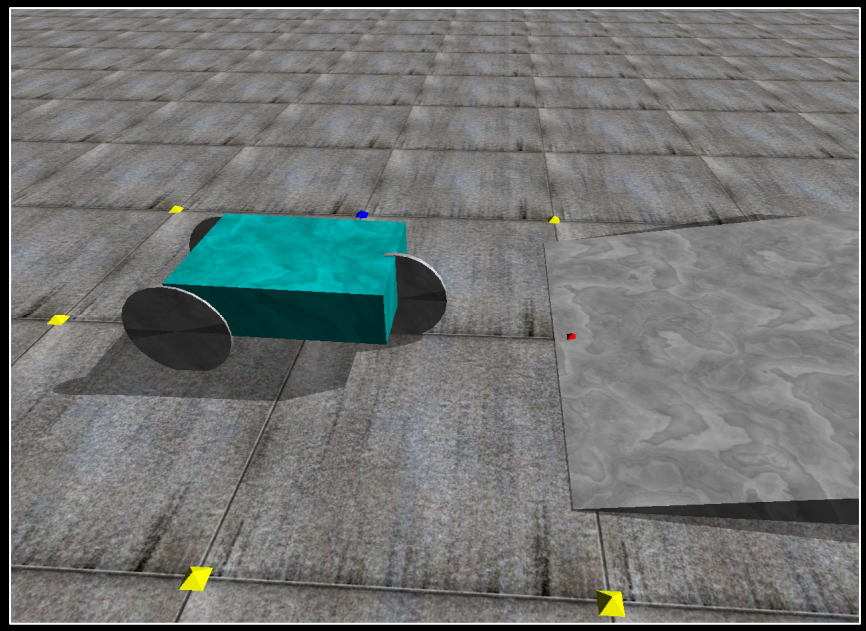
- A fast experimentation platform
- No damage to hardware
- Simulation can provide more robots, different sensors, new objects and environments that are not available in the real world
- Simulation and reality can be mixed (RAVE)
- For all but the simplest tasks, simulation is a necessity for evolutionary robotics

Issues

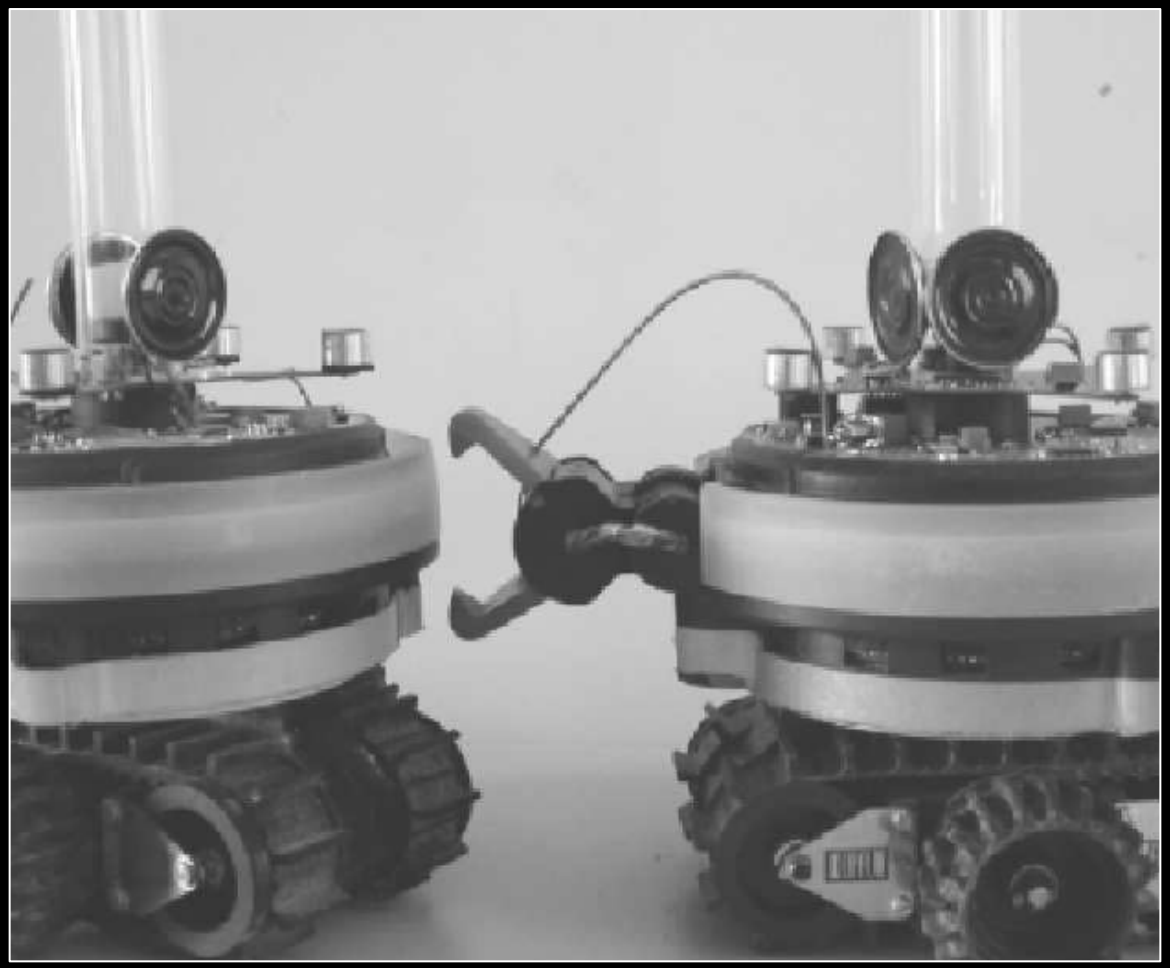
- Simulators take time to build
- No simulation is 100% accurate (the Reality Gap)
  - Controllers may not transfer
  - Solutions may not transfer
- Simulators often fail to take individual differences between robots into account

Complex simulators

- Dynamics
  - objects, collision spaces, joints and motors…
- Demonstration
- Dynamics is not going to save the world…



Try to simulate this (accurately)



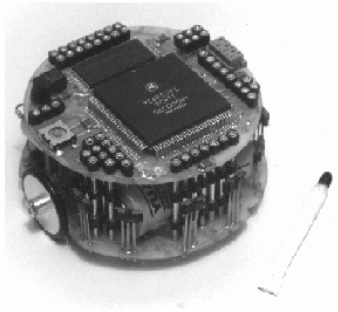
Complex simulators

- Complexity and dynamics does not mean *real*.
  - The world has to be modelled using primitives such as cylinders, spheres, boxes, etc.
  - Dynamics require discrete updates
  - The time and space complexity puts a limit to the accuratecy

Simple simulators

- Some simple sims. rely on kinematics
- Jacobi takes a different approach:
  - Model a *base set* of aspects accurately, and ensure that controllers are *base set exclusive*.
  - Noise during trails to ensure that controllers cannot rely on *implementation aspects*.
  - No simulator can model *anything* accurately: Controllers must be *base set robust*.
  - Noise between trails to ensure that controllers are *base set robust.*

Jacobi's minimal T-maze simulation



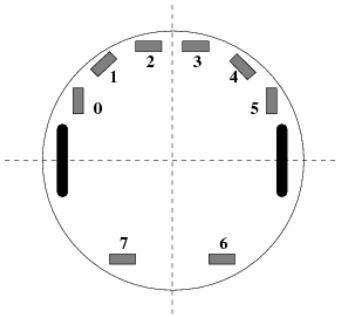


Figure 1: The Khepera robot.

300 lines of C code and a bunch of look-up tables to simulate the T-maze.

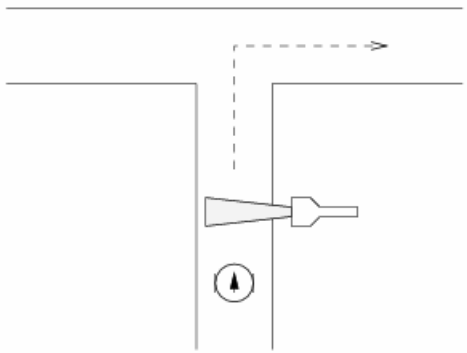


Figure 3: The task in the real world.

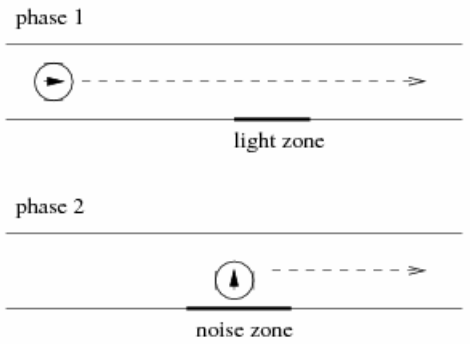
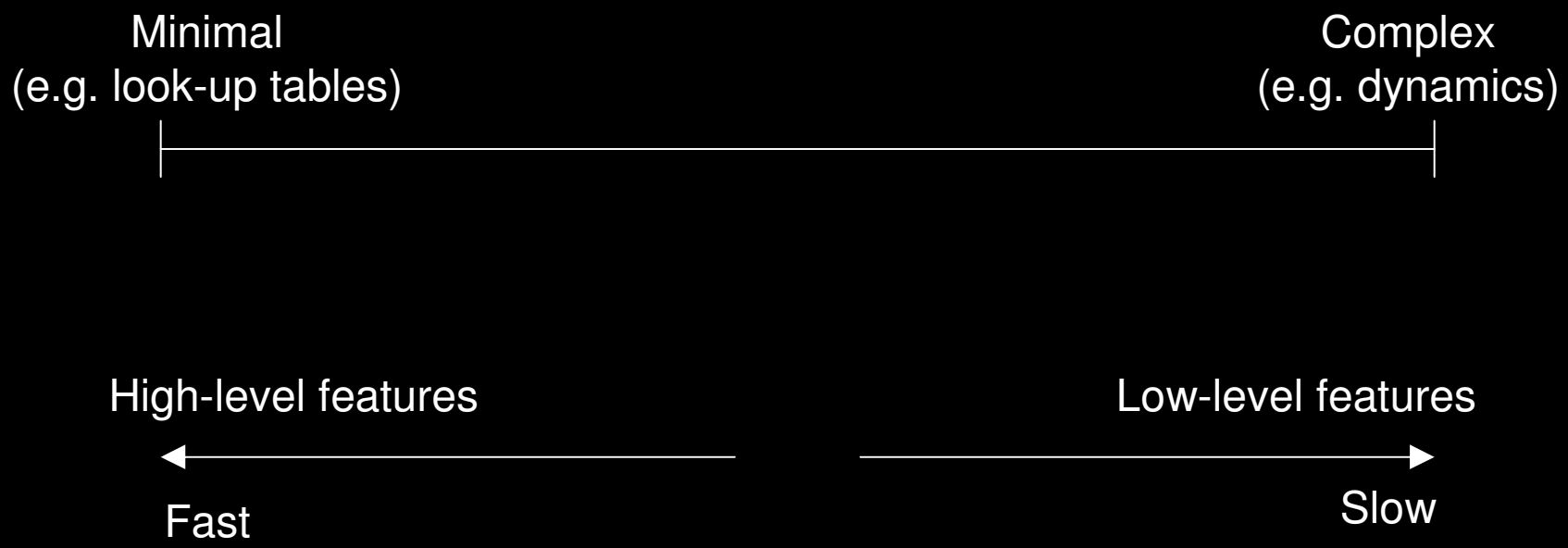


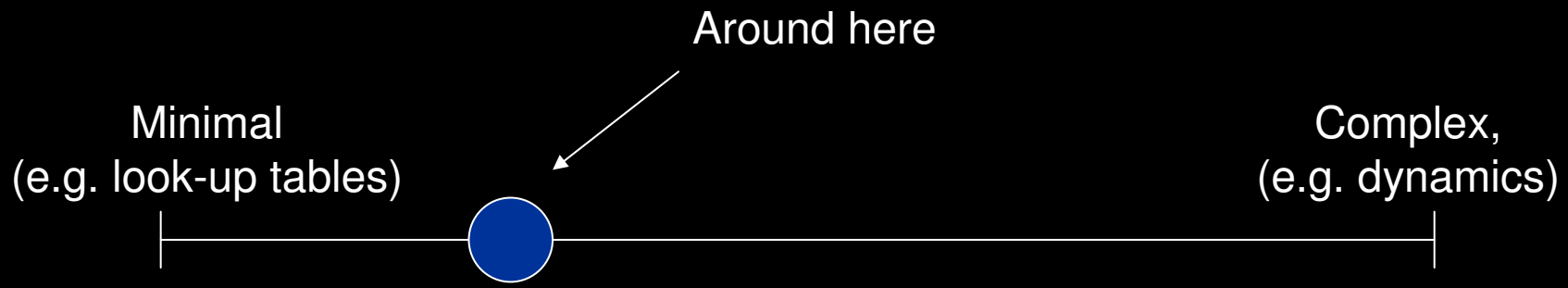
Figure 4: The task in simulation.

Strategies for crossing the Reality Gap



In almost all simulations noise is added

Where is TwoDee?



- Computations are done in 2D
  - But could be done in 3D
- High-level models and no *general* dynamics
- General architecture, e.g. environment, sensors, actuators and so on can easily be changed

Consequences

Pros:

- Speed
- Models can be changed easily
- Special cases can be treated individually
- Flexible within limitations

Cons:

- Lack of generality
- Each new, virtual object requires special treatment
- Inherit limitations built in
- Users need to understand the high-level models

Simulators also evolve

- Research and evolutionary robotics require *flexibility:*
  - The software should be constructed with this in mind.
  - Making components explicit through interfaces
    - Not just the Common Interface!
- External flexibility: *User interface*
  - Demonstration
- Internal flexibility: *Classes and interfaces*
  - The simulator should *grow* and *not change*
  - Demonstration

Coding an Experiment

- Experiments can be programmed too:
  - Get a simulator
    - Get and arena
    - Get swarm-bots
      - Get s-bots
        - » Get sensors
        - » Get actuators
        - » Get controller
    - Get a fitness function

That’s all folks

- Thanks for listening -