Artificial Intelligence

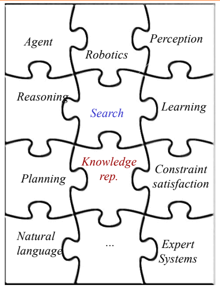
Module 5: Planning

Dr. Chandra Prakash
Assistant Professor
Department of Computer Science and Engineering

(Slides adapted from StuartJ. Russell, B Ravindran, Mausam, Dan Klein and Pieter Abbeel, Partha P Chakrabarti, Saikishor Jangiti

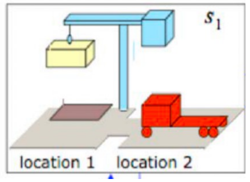
Main Areas of AI

- Agent architectures
- Knowledge representation (including formal logic)
- Search, especially heuristic search (puzzles, games)
- **Planning**
- Reasoning under uncertainty, including probabilistic reasoning
- Learning
- Robotics and perception
- Natural language processing

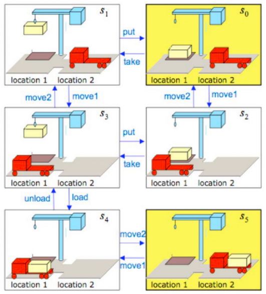


Planning

- According to Wikipedia:
  - "Planning is the process of thinking about an organizing the activities required to achieve a desired goal."

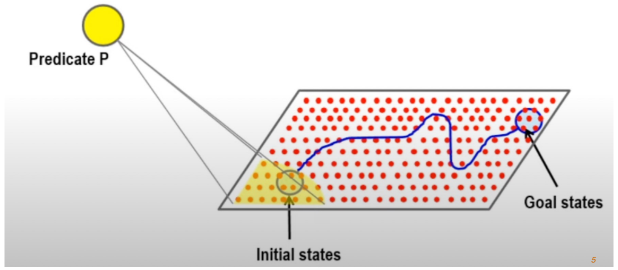


AI Planning

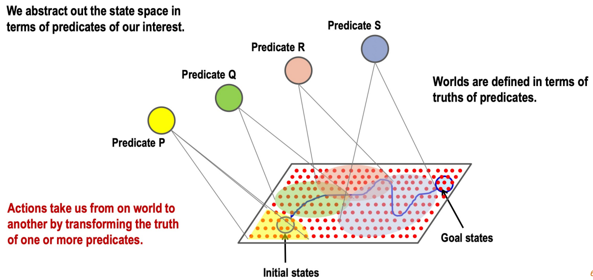


- Elements of a Planning Problem
  - A set of states (worlds) described in terms of predicates
  - A set of actions which transforms some parts of one world to take us to another world
  - An initial world
  - A goal in terms of the predicates that must hold in the final world
- Planning is widely used in robotics and automated control
- Modern AI explores techniques that combine planning with machine learning
  - Autonomous driving is one of many areas where such combinations are highly relevant

From State Spaces to Predicate Worlds

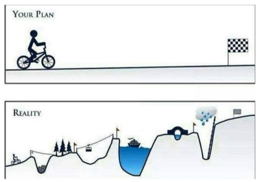


From State Spaces to Predicate Worlds



Planning in AI

- The intelligent way to do things
- Given a logical description of
  - The world states
  - Possible actions
  - Initial state
  - goal conditions
- Planning :
  - Find a sequence of actions (a plan of actions) that transforms the initial state to a state where the goal conditions are satisfied.



Real-World Problems: Application

- **Agents :**
  - Search-based problem-solving agent
  - Logical planning agent
  - Complex/large scale problems?
- Activity Planning for the Mars Exploration Rovers
- https://www.youtube.com/watch?v=WNrTtt vdIMc

Planning problem

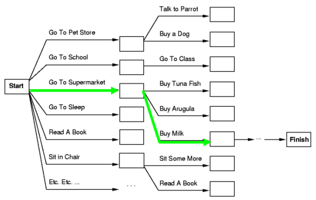
- Find a sequence of actions that achieves a given goal when executed from a given initial world state
- For the discussion, we consider classical planning environments
  - fully observable, deterministic, finite, static and discrete (in time, action, objects and effects)
- Given
  - a set of operator descriptions (defining the possible primitive actions by the agent),
  - an initial state description, and
  - a goal state description or predicate
- Compute a plan, which is
  - a sequence of operator instances, such that executing them in the initial state will change the world to a state satisfying the goal-state description.
- Goals are usually specified as a conjunction of goals to be achieved

Planning vs Problem Solving

- Planning agent is very similar to problem solving agent
  - Constructs plans to achieve goals, then executes them
- Planning agent is different from problem solving agent in:
  - Representation of goals, states, actions
    - States/Situations – logical description of the world that allows the agent to reason about
    - Goal Conditions – logical sentences vs goal test
    - Operators/Actions – transformations on logical sentences that allows the agent to reason about the effects of actions
  - Use of explicit, logical representations
  - Way it searches for solutions
    - Facilitates divide and conquer strategy – solve subgoals independently
- Planning is more powerful because of the representations and methods used
  - States, goals, and actions are decomposed into sets of sentences (usually in first-order logic)
  - Search often proceeds through plan space rather than state space (though there are also state-space planners)
  - Subgoals can be planned independently, reducing the complexity of the planning problem

Problems with Standard Search

- Overwhelmed by irrelevant actions
- Finding a good heuristic function is difficult
- Cannot take advantage of problem decomposition
- Consider the task: get milk, bananas, and a cordless drill
  - Standard search algorithms seem to fail miserably
  - Why? Huge branching factor & heuristics
- Perfectly decomposable problems are delicious but rare
  - Partial-order planner is based on the assumption that most real-world problems are nearly decomposable
  - Be careful, working on some subgoal may undo another subgoal

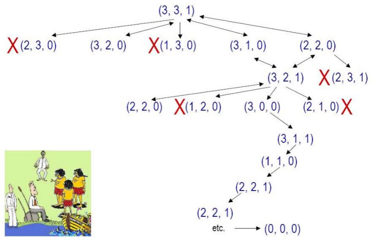


Typical assumptions

- Atomic time:
  - Each action is indivisible
- No concurrent actions are allowed
  - actions do not need to be ordered with respect to each other in the plan
- Deterministic actions:
  - The result of actions are completely determined—there is no uncertainty in their effects
- Agent is the sole cause of change in the world
- Agent is omniscient:
  - Has complete knowledge of the state of the world
- Closed World Assumption:
  - everything known to be true in the world is included in the state description.
  - Anything not listed is false.

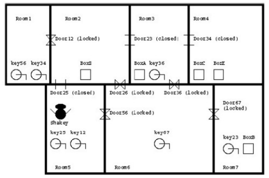
Famous Problem Solver Task

- **Missionaries and Cannibals**



Planning-Based Approach to Robot Control

- Job of planner:
  - generate a goal to achieve, and then construct a plan to achieve it from the current state
- Must define representations of:
  - Actions:
    - generate successor state descriptions by defining preconditions and effects
  - States:
    - data structure describing current situation
  - Goals:
    - what is to be achieved
  - Plans:
    - solution is a sequence of actions

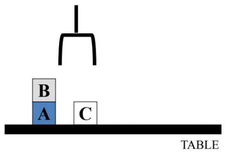


Planning systems

- Planning systems are problem-solving algorithms that operate on explicit propositional or relational representations of states and actions
- These represenatnions make possible the derivation of effective heuristics and development of powerful and flexible algorithms
- Planning Domain Defination Language (PDDL)
  - describes the initial and goal states as conjuctions of literals and actions in term sof their preconditons and effects .

Goal stack planning: Blocks world

- The blocks world is a micro-world that consists of a table, a set of blocks and a robot hand.
- Some domain constraints:
  - Only one block can be on another block
  - Any number of blocks can be on the table
  - The hand can only hold one block
- Typical representation:
  - ontable(A)
  - ontable(C)
  - on(B,A)
  - handempty
  - clear(B)
  - clear(C)



General Problem Solver

- The General Problem Solver (GPS) system was an early planner (Newell, Shaw, and Simon)
- GPS generated actions that reduced the difference between some state and a goal state
- GPS used Means-Ends Analysis
  - Compare what is given or known with what is desired and select a reasonable thing to do next
  - Use a table of differences to identify procedures to reduce types of differences
- GPS was a state space planner: it operated in the domain of state space problems specified by an initial state, some goal states, and a set of operations

Situation calculus planning

- Intuition: Represent the planning problem using first-order logic
  - Situation calculus lets us reason about changes in the world
  - Use theorem proving to "prove" that a particular sequence of actions, when applied to the situation characterizing the world state, will lead to a desired result
- **Initial** state: a logical sentence about (situation) S0
  - At(Home, S0) ^ ~Have(Milk, S0) ^ ~ Have(Bananas, S0) ^~Have(Drill, S0)
- **Goal** state:
  - (∃s) At(Home,s) ^ Have(Milk,s) ^ Have(Bananas,s) ^Have(Drill,s)
- **Operators** are descriptions of how the world changes as a result of the agent's actions:
  - ∀(a,s) Have(Milk,Result(a,s)) <=> ((a=Buy(Milk) ^ At(Grocery,s)) □ (Have(Milk, s) ^ a~=Drop(Milk)))
- Result(a,s) names the situation resulting from executing action a in situation s.
- Action sequences are also useful: Result'(l,s) is the result of executing the list of actions (l) starting in s:
  - (∀s) Result'([],s) = s
  - (∀a,p,s) Result'([a|p]s) = Result'(p,Result(a,s))

Situation calculus

- A solution is a plan that when applied to the initial
- state yields a situation satisfying the goal query:
  - At(Home,Result'(p,S0))
  - ^ Have(Milk,Result'(p,S0))
  - ^ Have(Bananas,Result'(p,S0))
  - ^ Have(Drill,Result'(p,S0))
- Thus we would expect a plan (i.e., variable assignment through unification) such as:
  - p = [Go(Grocery), Buy(Milk), Buy(Bananas), Go(HardwareStore), Buy(Drill), Go(Home)]

Non-Linear Planning

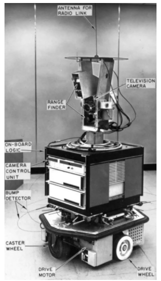
- A plan that consists of sub-problems, which are solved simultaneously is said to be a non-linear plan.
- In case of the goal stack planning, as discussed previously, it poses some problems.
  - Achieving a goal could possibly undo any of the already achieved goals and its called as Sussman`s anomaly.
  - In linear planning, just one goal is taken at a time and solved completely before the next one is taken.
- Example :
  - You want to take the car for servicing and have to make an important phone call.
  - In case of Linear planning, First you will achieve the goal of making a phone call and then will take the car for servicing.
  - Rather than completing both the tasks in a linear way, after completion of the task 1, as partial step, i.e., start the car and put on the Bluetooth, then complete the task 2 of phone call and then finally, complete the task 1 by leaving the car at the service station. This can be an example of non-linear planning.

Many AI Planners in History

- Well-known AI Planners:
  - STRIPS (Fikes and Nilsson, 1971): theoremproving system
  - ABSTRIPS (Sacerdoti, 1974): added hierarchy of abstractions
  - HACKER (Sussman, 1975): use library of procedures to plan
  - NOAH (Sacerdoti, 1975): problem decomposition and plan reordering

STRIPS-Based Approach to Robot Control

- Dartmouth Summer Research Project on Artificial Intelligence (1956)
- Use first-order logic and theorem proving to plan strategies from start to goal
- STRIPS language:
- STanford Research Institute Problem Solver
  - *Classical* approach that most planners use
  - Lends itself to efficient planning algorithms
- Environment: office environment with specially colored and shaped objects
- STRIPS planner: developed for this system to determine the actions of the robot should take to achieve goals
- Cost of Shakey: $100, 000



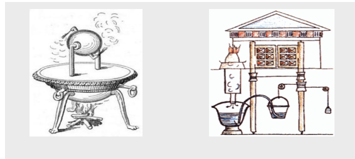
Source : http://www.ai.sri.com/shakey/

Planning is an integral part of Automation

- Recommended clip from Charlie Chaplin's Modern Times to see what can go wrong:
  - https://www.youtube.com/watch?v=n_1apYo6-Ow
  - Goal stack planning
  - Non-linear planning
  - Hierarchical planning
- What we intend to learn:
  1. Partial Order Planning
  2. GraphPlan and SATPlan

Origin of Automation: Replacing Human Muscle Power

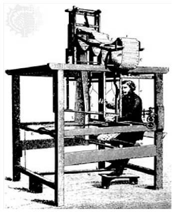
- 10,000 BC Stone tools used in early civilization: tools make better tools.
- Design of simple automation (150 BC) moving engine, Herons door etc. in Greece.



- 1780 AD saw the creation of automatic dolls which could write, draw pictures etc.
- Punch cards used in power looms in France in 1801 for manufacture of textiles Joseph-Marie-Jacquard.

Origin of Automation: Replacing Human Muscle Power

- Programmed textile loom: 1801 in France
- Hard Automation in Ford Motor Company 1904
  - Idea of transfer lines in which a car was assembled at different stations.
  - First use of hard automation – alignment devices, transfer devices etc.
  - 1904 Henry Ford's mass production of vehicles in the USA.

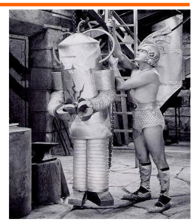


Robot : History

- 1921 Karel Kapec's play depicting human like mechanical man - robots.
- 1942 Isaac Asimov first used the term Robotics.
- 1945 master slave manipulator made for radioactive material handling for the Atom Bomb project.

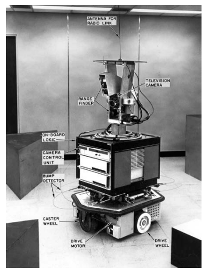
A strictly mechanical device

Motion transfer by wire rope and pulleys

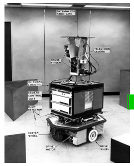


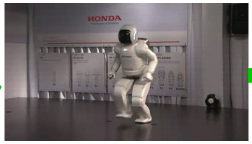
What changed everything?

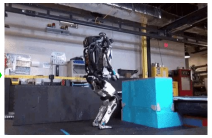
- Mechanical systems became electro-mechanical
- Microprocessor (1949) : concept of reprogram
  - 1950 SHAKY: First robot-Stanford University
  - 1952 George Dovel : teach / play back devices for NC machines/ robots.



Clumsy robots to sophisticated humanoids


1950


2000


2021

What is the definition of a Robot ?

- To be called a robot, it should do some or all of the following:
  - move around
  - sense and manipulate the environment.
  - display intelligent behavior
- Robots are physical agents that perform tasks by manipulating the physical world. They are equipped with :
  - Effectors : Leg, wheels , joints and grippers etc.
  - Sensors : Cameras, radars, lasers and microphone, gyroscopes, strain and torque sensors, accelerometers etc.

Robot Hardware

- Types of robots from hardware prospective
  - Anthropomorphic robots : eg. The Terminator
  - Manipulators : just robot arms
  - Mobile robots : with wheels, legs or rotors to move about the environment
  - Quadcopter drones
  - Unmanned Aerial Vehicle (UAV)
  - Autonomous underwater vehicles (AUV)
  - Autonomous car or rovers
  - Legged robots

Sensing the World

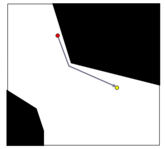
- **SENSORS**: are the perceptual interface between robot and environment.
- **Passive sensors :** True observers of the environment
  - Cameras, Stereo vision , Kinect (cameras+ structured light projector)
- **Active sensors :** send energy into the environment; rely on the fact that this energy is reflected back to the sensor
- **Range finders :**
  - Sonar , scanning lidars , Radar
  - Tactile sensors : whiskers, bump panels and touch sensitive skin.
- **Location sensors :**
  - Global Positioning system (GPS) , Differential GPS
- **Proprioceptive sensors :** inform robot of its own motion
  - eg : shaft decoders , odometry, inertial sensors, force sensors or torque sensors

Producing Motion

- The mechanism that initiates the motion of an effector is called an **actuator**
- Electric actuator : used in system with rotational motion like joints on a robot arm
- Hydraulic actuators
- Pneumatic actuators
- Revolute joints
- Prismatic joints
- Grippers : parallel jaw gripper

What is Planning for Robotics?

- Given
  - model (states and actions) of the robot(s) $M^R = <S^R,A^R>$
  - a model of the world M^W
  - current state of the robot $s^W_{current}$
  - current state of the world $s^W_{current}$
  - cost function C of robot actions
  - desired set of states for robot and world G



- Compute a plan π that
  - prescribes a set of actions $a_1 ,...a_K$ in A^R the robot should execute
  - reaches one of the desired states in G
  - (preferably) minimizes the cumulative cost of executing actions $a_1 ,...a_K$

Differences between Robotics and Automation ?

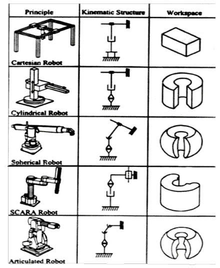
- Robotics focuses on systems incorporating sensors and actuators that operate autonomously or semi- autonomously in cooperation with humans.
  - Environment is partially observable and stochastic.
- Robotics research emphasizes intelligence and adaptability to cope with unstructured environments.
- Automation research emphasizes efficiency, productivity, quality, and reliability, focusing on systems that operate autonomously, often in structured environments over extended periods, and on the explicit structuring of such environments.

Three generations of robotics / engineering

- First generation of robots: simple pick and place devices with no external sensors.
- Second generation robots: external Sensors (vision, tactile, etc) for interaction with the environment.
- Third generation robots: intelligence, smart materials, bio , etc.
- Future robots: bio-robots, micro , nano cybogs, aneroids etc.

First Generation Robots: 1950-1970 NC Technology

- ➢ Simple motion capabilities for pick and place applications
- ➢ Robots made of revolute joints actuated by open loop or closed loop control.



Second Generation of Robots: 1970-1990

- Electronics: smaller, faster and cheaper processors
  - External sensors : interaction with the environment
    - vision
    - advanced sensors : gyros, inclination, force, slip.
    - advanced controllers : microcontroller, DSP
    - speech recognition
    - AI

Third generation robots 1990 - 2000

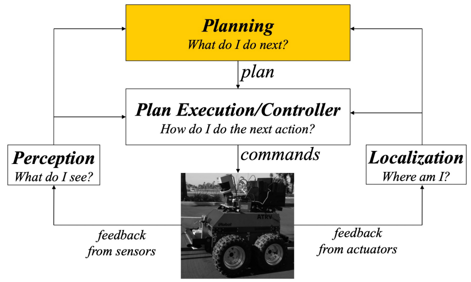
- New materials – smart materials, smart actuators.
- Interest in emulating biological design paradigms.
- New areas like: Micro, Nano-robotics, Vision, bio-robotics, etc.

Autonomous Robotic Systems

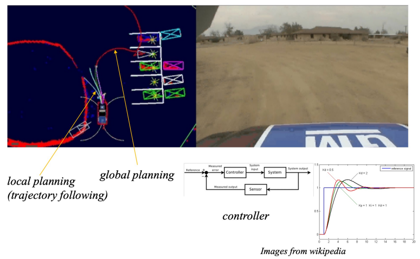
- Autonomous system ?
- Three level hierarchy in robotics :
  - Task planning
  - Motion planning
  - Control

Motion planning is the ability for an agent to compute its own motions in order to achieve certain goals.
All autonomous robots and digital actors should eventually have this ability.

Planning within a Typical Autonomy Architecture

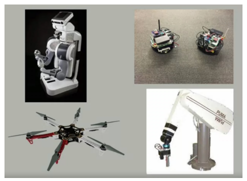


Planning vs. Trajectory Following vs. Control



Motion Planning Problem

- A special case of more general planning problem
- Goal is to develop techniques that would allow a robot or robots to automatically decide ow to move from one position or configuration to another



Robot Motion Planning

- Enables an autonomous mobile robot to determine its movements in a cluttered environment so as to achieve a variety of goals while avoiding collisions.
- The ability of a robot to plan its motions without explicit human guidance is a basic prerequisite for robotic autonomy.
- **Classical robot motion planning**–
  - planning when the geometry of the robot's stationary surroundings is known in advance.
- **Sensor-based motion planning**–
  - planning in the presence of a priori unknown or poorly known geometry of the robot's surroundings.
  - advanced sensor-based planning algorithms, and
  - robotic mapping and localization methods.

Planning : Robot Motion

- How do you represent the environment?
- How do you find a collision-free path?
- How can you know that it is the best possible path?
- What if you need to find a solution faster?
- How can you ensure that you will always find a path that the robot can follow with limitations in sensing and actuation?

Planning : Robot Motion

- Characterization of a motion planner is according to the problem it solves.
- Robotic planning considers four tasks:
  - **Navigation**
    - is the problem of finding a collision-free motion for the robot system from one configuration (or state) to another.
    - The robot could be a robot arm, a mobile robot, or something else.
  - **Coverage**
    - is the problem of passing a sensor or tool over all points in a space, such as in demining or painting.
  - **Localization**
    - is the problem of using a map to interpret sensor data to determine the configuration of the robot.
  - **Mapping**
    - is the problem of exploring and sensing an unknown environment to construct a representation that is useful for navigation, coverage, or localization.
  - **Localization and mapping can be combined, as in SLAM (Simultaneous Localization and Mapping)**

Planning as Graph Search Problem

- Planning
  1. Construct a graph representing the planning problem
  2. Search the graph for a (hopefully, close-to-optimal) path

The two steps above are often interleaved

Planning Representations:

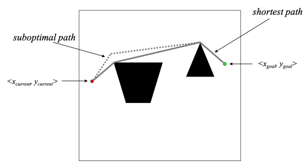
Skeleton- and Grid-based Graphs, Explicit vs. Implicit Graphs

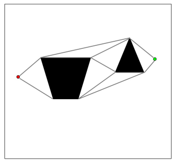
Two Classes of Graph Construction Methods

- Skeletonization
  - Visibility Graphs [Wesley & Lozano-Perez '79]
  - Voronoi diagrams
  - Probabilistic roadmaps
- Cell decomposition
  - X-connected grids
  - lattice-based graphs

Skeletonization-based Graphs

- Visibility Graphs [Wesley & Lozano-Perez '79]
  - based on idea that the shortest path consists of obstacle-free straight line segments connecting all obstacle vertices and start and goal
  - construct a graph by connecting all vertices, start and goal by obstacle-free straight line segments (graph is $O(n^2)$), where n - # of vert.)



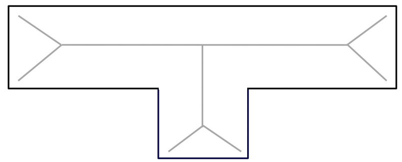


Visibility Graphs : Skeletonization-based Graphs

- advantages:
  - independent of the size of the environment
- disadvantages:
  - path is too close to obstacles
  - hard to deal with the cost function that is not distance
  - hard to deal with non-polygonal obstacles
  - hard to maintain the polygonal representation of obstacles
  - can be expensive in spaces higher than 2D

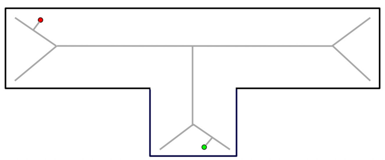
Voronoi diagram : Skeletonization-based Graphs

- Voronoi diagram [Rowat '79]
  - set of all points that are equidistant to two nearest obstacles (can be computed O (n log n), where n - # of points that represent obstacles)



Skeletonization-based Graphs

- Voronoi diagram-based graph
  - Edges: Boundaries in Voronoi diagram
  - Vertices: Intersection of boundaries
  - Add start and goal vertices
  - Add edges that correspond to:
    - shortest path segment from start to the nearest segment on the Voronoi diagram
    - shortest path segment from goal to the nearest segment on the Voronoi diagram



Skeletonization-based Graphs

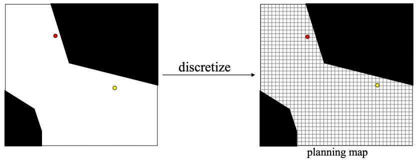
- Voronoi diagram-based graph
- Advantages:
  - tends to stay away from obstacles
  - independent of the size of the environment
  - can work with any obstacles represented as set of points
- Disadvantages:
  - can result in highly suboptimal paths
  - hard to deal with the cost function that is not distance
  - hard to use/maintain beyond 2D

Two Classes of Graph Construction Methods

- Skeletonization
  - Visibility Graphs [Wesley & Lozano-Perez '79]
  - Voronoi diagrams
  - Probabilistic roadmaps
- Cell decomposition
  - X-connected grids
  - lattice-based graphs

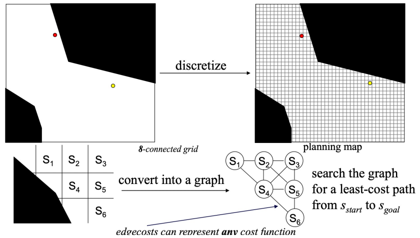
Grid-based Graphs

- Approximate Cell Decomposition:
  - overlay uniform grid (discretize)



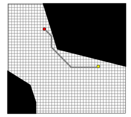
Grid-based Graphs

- Approximate Cell Decomposition:
  - construct a graph



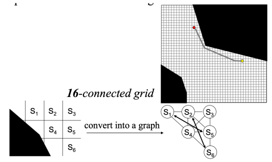
Grid-based Graphs

- Graph construction:
  - connect neighbours
  - path is restricted to 45º degrees



- connect cells to neighbor of neighbors
- path is restricted to 22.5º degrees

- path is restricted to 26.6º/63.4º degrees



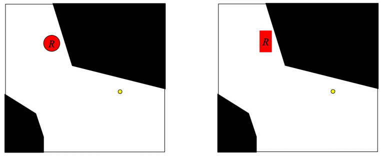
Cell Decomposition-based Graphs

- Grid-based graph
- advantages:
  - very simple to implement (super popular)
  - can represent any dimensional space
  - works well with obstacles represented as set of points
  - works with any cost function
- disadvantages:
  - size does depend on the size of the environment
  - can be expensive to compute/store if # of dimensions > 3

2D Planning for Omnidirectional Non-Circular Non-point Robot

- Planning for omnidirectional point robot:

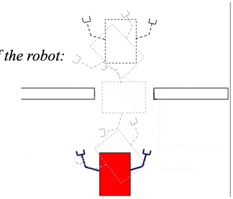
What is $M^R = <x,y>$
What is $M^W = <obstacle/free\ space>$
What is $s^R_{current} = <x_{current}, y_{current}>$
What is $s^W_{current}$ = constant
What is C = Euclidean Distance
What is $G = <x_{goal}, y_{goal}>$



Configuration Space (C- Space)

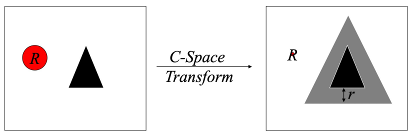
- Configuration is legal if it does not intersect any obstacles and is valid
- Configuration Space is the set of legal configurations

Legal configurations for the base of the robot:



C-Space Transform

- Configuration space for a robot base in 2D world is:
  - 2D if robot's base is circular

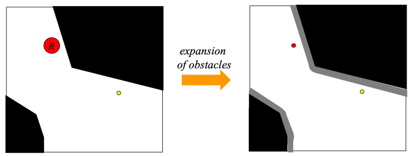


- expand all obstacles by radius r of the robot's base
- graph construction can then be done assuming point robot

2D Planning for Omnidirectional Non-Circular Non-point Robot

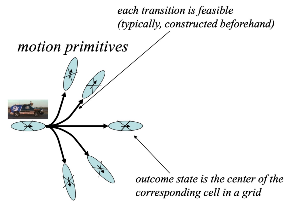
- Planning for omnidirectional point robot:

What is $M^R = <x,y>$
What is $M^W = <obstacle/free\ space>$
What is $s^R_{current} = <x_{current}, y_{current}>$
What is $s^W_{current}$ = constant
What is C = Euclidean Distance
What is $G = <x_{goal}, y_{goal}>$



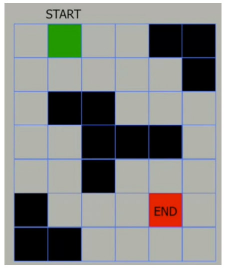
Lattice Graphs [Pivtoraiko & Kelly '05]

- Graph {V, E} where
  - V: centers of the grid-cells
  - E: motion primitives that connect centers of cells via short-term feasible motions



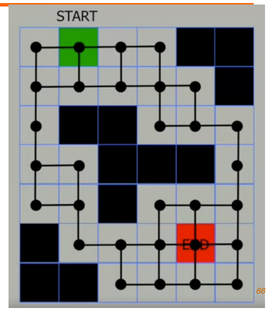
Planning on grid

- Robot can move between adjacent cells on the grid
- Dark part – obstacles

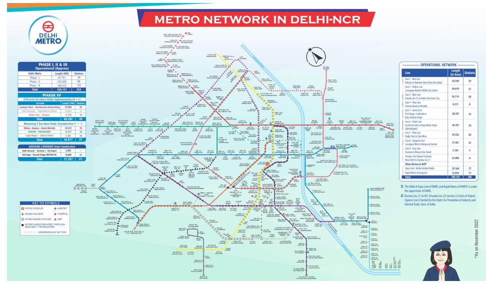


AI Problem formulation : Graph Structure

- Graph :
  - Node
  - Edges
    - Annotated with numerical value
    - Indicate relevant quantities like distance or cost
  -

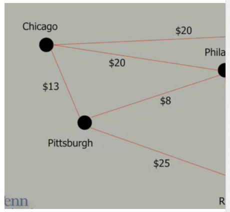


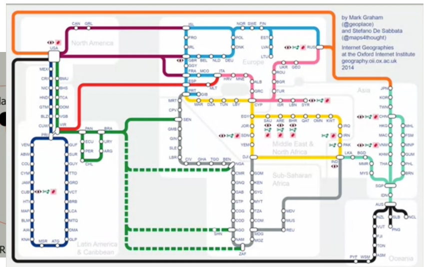
AI Problem formulation :Delhi metro



AI Problem formulation

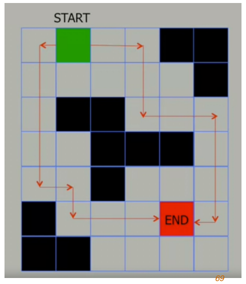
- Toll chart
- WWW





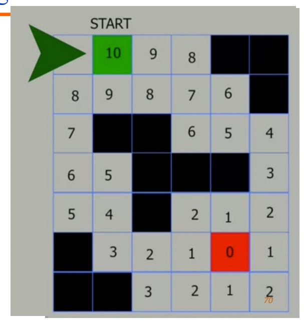
Planning on grid

- Cost or distance of 1 with every edge in the graph
- Goal is to construct a path through the grid /graph from the start to the goal
- Many possible paths
- Interested in the shortest path



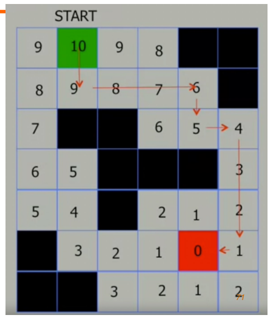
Grassfire :Algorithm

- Grassfire algorithm
- Begin the Goal as distance 0
- Then 1 step from the goal (+1)

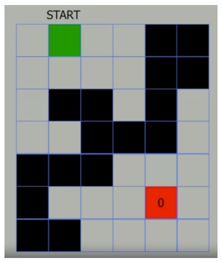


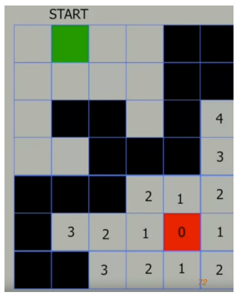
Grassfire

- For each node n in the graph
  - n.distance = Infinity
- Create an empty list.
- goal.distance = 0, add goal to list.
- While list not empty
  - Let current = first node in list, remove current from list
  - For each node, n that is adjacent to current
    - If n.distance = Infinity
      - n.distance = current.distance + 1
      - add n to the back of the list

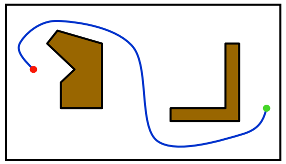


If path not exist



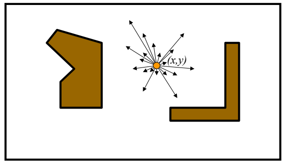


Basic problem



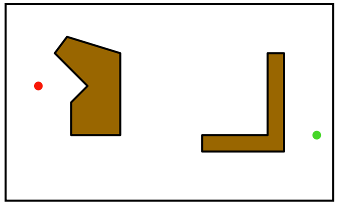
- Point robot in a 2-dimensional workspace with obstacles of known shape and position
- Find a collision-free path between a start and a goal position of the robot

Basic problem



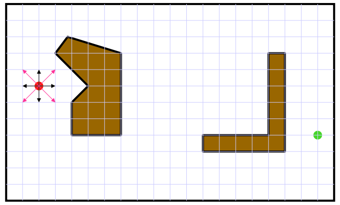
- Each robot position (x,y) can be seen as a state
- → Continuous state space
- Then each state has an infinity of successors
- We need to discretize the state space

Path Planning



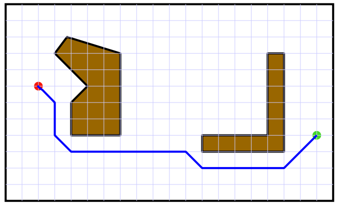
What is the state space?

Formulation #1



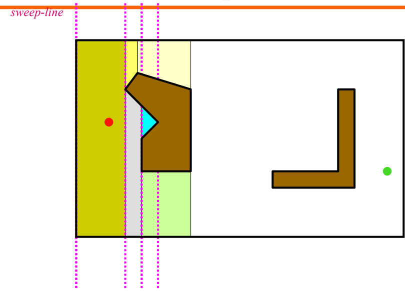
Cost of one horizontal/vertical step = 1
Cost of one diagonal step = √2

Optimal Solution

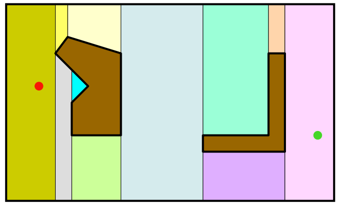


This path is the shortest in the discretized state space, but not in the original continuous space

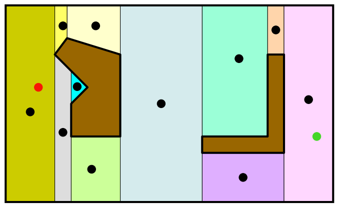
Formulation #2- Trapezoidal Decomposition



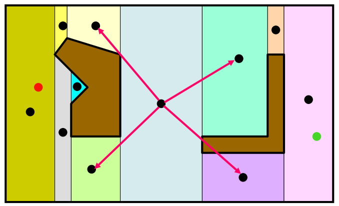
Formulation #2



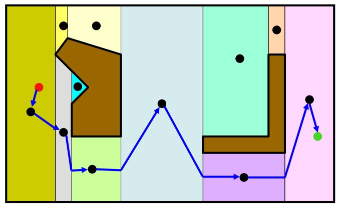
States



Successor Function

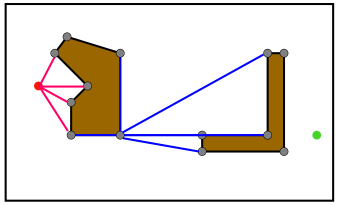


Solution Path



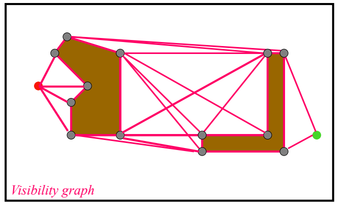
A path-smoothing post-processing step is usually needed to shorten the path further

Formulation #3: Visibility Graph



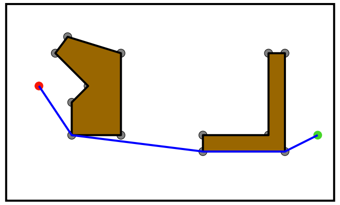
Cost of one step: length of segment

Formulation #3



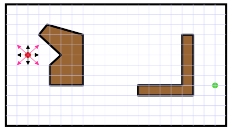
Cost of one step: length of segment

Solution Path



The shortest path in this state space is also the shortest in the original continuous space

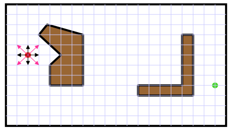
Robot Navigation Heuristics



Cost of one horizontal/vertical step = 1
Cost of one diagonal step = $\sqrt{2}$

$h_1(N) = \sqrt{(x_N-x_g)^2+(y_N-y_g)^2}$ is admissible

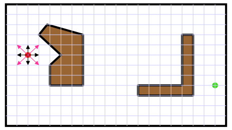
Robot Navigation Heuristics



Cost of one horizontal/vertical step = 1
Cost of one diagonal step = $\sqrt{2}$

$h_2(N) = |x_N-x_g| + |y_N-y_g|$ is ???

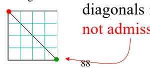
Robot Navigation Heuristics



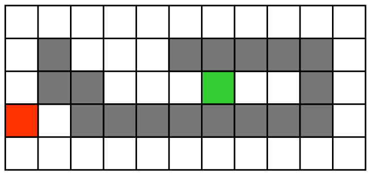
Cost of one horizontal/vertical step = 1
Cost of one diagonal step = $\sqrt{2}$

$h_2(N) = |x_N-x_g| + |y_N-y_g|$ is admissible if moving along diagonals is not allowed, and not admissible otherwise

$h^*(I) = 4\sqrt{2}$
$h_2(I) = 8$



Robot Navigation



Robot Navigation

f(N) = h(N), with h(N) = Manhattan distance to the goal
(not A)*

8	7	6	5	4	3	2	3	4	5	6
7		5	4	3						5
6			3	2	1	0	1	2		4
7	6									5
8	7	6	5	4	3	2	3	4	5	6

Robot Navigation

f(N) = h(N), with h(N) = Manhattan distance to the goal
(not A)*

8	7	6	5	4	3	2	3	4	5	6
7		5	4	3						5
6			3	2	1	0	1	2		4
7	6									5
8	7	6	5	4	3	2	3	4	5	6

Robot Navigation

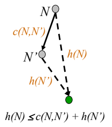
f(N) = g(N)+h(N), with h(N) = Manhattan distance to goal
(A)*

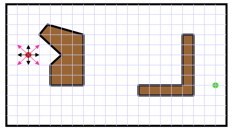
8+3	7+4	6+3	5+6	4+7	3+8	2+9	3+10	4	5	6
7+2		5+6	4+7	3+8						5
6+1			3	2+9	1+10	0+11	1	2		4
7+0	6+1									5
8+1	7+2	6+3	5+4	4+5	3+6	2+7	3+8	4	5	6

Best-First Search

- An evaluation function f maps each node N of the search tree to a real number
  f(N) ≥ 0
- Best-first search sorts the FRINGE in increasing f

Robot Navigation



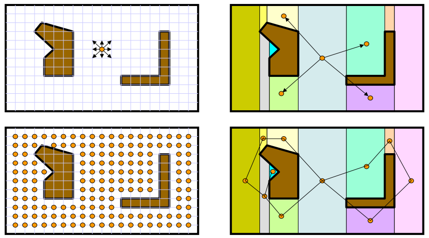


Cost of one horizontal/vertical step = 1
Cost of one diagonal step = $\sqrt{2}$

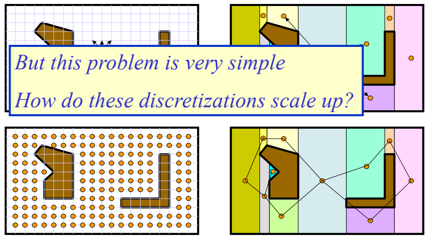
$h_1(N) = \sqrt{(x_N-x_g)^2+(y_N-y_g)^2}$ *is consistent*

$h_2(N) = |x_N-x_g| + |y_N-y_g|$ *is consistent if moving along diagonals is not allowed, and not consistent otherwise*

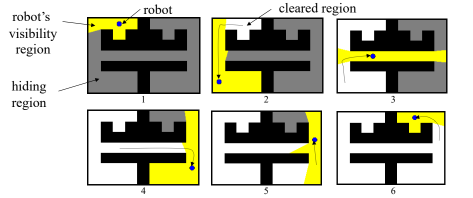
Two Possible Discretizations



Two Possible Discretizations



Intruder Finding Problem

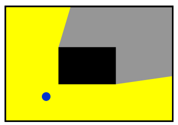


- A moving intruder is hiding in a 2-D workspace
- The robot must "sweep" the workspace to find the intruder
- Both the robot and the intruder are points

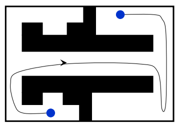
Does a solution always exist?

Does a solution always exist?

No !

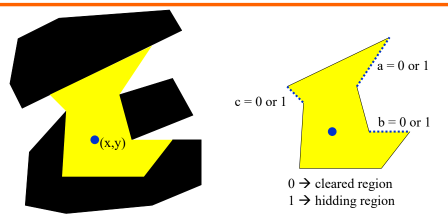


Easy to test:
"Hole" in the workspace



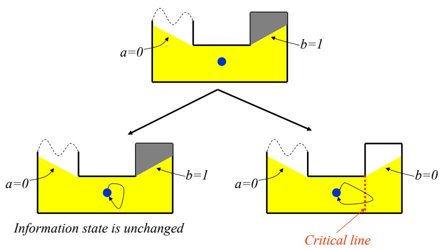
Hard to test:
No "hole" in the workspace

Information State

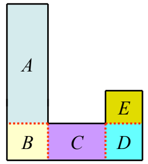


- Example of an information state = (x,y,a=1,b=1,c=0)
- An initial state is of the form (x,y,1, 1, ..., 1)
- A goal state is any state of the form (x,y,0,0, ..., 0)

Critical Line

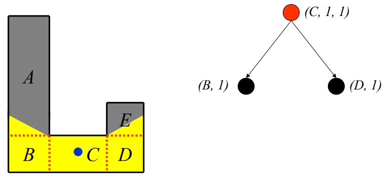


Criticality-Based Discretization

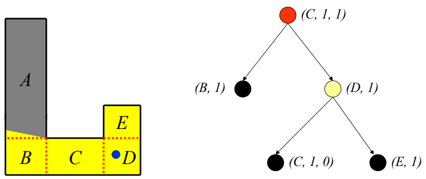


Each of the regions A, B, C, D, and E consists of "equivalent" positions of the robot, so it's sufficient to consider a single position per region

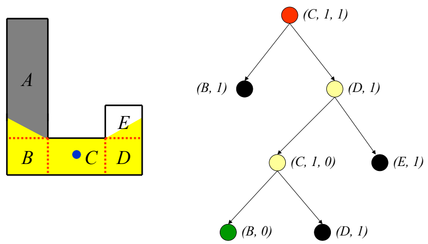
Criticality-Based Discretization



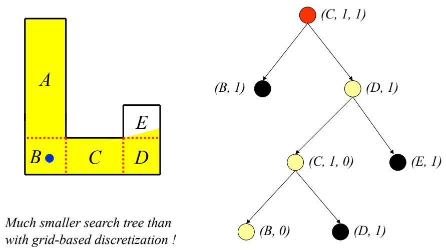
Criticality-Based Discretization



Criticality-Based Discretization

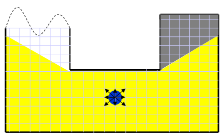


Criticality-Based Discretization



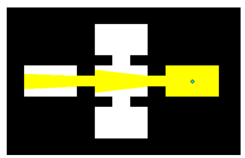
Much smaller search tree than with grid-based discretization !

Grid-Based Discretization



- Ignores critical lines → Visits many "equivalent" states
- Many information states per grid point
- Potentially very inefficient

Example of Solution

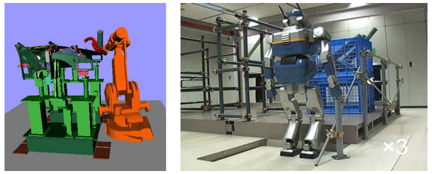


But ...

Criticality-based discretization does not scale well in practice when the dimensionality of the continuous space increases

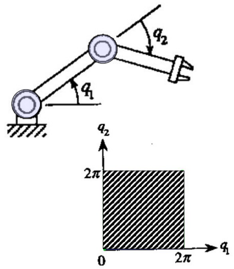
(It becomes prohibitively complex to define and compute)

Motion Planning for an Articulated Robot



Find a path to a goal configuration that satisfies various constraints: collision avoidance, equilibrium, etc...

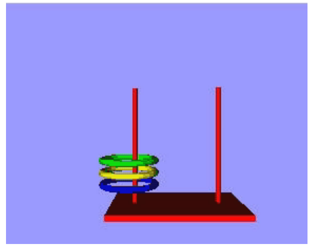
Configuration Space of an Articulated Robot



- A configuration of a robot is a list of non-redundant parameters that fully specify the position and orientation of each of its bodies
- In this robot, one possible choice is: (q_1, q_2)

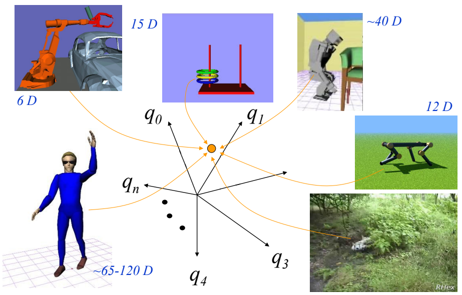
The configuration space (C-space) has 2 dimensions

How many dimensions has the C-space of these 3 rings?

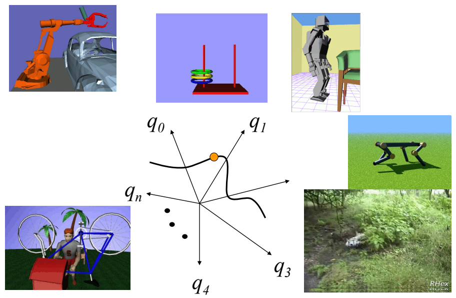


Answer:
3×5 = 15

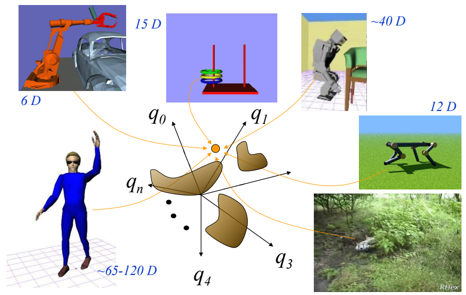
Every robot maps to a point in its configuration space ...



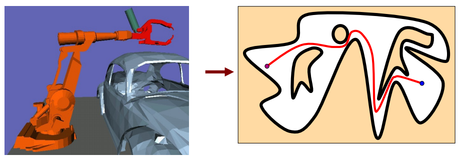
... and every robot path is a curve in configuration space



But how do obstacles (and other constraints) map in configuration space?

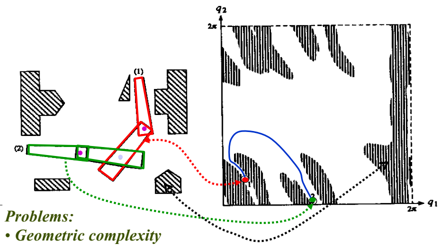


C-space "reduces" motion planning to finding a path for a point



But how do the obstacle constraints map into C-space ?

A Simple Example: Two-Joint Planar Robot Arm



Problems:
- ***Geometric complexity***
- ***Space dimensionality***

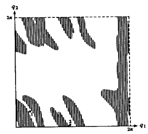
Continuous state space

↓

Discretization

↓

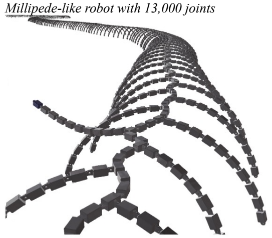
Search



C-space

Robots with many joints:
Modular Self-Reconfigurable Robots

Millipede-like robot with 13,000 joints

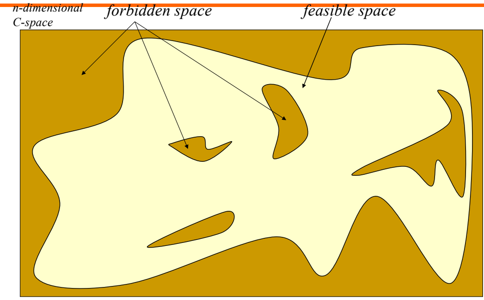


(M. Yim)

(S. Redon)

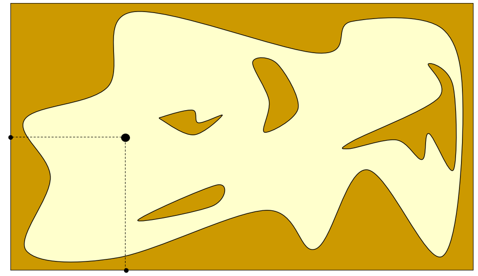
119

Probabilistic Roadmap (PRM)



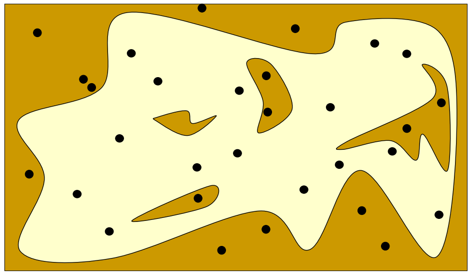
Probabilistic Roadmap (PRM)

Configurations are sampled by picking coordinates at random



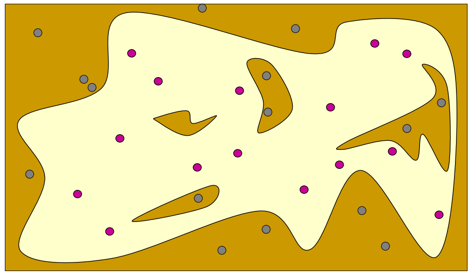
Probabilistic Roadmap (PRM)

Configurations are sampled by picking coordinates at random



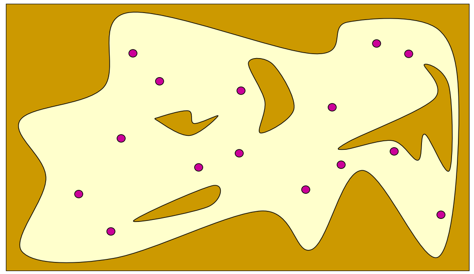
Probabilistic Roadmap (PRM)

Sampled configurations are tested for collision (feasibility)



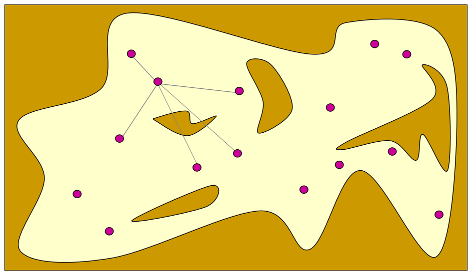
Probabilistic Roadmap (PRM)

The collision-free configurations are retained as "milestones" (states)



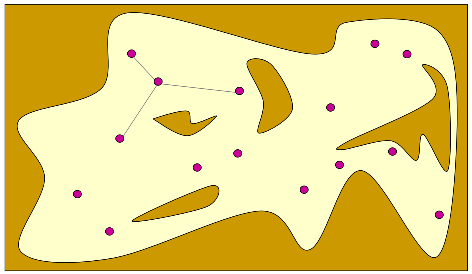
Probabilistic Roadmap (PRM)

Each milestone is linked by straight paths to its k-nearest neighbors



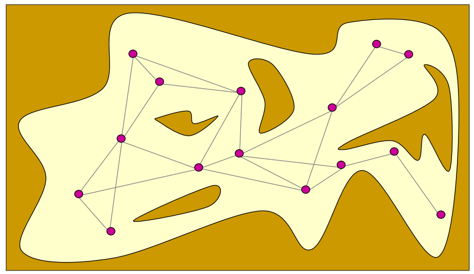
Probabilistic Roadmap (PRM)

Each milestone is linked by straight paths to its k-nearest neighbors



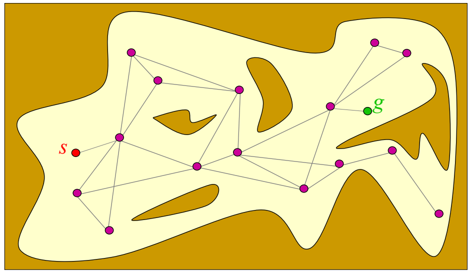
Probabilistic Roadmap (PRM)

The collision-free links are retained to form the PRM (state graph)



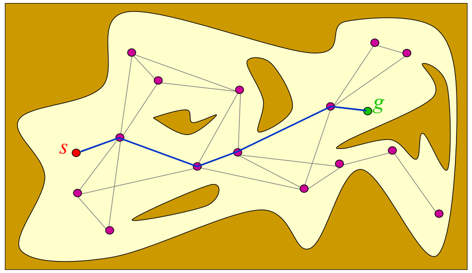
Probabilistic Roadmap (PRM)

The start and goal configurations are connected to nodes of the PRM



Probabilistic Roadmap (PRM)

The PRM is searched for a path from s to g



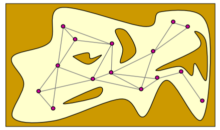
Continuous state space

↓

Discretization

↓

Search *A**

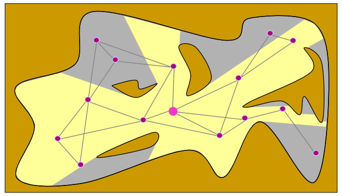


Why Does PRM Work?

Because most feasible spaces verifies some good geometric (visibility) properties

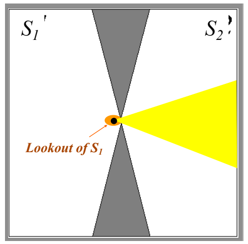
Why Does PRM Work?

In most feasible spaces, every configuration "sees" a significant fraction of the feasible space



→ A relatively small number of milestones and connections between them are sufficient to cover most feasible spaces with high probability

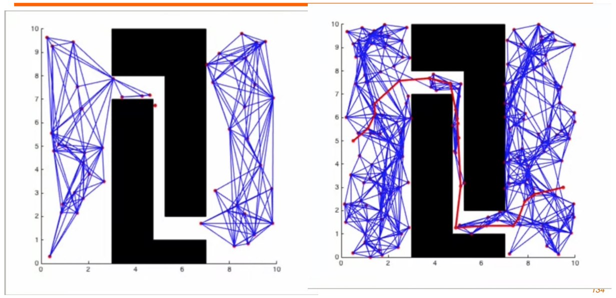
Narrow-Passage Issue



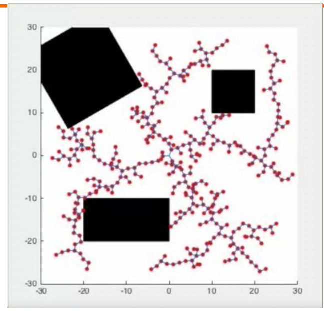
The lookout of a subset S of the feasible space is the set of all configurations in S from which it is possible to "see" a significant fraction of the feasible space outside S

*The feasible space is **expansive** if all of its subsets have a large lookout*

Issue



Rapidly Exploring Random Trees (RRT)



Rapidly Exploring Random Trees (RRT)

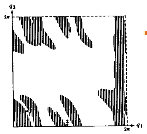
- Add start node to tree
- Repeat n times
  - o Generate a random configuration, x
  - o If x is in freespace using the CollisionCheck function
    - Find y, the closest node in the tree to the random configuration x
    - If (Dist (x, y) > delta) – Check if x is too far from y
      - Find a configuration, z, that is along the path from x to y such that Dist(z,y) <= delta
      - x = z;
    - If (LocalPlanner (x,y)) – Check if you can get from x to y
      - Add x to the tree with y as its parent

Probabilistic Completeness of a PRM Motion Planner

In an expansive feasible space, the probability that a PRM planner with uniform sampling strategy finds a solution path, if one exists, goes to 1 exponentially with the number of milestones (~ running time)

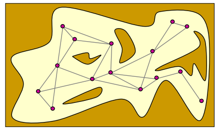
A PRM planner can’t detect that no path exists. Like A*, it must be allocated a time limit beyond which it returns that no path exists. But this answer may be **incorrect**. Perhaps the planner needed more time to find one !

Continuous state space



↓

Discretization

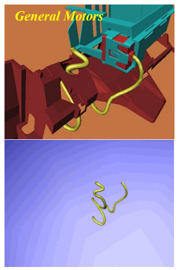


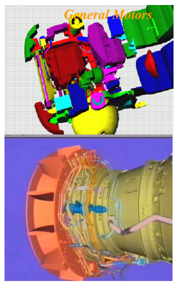
↓

Search

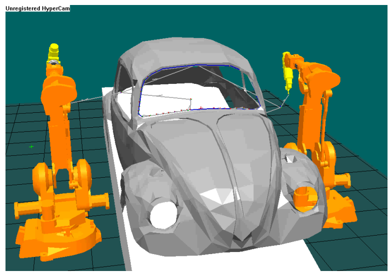
Some Applications of Motion Planning

Design for Manufacturing and Servicing



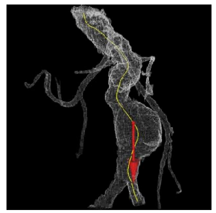


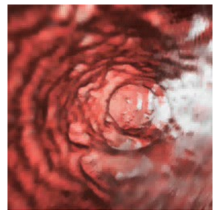
Automatic Robot Programming



ABB

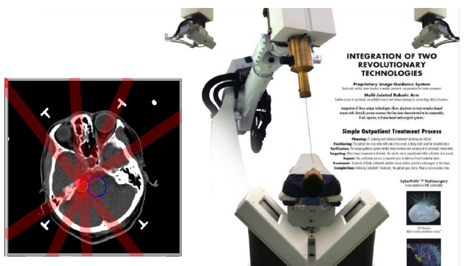
Virtual Angiography





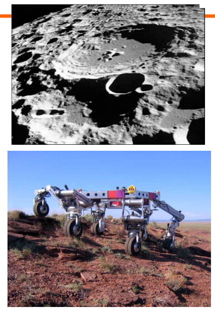
[S. Napel, 3D Medical Imaging Lab, Stanford]

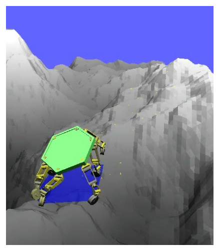
Radiosurgery



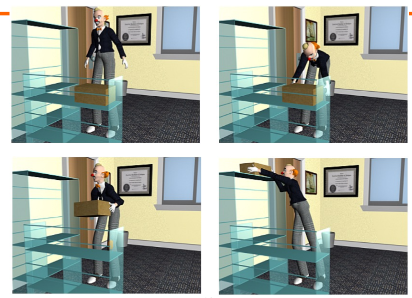
CyberKnife (Accuray)

Planet Exploration





Autonomous Digital Actors



[Yamane, Kuffner and Hodgins]

Next :

- Module 6: Reasoning Under Uncertainty

References

- ***Artificial Intelligence*** *by Elaine Rich & Kevin Knight*, Third Ed, Tata McGraw Hill
- ***Artificial Intelligence and Expert System*** *by Patterson*
- http://www.cs.rmit.edu.au/AI-Search/Product/
- http://aima.cs.berkeley.edu/demos.html (for more demos)
- ***Artificial Intelligence and Expert System*** *by Patterson*
- Slides adapted from CS188 Instructor: Anca Dragan, University of California, Berkeley
- Slides adapted from CS60045 ARTIFICIAL INTELLIGENCE