CS 4649/7649
Robot Intelligence: Planning

PRM/RRT, Motion Planning Summary

Sungmoon Joo

School of Interactive Computing
College of Computing
Georgia Institute of Technology

*Slides based in part on Dr. Mike Stilman and Dr. Howie Choset's lecture slides

Administrative – HW#2

- HW#2
  - due Nov. 10
  - similar protocol as HW#1 – use Wiki for grouping
  - **Deliverables:**
    (i) A PDF summary
    (ii) A repository(git, dropbox, etc.): Contains the relevant files (summary, source code, movies, README, etc.)
    * Email the PDF summary and the link to your repository.
  - **Participation:** Include a page in your summary describing what each group member did to participate in the project, in detail.
  - **Printing:** On Nov. 11, bring a printout of your summary to the class.

Administrative – Final Project

- CS7649
- project topic decision, grouping: Due Oct. 23 → update Wiki (group, description)
- project proposal: Due Oct. 30, 2-3page (motivation, technical gap, approach, expected result)
- project final report: Due Dec. 4, 23:59pm, conference-style paper (format is on the course web)
- project presentation: Dec. 11, 11:30am - 2:20pm

*there may be meetings between project teams and the instructor to see if projects are progressing as scheduled.

- CS4649
- project reviewer assignment: Oct. 28
- proposal review report: Due Nov. 6
- project review report(for the assigned project): Due Dec. 11, 11:30am
- project presentation review*(for all presentation): Due Dec. 11, 2:20pm

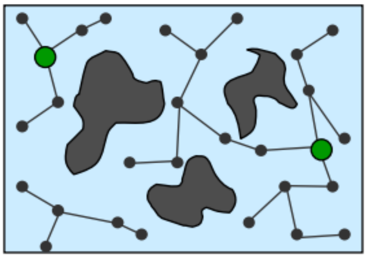
*presentation review sheets will be provided

Probabilistic Roadmap

- Roadmap is a graph G(V,E) where a robot configuration $q \in Q_{free}$ is a vertex $\in$ V, edge $(q_1, q_2) \in$ E implies collision-free path between these configurations
- Create a roadmap once (for static environment)
- Learning the map - Construction and Expansion
  - Initially empty graph G
  - A configuration q is randomly chosen, if $q \in Q_{free}$, then added to G
  - Repeat until N vertices chosen
  - For each q, select k closest neighbors
  - Local planner connects q to its neighbors
  - If connect is successful (exists a collision free local path), add edge(q,q') to G
  - If there are disconnected 'roadmaps', expand locally to connect them

PRM

- Query - Finding a path
  - Given q_{init} and q_{goal}
  - Find k nearest neighbors of q_{init} and q_{goal} in the roadmap, and plan local paths from q_{init} and q_{goal} to the roadmap, respectively
  - Find connections from q_{init} to q_{goal}
  - Once we have a roadmap, search !



PRM samples the entire space!

Spreads out like uniformity but need lots of sample to cover space(Multy-query)

PRM: Challenges

1. **Finding & Connecting neighboring points**
   - Only easy for holonomic systems (e.g. linked manipulators) → why?
   (i.e., for which you can move each degree of freedom at will at any time).
   - Typically solved w/o collision checking; later verified if valid by collision checking
2. **Collision checking**
   - Often takes majority of time in applications
3. **Sampling**
   - How to sample uniformly (or biased according to prior information) over configuration space?
4. **Local Planner**
   - How to generate local path? – incremental, …

***distance metric – Euclidean, …   *post processing – shortening, smoothing**

Making PRM Efficient

- Two procedures need to be extremely efficient:
  - Find Nearest Neighbor
    → Identifies goals for local planner
  - Collision Detection
    → Check if a sampled configuration is in free space
    → Validate local plan

PRM: Analysis

- **Sound**
  Yes

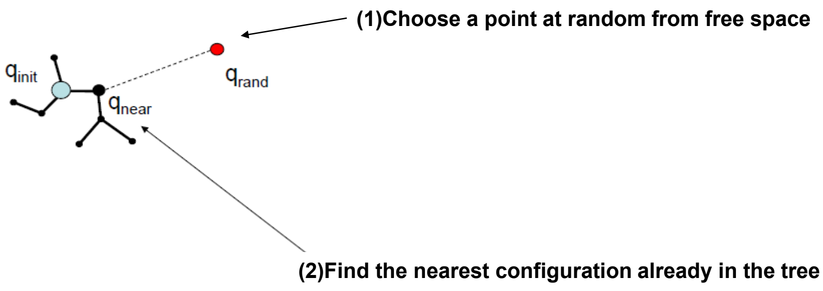
- **Complete**
  No
  Probabilistically Complete
  – The probability of success increases exponentially with the number of samples generated.

Completeness

- Completeness
  - Complete planner: always answers a path planning query correctly in bounded time
  - Probabilistic complete planner: if a solution exists, planner will 'eventually' find it, using random sampling (e.g. Monte-Carlo sampling)
  - Resolution complete planner: similar concept as PCP but based on a deterministic sampling (e.g. sampling on a fixed grid), and the 'resolution' of the grid matters while the number of samples matters in PCP

Rapidly-Exploring Random Trees (RRT)

- Planning is search, and search happens over a search tree
- RRT defines a simple rule for growing high quality trees
- Slightly different than random sampling idea in generic PRM

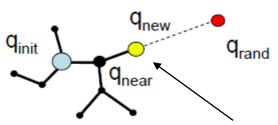


[LaVallе '98, LaValle & Kuffner '00]

Rapidly-Exploring Random Trees (RRT)

- Planning is search, and search happens over a search tree
- RRT defines a simple rule for growing high quality trees
- Slightly different than random sampling idea in generic PRM

EXTEND(T, q_{rand})



(3)Extend the tree in the direction of the new configuration

BUILD_RRT (q_{init}) {
T.init(q_{init});
for k = 1 to K do
q_{rand} = RANDOM_CONFIG();
EXTEND(T, q_{rand})
}

Extend returns
1. **Trapped, can't make it**
2. **Extended, steps toward q_{rand}**
3. **Reached, connects to q_{rand}**

RRT: Naïve Implementation

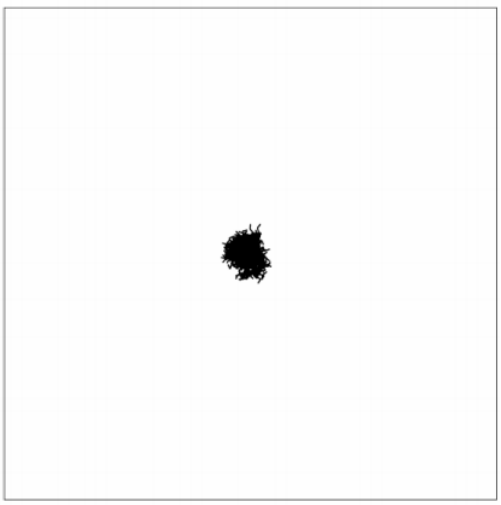
Start with middle

Sample near this node

Then pick a node at random in tree

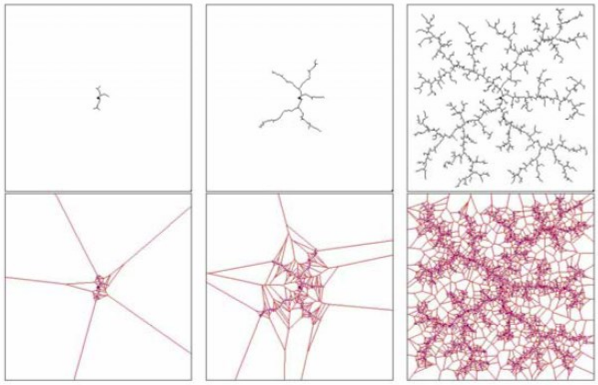
Sample near it

End up Staying in middle



RRT: Voronoi Bias

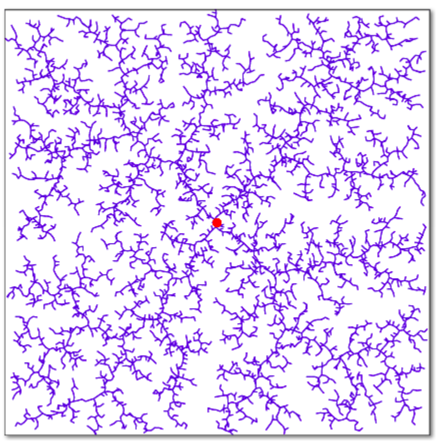
Monte-Carlo way of biasing search into largest Voronoi regions



The probability that a path is found increases exponentially with the number of iterations.

[Kuffner & LaValle '00]

RRT



http://msl.cs.uiuc.edu/rrt/gallery_2drrt.html

RRT

- A data structure and algorithm that is designed for efficiently searching nonconvex high-dimensional spaces.
- RRTs are constructed incrementally in a way that quickly reduces the expected distance of a randomly-chosen point to the tree.
- RRTs are particularly suited for path planning problems that involve obstacles and constraints (nonholonomic or kinodynamic).

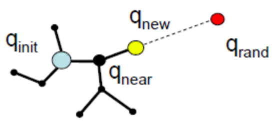
Merging Trees: Bidirectional

- 2 trees: T_{init} rooted at q_{init} and T_{goal} rooted at q_{goal}
- Each tree is expanded by
  - q_{rand} is generated from uniform distribution
  - q_{near} is found, nearest tree node to q_{rand}
  - move by a step-size along line (q_{near}, q_{rand}) to q_{new}. If no collision, add q_{new} to tree
- If trees merge, path is found

[Kuffner & LaValle ‘99]

RRT Connect

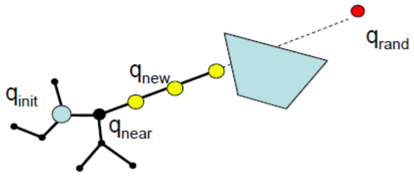
RRT algorithm is sensitive to step-size
- How far do we move along line (q_{near}, q_{rand})?
- Can a greedier algorithm work better?
- Why not move all the way to q_{rand}?



[LaValle '98, LaValle & Kuffner '01]

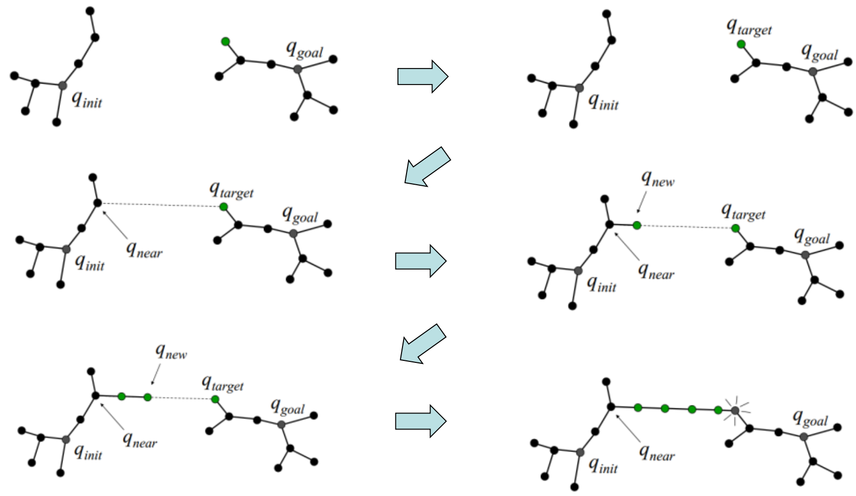
RRT Connect

RRT algorithm is sensitive to step-size
- How far do we move along line (q_{near}, q_{rand})?
- Can a greedier algorithm work better?
- Why not move all the way to q_{rand}?



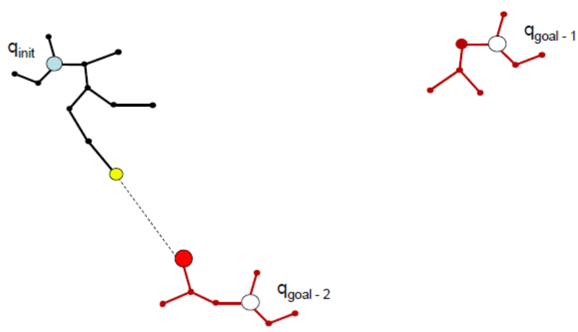
[LaValle '98, LaValle & Kuffner '01]

Merge Two Trees with RRT Connect



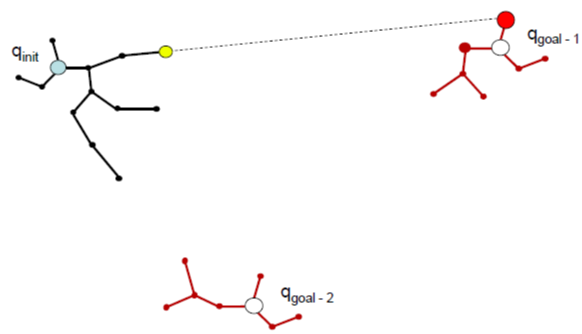
[LaValle '98, LaValle & Kuffner '01]

Multi-Tree RRT Connect



[Hirano et. al. '05]

Multi-Tree RRT Connect



[Hirano et. al. '05]

RRT shaping

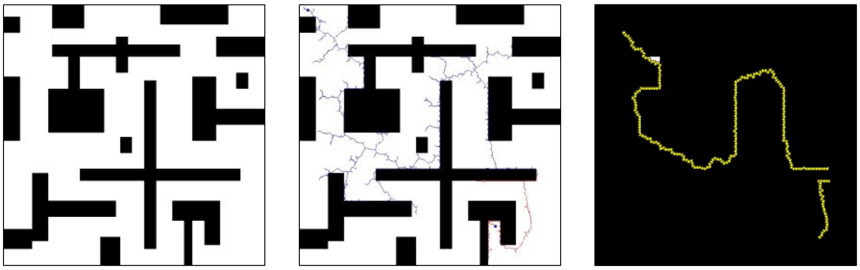
- If step-size is small, many nodes are generated, close together
- As number of nodes increases, nearest neighbor search slows down
  → Maybe better to only add the last sample along the line (q_{near}, q_{rand})?

- q_{rand} determines what direction we go
- What if $q_{rand} = q_{goal}$?
  → Very greedy algorithm (too much bias), Get stuck in local minima
  → Maybe use uniform q_{rand} with occasional(how often?) $q_{rand} = q_{goal}$?

* Bias toward goal
– When generating a random sample, with some probability pick the goal instead of a random node when expanding
– This introduces another parameter
– 5-10% is the right choice
– If you do this 100%, then you may easily get stuck in local minima

RRT-based Planning in Action

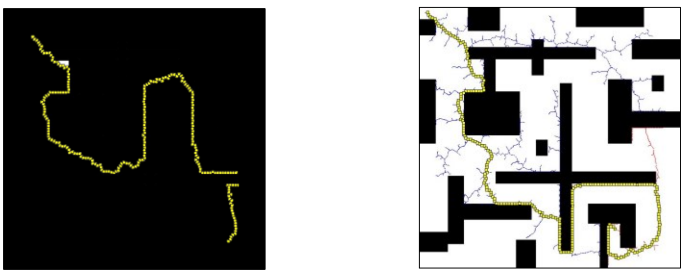
2D Maze: Point Robot



http://msl.cs.uiuc.edu/rrt

RRT-based Planning in Action

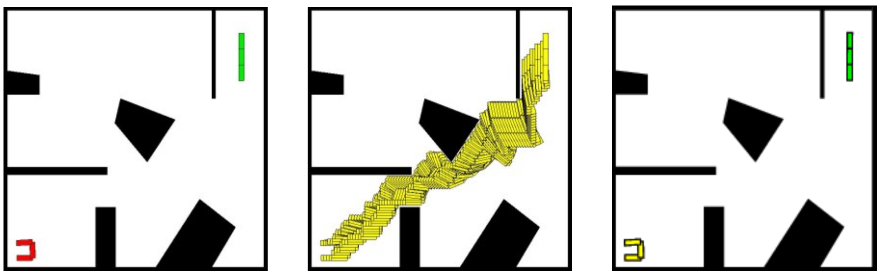
Randomness



http://msl.cs.uiuc.edu/rrt

RRT-based Planning in Action

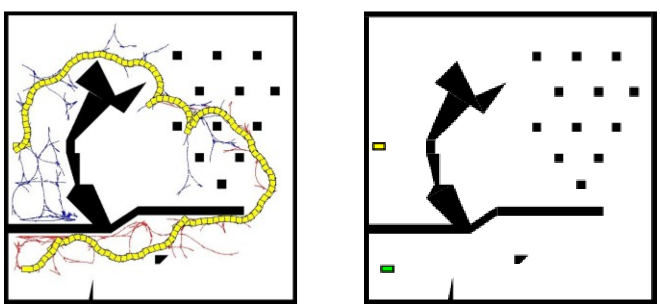
Articulated Linkage



http://msl.cs.uiuc.edu/rrt

RRT-based Planning in Action

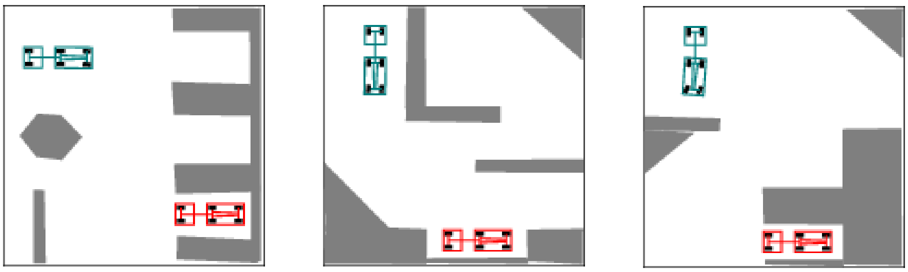
Car-like Robot



http://msl.cs.uiuc.edu/rrt

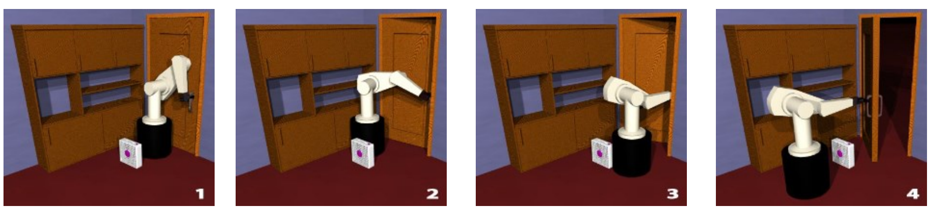
RRT-based Planning in Action

Trailer Parking

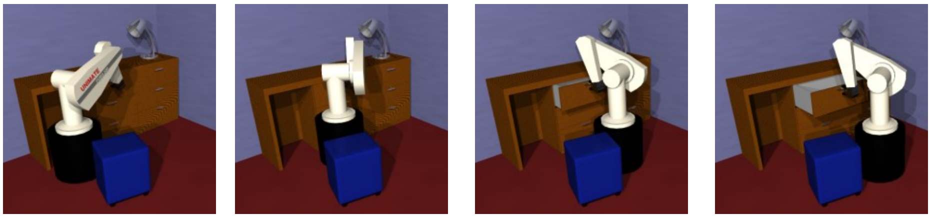


http://msl.cs.uiuc.edu/rrt

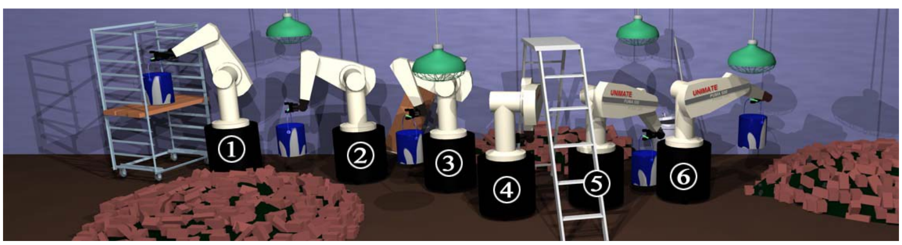
Task Constraints



Task Constraints

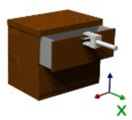


Task Constraints



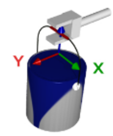
Task Constraints

Workspace



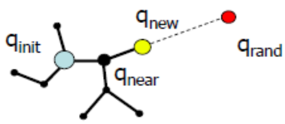
(a) Fixed C = [0 1 1 1 1 1]^T

(b) Fixed C = [1 1 1 1 1 0]^T



(c) Para. C = [0 0 0 1 1 0]^T

Joint space



Probability of Satisfying Task Constraints ~ 0

Task Constraints

- **Projection methods:**
  - **Exact Task Constraints ✓**

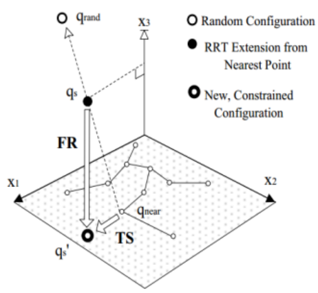
    [Stilman 2007, 2010]
  - **Hard Task Constraints ✓**

    [Berenson et al. 2009]
- **Tangent-space sampling**

  [Um et al. 2010]
- **Piecewise approximation of constraint manifold**

  [Porta et al. 2011]
- **Soft Constraint ✓**

  [Kunz & Stilman 2012]



Task Constraints: First-Order Retraction

Stilman '07,'10

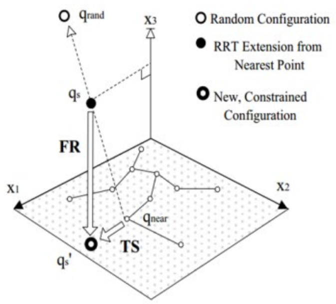
Translation or rotation can only be fixed completely within

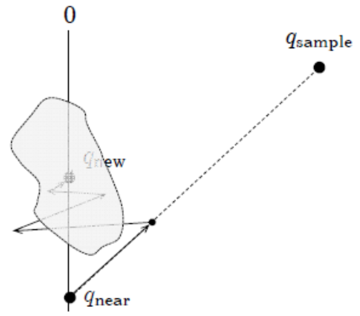
$$|\triangle x_{error}| < \epsilon$$

$$\Delta x_{err}(q) = C\Delta x(q)$$

$$\Delta q_{\text{err}} = J^{\dagger}_{\Delta x}\ \Delta x_{\text{err}}$$

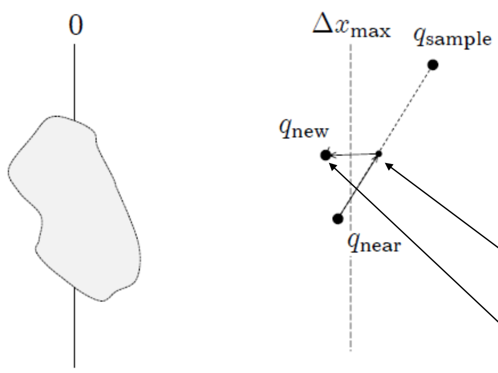
$$q'_s = q_s - \Delta q_{\text{err}}$$





Task Constraints: Hard Constraints

Berenson '09

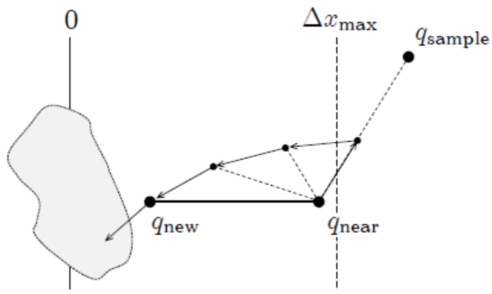


$$\Delta x_{\text{min}} \leq \Delta x_{err}(q) \leq \Delta x_{\text{max}}$$

- Constraints are allow for an interval of values
- All configurations satisfying the constraints are equally good with no bias toward the center of the constraint, rather bias toward the boundary of the constraint
- Infeasible samples are projected toward the nearest constraint boundary.

Task Constraints: Soft Constraints

- Soft Task constraints
  - Allow range of values
  - Biased toward a preferred value



$$q'_s = q_s - \min\{\delta, ||\Delta q_{\text{err}}||\}\frac{\Delta q_{\text{err}}}{||\Delta q_{\text{err}}||}$$

$q_{new} \leftarrow q'_s$

Repeat until

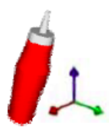
- **Collision**
- **Reached joint limits**
- **No progress toward preferred value**
- **Reached preferred value**

Accept q_{new} **if**

- **Progress toward preferred value**
- **Progress toward sample**

Constraint Representation

Selection matrix (Stilman)

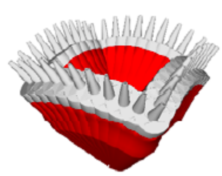


$$\Delta x = \begin{bmatrix} t^t_{obj} \\ \psi \\ \theta \\ \phi \end{bmatrix}$$

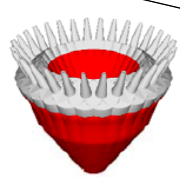
$$\Delta x_{err}(q) = C\Delta x(q)$$

$$\Delta x_{\min} \le \Delta x_{err}(q) \le \Delta x_{\max}$$

Error range (Berenson)

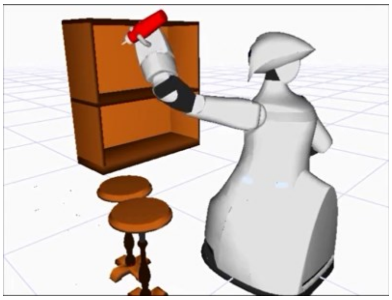


X-Y-Z Euler angles [Berenson et al. 2009]

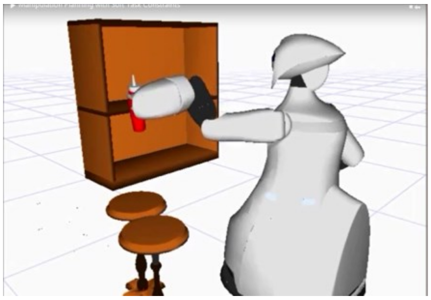


Z-Y-Z Euler angles

Soft Task Constraints



No constraint



Soft task constraint

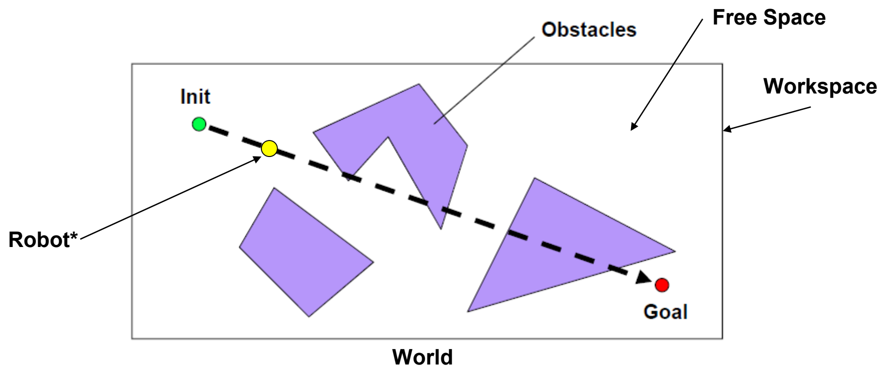
[Kunz et al. '12]

Motion Planning Summary

- Motion planning is the ability for a robot to compute its own motions in order to achieve certain goals.
  - (i) To compute 'motion strategies'
    - geometric path
    - time-parameterized trajectories
    - sequence of sensor-based motion commands, …
  - (ii) To achieve high-level goals
    - go to A without colliding with obstacles
    - pick up the mug, …
- Make decisions in continuous space!!
- All (autonomous) robots should eventually have this ability

Basic Path Planning

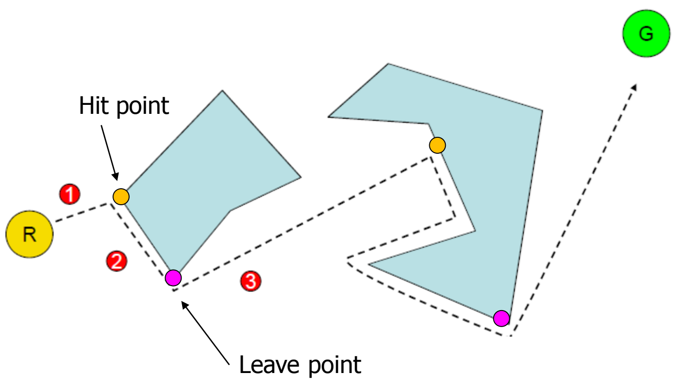
"Compute a continuous sequence of collision-free robot configurations connecting the initial and goal configurations."



***Point robot**

"Bug 0" Algorithm

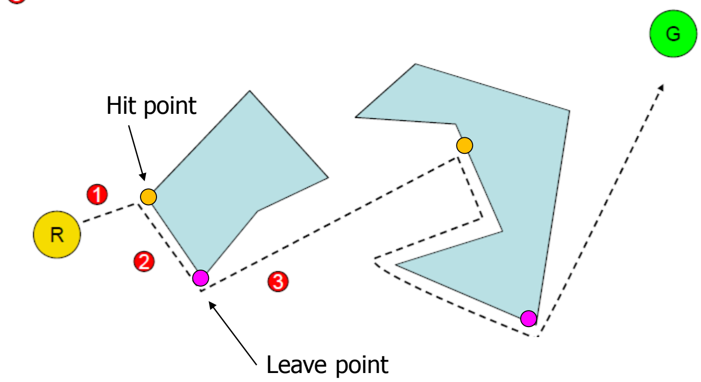
1. Move to Goal
2. Follow obstacle until you can go to goal again
3. Continue



“Bug 0” Algorithm

Not complete!

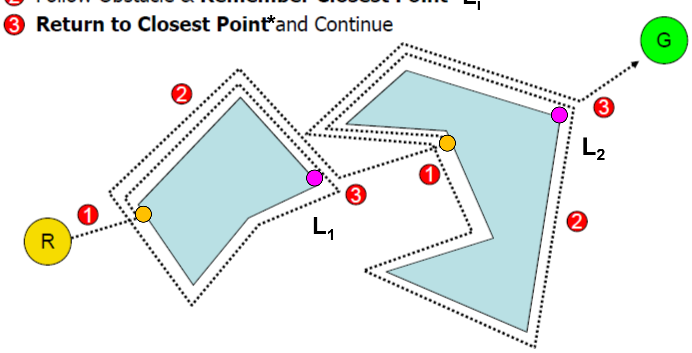
1. Move to Goal
2. Follow obstacle until you can go to goal again
3. Continue



“Bug 1” Algorithm

circumnavigate

1. Move to Goal
2. Follow Obstacle & **Remember Closest Point** L_i
3. **Return to Closest Point*** and Continue

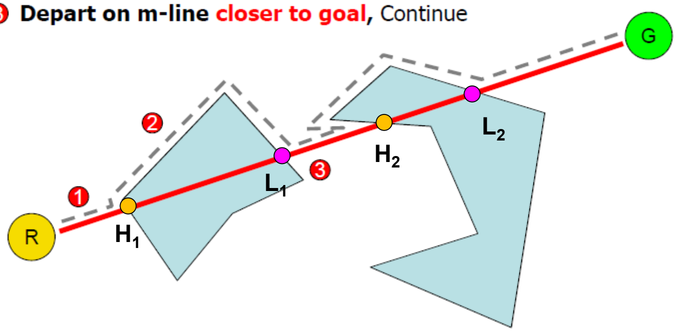


Terminate when goal is found or no progress.

*By following the shortest path along the object boundary

"Bug 2" Algorithm

1. Move to Goal (following m-line)
2. Follow Obstacle to m-line (closer to goal)
3. **Depart on m-line closer to goal**, Continue



Terminate when goal is found or no progress.

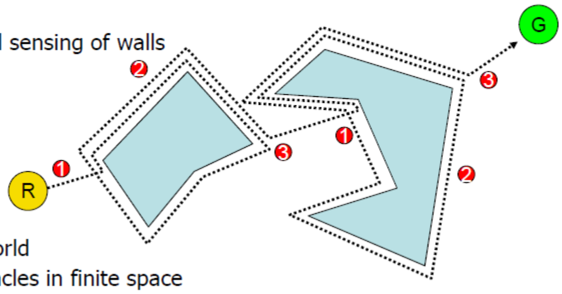
Comparison of Bug 1 & Bug 2

- Bug 1
  - Exhaustive Search
  - Evaluates Choices before choosing
- Bug 2
  - Greedy, Heuristic Algorithm
- Mostly, Bug 2 performs better and Bug 1 is more predictable
- Both are complete, neither is optimal!
- There exist variants for more complex sensors (see Tangent Bug)

Bug Algorithms Summary

local environment knowledge & **global goal**
Robot can tell the distance (smell the goal)

- Otherwise local sensing of walls



- Reasonable World
  - Finite obstacles in finite space
  - Workspace is bounded

Bug Algorithms Summary

Completeness is Desirable

- Completeness does not always require complexity
- Bug Algorithms achieve global completeness with local planning

Optimality is Desirable

- Bug Algorithms are not Optimal

Bug Algorithms Summary

Algorithm	Bug 0	Bug 1	Bug 2
Completeness	X	0, Exhaustive	0, Greedy
Characteristic	-	Safe, Reliable	Better in some cases. But worse in other cases

***None of them is optimal**

Roadmap Approach to Navigation Planning

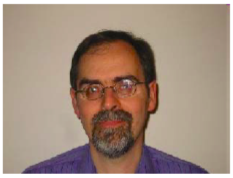
- Assumption 1: Static environment
- Assumption 2: World is known

- General Idea:

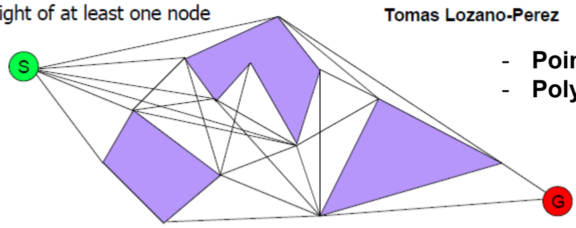
- Avoid searching the entire space
- Pre-compute a (hopefully small) graph (i.e. the roadmap) s.t. staying on the roads is guaranteed to avoid the 'obstacles' (& to take us to the goal)
- Search a path between q_{init} and q_{goal} on the roadmap

Visibility Graphs*

- Early Motion Planning Algorithm
- Nodes share an edge if they are within "line of sight"
- All points in free space are within sight of at least one node



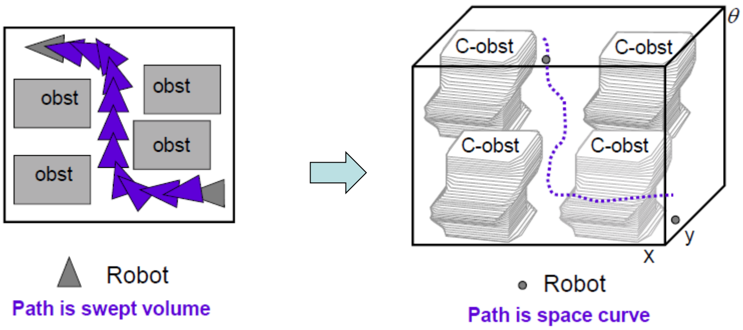
Tomas Lozano-Perez



- Point robot
- Polygonal obstacles

* "An algorithm for planning collision-free paths among polyhedral obstacles" 1979 T. Lozano-Perez & M. A. Wesley
* "A mobile automaton: An application of artificial intelligence techniques" 1969 N.J Nilson

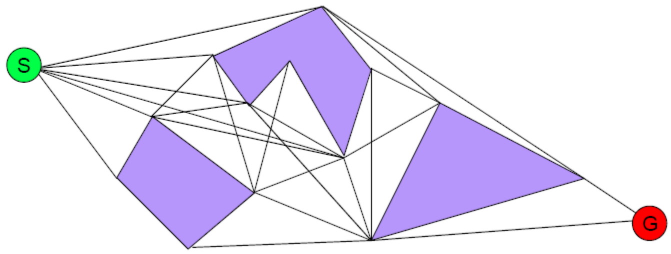
Path Planning for Robots with Geometric Shapes



Step 1: Reduce robot to a point in the configuration space
Step 2: Compute configuration-space obstacles (not a trivial job)
Step 3: Search for a path in the collision-free configuration space

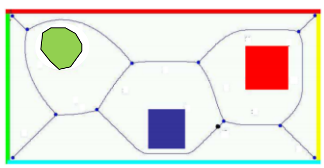
Visibility Graph Analysis

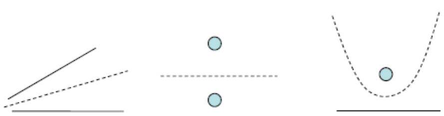
- Visibility Graphs are complete? Yes (Assuming Polygonal Obstacles)
- Visibility Graphs are optimal? Yes   Metric: Distance Traveled



Voronoi Diagrams

- Edges maximally separate domain features
- Nodes are critical points where edges intersect

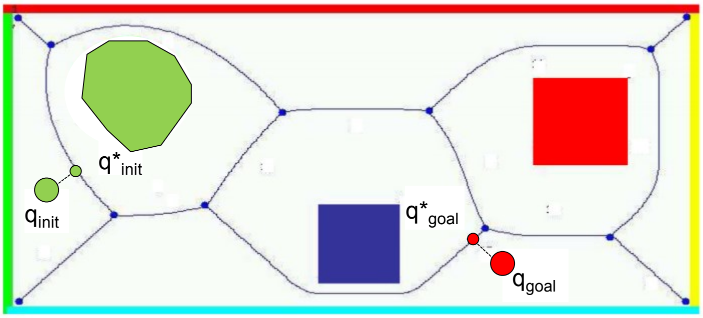




Voronoi Diagram → Navigation Planning

- Idea: Construct a path between q_{init} and q_{goal} by following edges on the Voronoi diagram

Voronoi diagram = Roadmap



Step1. Find the point q^*_{init} of the Voronoi diagram closest to q_{init}

Step2. Find the point q^*_{goal} of the Voronoi diagram closest to q_{goal}

Step3. Compute shortest path from q^*_{init} to q^*_{goal} on the Voronoi diagram

Interim Summary

Roadmap Approach

- Static environment, World is known
- Avoid searching the entire space

- Pre-compute a graph (i.e. roadmap) → Search space reduction

Roadmap approach for Navigation Planning

- Visibility Graph → Short path
- Voronoi Diagram → Safe/Conservative path

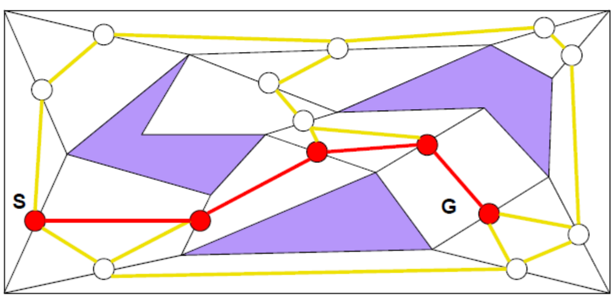
Other Options

- Exact Cell Decomposition
- Approximate Decomposition
- Potential Fields (Not really grids, but relevant)

Exact Cell Decomposition: Convex Polygons

- Collection of non-overlapping cells: Union(Cells) = Free Space

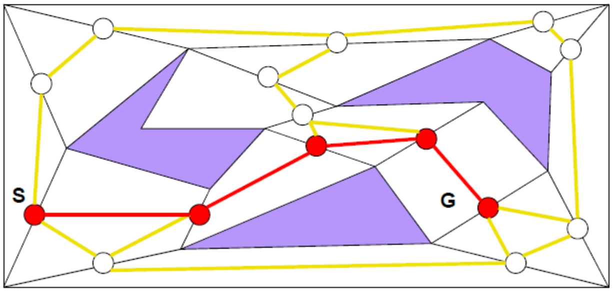
"The graph of midpoints of edges between adjacent cells defines a roadmap"



Exact Cell Decomposition: Convex Polygons

- Collection of non-overlapping cells: Union(Cells) = Free Space

"The graph of midpoints of edges between adjacent cells defines a roadmap"

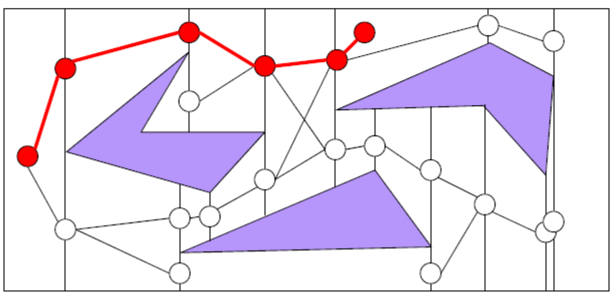


Exact Cell Decomposition: Trapezoidal

- Collection of non-overlapping cells: Union(Cells) = Free Space

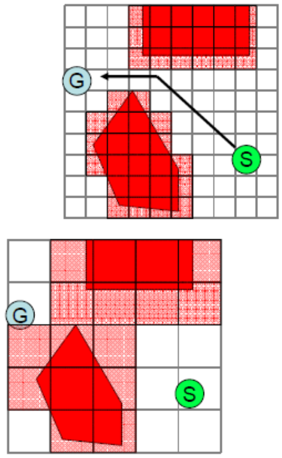
Extend a bi-directional vertical line from each vertex until collision

- This **is** a convex polygonal decomposition
- Again a graph search



Approximate Cell Decomposition

- Use grid
- Is it complete?
  - Yes, up to grid size
  - **Resolution Complete**
- Is it optimal?
  - Metric = # grid cells traversed
  - Search Method = A*
  - Yes IF **heuristic is admissible**



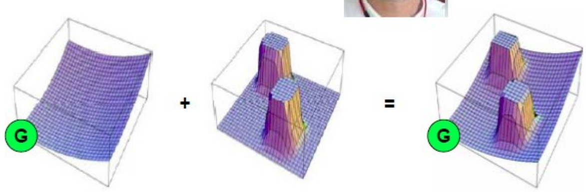
Potential Fields

Avoid search

- Potential Function
  - $U_a(q)$ Attracts to goal
  - $U_r(q)$ Repels from obstacles

Oussama Khatib

‘86



$U_a(q)$ + $U_r(q)$ = $U(q)$

– Typically smooth **Because we follow gradient: $\nabla U(q)$**

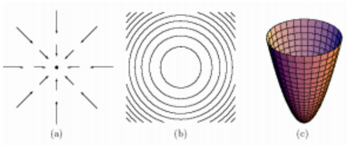
Potential Fields

Conical Potential

$U(q) = \zeta d(q, q_{\text{goal}}).$

$\nabla U(q) = \frac{\zeta}{d(q,q_{\text{goal}})}(q - q_{\text{goal}}).$

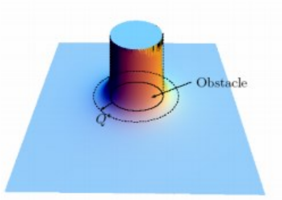
Quadratic Potential



$$U_{\text{att}}(q) = \frac{1}{2}\zeta d^2(q, q_{\text{goal}}),$$

$$\begin{aligned} F_{\text{att}}(q) = \nabla U_{\text{att}}(q) &= \nabla\left(\frac{1}{2}\zeta d^2(q, q_{\text{goal}})\right), \\ &= \frac{1}{2}\zeta \nabla d^2(q, q_{\text{goal}}), \\ &= \zeta(q - q_{\text{goal}}), \end{aligned}$$

Potential Fields



$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{D(q)} - \frac{1}{Q^*}\right)^2, & D(q) \le Q^*, \\ 0, & D(q) > Q^*, \end{cases}$$

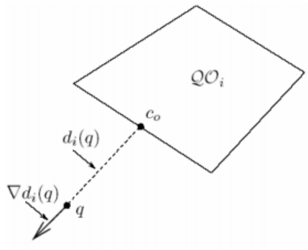
whose gradient is

$$\nabla U_{\text{rep}}(q) = \begin{cases} \eta\left(\frac{1}{Q^*} - \frac{1}{D(q)}\right)\frac{1}{D^2(q)}\nabla D(q), & D(q) \le Q^*, \\ 0, & D(q) > Q^*, \end{cases}$$

Potential Fields

$$d_i(q) = \min_{c \in \mathcal{QO}_i} d(q, c) \qquad \nabla d_i(q) = \frac{q - c}{d(q, c)}$$

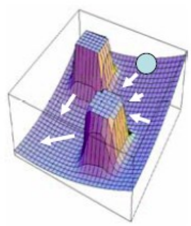
$$U_{\mathrm{rep}_i}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{d_i(q)} - \frac{1}{Q_i^*}\right)^2, & \text{if } d_i(q) \le Q_i^* \\ 0, & \text{if } d_i(q) > Q_i^* \end{cases}$$

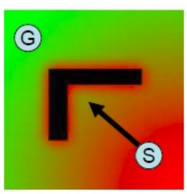


$$U_{\mathrm{rep}}(q) = \sum_{i=1}^{n} U_{\mathrm{rep}_i}(q)$$

Potential Fields

- Gradient Descent: $\dot{q} = -\nabla U(q)$
- Complete? No (local minima)*





*Navigation function: To make sure only one global minimum

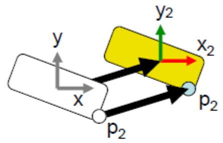
Summary: Navigation Algorithms

	Complete	Optimal	Efficiency	Model Required
Bug 1	Yes	No	~	No
Bug 2	Yes	No	Usually > B1	No
Visibility	Yes	Goal Dist	n^2 log n + A*	Yes
Voronoi	Yes	Obs Dist	n log n? + A*	Yes
Voronoi Bug	Yes	Obs Dist	~	No
Voronoi Brushfire	Resolution	Obs Dist	~ # cells	Yes
Exact Cell	Yes	No	n log n + A*	Yes
Approximate Cell	Resolution	Manh. Dist.	~ # cells	Yes
Potential Fields	No	Locally	Linear	Yes

Rigid Body Displacements

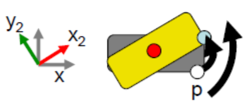
Must preserve **rigid** property, reflections are not allowed.

- **Translation:** Every point moves a fixed distance in a specified direction.



$$x_2 = x_1 + d_x \qquad y_2 = y_1 + d_y \qquad \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

- **Rotation**: One point is fixed. Others move a specified **angle** relative to fixed point.



$$x_2 = x_1 \cos\theta - y_1 \sin\theta \qquad y_2 = x_1 \sin\theta + y_1 \cos\theta \qquad \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} c\theta & -s\theta \\ s\theta & c\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

- Every displacement can be represented as **1 Translation** and/or **1 Rotation**

Representations of Rotation (Coordinates)

- Fixed Axis (x,y,z) or (roll, pitch, yaw)
  - Convenient and intuitive, 12 variations
- Euler Angles (z,y,x), (z,y,z) - (moving axes)
  - Equivalent to reverse order fixed-axis
- Unit Quaternions (4 numbers)
  - Easy to compose
  - Meaningful Interpolation
  - Useful for numerical stability, sampling, optimization
- Angle-Axis (4 numbers = 3 axis + 1 angle)
- **Rotation Matrices (9 numbers – Orthonormal Matrix)**

Homogeneous Transform 3D

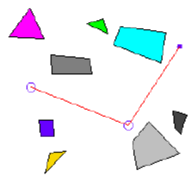
$$\mathbf{T}_B^A = \begin{bmatrix} & \mathbf{R}_B^A & & \mathbf{t}_B^A \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

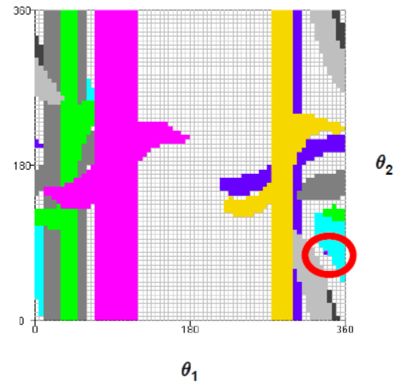
Interpretations:

- Maps p^B to p^A $\quad p^A = \mathrm{T}_\mathrm{B}^\mathrm{A} p^B$
- Transform operator: creates p_2^A from p_1^A
- Describes frame B relative to frame A
  t_B^A = position of the frame

$$\mathbf{R}_B^A = \begin{bmatrix} \mathbf{x}_B^A & \mathbf{y}_B^A & \mathbf{z}_B^A \end{bmatrix}$$

Configurations Space





Java Applet: Jeff Wiegley, Eric Lee, Ken Goldberg

Kinematics

- Forward Kinematics
  Mapping **from** joint angles **to** link positions
- **Inverse Kinematics**
  Mapping **from** link positions **to** joint angles

  Goals are defined in world coordinates, not joints coordinates.

Differential Kinematics

What is the robot Jacobian?

$$\mathbf{J}(\theta_1, \ldots, \theta_n) = \begin{bmatrix} \frac{\delta x}{\delta\theta_1} & \frac{\delta x}{\delta\theta_2} & \cdots & \frac{\delta x}{\delta\theta_n} \\ \frac{\delta y}{\delta\theta_1} & \frac{\delta y}{\delta\theta_2} & & \frac{\delta y}{\delta\theta_n} \\ \frac{\delta z}{\delta\theta_1} & \frac{\delta z}{\delta\theta_2} & & \frac{\delta z}{\delta\theta_n} \\ \frac{\delta \omega_x}{\delta\theta_1} & \frac{\delta \omega_x}{\delta\theta_2} & & \frac{\delta \omega_x}{\delta\theta_n} \\ \frac{\delta \omega_y}{\delta\theta_1} & \frac{\delta \omega_y}{\delta\theta_2} & & \frac{\delta \omega_y}{\delta\theta_n} \\ \frac{\delta \omega_z}{\delta\theta_1} & \frac{\delta \omega_z}{\delta\theta_2} & & \frac{\delta \omega_z}{\delta\theta_n} \end{bmatrix}$$

Matrix of Partial Derivatives of Kinematics w.r.t. each joint variable.

Why is it useful?

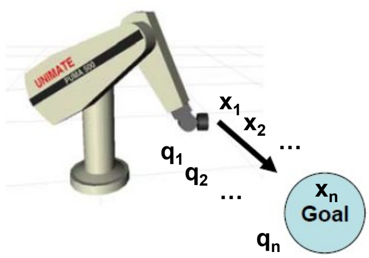
$$\boxed{\frac{\delta x}{\delta t}} = \frac{\delta x}{\delta \theta_1} \boxed{\frac{\delta \theta_1}{\delta t}}$$

Workspace velocity **Joint Velocity**

S. Joo (sungmoon.joo@cc.gatech.edu) 10/16/2014 71

Gradient IK - How do we use J?

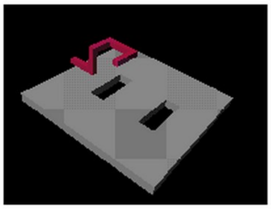
- Workspace goal
- How do we get a joint space goal?

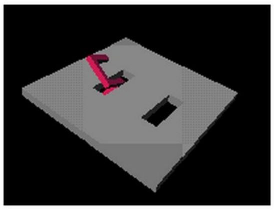


- Assuming 6 D.O.F and **J** is full rank: $\Delta q = J^{-1}\Delta x$
  Iterate until convergence $\mathbf{x}_i = \mathbf{f}(\mathbf{q}_i)$
- Otherwise Still Possible (Pseudo-Inverse & Variants): $J^+ = JT\,(JJ^T)^{-1}$

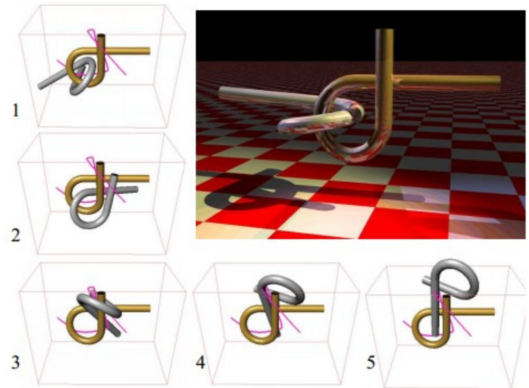
S. Joo (sungmoon.joo@cc.gatech.edu) 10/16/2014 72

Can we solve these planning problems?





http://www.kavrakilab.org/robotics/prm.html



"Planning Algorithms", S. Lavalle

Key Idea

- What did Visibility, Voronoi, Cells, Fields have in common?
  - Some form of explicit environment representation
  - Attempt at some form of optimality

- New concepts from 1990s:
  - Forget optimality altogether
  - Focus on Completeness
  - Think about Free Space

A New Kind of Roadmap

• Lydia Kavraki ‘94, ‘96 – Present
• Mark Overmars ’92, ‘96 - Present

- Previous roadmaps used features related to actual obstacle features.

- Probabilistic Roadmaps (PRM)
  - Features: Sampled free points
  - Edges: Verified connections

“Probabilistic roadmaps for path planning in high-dimensional configuration spaces” By Kavraki, Svestka, Latombe, and Overmars, 1996, IEEE Transactions on Robotics and Automation