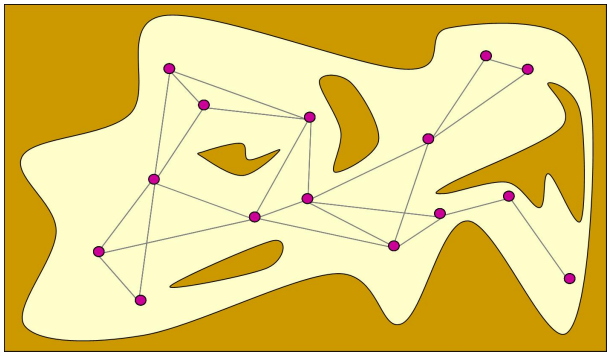
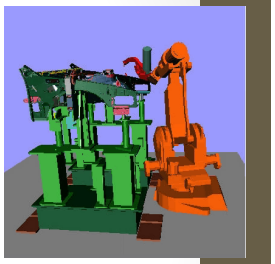
Tree-Growing Sample-Based Motion Planning

Probabilistic Roadmaps





What if only a small portion of the space needs to be explored?

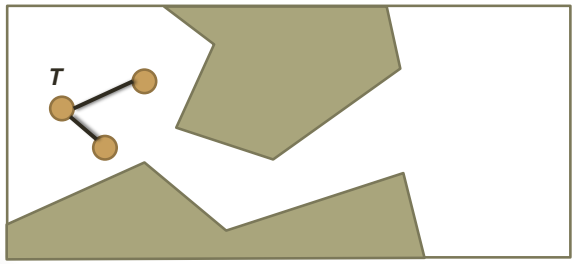
What if omnidirectional motion in C-space is not permitted?

Tree-growing planners

- Idea: grow a **tree** of feasible paths from the start until it reaches a neighborhood of the goal
  - *Sampling bias* toward "boundary" of currently explored region, helps especially if the start is in a region with poor visibility
- Many variants:
  - Rapidly-exploring Random Trees (RRTs) are popular, easy to implement (LaValle and Kuffner 2001)
  - Expansive space trees (ESTs) use a different sampling strategy (Hsu et al 2001)
  - SBL (Single-query, Bidirectional, Lazy planner) often efficient in practice (Sanchez-Ante and Latombe, 2005)

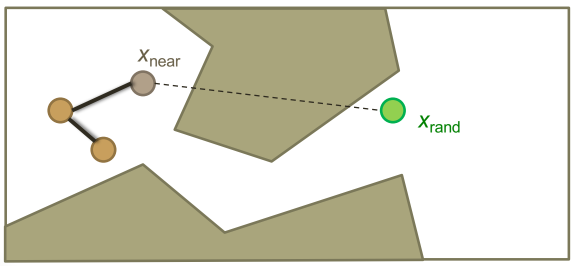
RRT

- Build a tree *T* of configurations, starting at x_{start}
- Extend:
  - Sample a configuration x_{rand} from *C* at random
  - Find the node x_{near} in *T* that is closest to x_{rand}
  - Extend a short path from x_{near} toward x_{rand}



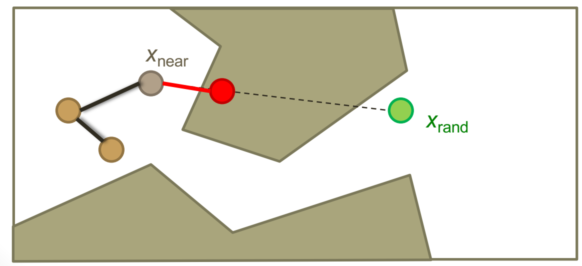
RRT

- Build a tree *T* of configurations, starting at x_{start}
- Extend:
  - Sample a configuration x_{rand} from *C* at random
  - Find the node x_{near} in *T* that is closest to x_{rand}
  - Extend a short path from x_{near} toward x_{rand}



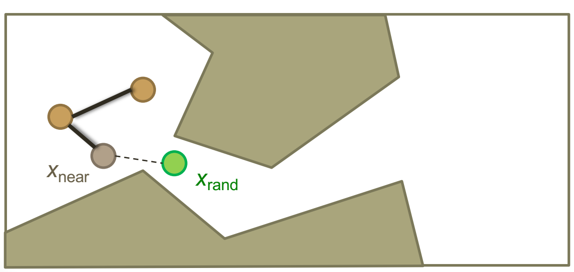
RRT

- Build a tree *T* of configurations, starting at x_{start}
- Extend:
  - Sample a configuration x_{rand} from *C* at random
  - Find the node x_{near} in *T* that is closest to x_{rand}
  - Extend a short path from x_{near} toward x_{rand}



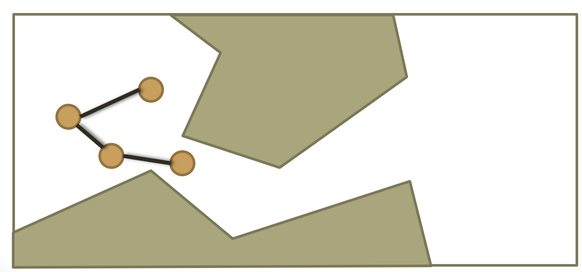
RRT

- Build a tree *T* of configurations, starting at x_{start}
- Extend:
  - Sample a configuration x_{rand} from *C* at random
  - Find the node x_{near} in *T* that is closest to x_{rand}
  - Extend a short path from x_{near} toward x_{rand}



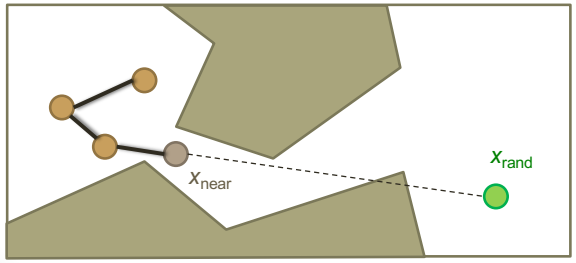
RRT

- Build a tree *T* of configurations, starting at x_{start}
- Extend:
  - Sample a configuration x_{rand} from *C* at random
  - Find the node x_{near} in *T* that is closest to x_{rand}
  - Extend a short path from x_{near} toward x_{rand}



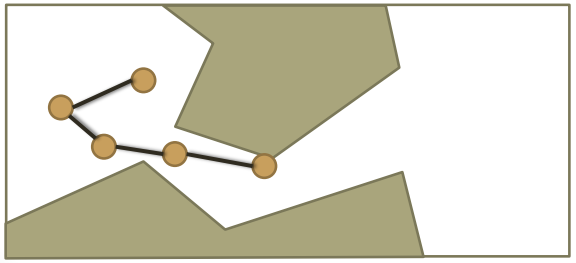
RRT

- Build a tree *T* of configurations, starting at x_{start}
- Extend:
  - Sample a configuration x_{rand} from *C* at random
  - Find the node x_{near} in *T* that is closest to x_{rand}
  - Extend a short path from x_{near} toward x_{rand}



RRT

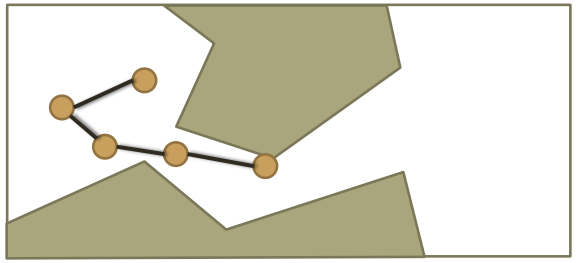
- Build a tree *T* of configurations, starting at x_{start}
- Extend:
  - Sample a configuration x_{rand} from *C* at random
  - Find the node x_{near} in *T* that is closest to x_{rand}
  - Extend a short path from x_{near} toward x_{rand}



RRT

- Build a tree *T* of configurations, starting at x_{start}
- Extend:
  - Sample a configuration x_{rand} from *C* at random
  - Find the node x_{near} in *T* that is closest to x_{rand}
  - Extend a short path from x_{near} toward x_{rand}

Governed by a step size parameter δ

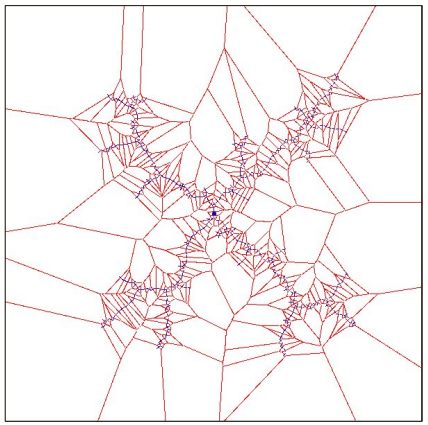


Implementation Details

- Bottleneck: finding nearest neighbor each step
  - O(n) with naïve implementation => $O(n^2)$ overall
  - KD tree data structure $O(n^{1-1/d})$
  - Approximate nearest neighbors often very effective
- Terminate when extension reaches a goal set
  - Usually visibility set of goal configuration
- Bidirectional strategy
  - Grow a tree from goal as well as start, connect when closest nodes in either tree can “see” each other

What is the sampling strategy?

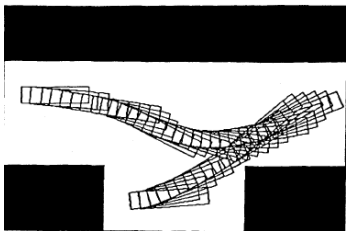
- Probability that a node gets selected for expansion is proportional to the volume of its Voronoi cell

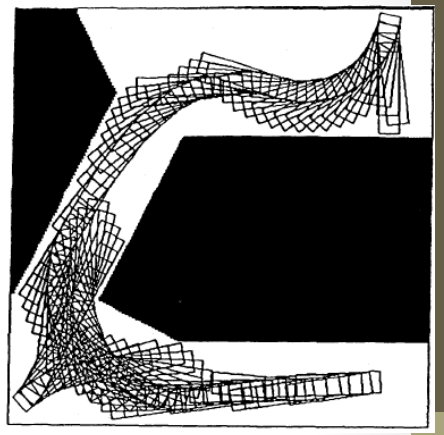


Planning with Differential Constraints

(Kinodynamic Planning)

Paths for a Car-Like Robot





Setting

- **Differential constraints**
- Dynamics, nonholonomic systems

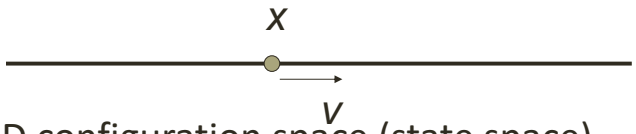
$$dq/dt = \Sigma_k f_k(q) u_k$$

Vector fields Controls

We'll consider fewer control dimensions than state dimensions

Example: 1D Point Mass

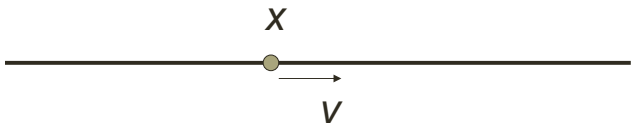
- Mass M

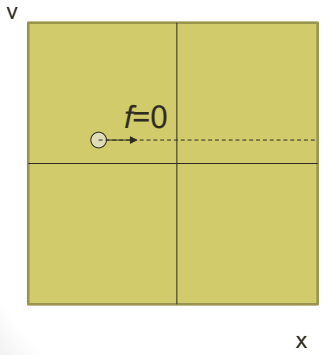


- 2D configuration space (state space)
- Controlled force f
- Equations of motion:

$$dx/dt = v$$
$$dv/dt = f / M$$

Example: 1D Point Mass



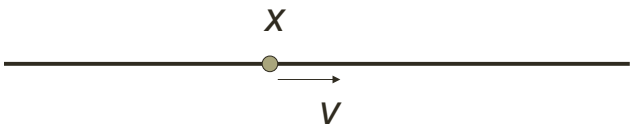


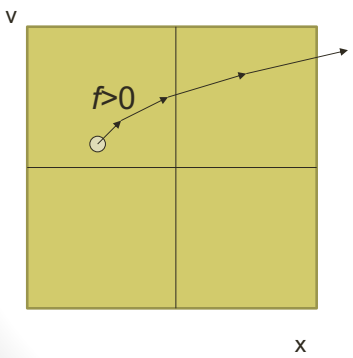
$$dx/dt = dv$$
$$dv/dt = f / M$$

Solution:

$$v(t) = v(0)+t\ f\ /M$$
$$x(t) = x(0)+t\ v(0) + \tfrac{1}{2}\ t^2\ f\ /\ M$$

Example: 1D Point Mass



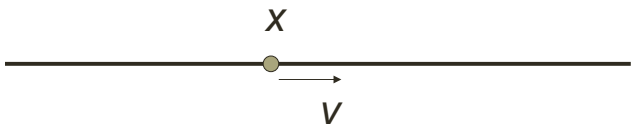


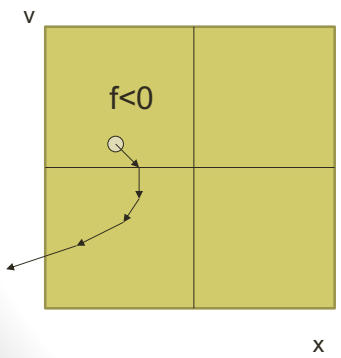
$$dx/dt = dv$$
$$dv/dt = f / M$$

Solution:

$$v(t) = v(0)+t\ f\ /M$$
$$x(t) = x(0)+t\ v(0) + \tfrac{1}{2}\ t^2\ f\ /\ M$$

Example: 1D Point Mass



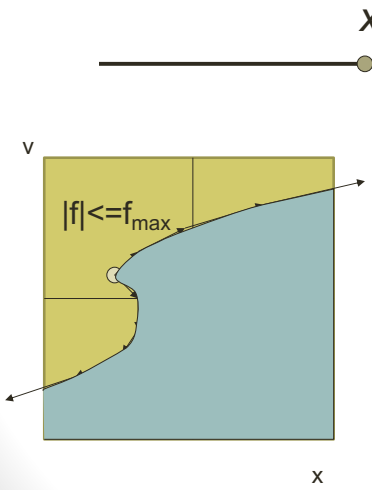


$$dx/dt = dv$$
$$dv/dt = f / M$$

Solution:

$$v(t) = v(0)+t\ f\ /M$$
$$x(t) = x(0)+t\ v(0) + \tfrac{1}{2}\ t^2\ f\ /\ M$$

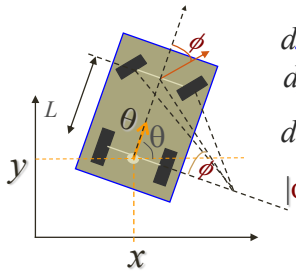
Example: 1D Point Mass



dx/dt = dv
dv/dt = *f* / *M*

Solution:
v(t) = v(0)+t f /M
x(t) = x(0)+t v(0) + ½ t² f / M

Example: Car-Like Robot



$dx/dt = v\cos\theta$
$dy/dt = v\sin\theta$
$\rightarrow dx\sin\theta - dy\cos\theta = 0$

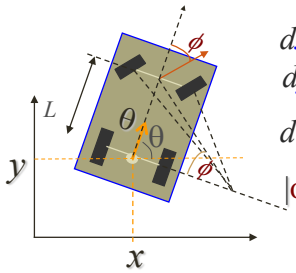
$d\theta/dt = (v/L)\tan\phi$

$|\phi| \leq \Phi$

Configuration space is 3-dimensional: q = (*x*, *y*, *θ*)

But control space is 2-dimensional: (*v*, ϕ) with
|*v*| = sqrt[(dx/dt)²+(dy/dt)²]

Example: Car-Like Robot



$dx/dt = v\cos\theta$
$dy/dt = v\sin\theta$
$\rightarrow dx\sin\theta - dy\cos\theta = 0$

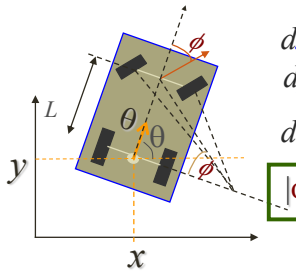
$d\theta/dt = (v/L)\tan\phi$

$|\phi| \leq \Phi$

q = (x,y,q)
q'= dq/dt = (dx/dt,dy/dt,dq/dt)
dx sinq – dy cosq = 0 is a particular form of f(q,q')=0

A robot is **nonholonomic** if its motion is constrained by a non-integrable equation of the form f(q,q') = 0

Example: Car-Like Robot



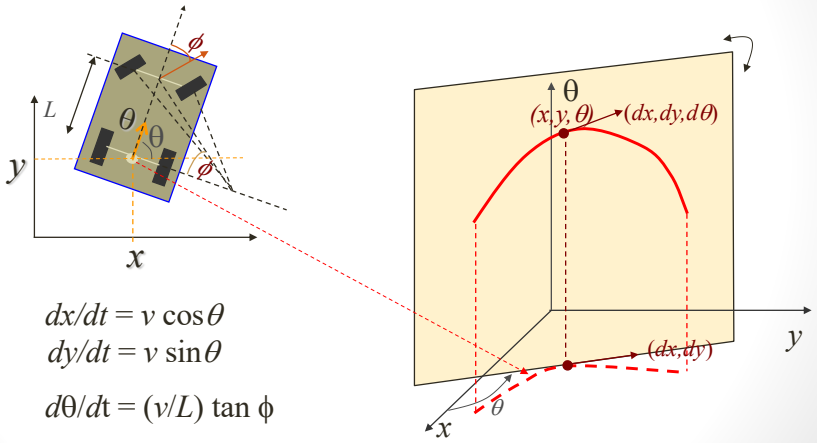
$dx/dt = v\cos\theta$
$dy/dt = v\sin\theta$
$\rightarrow dx\sin\theta - dy\cos\theta = 0$

$d\theta/dt = (v/L)\tan\phi$

$|\phi| \leq \Phi$

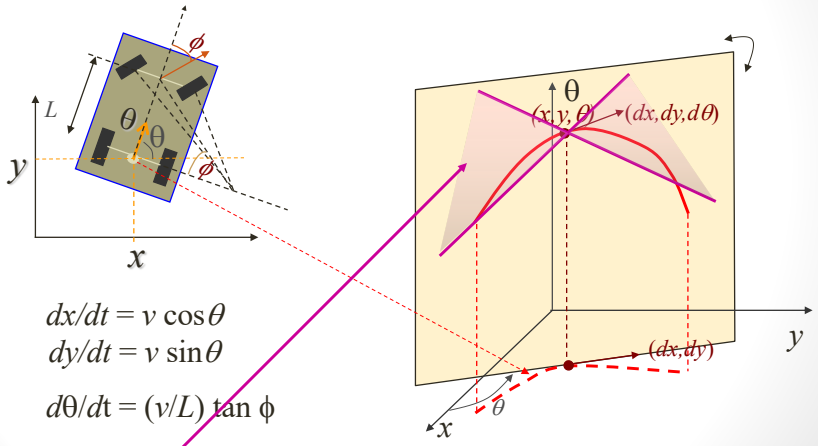
Lower-bounded turning radius

How Can This Work? Tangent Space/Velocity Space



$dx/dt = v\cos\theta$
$dy/dt = v\sin\theta$
$d\theta/dt = (v/L)\tan\phi$
$|\phi| \leq \Phi$

How Can This Work? Tangent Space/Velocity Space



$dx/dt = v\cos\theta$
$dy/dt = v\sin\theta$
$d\theta/dt = (v/L)\tan\phi$
$|\phi| \leq \Phi$

Nonholonomic Path Planning Approaches

- Two-phase planning (path deformation):
  - Compute collision-free path ignoring nonholonomic constraints
  - Transform this path into a nonholonomic one
  - Efficient, but possible only if robot is "controllable"
  - Need for a "good" set of maneuvers
- Direct planning (control-based sampling):
  - Use "control-based" sampling to generate a tree of milestones until one is close enough to the goal (deterministic or randomized)
  - Robot need not be controllable
  - Applicable to high-dimensional c-spaces

Control-Based Sampling

- PRM sampling technique: Pick each milestone in some region
- Control-based sampling:
  1. Pick control vector (at random or not)
  2. Integrate equation of motion over short duration (picked at random or not)
  3. If the motion is collision-free, then the endpoint is the new milestone

→ Tree-structured roadmaps
→ Need for endgame regions

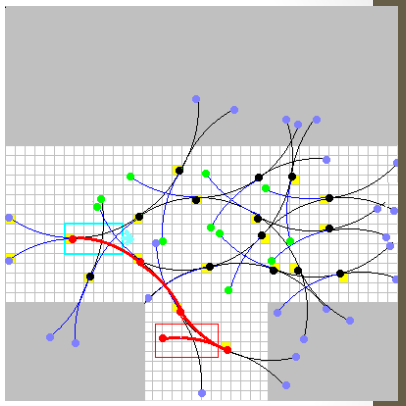
Example

$dx/dt = v\cos\theta$
$dy/dt = v\sin\theta$

$d\theta/dt = (v/L)\tan\phi$

$|\phi| \leq \Phi$

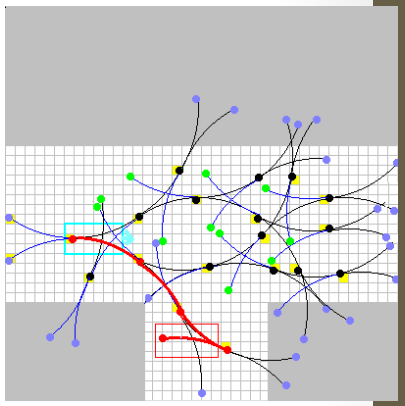
1. Select a milestone m
2. Pick ***v***, ϕ, and δt
3. Integrate motion from m
   → new milestone m'



Example

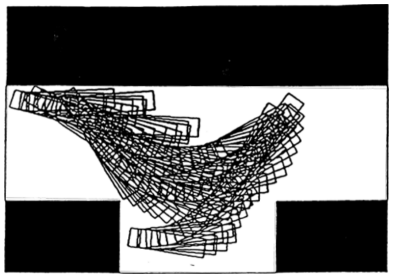
Indexing array:
A 3-D grid is placed over the configuration space. Each milestone falls into one cell of the grid. A maximum number of milestones is allowed in each cell (e.g., 2 or 3).

Asymptotic completeness:
If a path exists, the planner is guaranteed to find one if the resolution of the grid is fine enough.



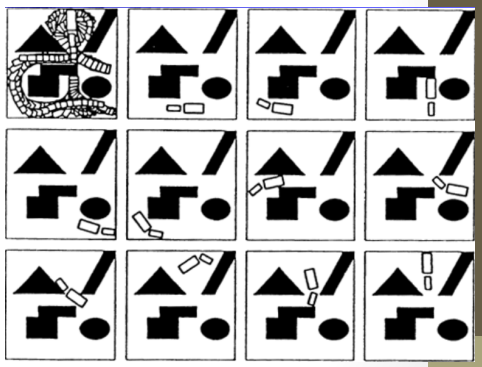
Computed Paths

Car That Can Only Turn Left



$\varphi_{max}=45°$, $\varphi_{min}=22.5°$

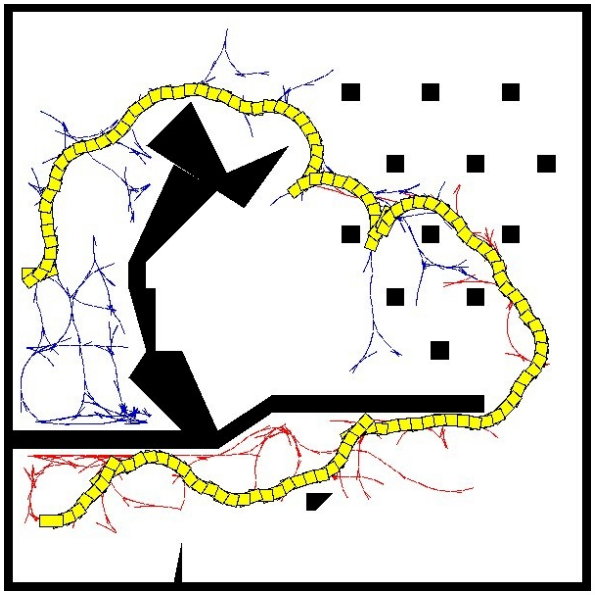
Tractor-trailer



$\varphi_{max}=45°$

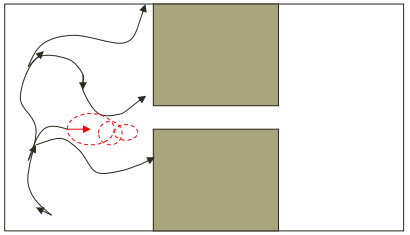
Control-Sampling RRT

- Configuration **generator** f(x,u)
- Build a tree T of configurations
- Extend:
  - Sample a configuration x_{rand} from X at random
  - Find the node x_{near} in T that is closest to x_{rand}
  - Pick a control u that brings f(x_{near},u) close to x_{rand}
  - Add f(x_{near},u) as a child of x_{near} in T



Weaknesses of RRT's strategy

- Depends on the domain from which x_{rand} is sampled
- Depends on the notion of "closest"
- A tree that is grown "badly" by accident can greatly slow convergence



Other strategies

- Dynamic-domain RRTs (Yershova et al 2008)
- Perturb samples in a neighborhood (EST, SBL)
- Lazy planning: delay expensive collision checks until end