Search Spaces II

ME/CS 132b
Advanced Robotics: Navigation and Perception
4/07/2011

Before we get started...

Homework Set #1 due date pushed back one week

Homework Set #2 assigned today

- Book chapter review and Matlab exercises
- Due one week from today (4/14)

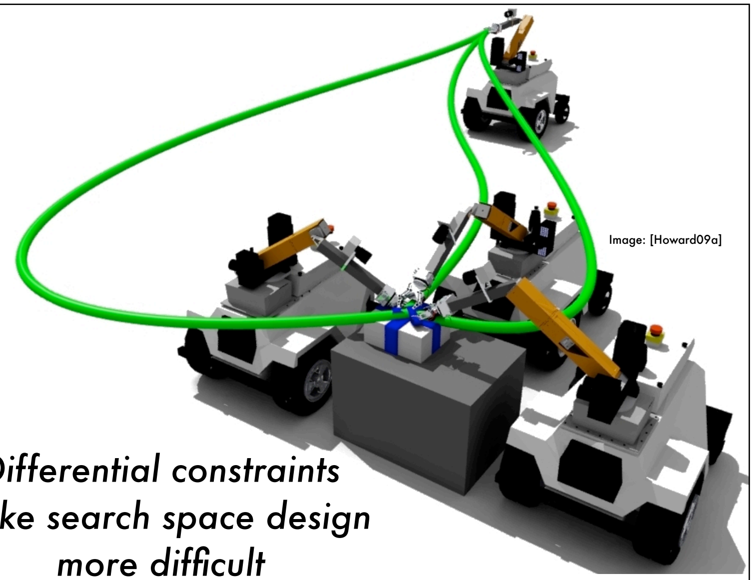


Image: [Howard09a]

Differential constraints make search space design more difficult

Search Space Lectures Outline

Part I

- Introductory Topology
- Configuration Spaces
- Search Space Design

Part II

- Search Space Design with Differential Constraints
- Applications
- Research Topics

Covers topics in Chapters 3, 4, 5, and 14 in S. LaValle's *Planning Algorithms*

Recap of Search Spaces I

Topology

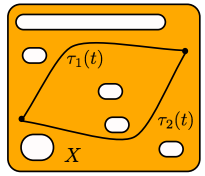
- Topological Space, Manifold
- Homotopy

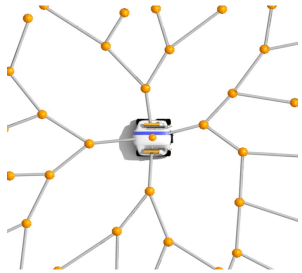
Geometric Transformations

- 2D, 3D, Quaternions

Search Spaces

- Grid, Lattices, RTD, PRM

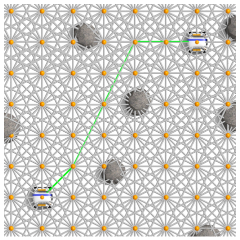




Search Space Design with Differential Constraints

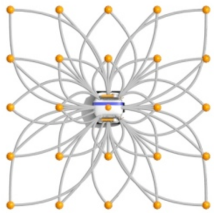
Concepts

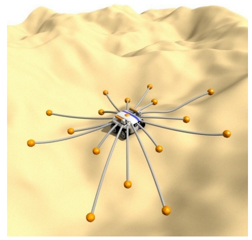
Challenges of Search Space Design with Differential Constraints



Efficiency

Expressiveness
and
Reachability





Feasibility

Motion Planning Problem Formulation

A world exists with a obstacle region $\mathcal{O}$ and configuration space $\mathcal{C}$ that maps into a state space X

The state space is a smooth manifold $\mathbf{x} \in \mathbf{X}$

An action space is defined for all states $\mathbf{u} \in \mathbf{U}(\mathbf{x})$

Obstacles are a subset of the state space

A state transition equation is known $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$

The initial state and goal region are known

Differences in Planning Terminology

Nonholonomic Planning
Planning that deals only with non-integrable kinematic constraints

Kinodynamic Planning
Planning with kinematic and dynamic constraints

Trajectory Planning
Similar to kinodynamic planning but typically involves separating path and velocity planning

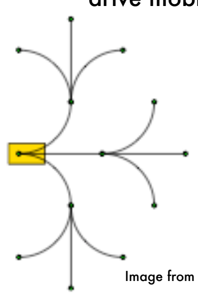
Planning with Differential Constraints

Edge Generation
Action/Trajectory Planning

Trajectory generation for a differential drive mobile manipulator

Search Space Generation
Reachable Sets

Sampling-Based Techniques



Dubins car reachable set with depth 2

Image from Figure 14.6 in [LaValle 06a]

Search Space Design with Differential Constraints

Edge Generation

Model-Predictive Trajectory Planning

One method for generating actions that satisfy general mobility constraints (many other methods exist)

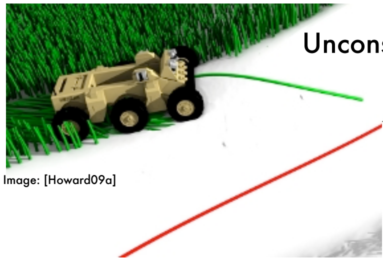
Parameterize the space of inputs

$$\mathbf{u}(\mathbf{x}, t) \rightarrow \mathbf{u}(\mathbf{p}, \mathbf{x}, t)$$

Generate parameter corrections based on gradients of simulated motion

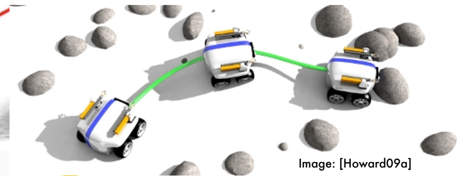
Numerically estimate partial derivatives to achieve applicability to various motion models

Three Classes of Problems



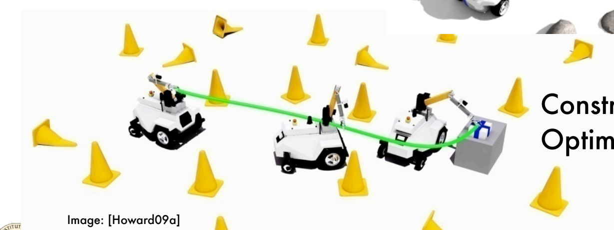
Unconstrained

Image: [Howard09a]



Constrained

Image: [Howard09a]



Constrained Optimization

Image: [Howard09a]

Unconstrained Trajectory Generation

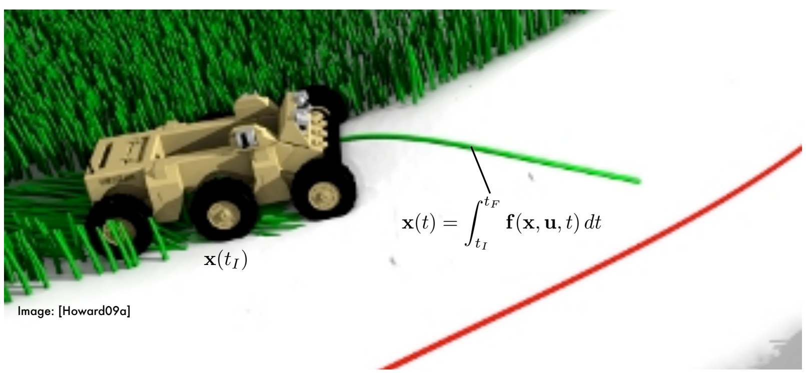


Image: [Howard09a]

Minimize $J(\mathbf{x}, \mathbf{u}, t_I, t_F)$ Satisfy

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \\ \mathbf{x}(t_I) &= \mathbf{x_I} \end{aligned}$$

Trajectory Cost

The trajectory cost is defined as a functional J consisting of a boundary state penalty Φ and an integrated Lagrangian $\mathcal{L}$

$$J\left(\mathbf{x},\mathbf{u},t_I,t_F\right) = \Phi\left(\mathbf{x}\left(t_I\right),t_I,\mathbf{x}\left(t_F\right),t_F\right) + \int_{t_I}^{t_F} \mathcal{L}\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right),t\right)\,dt$$

Cost functional represents a weighted combination of penalties associated with trajectory state or actions

Unconstrained Trajectory Generation

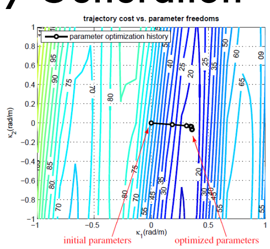
Minimize $J\left(\mathbf{x},\mathbf{u},t_I,t_F\right)$

Satisfy
$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}\left(\mathbf{x},\mathbf{u},t\right)\\ \mathbf{x}\left(t_I\right) &= \mathbf{x_I}\end{aligned}$$

Apply Gradient-Based Optimization

$$\mathbf{p_{i+1}} = \mathbf{p_i} - \delta\nabla J\left(\mathbf{x},\mathbf{u}\left(\mathbf{p}\right),t\right)$$

$$i \geq 0 \qquad \delta > 0$$



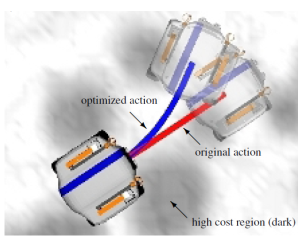
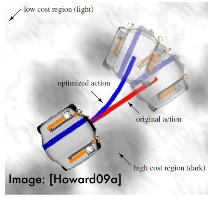


Image: [Howard09a]

Numerical Cost Jacobian Esimtation

Often the integral of the state transition equation is non-differentiable

Estimate the Jacobian numerically



$$\nabla J\left(\mathbf{x}, \mathbf{u}\left(\mathbf{p}\right), t\right) = \left[\begin{array}{ccc} \frac{\delta J(\mathbf{x},\mathbf{u}(\mathbf{p}),t)}{\delta p_0} & \ldots & \frac{\delta J(\mathbf{x},\mathbf{u}(\mathbf{p}),t)}{\delta p_n} \end{array} \right]$$

$$\frac{\delta J\left(\mathbf{x}, \mathbf{u}\left(\mathbf{p}\right), t\right)}{\delta p_n} = \frac{J\left(\mathbf{x}, \mathbf{u}\left(\mathbf{p}, p_n + \eta\right), t\right) - J\left(\mathbf{x}, \mathbf{u}\left(\mathbf{p}\right), t\right)}{\eta}$$

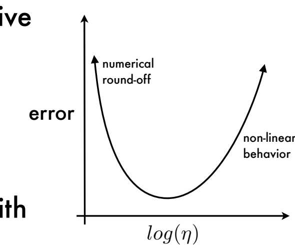
$$\eta \rightarrow 0$$

Issues with Numerical Estimation

This technique for estimation of the Jacobian is computationally expensive and numerically sensitive

Cost Jacobian requires $n+1$ simulations

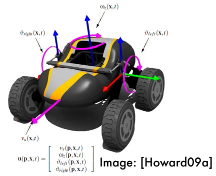
Partial derivative accuracy varies with value of η



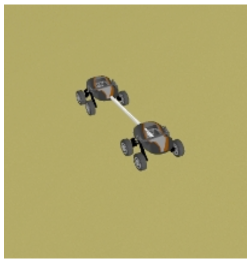
$$\frac{\delta J\left(\mathbf{x}, \mathbf{u}\left(\mathbf{p}\right), t\right)}{\delta p_n} = \frac{J\left(\mathbf{x}, \mathbf{u}\left(\mathbf{p}, p_n + \eta\right), t\right) - J\left(\mathbf{x}, \mathbf{u}\left(\mathbf{p}\right), t\right)}{\eta}$$

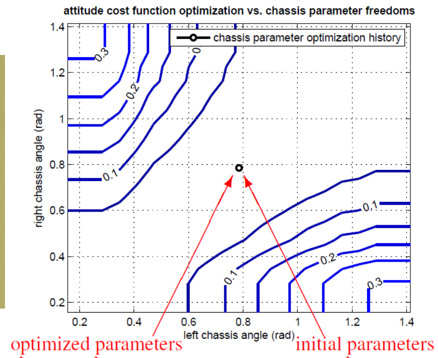
$$\eta \rightarrow 0$$

Another Example



Suspension Optimization





Constrained Trajectory Generation

Satisfy
$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x},\mathbf{u},t)\\ \mathbf{x}(t_I) &= \mathbf{x_I}\\ \mathbf{x}(t_F) &= \mathbf{x_F}\end{aligned}$$

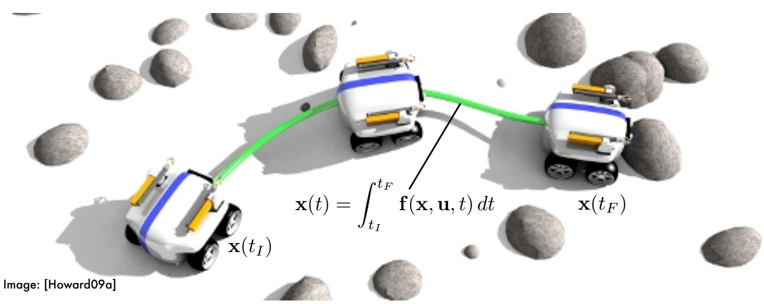


Image: [Howard09a]

Constrained Trajectory Generation

Satisfy
$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}, t)\\ \mathbf{x}(t_I) &= \mathbf{x_I}\\ \mathbf{x}(t_F) &= \mathbf{x_F}\end{aligned}$$

Define a constraint error

$$\Delta \mathbf{x_T}(\mathbf{p}) = \mathbf{X}_F - \left(\mathbf{x_I} + \int_{t_I}^{t_F} \mathbf{f}(\mathbf{x}, \mathbf{u}(\mathbf{p}), t)\, dt\right)$$

Solve using root finding techniques to drive $\Delta \mathbf{x_T}(\mathbf{p})$ to zero

Constraint Satisfaction

Iterative refinement of action parameters

$$\mathbf{p_{i+1}} = \mathbf{p_i} - \left[\frac{\delta \Delta \mathbf{x_T}(\mathbf{p})}{\delta \mathbf{p}}\right]^{-1} \Delta \mathbf{x_T}(\mathbf{p_i}) \qquad i \geq 0$$

Estimate the constraint Jacobian numerically

$$\frac{\delta \Delta \mathbf{x_T}(\mathbf{p})}{\delta \mathbf{p}} = \begin{bmatrix} \delta \Delta \mathbf{x_{T0}}(\mathbf{p}) \\ \vdots \\ \delta \Delta \mathbf{x_{T}}_n(\mathbf{p}) \end{bmatrix} = \begin{bmatrix} \delta \Delta \mathbf{x_{T0}}(p_0) & \delta \Delta \mathbf{x_{T0}}(p_1) & \dots & \delta \Delta \mathbf{x_{T0}}(p_m) \\ \delta \Delta \mathbf{x_{T1}}(p_0) & \delta \Delta \mathbf{x_{T1}}(p_1) & \dots & \delta \Delta \mathbf{x_{T1}}(p_m) \\ \vdots & \ddots & \dots & \vdots \\ \delta \Delta \mathbf{x_{T}}_n(p_0) & \delta \Delta \mathbf{x_{T}}_n(p_1) & \dots & \delta \Delta \mathbf{x_{T}}_n(p_m) \end{bmatrix}$$

Constraint Jacobian requires $n+1$ simulations
simulations

Trajectory Generation Example

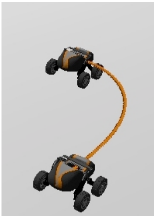
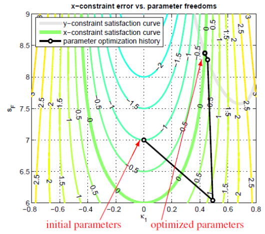


Image: [Howard09a]



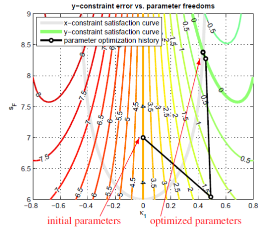
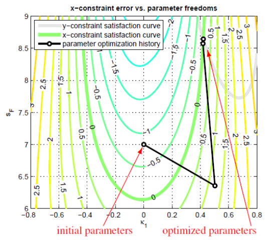
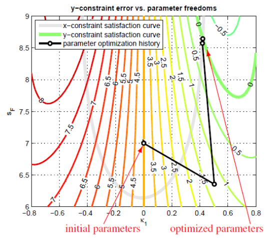


Image: [Howard09a]





Action Parameterization

It is good to match the number of degrees of freedom in the action with the number of boundary state constraints in the system (square constraint Jacobian)

For example,

$$\mathbf{x_I} = (\kappa_I) \qquad \mathbf{p} = (\kappa_0, \kappa_1, \kappa_2, \kappa_3, s_F)$$

$$\mathbf{x_F} = (x_F, y_F, \theta_F, \kappa_F) \qquad \text{for } s_I \leq s \leq s_F$$

$$\mathbf{u}(\mathbf{p}, \mathbf{x}, t) = \kappa(\mathbf{p}, \mathbf{x}, t) = a(\mathbf{p}, \mathbf{x}, t) + b(\mathbf{p}, \mathbf{x}, t)\, s + c(\mathbf{p}, \mathbf{x}, t)\, s^2 + d(\mathbf{p}, \mathbf{x}, t)\, s^3$$

$$a(\mathbf{p}) = \kappa 0$$

$$b(\mathbf{p}) = -\frac{1}{2}\frac{-2\kappa_3 + 11\kappa_0 - 18\kappa_1 + 9\kappa_2}{s_F - s_I}$$

$$c(\mathbf{p}) = \frac{9}{2}\frac{-\kappa_3 + 2\kappa_0 - 5\kappa_1 + 4\kappa_2}{(s_F - s_I)^2}$$

$$d(\mathbf{p}) = -\frac{9}{2}\frac{-\kappa_3 + \kappa_0 - 3\kappa_1 + 3\kappa_2}{(s_F - s_I)^3}$$

Parameter Initialization

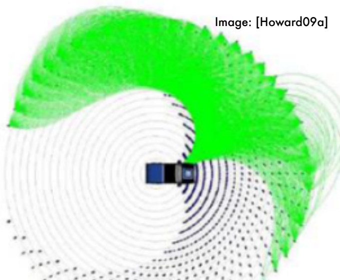
Since the technique is local optimization or root finding technique, a good initial guess of parameters is necessary

Methods:

Precomputed lookup tables

Machine learning techniques

Image: [Howard09a]



5-dimensional lookup table in position, orientation, velocity, and curvature

Constrained Optimization Trajectory Generation

Minimize $J\left(\mathbf{x},\mathbf{u},t_I,t_F\right)$ Satisfy

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}\left(\mathbf{x},\mathbf{u},t\right)\\ \mathbf{x}\left(t_I\right) &= \mathbf{x_I}\\ \mathbf{x}\left(t_F\right) &= \mathbf{x_F}\end{aligned}$$

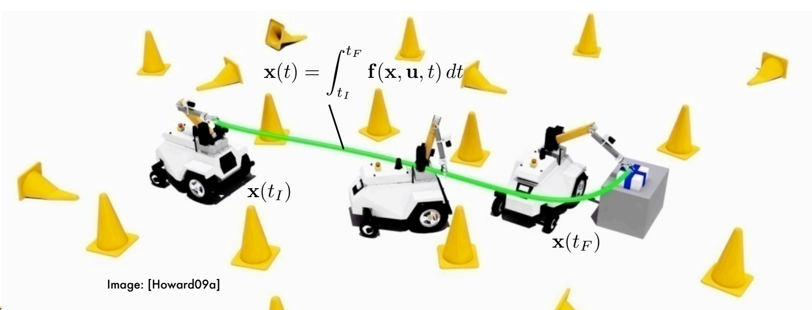


Image: [Howard09a]

Solution Technique

Form a Hamiltonian that represents the path cost and constraint error

$$\mathcal{H}(\mathbf{x}, \mathbf{u}, t_I, t_F, \lambda) = J(\mathbf{x}, \mathbf{u}, t_I, t_f) + \lambda^T \Delta x_T(\mathbf{x_I}, \mathbf{u}(\mathbf{p}))$$

Solve the system of first-order necessary conditions for optimality numerically

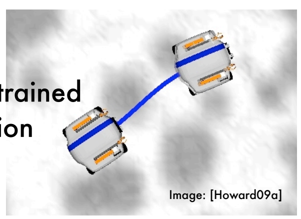
$$\frac{\mathcal{H}(\mathbf{x}, \mathbf{u}, t_I, t_F, \lambda)}{\delta \mathbf{p}} = \frac{J(\mathbf{x}, \mathbf{u}, t_I, t_f)}{\delta \mathbf{p}} + \lambda^T \frac{\Delta x_T(\mathbf{x_I}, \mathbf{u}(\mathbf{p}))}{\delta \mathbf{p}} = 0$$

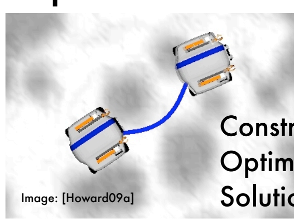
$$\frac{\mathcal{H}(\mathbf{x}, \mathbf{u}, t_I, t_F, \lambda)}{\delta \lambda} = \lambda^T \Delta x_T(\mathbf{x_I}, \mathbf{u}(\mathbf{p})) = 0$$

Hamiltonian Hessian requires $n^2 + 1$ simulations

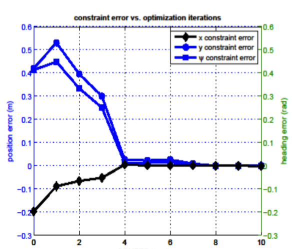
An Example

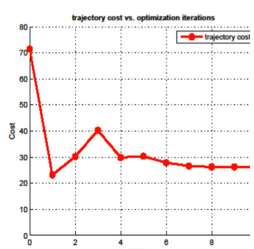
Constrained Solution

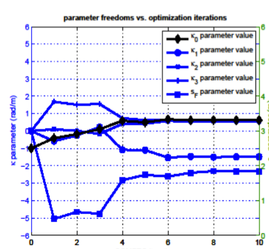




Constrained Optimization Solution







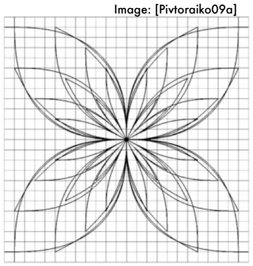
Search Space Design with Differential Constraints

Graph Generation

Reachable Sets

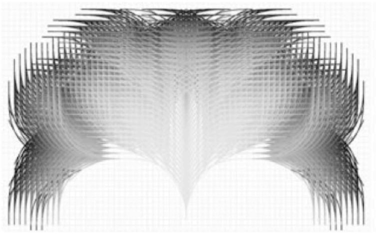
Must search densely in state space (a function of the sampling of action space)

Image: [Pivtoraiko09a]



control set

Image: [Pivtoraiko09a]



first 1400 expansions of a state lattice

Dubins Car Reachable Set

[Dubins57a]

Dubins Car Mobility: A car that only moves forward or at the maximum steering angle in either direction

Reachability tree for a Dubins car control set (depth = 4)

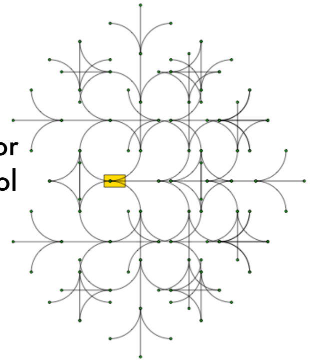


Image from Figure 14.6 in [LaValle 06a]

Tree Pruning

By discretizing the space and allowing only a single state at each node a smaller tree can be formed

w/o pruning

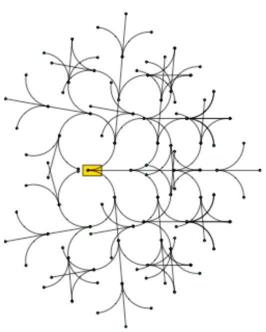


Image from Figure 14.16 in [LaValle 06a]

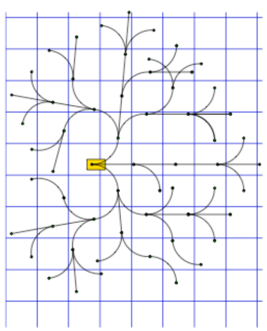


Image from Figure 14.16 in [LaValle 06a]

w/ pruning

Recombinant State Lattices

By defining a repeatable control set for each state in a discretized graph a feasible search that recombines can be generated

discretized and continuous in x,y,θ

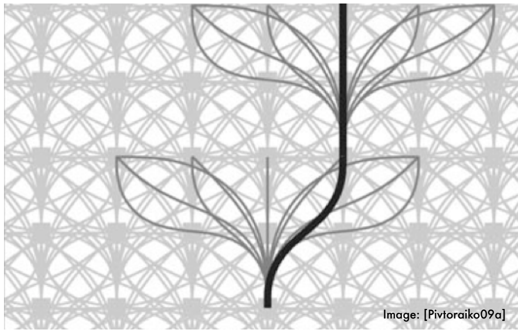


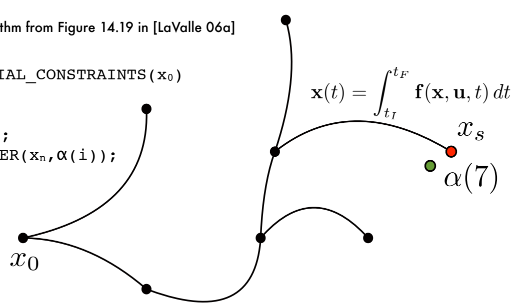
Image: [Pivtoraiko09a]

Rapidly Exploring Dense Trees [LaValle99a]

The same general technique applies but requires a local planner to determine an action that drives towards the sample

Algorithm from Figure 14.19 in [LaValle 06a]

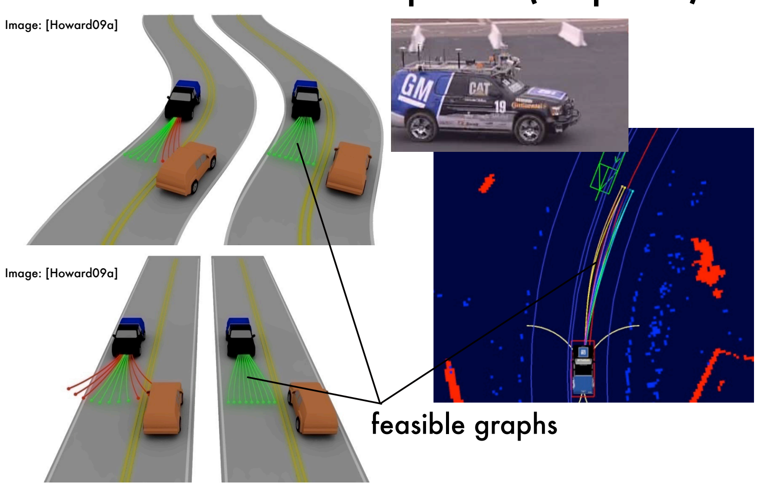
```
SIMPLE_RDT_WITH_DIFFERENTIAL_CONSTRAINTS(x0)
G.init(x0);
for i = 1 to k do
  xn ← NEAREST(S,α(i));
  (u,xs) ← LOCAL_PLANNER(xn,α(i));
   G.add_vertex(xs);
   G.add_edge(u);
```



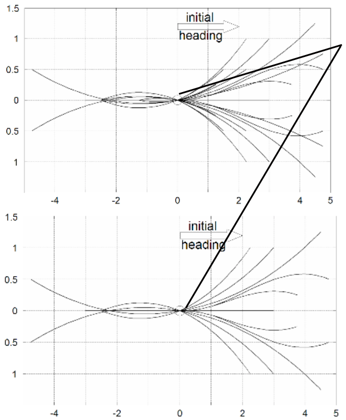
Applications

Local/Global Navigation

Feasible Search Spaces (Depth 1)

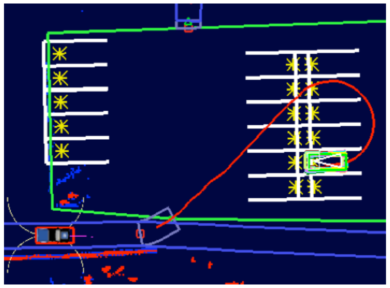


Feasible Search Spaces (Lattices)



multi-resolution control sets

Image: [Ferguson08b]



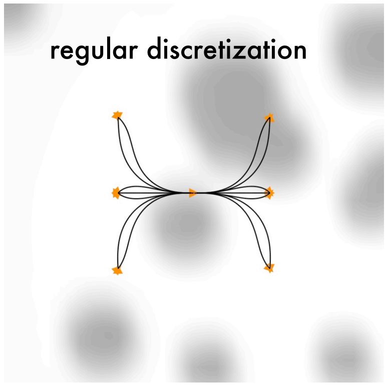
motion plan generated on a feasible lattice

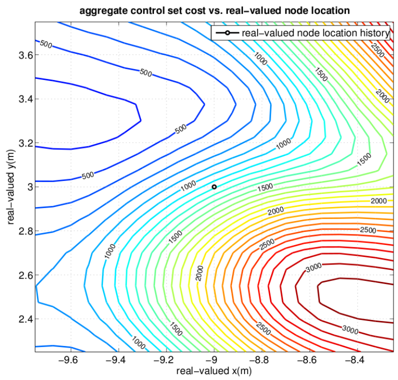
Image: [Ferguson08b]

Research Topics

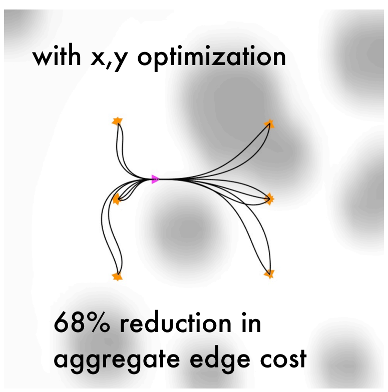
Graph Adaptation

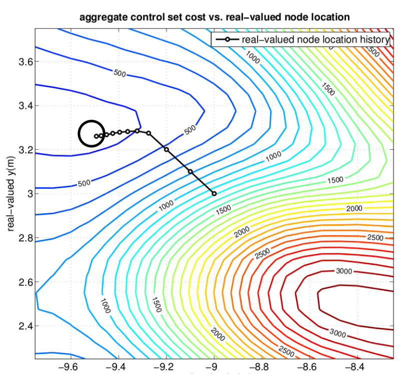
Exploiting Environmental Information



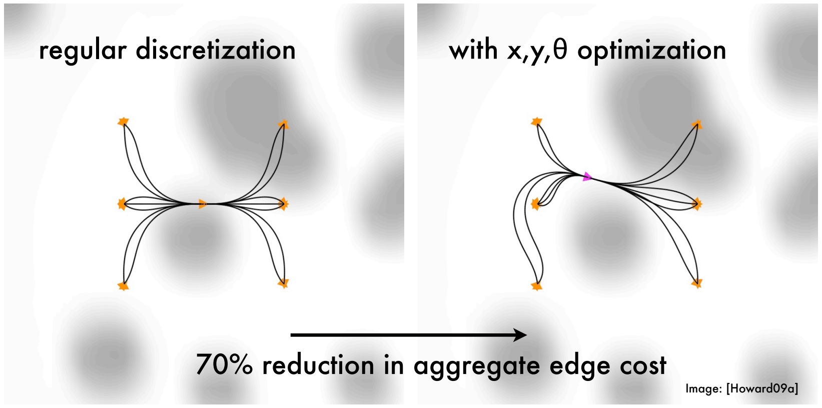


Exploiting Environmental Information

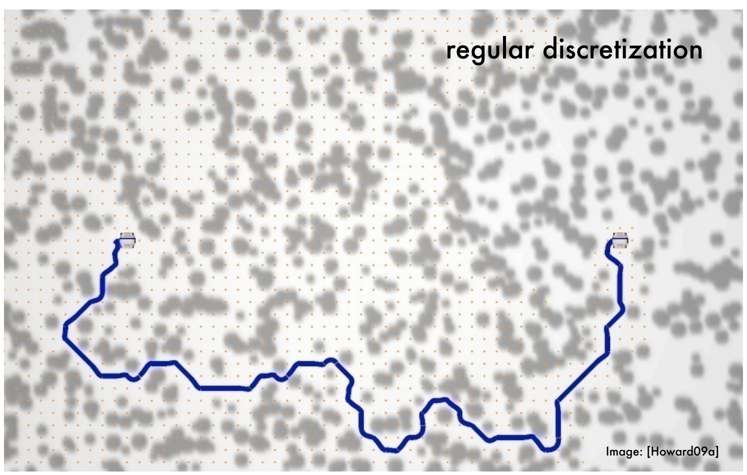




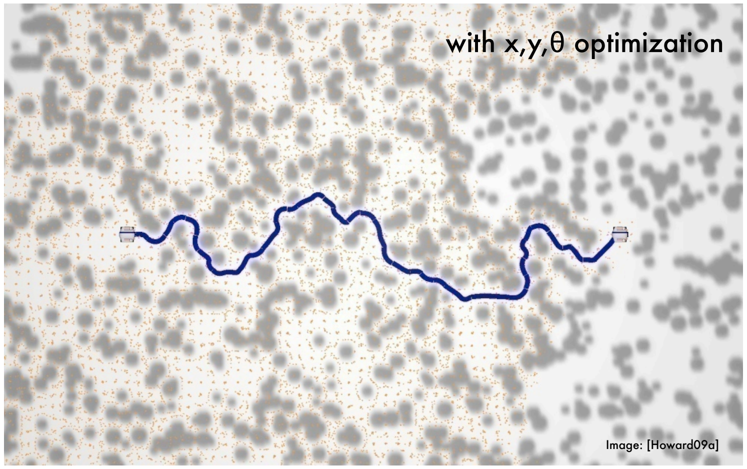
Exploiting Environmental Information



Graph Adaptation



Graph Adaptation



Summary

Edge generation requires solving for actions to satisfy boundary state constraints subject to a variety of kinematic and dynamic constraints

Extensions of existing configuration space sampling techniques can be extended to planning with differential constraints by integrating a local motion planner or trajectory generator

Planning with differential constraints is more computationally expensive that without but it sometimes necessary to guarantee safety

Next Lecture (4/7)

Obstacles and Cost

- C-Space Expansion
- Continuous Valued Obstacle Representation

Document/Image References

[Howard 09a] T.M. Howard, "Adaptive Model-Predictive Motion Planning for Navigation in Complex Environments". Ph.D. Thesis, Carnegie Mellon University, August 2009

[LaValle99a] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In Proceedings IEEE International Conference on Robotics and Automation, pages 473–479, 1999.

[Ferguson08b] David Ferguson, Thomas Howard, and Maxim Likhachev, "Motion Planning in Urban Environments: Part II," Proceedings of the IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems, September, 2008.

[Pivtoraiko09a] M. Pivtoraiko, R. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, Vol. 26, No. 3, March, pages 308-333, 2009

[LaValle 06a] S. LaValle, "Planning Algorithms". Cambridge: Cambridge University Press. ISBN 0521862051.

[Dubins57a] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. American Journal of Mathematics, 79, pages 497–516, 1957