Planning for Robotics

Christian Dornhege

dornhege@informatik.uni-freiburg.de

6. Februar 2012

Motivation

High-Level Robot Control

- Decide which high-level action to execute and how (which parameters)
- Examples: drive to location3, move arms into drive position, grasp an object, move head towards object

Why Planning?

- Other control methods like scripting cumbersome, involve manual design of every part of the plan
- Recent advances in robotics:
- Until a couple of years ago robots could execute only one task (e.g. drive in unknown environment)
- Now: multiple capabilities in one system (e.g. mobile manipulators) $\rightarrow$ need high-level module to decide what to do to achieve goal

Contact: Christian Dornhege (*dornhege@informatik.uni-freiburg.de*)

High-Level Robot Control

Challenges

- Representation of real world as logical formulation
- Plans should be executable
- Dealing with execution failures
- Performance: max. 30 secs vs 30 mins

Techniques

- Continual planning
- Integration of task and motion planning

Contact: Christian Dornhege (*dornhege@informatik.uni-freiburg.de*)

Continual Planning

Integrating a planner into a real-world system

- Usually: Domain + Problem $\rightarrow_{\text{Planner}}$ Plan
- How do we get the problem: Matching (huge amounts of) continuous data to symbolic data
- How to execute a plan: Map textual actions to real actions
- Need to be able to react to (changing) real-world data, execution failures: Planning-Execution-Monitoring loop
- Other useful features: Numerical fluents, „syntactic sugar“, feature-rich planning language

Contact: Christian Dornhege (*dornhege@informatik.uni-freiburg.de*)

Integration of task and motion planning

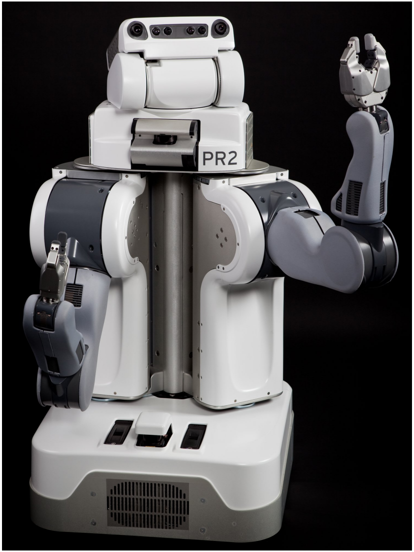
Integrating the real-world into planning systems

- Symbolic plans should be executable in the real world
- Symbolic abstractions easily relax problem beyond that point
- Semantic attachments: a fluent's semantics in the planner might be determined by an external procedure
- Ex.: predicate `(candrive Loc0 Loc27)` is not present in the state, but a planner module calls a motion planner to determine this fact
- Generic domain-independent interface, currently supporting: condition checkers (predicates), action costs (durations), numerical effects

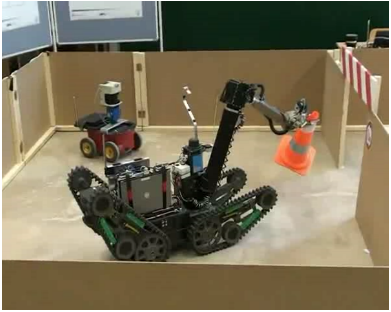
Contact: Christian Dornhege (*dornhege@informatik.uni-freiburg.de*)

Applications

Mobile manipulation



Multi-robot coordination



Contact: Christian Dornhege (*dornhege@informatik.uni-freiburg.de*)