Introduction to Motion Planning

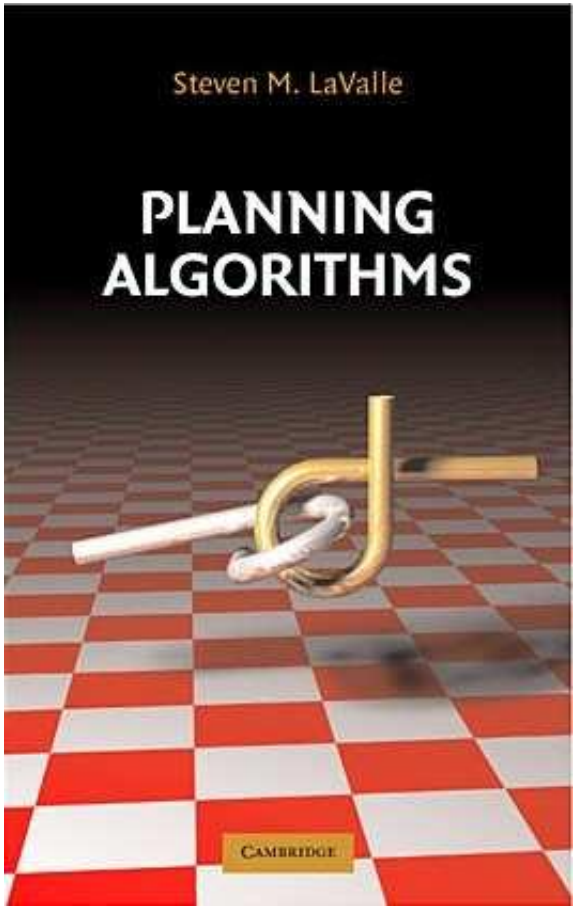
Ioannis Havoutis

Italian Institute of Technology

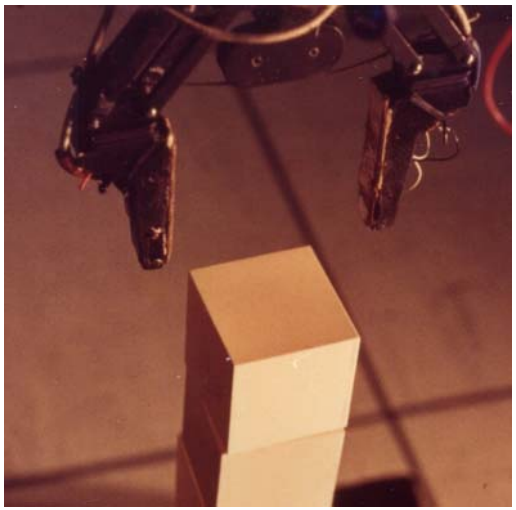
November 3, 2011

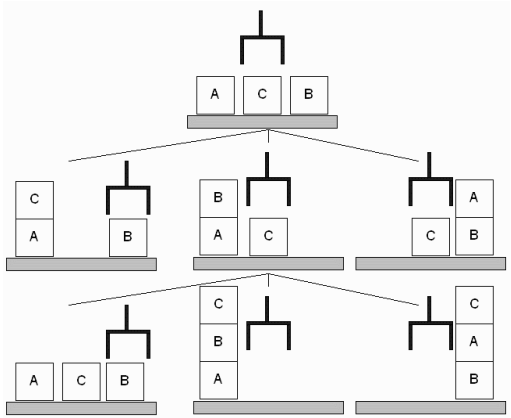
Overview

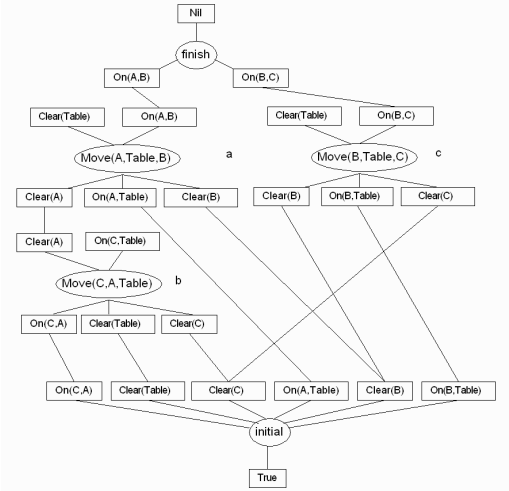
1. Motivation
   - Planning history
2. General concepts
3. Discrete planning
   - ...searching really
4. Modelling
   - configuration space
   - cell decompositions
5. Continuous planning
   - feedback motion planning
   - sampling based motion planning
   - extensions
6. Planning under differential constraints



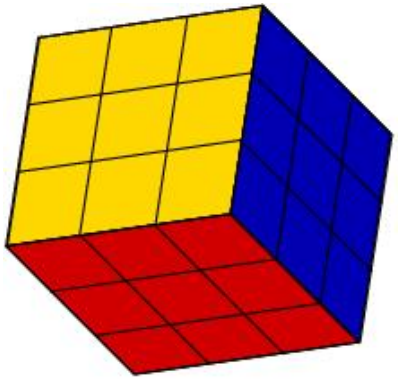
- STRIPS Action Planning (60s!)







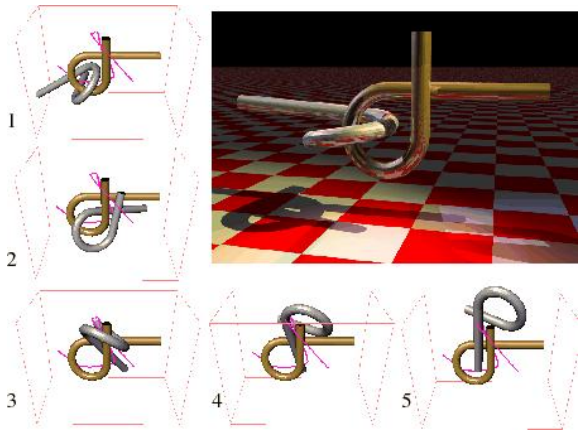
- Puzzles



(a)

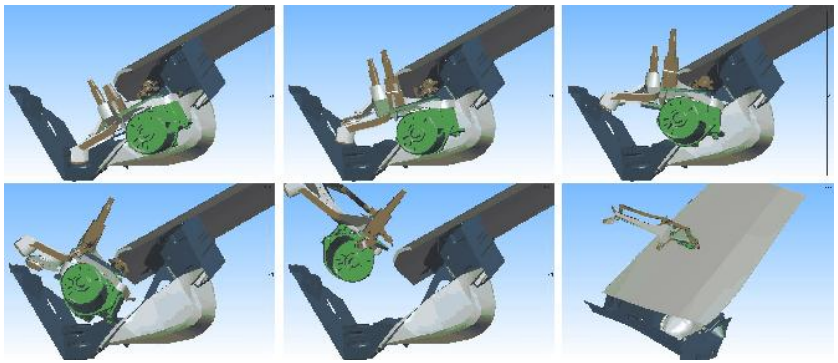
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

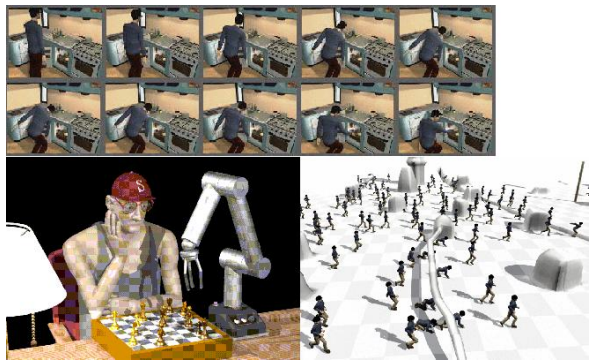
(b)



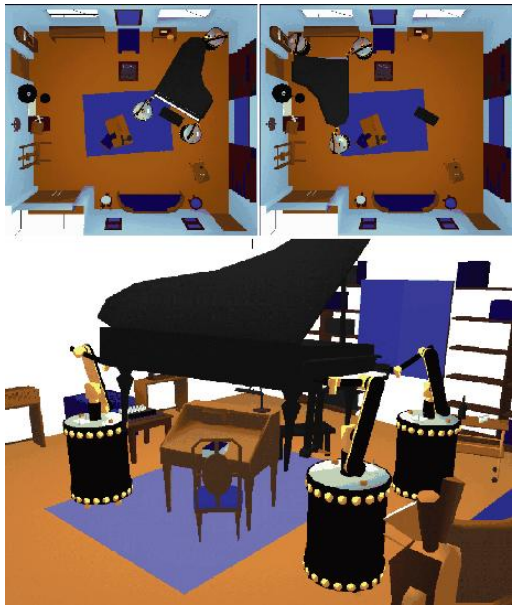
Motivation
More practical applications

- Industrial assemblies
- General motion planning

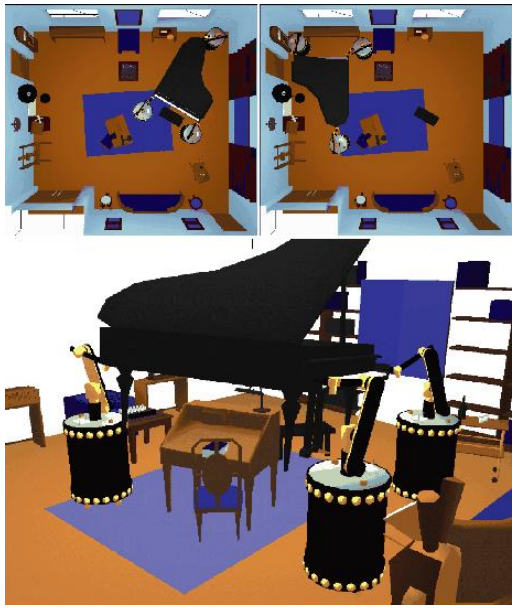




- The piano mover's problem



- The piano mover's problem



Basic Path Planning

Compute a continuous sequence of collision-free robot configurations connecting the initial and goal configurations.

Why bother?

Original 'piano mover's problem', now includes uncertainties, multiple bodies and dynamics.

Why bother?

Original 'piano mover's problem', now includes uncertainties, multiple bodies and dynamics.

1. Robotics
   - generate useful motions by processing complicated geometric models

Why bother?

Original 'piano mover's problem', now includes uncertainties, multiple bodies and dynamics.

1. Robotics
   - generate useful motions by processing complicated geometric models
2. Artificial Intelligence
   - decision-theoretic models to compute appropriate actions. (not covered in this lecture)

Why bother?

Original 'piano mover's problem', now includes uncertainties, multiple bodies and dynamics.

1. Robotics
   - generate useful motions by processing complicated geometric models
2. Artificial Intelligence
   - decision-theoretic models to compute appropriate actions. (not covered in this lecture)
3. Control theory
   - compute feasible trajectories for systems, with some additional coverage of feedback and optimality.

Why bother?

Original 'piano mover's problem', now includes uncertainties, multiple bodies and dynamics.

1. Robotics
   - generate useful motions by processing complicated geometric models
2. Artificial Intelligence
   - decision-theoretic models to compute appropriate actions. (not covered in this lecture)
3. Control theory
   - compute feasible trajectories for systems, with some additional coverage of feedback and optimality.

The user of the plan

Robot, decision maker, agent, software agent, controller, policy, control law, player...

Basic Ingredients

- State
  - discrete, continuous (partially known, dynamic)
- Time
  - explicitly modelled, implicit
- Actions
  - (a.k.a. inputs, controls)
- Initial and Goal state(s)
  - single or set
- Criteria
  - feasibility, optimality, ...
- Plan
  - a specific strategy or behaviour that the agent follows

Basic representation

- A world example

- Find a path between two locations in an unknown, partially known, or known environment
- Performance
  - Completeness
  - Optimality (operating cost)
  - Complexity

Strong Assumption

Some form of discretization on your state space.

Breadth first
Depth first

- Uninformed search
  - No information from environment
  - Blind search
- Informed search
  - Use evaluation function
  - More efficient (can tune)
  - Requires a heuristic

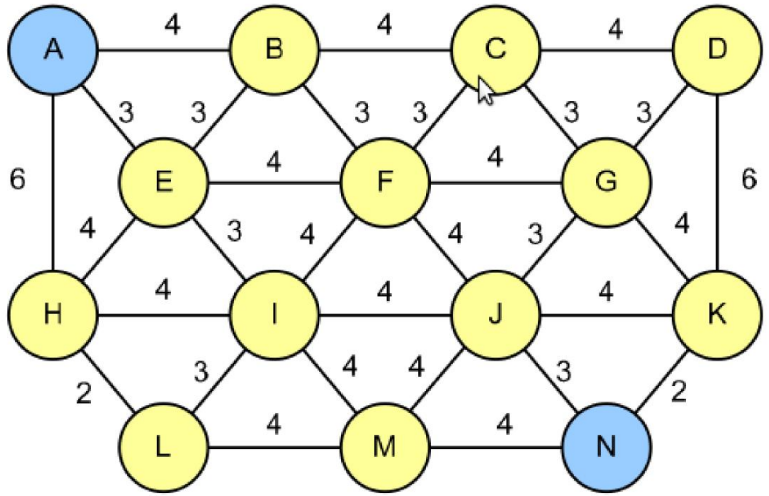
- Uninformed search
  - No information from environment
  - Blind search
- Informed search
  - Use evaluation function
  - More efficient (can tune)
  - Requires a heuristic

Breadth first
Depth first
Dijkstras algorithm (introduced notion of cost)
A*,D*,etc. (incorporation of heuristic estimate of the cost)

- Uninformed search
  - No information from environment
  - Blind search
- Informed search
  - Use evaluation function
  - More efficient (can tune)
  - Requires a heuristic

Breadth first
Depth first
Dijkstras algorithm (introduced notion of cost)
A*,D*,etc. (incorporation of heuristic estimate of the cost)
Best first
Iterative deepening
Backwards search
Incremental/Anytime versions of all of the above..
Bidirectional search

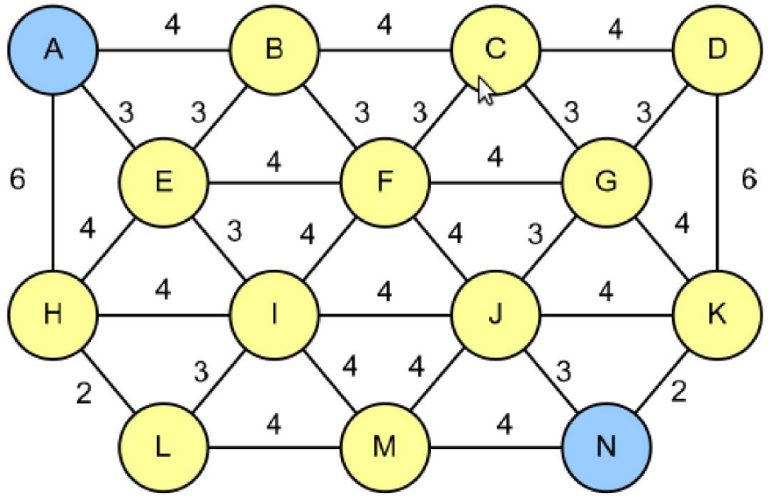
- Evaluation function: $f(n) = g(n) + h(n)$
- Operating cost: $g(n)$
  -Actual operating cost having been already traversed
- Heuristic function: $h(n)$
  - Info used to find the most promising node to traverse
  - Admissible: never overestimate the actual path cost



Heuristics

A = 14	H = 8
B = 10	I = 5
C = 8	J = 2
D = 6	K = 2
E = 8	L = 6
F = 7	M = 2
G = 6	N = 0

- Evaluation function: $f(n) = g(n) + h(n)$
- Operating cost: $g(n)$
  -Actual operating cost having been already traversed
- Heuristic function: $h(n)$
  - Info used to find the most promising node to traverse
  - Admissible: never overestimate the actual path cost



Heuristics	
A = 14	H = 8
B = 10	I = 5
C = 8	J = 2
D = 6	K = 2
E = 8	L = 6
F = 7	M = 2
G = 6	N = 0

-Effect of heuristic Dijkstra,A*,wA*.

Basic representation

- A world example
- ..but what if your robot is not a point?

Basic representation

- A world example
- ..but what if your robot is not a point?
  - Blow up the world (i.e. take the Minkowski sum of the convex polygons)
  - Issues with rotations and translations
- 2-link robot example

Cell decompositions to get to adjacency graphs
- Two cells are adjacent if they share a common bound
- Exact and approximate
- Techniques

  - Trapezoidal Decomposition
  - Morse Cell Decomposition
    - Boustrophedon decomposition
    - Morse decomposition definition
    - Sensor-based coverage
    - Examples of Morse decomposition
  - Visibility-based Decomposition

Sampling based motion planning

The main idea is to avoid the explicit construction of C_{obs} and instead probe the C-space with a sampling scheme

Sampling based motion planning

The main idea is to avoid the explicit construction of C_{obs} and instead probe the C-space with a sampling scheme

- probabilistic completeness

Sampling based motion planning

The main idea is to avoid the explicit construction of C_{obs} and instead probe the C-space with a sampling scheme

- probabilistic completeness
- Collision detection

Sampling based motion planning

The main idea is to avoid the explicit construction of C_{obs} and instead probe the C-space with a sampling scheme

- probabilistic completeness
- Collision detection
- Unidirectional/Bidirectional/Multi-directional methods

Sampling based motion planning

The main idea is to avoid the explicit construction of C_{obs} and instead probe the C-space with a sampling scheme

- probabilistic completeness
- Collision detection
- Unidirectional/Bidirectional/Multi-directional methods
- Nearest-neighbour computation

Sampling based motion planning

The main idea is to avoid the explicit construction of C_{obs} and instead probe the C-space with a sampling scheme

- probabilistic completeness
- Collision detection
- Unidirectional/Bidirectional/Multi-directional methods
- Nearest-neighbour computation

-Vanilla RRT example

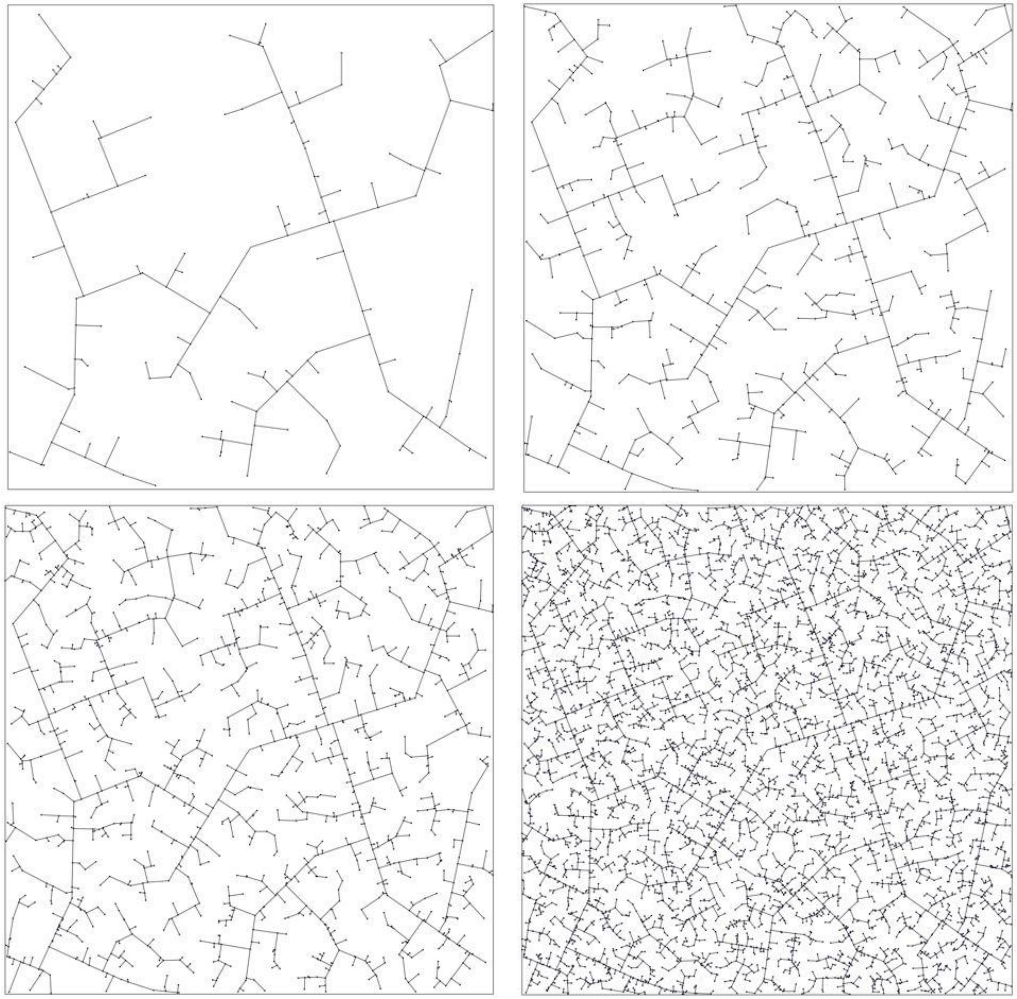
Sampling based motion planning

The main idea is to avoid the explicit construction of C_{obs} and instead probe the C-space with a sampling scheme

- probabilistic completeness
- Collision detection
- Unidirectional/Bidirectional/Multi-directional methods
- Nearest-neighbour computation

-Vanilla RRT example -Problems with RRTs

RRT example



Feedback motion planning

- Grids

Feedback motion planning

- Grids
- Vector fields

Feedback motion planning

- Grids
- Vector fields
- Barycentric interpolations

Feedback motion planning

- Grids
- Vector fields
- Barycentric interpolations
- Funnel covering

Feedback motion planning under differential constraints

Differences: 1) the introduction of time, 2) the state or phase space, and 3) the state transition equation

- Motion primitives
  e.g. manoeuvre automatons, local control laws with well defined in/out states and connectivity graph
- Reachability sets
  e.g in phase space and lattices
- Regions of inevitable collision
  e.g. it is in phase space and state dependent, depends on velocity for instance
- System simulations
  e.g. sampling control space

The beyond

System theoretic and analytical techniques

- Dynamic programming
- LQR, LQR-trees
- iLQG
- Pi^2