Self-Reconfiguring Modular Robots

Robert Fitch
ARC Centre of Excellence for Autonomous Systems
Australian Centre for Field Robotics

Vision
Research Problems

Examples: Parallel, Decentralized, Compliant
Abstract Module: the Sliding Cube

- Lattice-based
- Cube
- Motion primitives:
  - Connect with adjacent modules
  - Point-to-point communication

Reconfiguration Planning

How to move in parallel?

Path planning: where to move?

What is goal shape and position?

How to move compliantly?

Connectivity: Can a module move without disconnecting global structure?
Reconfiguration Planning

- How to move in parallel? **Soft Locking**
- Path planning: where to move? **Real-Time Dynamic Programming**
- What is goal shape and position? **Bounding Box**
- How to move compliantly? **Sense Obstacles**
- Connectivity: Can a module move without disconnecting global structure? **Parallel Connectivity Check**

The Connectivity Problem: Find a Set of Mobile Modules

- Assume SlidingCube module abstraction
- Single module case:
The Connectivity Problem: Find a Set of Mobile Modules

- Assume SlidingCube module abstraction
- Single module case:
  - connecting cycle

- Multiple module case:
  - connecting cycle
The Connectivity Problem: Find a Set of Mobile Modules

- Assume SlidingCube module abstraction
- Single module case:
  ![SlidingCube module](image)
- Multiple module case:
  ![Connecting cycle](image)

Dense configurations have short connecting cycles!

Local, Parallel Algorithm for Mobility Check

- Algorithm for one module
  1. **DfsSend** search message to all neighbors (preLock neighbors)
  2. Maintain disjoint set to track connectivity
  3. **When** a
     - If neighbors connected, lock connecting cycle
  4. Release locks after done moving
  5. **If** failure, release locks
- Coordination and locking
  - Arbitrary module priority
  - If locked, accept all lock requests
Planning: MDP Formulation (GridWorld Example)

Standard 2D gridworld: cell has a computer in it:

3D gridworld where every cell has a computer in it:

MDP Formulation

- States
  - Set of all free module faces in current configuration
- Actions
  - Module motions (6 faces x 4 moves = 24)
  - Available subset determined by local neighborhood
- Transition model
  - SlidingCube motions (sliding, convex)
- Reward function
  - -1 (not in goal), or -k*height (in goal)
- Will use greedy policy
**Distributed DP Updates**

- **Value functions**
  
  \[ V(s) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \right] \]

  \[ Q(s, a) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a \right] \]

- **Update rule**
  
  \[ Q(s_t, a_t) \leftarrow r_t + \gamma \max_{a' \in A} Q(s_{t+1}, a') \]

- **Implementation**
  - **One-step look ahead**
  - Triggered after a move

**Complete algorithm**

- Handle connectivity check messages
- Handle DP update messages
- Check mobile
  - Lock connecting cycle
  - Move
Example: Finger Obstacles

Example: Concave Obstacle
From Abstract Cubes to Native Kinematics

SuperBot
New Challenges

Connector mechanism
Communication system
Integrated platform

Planning Algorithms

Static stability
Assembly order
Configuration determination
Planning for a deformable box

Custom UWB radar
Localisation
Decentralized ranging and imaging

Hardware
Sensing and Perception