MISC: Motion Planning & Control + Shaping the Dynamic Response

METR 4202: Advanced Control & **Robotics**

Dr Surya Singh -- Lecture # 11

October 15, 2014

metr4202@itee.uq.edu.au
http://robotics.itee.uq.edu.au/~metr4202/

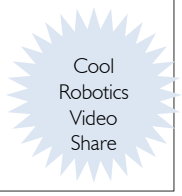
© 2014 School of Information Technology and Electrical Engineering at the University of Queensland

Schedule

Week	Date	Lecture (W: 11:10-12:40, 24-402)
1	30-Jul	Introduction
2	6-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	13-Aug	Robot Kinematics (& Ekka Day)
4	20-Aug	Robot Dynamics & Control
5	27-Aug	Robot Motion
6	3-Sep	Robot Sensing: Perception & Multiple View Geometry
7	10-Sep	Robot Sensing: Features & Detection using Computer Vision
8	17-Sep	Navigation (+ Prof. M. Srinivasan)
9	24-Sep	Localization & Motion Planning + Control
	1-Oct	*Study break*
10	8-Oct	**State-Space Modelling**
11	15-Oct	Shaping the Dynamic Response
12	22-Oct	Linear Observers & LQR
13	29-Oct	Applications in Industry & Course Review

Announcements: Lab 2 Extended

- **Feedback from Lab 1!**
- **Lab 2 Report:**
  - Same as last lab
  - This time it's public
  - "Private" (PAF) comments are in the **online Lab2 PAF**
- **Lab 3:**
  - Cup stack/unstacking
  - Hope to get it written by this afternoon.
  - Might be a "live document" like last year
- Cool Robotics Share Site
  ➔ http://metr4202.tumblr.com/



State-Space Modelling II

State Space Definition

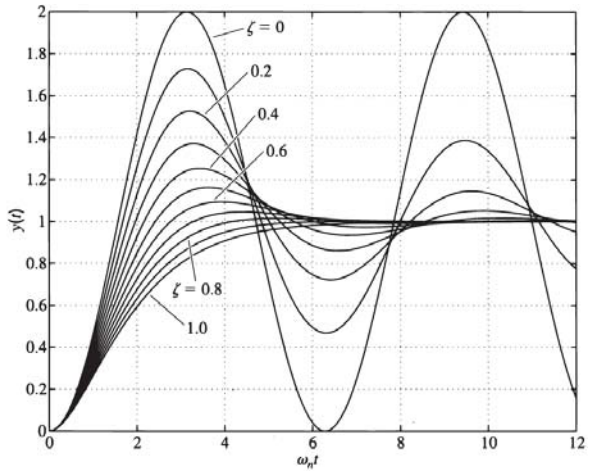
State:

- The state of a dynamic system is a set of physical quantities, the specification of which (in the absence of external excitation) completely determines the evolution of the system.
  - *specific* quantities that define the system state are not unique
  - their number (called the system *order)* is unique
  - ∴ obvious choice of the variables *(state variables)* to define the system state, but there are also many cases in which the choice of state variables is by no means obvious

2nd Order System Response

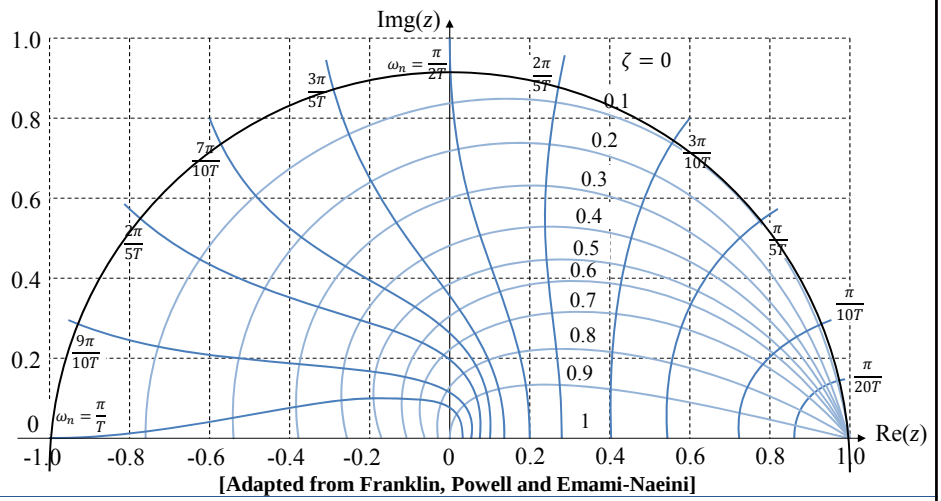
2nd Order System Response

- Response of a 2nd order system to increasing levels of damping



Damping and natural frequency

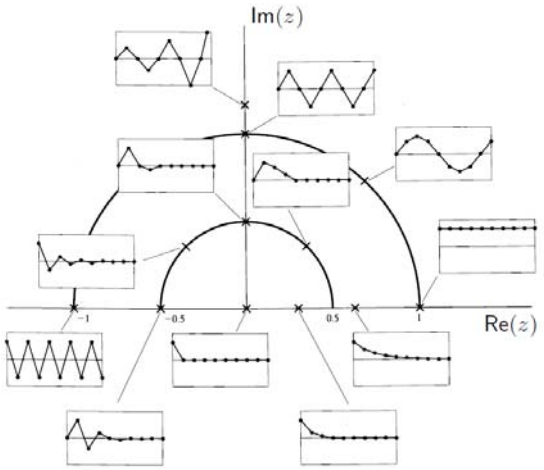
$$z = e^{sT} \text{ where } s = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}$$



[Adapted from Franklin, Powell and Emami-Naeini]

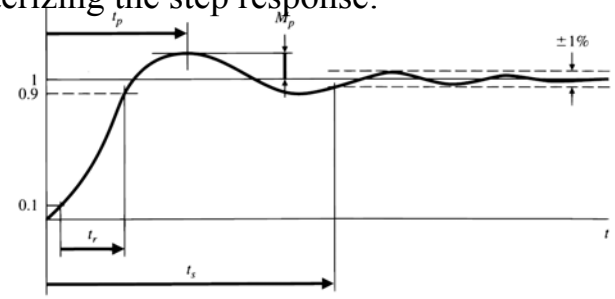
Pole positions in the z-plane

- Poles inside the unit circle are **stable**
- Poles outside the unit circle **unstable**
- Poles on the unit circle are oscillatory
- Real poles at 0 < z < 1 give exponential response
- Higher frequency of oscillation for larger
- Lower apparent damping for larer and r



2nd Order System Specifications

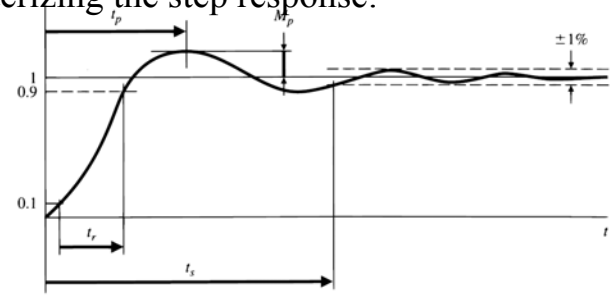
Characterizing the step response:



- Rise time (10% → 90%): $t_r \approx \frac{1.8}{\omega_0}$
- Overshoot: $M_p \approx \frac{e^{-\pi\zeta}}{\sqrt{1-\zeta^2}}$
- Settling time (to 1%): $t_s = \frac{4.6}{\zeta\omega_0}$
- Steady state error to unit step: e_{ss}
- Phase margin: $\phi_{PM} \approx 100\zeta$

2nd Order System Specifications

Characterizing the step response:



- Rise time (10% → 90%) & Overshoot:
  t_r, M_p → ζ, ω_0 : Locations of dominant poles
- Settling time (to 1%):
  t_s → radius of poles: $|z| < 0.01^{\frac{T}{t_s}}$
- Steady state error to unit step:
  e_{ss} → final value theorem $e_{ss} = \lim_{z \to 1} \{(z-1) F(z)\}$

Ex: System Specifications → Control Design [1/4]

Design a controller for a system with:

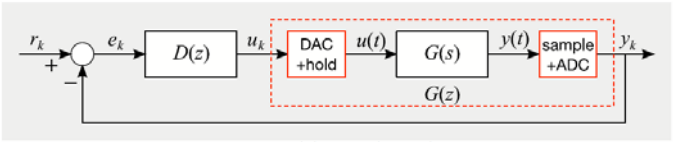
- A continuous transfer function: $G(s) = \frac{0.1}{s(s+0.1)}$
- A discrete ZOH sampler
- Sampling time (T_s): T_s= 1s
- $C \; u_k = -0.5u_{k-1} + 13\,(e_k - 0.88e_{k-1})$

The closed loop system is required to have:

- $M_p < 16\%$
- $t_s < 10$ s
- $e_{ss} < 1$

Ex: System Specifications → Control Design [2/4]

1. (a) Find the pulse transfer function of $G(s)$ plus the ZOH



$$G(z) = (1 - z^{-1})\mathcal{Z}\left\{\frac{G(s)}{s}\right\} = \frac{(z-1)}{z}\mathcal{Z}\left\{\frac{0.1}{s^2(s+0.1)}\right\}$$

e.g. look up $\mathcal{Z}\{a/s^2(s+a)\}$ in tables:

$$G(z) = \frac{(z-1)}{z}\,\frac{z\Big((0.1-1+e^{-0.1})z + (1-e^{-0.1}-0.1e^{-0.1})\Big)}{0.1(z-1)^2(z-e^{-0.1})}$$
$$= \frac{0.0484(z+0.9672)}{(z-1)(z-0.9048)}$$

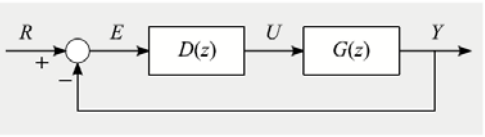
(b) Find the controller transfer function (using z = shift operator):

$$\frac{U(z)}{E(z)} = D(z) = 13\frac{(1-0.88z^{-1})}{(1+0.5z^{-1})} = 13\frac{(z-0.88)}{(z+0.5)}$$

Ex: System Specifications → Control Design [3/4]

2. Check the steady state error e_{ss} when r_k = unit ramp

$$e_{ss} = \lim_{k\to\infty} e_k = \lim_{z\to 1}(z-1)E(z)$$



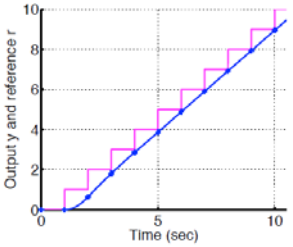
$$\frac{E(z)}{R(z)} = \frac{1}{1+D(z)G(z)}$$
$$R(z) = \frac{Tz}{(z-1)^2}$$

so $e_{ss} = \lim_{z\to 1}\left\{(z-1)\frac{Tz}{(z-1)^2}\frac{1}{1+D(z)G(z)}\right\} = \lim_{z\to 1}\frac{T}{(z-1)D(z)G(z)}$

$$= \lim_{z\to 1}\frac{T}{(z-1)\dfrac{0.0484(z+0.9672)}{(z-1)(z-0.9048)}D(1)}$$

$$= \frac{1-0.9048}{0.0484(1+0.9672)D(1)} = 0.96$$

$\Longrightarrow$ $e_{ss} < 1$ (as required)



Ex: System Specifications → Control Design [4/4]

3. Step response: overshoot $M_p < 16\% \implies \zeta > 0.5$

   settling time $t_s < 10 \implies |z| < 0.01^{1/10} = 0.63$

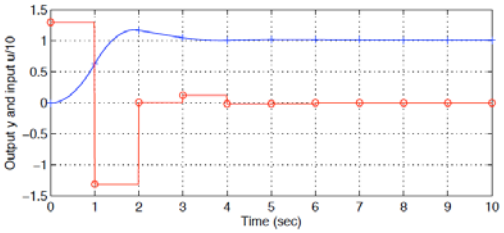
The closed loop poles are the roots of $1 + D(z)G(z) = 0$, i.e.

$$1 + 13\frac{(z-0.88)}{(z+0.5)}\frac{0.0484(z+0.9672)}{(z-1)(z-0.9048)} = 0$$

$$\implies z = 0.88,\ -0.050 \pm j0.304$$

But the pole at $z = 0.88$ is cancelled by controller zero at $z = 0.88$, and

$$z = -0.050 \pm j0.304 = re^{\pm j\theta} \implies \begin{cases} r = 0.31,\ \theta = 1.73 \\ \zeta = 0.56 \end{cases}$$



all specs satisfied!

Example:
(Back To) Robot Arms

Slides 17-27 Source: R. Lindeke, ME 4135, "Introduction to Control"

Remembering the Motion Models:

- Recall from Dynamics, the Required Joint Torque is:

$$\tau_i = D_i(q)\,\ddot{q}_i + C_i(q, \dot{q}_i) + h(q) + b(\dot{q}_i)$$

- $D_i(q)\,\ddot{q}_i$: Dynamical Manipulator Inertial Tensor – a function of position and acceleration
- $C_i(q, \dot{q}_i)$: Coupled joint effects (centrifugal and coriolis) issues due to multiple moving joints
- $h(q)$: Gravitational Effects
- $b(\dot{q}_i)$: Frictional Effect due to Joint/Link movement

Lets simplify the model

- This torque model is a 2nd order one (in position) lets look at it as a velocity model rather than positional one then it becomes a system of highly coupled 1st order differential equations

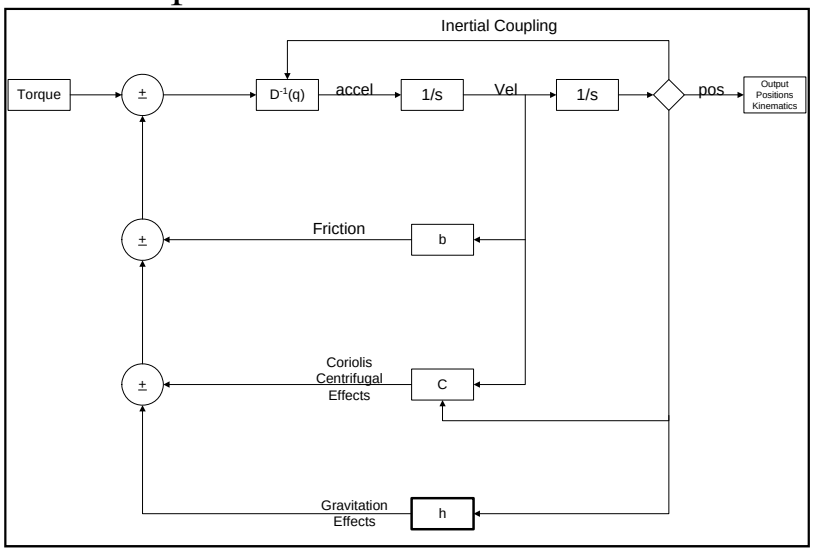
- We will then isolate Acceleration terms (acceleration is the 1st derivative of velocity)

$$a = \dot{v} = \ddot{q} = D_i^{-1}(q)\,(\tau_i - C_i(q, \dot{q}_i) - h(q) - b(\dot{q}_1))$$

Considering Control:

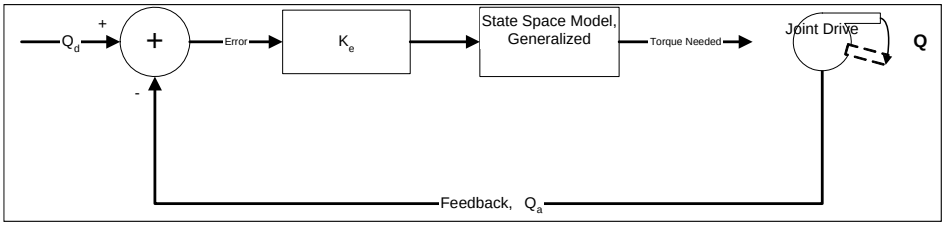
- Each Link's torque is influenced by each other links motion
  - We say that the links are highly coupled
- Solution then suggests that control should come from a simultaneous solution of these torques
- We will model the solution as a "State Space" design and try to balance the torque-in with *positional control*-out – the most common way it is done!
  - But we could also use 'force control' to solve the control problem!

The State-Space Control Model:



Setting up a Real Control

- We will (start) by using positional error to drive our torque devices



- This simple model is called a PE (proportional error) controller

PE Controller:

- To a 1st approximation, $\tau = K_m * I$
  - Torque is proportional to motor current

- And the Torque required is a function of 'Inertial' (Acceleration) and 'Friction' (velocity) effects as suggested by our L-E models

$$\tau_m \simeq J_{eq}\ddot{q} + F_{eq}\dot{q}$$

→ Which can be approximated as:

$$K_m I_m = J_{eq}\ddot{q} + F_{eq}\dot{q}$$

Setting up a "Control Law"

- We will use the positional error (as drawn in the state model) to develop our torque control
- We say then for PE control:

$$\tau \propto k_{pe}(\theta_d - \theta_a)$$

- Here, k_{pe} is a "gain" term that guarantees sufficient current will be generated to develop appropriate torque based on observed positional error

Using this Control Type:

- It is a representation of the physical system of a mass on a spring!
- We say after setting our target as a 'zero goal' that:

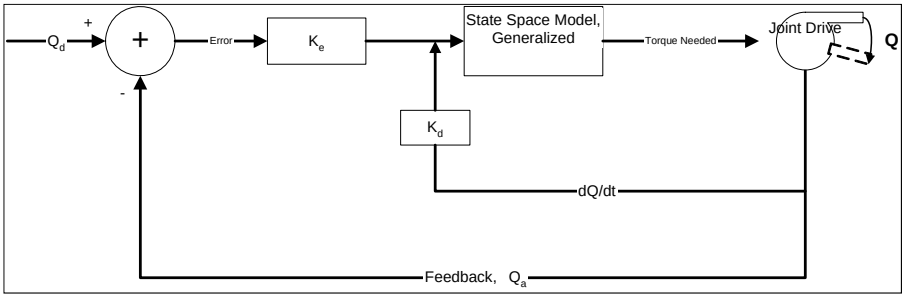
$$-k_{pe} * \theta_a = J\ddot{\theta} + F\dot{\theta}$$

the solution of which is:

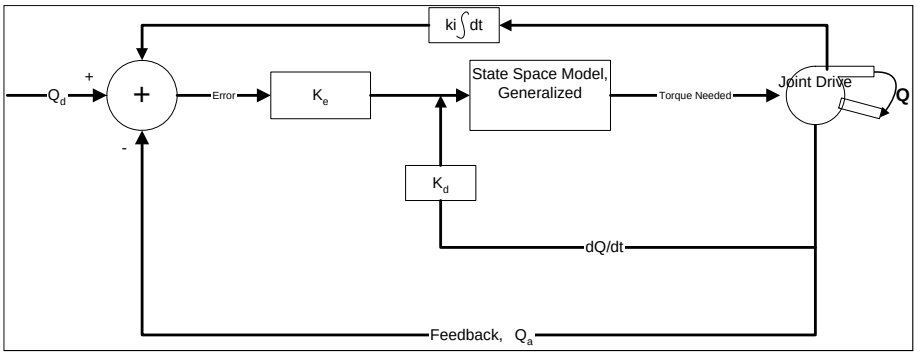
θ_a is a function of the servo feedback as a function of time!

$$\theta_a = e^{-(F/2J)t}\left[C_1 e^{(1/2)\omega t} + C_2 e^{-(1/2)\omega t}\right]$$

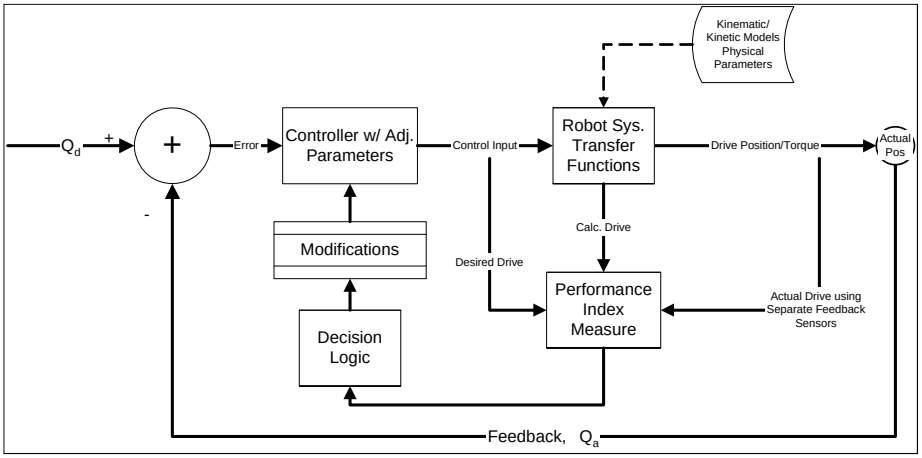
State Space Model of PD:



PID State Space Model:



State Model of Adjustable Controller



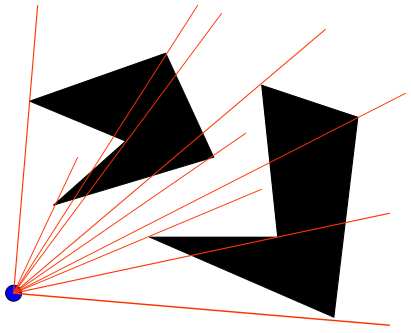
From Last Week: Motion Planning

Path-Planning Approaches

- Roadmap
  Represent the connectivity of the free space by a network of 1-D curves
- Cell decomposition
  Decompose the free space into simple cells and represent the connectivity of the free space by the adjacency graph of these cells
- Potential field
  Define a function over the free space that has a global minimum at the goal configuration and follow its steepest descent

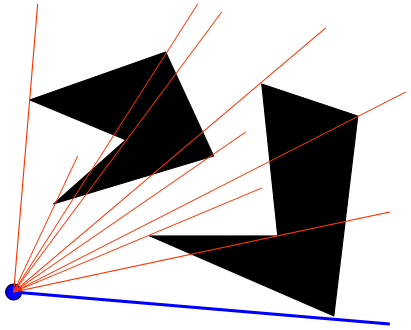
Slide from Latombe, CS326A

I. Rotational Sweep



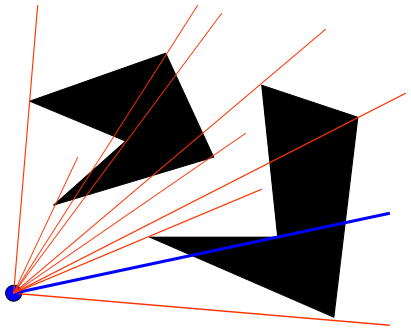
Slide from Latombe, CS326A

Rotational Sweep



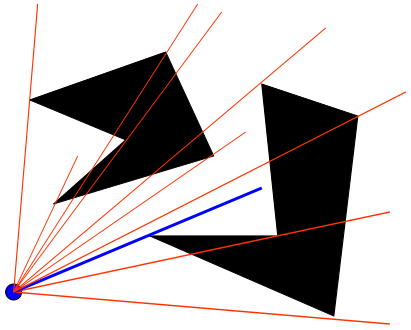
Slide from Latombe, CS326A

Rotational Sweep



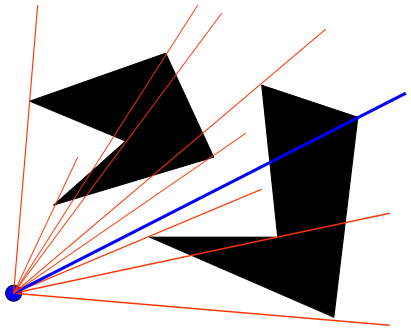
Slide from Latombe, CS326A

Rotational Sweep



Slide from Latombe, CS326A

Rotational Sweep



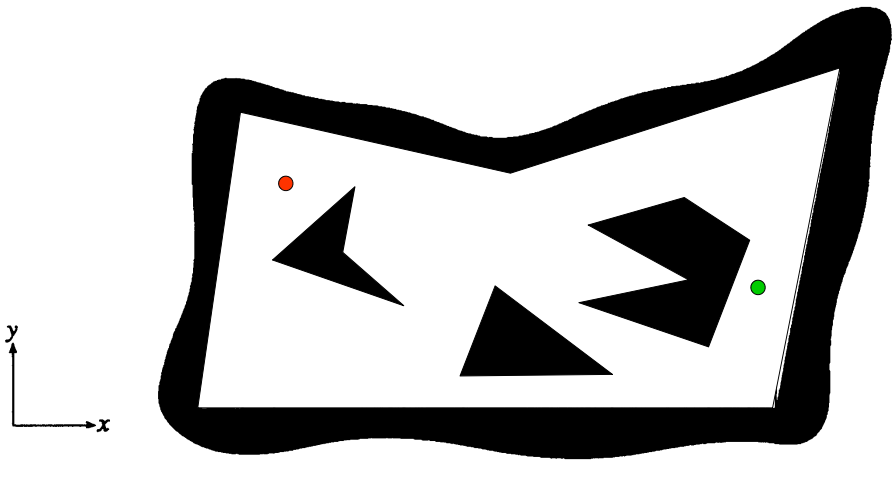
Slide from Latombe, CS326A

II. Cell-Decomposition Methods

Two classes of methods:

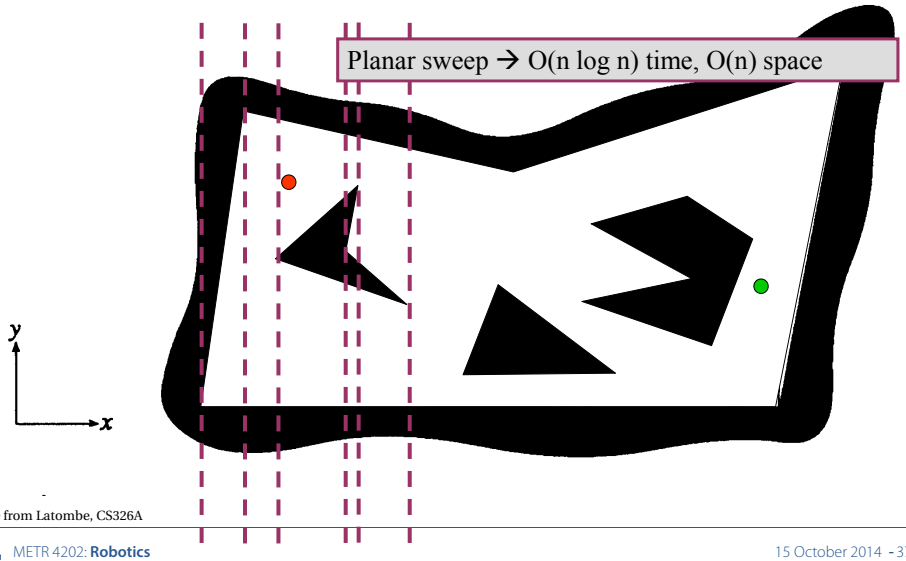
- Exact cell decomposition
  - The free space **F** is represented by a collection of non-overlapping cells whose union is exactly **F**
  - Example: trapezoidal decomposition
- Approximate cell decomposition
  - **F** is represented by a collection of non-overlapping cells whose union is contained in **F**
    Examples: quadtree, octree, 2n-tree

Trapezoidal decomposition



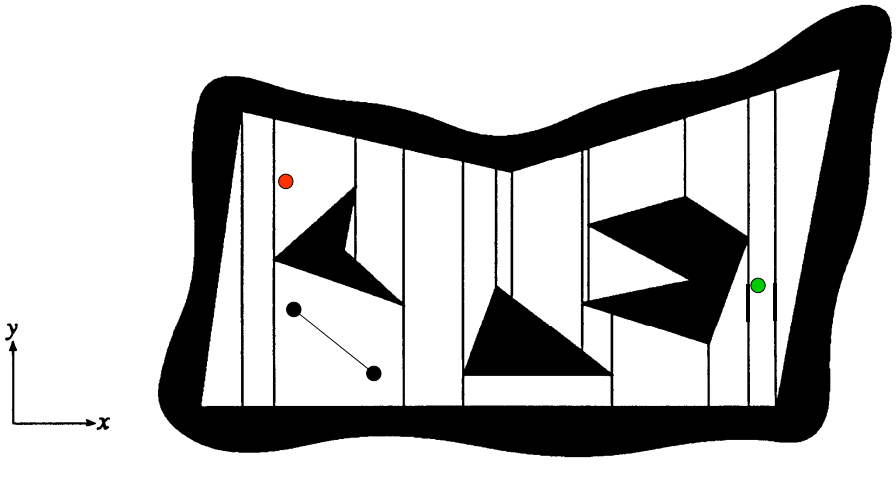
Slide from Latombe, CS326A

Trapezoidal decomposition



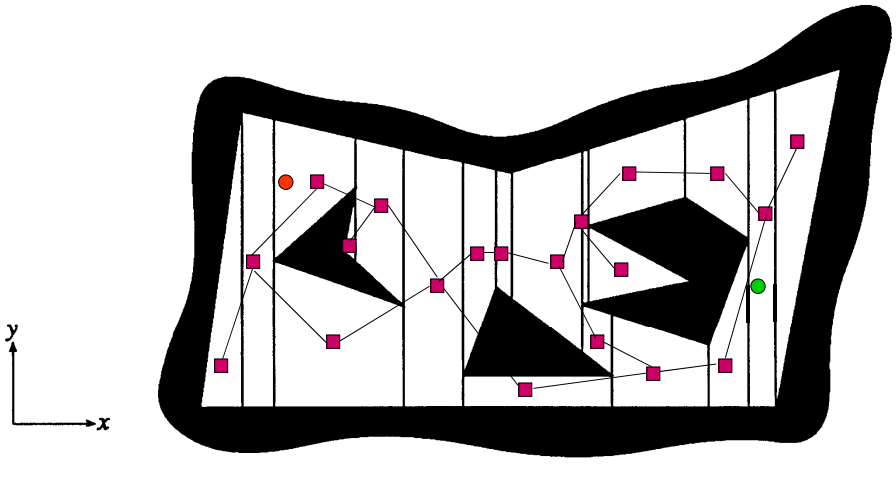
Slide from Latombe, CS326A

Trapezoidal decomposition



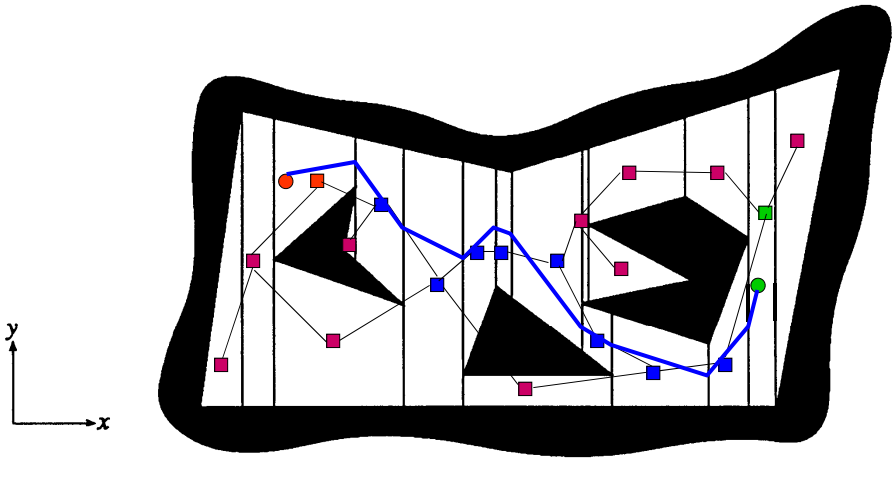
Slide from Latombe, CS326A

Trapezoidal decomposition



Slide from Latombe, CS326A

Trapezoidal decomposition



Slide from Latombe, CS326A

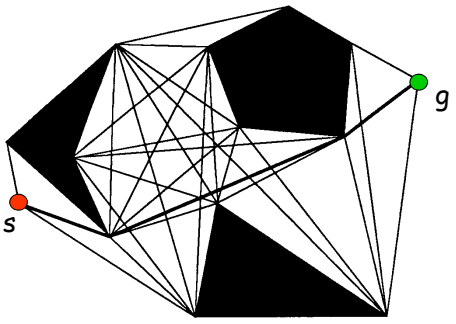
III. Roadmap Methods

- **Visibility graph**
- **Voronoi diagram**
- Silhouette
  First complete general method that applies to spaces of any dimension and is singly exponential in # of dimensions [Canny, 87]
- **Probabilistic roadmaps (PRMS) and Rapidly-exploring Randomized Trees (RRTs)**

Slide from Latombe, CS326A

Roadmap Methods

- **Visibility graph**
  Introduced in the Shakey project at SRI in the late 60s.
  Can produce shortest paths in 2-D configuration spaces

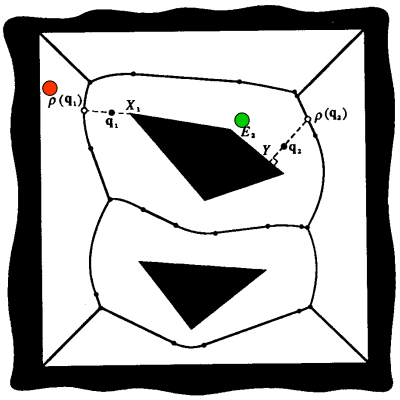


Slide from Latombe, CS326A

Roadmap Methods

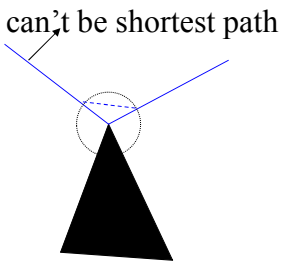
- Voronoi diagram Introduced by Computational Geometry researchers. Generate paths that maximizes clearance.

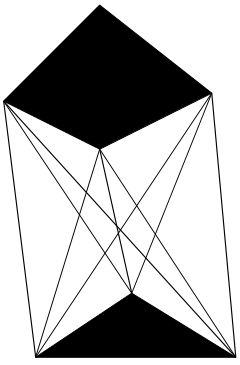
  O(n log n) time
  O(n) space

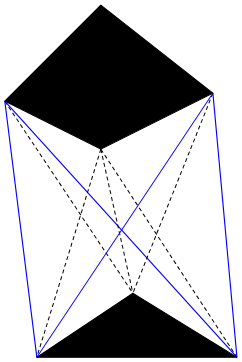


Slide from Latombe, CS326A

II. Visibility Graph





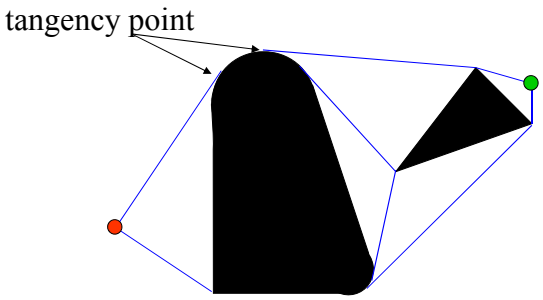


tangent segments

→ Eliminate concave obstacle vertices

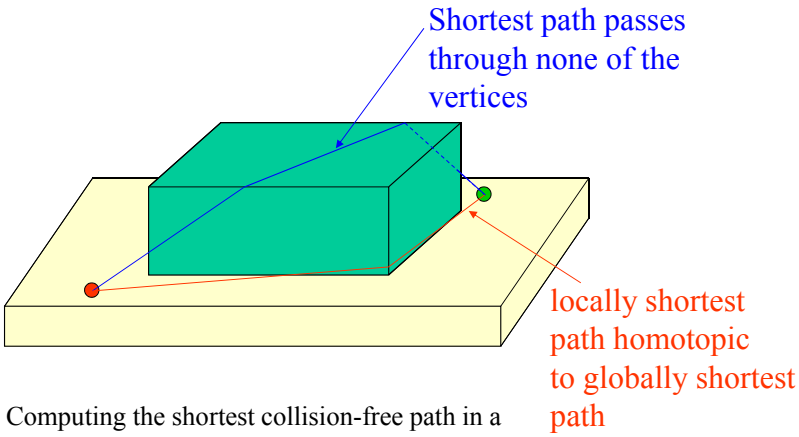
Slide from Latombe, CS326A

Generalized (Reduced) -- Visibility Graph



Slide from Latombe, CS326A

Three-Dimensional Space

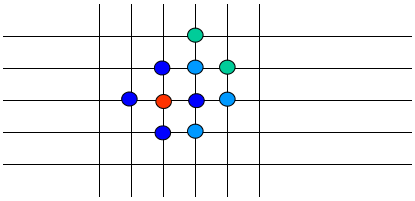


Computing the shortest collision-free path in a polyhedral space is NP-hard

Slide from Latombe, CS326A

Sketch of Grid Algorithm (with best-first search)

- Place regular grid G over space
- Search G using best-first search algorithm with potential as heuristic function



Slide from Latombe, CS326A

Simple Algorithm (for Visibility Graphs)

- Install all obstacles vertices in VG, plus the start and goal positions
- For every pair of nodes u, v in VG

```
If segment(u,v) is an obstacle edge then
        insert (u,v) into VG
else
for every obstacle edge e
        if segment(u,v) intersects e
                then go up to segment
insert (u,v) into VG
```

- Search VG using A*

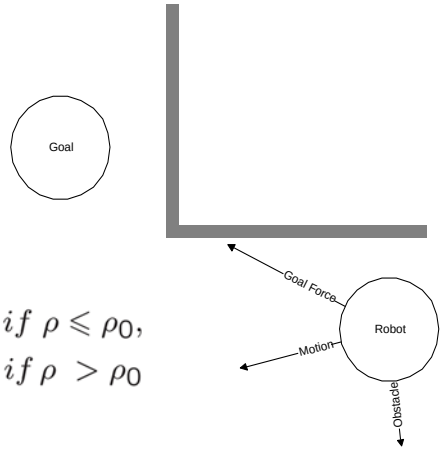
Slide based on Latombe, CS326A

IV. Potential Field Methods

- Approach initially proposed for real-time collision avoidance [Khatib, 86]

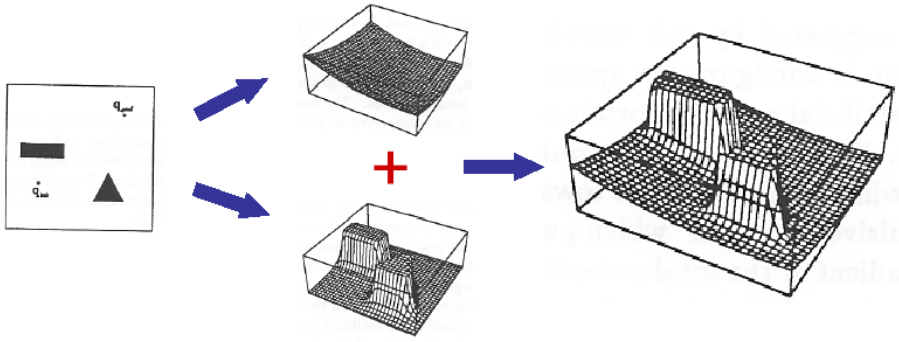
$$F_{Goal} = -k_p(x - x_{Goal})$$

$$F_{Obstacle} = \begin{cases} \eta \left(\frac{1}{\rho} - \frac{1}{\rho_0}\right) \frac{1}{\rho^2} \frac{\partial \rho}{\partial x} & if\ \rho \leqslant \rho_0, \\ 0 & if\ \rho > \rho_0 \end{cases}$$



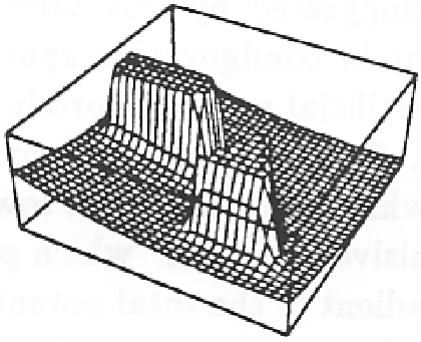
Slide based on Latombe, CS326A

Attractive and Repulsive fields



Slide from Latombe, CS326A

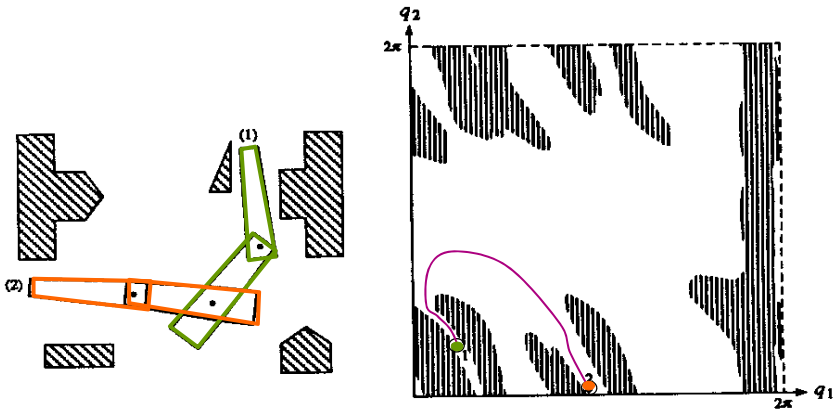
Local-Minimum Issue



- Perform best-first search (possibility of combining with approximate cell decomposition)
- Alternate descents and random walks
- Use local-minimum-free potential (navigation function)

Slide from Latombe, CS326A

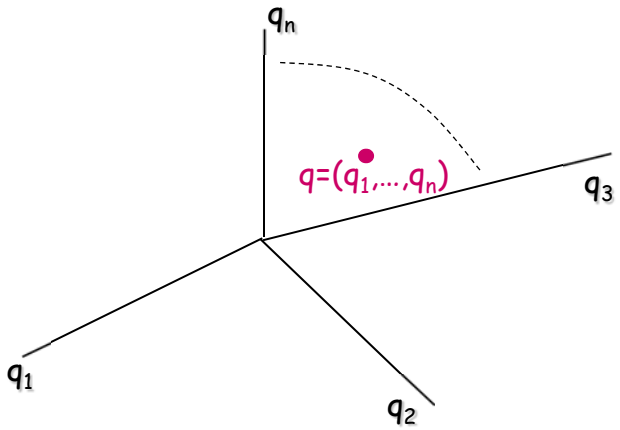
Configuration Space



- A robot configuration is a specification of the positions of all robot points relative to a fixed coordinate system
- Usually a configuration is expressed as a "vector" of position/orientation parameters

Slide from Latombe, CS326A

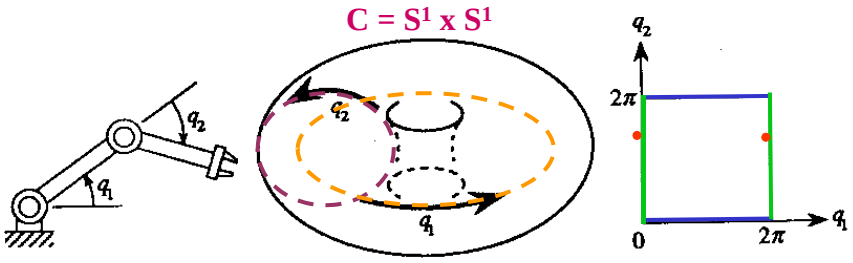
Motion Planning in C-Space



Slide from Latombe, CS326A

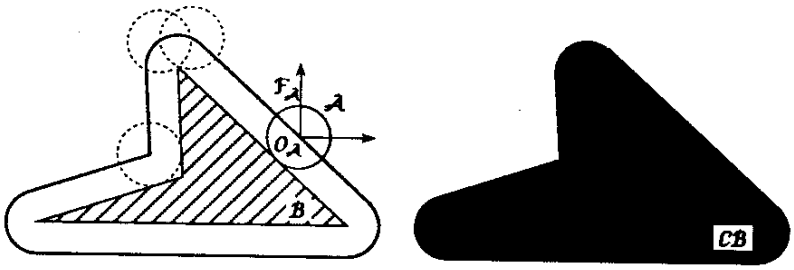
Configuration Space of a Robot

- Space of all its possible configurations
- But the topology of this space is usually not that of a Cartesian space



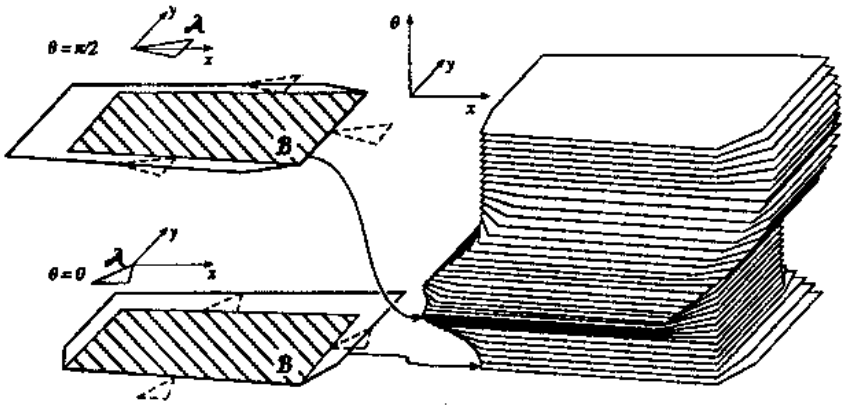
Slide from Latombe, CS326A

Disc Robot in 2-D Workspace



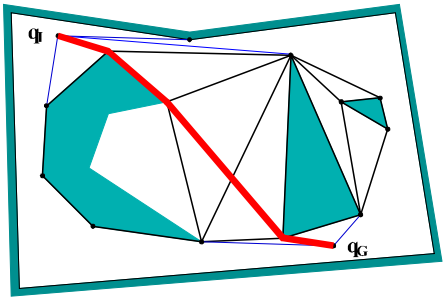
Slide from Latombe, CS326A

Rigid Robot Translating and Rotating in 2-D



Slide from Latombe, CS326A

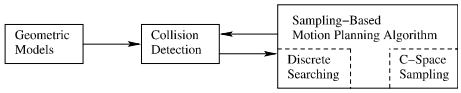
Geometric Planning Methods



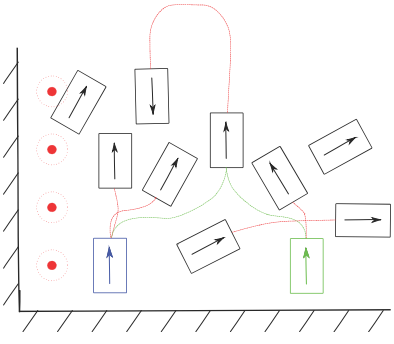
- Several Geometric Methods:
  - Vertical (Trapezoidal) Cell Decomposition
  - **Roadmap Methods**
    - **Cell (Triangular) Decomposition**
    - Visibility Graphs
    - Veroni Graphs

Artwork from LaValle, Ch. 6

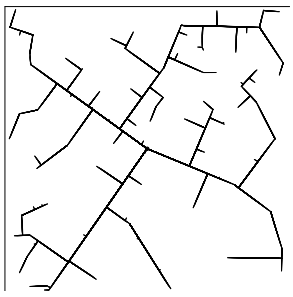
Sample-Based Motion Planning



- PRMs

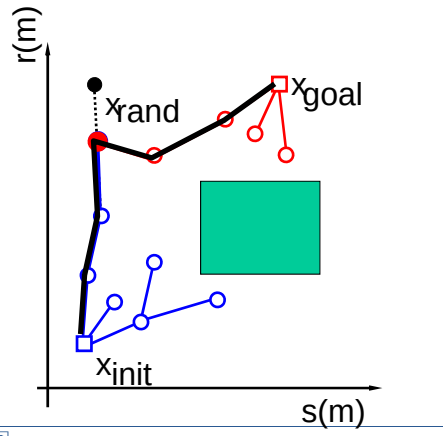


- RRTs

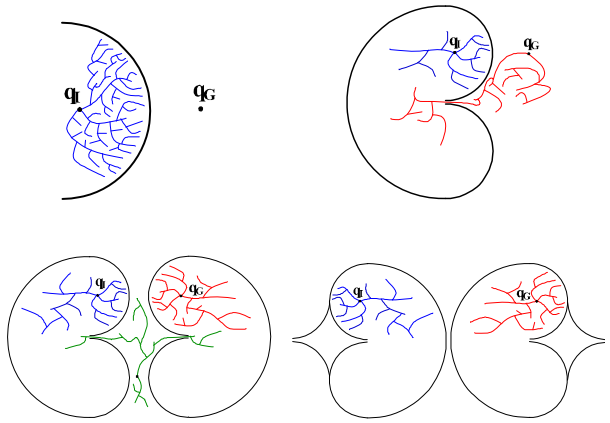


Artwork based on LaValle, Ch. 5

Rapidly Exploring Random Trees (RRT)

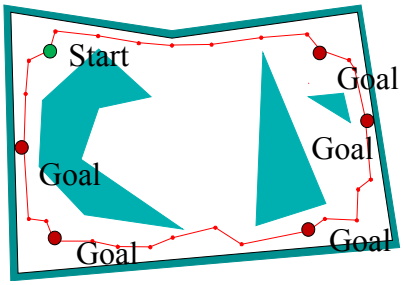


Sampling and the "Bug Trap" Problem



Artwork based on LaValle, Ch. 5

Multiple Points & Sequencing



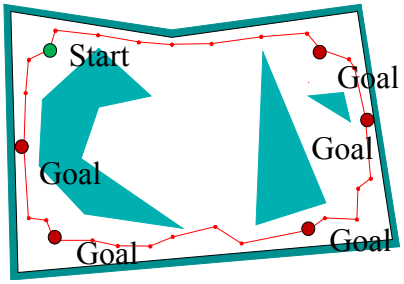
- Sequencing
  - Determining the "best" order to go in

➔ Travelling Salesman Problem

A salesman has to visit each city on a given list exactly once. In doing this, he **starts** from his home city and in the **end he has to return to his home** city. It is plausible for him to select the order in which he visits the cities so that the **total of the distances travelled** in his tour is as small as possible.

Artwork based on LaValle, Ch. 6

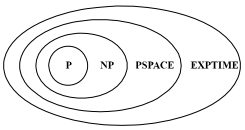
Travelling Salesman Problem



- Given a $n \times n$ distance matrix $\mathbf{C}=(c_{ij})$
- Minimize:

$$c(\pi) = \sum_{i=1}^{n} c_{i\pi(i)}$$

- Note that this problem is NP-Hard

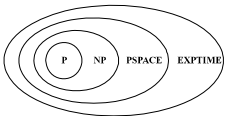


➔ BUT, Special Cases are Well-Solvable!

Artwork based on LaValle, Ch. 6

Travelling Salesman Problem [2]

- This problem is NP-Hard



➔ BUT,
Special Cases are
Well-Solvable!

For the Euclidean case
(where the points are on the 2D Euclidean plane) :

- The shortest TSP tour does not intersect itself, and thus geometry makes the problem somewhat easier.
- If all cities lie on the boundary of a convex polygon, the optimal tour is a cyclic walk along the boundary of the polygon (in clockwise or counterclockwise direction).

***The k*-line TSP**

- The a special case where the cities lie on *k* parallel (or almost parallel) lines in the Euclidean plane.
- EG: Fabrication of printed circuit boards
- Solvable in $\mathbf{O}(n^3)$ time by Dynamic Programming (Rote's algorithm)

The necklace TSP

- The special Euclidean TSP case where there exist *n* circles around the *n* cities such that every cycle intersects exactly two adjacent circles