Today

- Robot control architectures
- Deliberative control: motion planning
- Applications: industrial assembly, exploration, drug design
- Reading: chapter 6

Controlling in the large

- We have seen feedback control
- How do we put together multiple feedback controllers?
  - in what order?
  - with what priority?
- How do we generate reliable and correct robot behavior?
A control architecture provides a set of principles for organizing a robot (software) control system. Like in computer architecture, it specifies building blocks. It provides:

- Structure
- Constraints

Control Architecture Types

- Deliberative control
- Reactive control
- Hybrid control
- Behavior-based control

Deliberative Architecture

- Maps, lots of state
- Look-ahead

Reactive Architecture

- No maps, no state
- No look ahead
Behavior-based Architecture

- Some state
- Look ahead only while acting
- Reactive + state

Hybrid architectures

- State
- Look ahead but react
- Combines long and short time scales

Criteria For Selection

<table>
<thead>
<tr>
<th></th>
<th>deliberative</th>
<th>reactive</th>
<th>behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task and environment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run-time constraints</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Correctness/Completeness</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Motion Planning

How do we command the robot to move from A to B despite complications?

Complications: error in maps, sensing, control, unexpected obstacles, etc.
Spatial Planning: Shakey and Stanford Cart (1969)

TV camera
Triangulating range finger
Bump sensors
DEC PDP-10, PDP-15 via radio
(192K 36-bit)

15 mins processing for video planning per meter of travel

Deliberative Architecture

Sense

Map

Plan

Act

Motion Planning

Motion Planner

Trajectory Generator

Controller

Localization

Calibration

Robot motors, sensors + External world

Signals to joint controllers/drivers

Trajectory generation from waypoints

Different interpolations
Depending on robot constraints
Motion Planning

- Known Environments (Model)
- Unknown Environments (No Model)

OFFLINE ALGORITHMS
ONLINE ALGORITHMS

Example: how do we find a bridge in the fog?

Online Motion Planning

Always finds a path (if it exists)

Off-line Motion Planning

Visibility Graphs

Vertices: Start, Goal, obstacle vertices
Edges: all combinations \((v_i, v_j)\) that do not intersect any obstacle
Search Path: Dijkstra’s Algorithm

```plaintext
1  function Dijkstra(G, w, s)
2     for each vertex v in V[G]                        // Initializations
3           d[v] := infinity
4           previous[v] := undefined
5     d[s] := 0
6     S := empty set
7     Q := set of all vertices
8     while Q is not an empty set                 // The algorithm itself
9           u := Extract_Min(Q)
10           S := S union {u}
11           for each edge (u,v) outgoing from u
12                  if d[v] > d[u] + w(u,v)             // Relax (u,v)
13                        d[v] := d[u] + w(u,v)
14                        previous[v] := u
```

Visibility Graphs Summary

For what robot shapes does this work?

What if the robot is not a point?

Configuration space

C-space: (x, y, θ) DOF
Configuration space

Configuration space = the set of all feasible configurations
3-D space for planar, mobile robots

C-space: \((x, y, \theta)\) DOF

Transforming to C-Space

Transform to equivalent simpler problem
Higher dimension

Simpler problem
Point
Spatial interaction
Shape

Robot Configuration Space

Transforming to C-Space

start
goal
Spatial interaction
Shape

start
goal
Allowable Robot positions (no rotations)

Allowable Robot positions (no rotations)

Allowable Robot positions (for some robot rotation)

C-space Algorithm

Step 1: Reflect Robot
C-space Algorithm

Step 2: Vert (Robot) + Vert (Obstacle)

C-space Algorithm

Step 3: ConvexHull (Vert (Robot) + Vert (Obstacle))

Convex Hull Algorithm

Convex Hull Algorithm

D  E  C  B

A
Convex Hull Algorithm

Algorithm Summary
- Compute c-space for each obstacle
- Compute v-graph
- Find path from start to goal

V-graph complete; gives optimal shortest path in 2d
What about 3d? What else can we optimize?

Configuration Space with Rotations

Piano Movers’ Problem

Donald et al 93