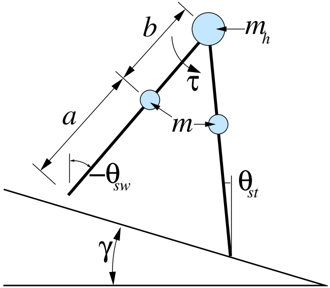
LQR-Trees: Feedback Motion Planning on Sparse Randomized Trees

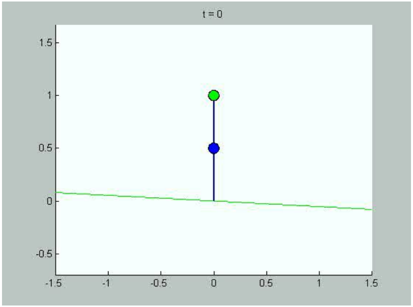
Russ Tedrake

Associate Professor
MIT Computer Science and Artificial Intelligence Lab

RSS, Seattle
June 29, 2009

A Motivating Example: The Compass Gait

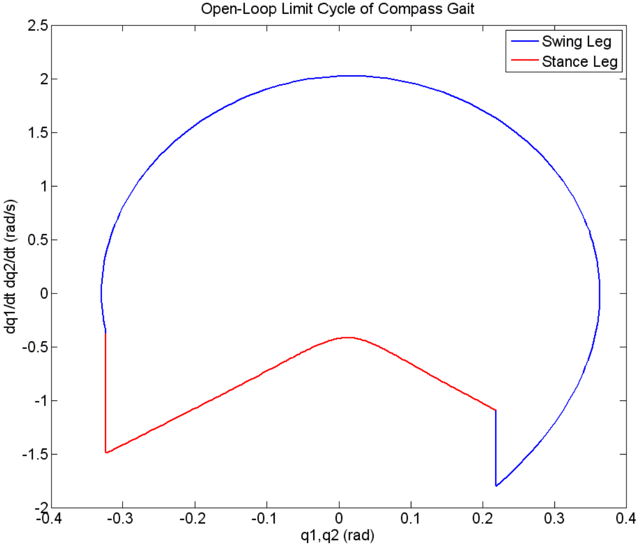




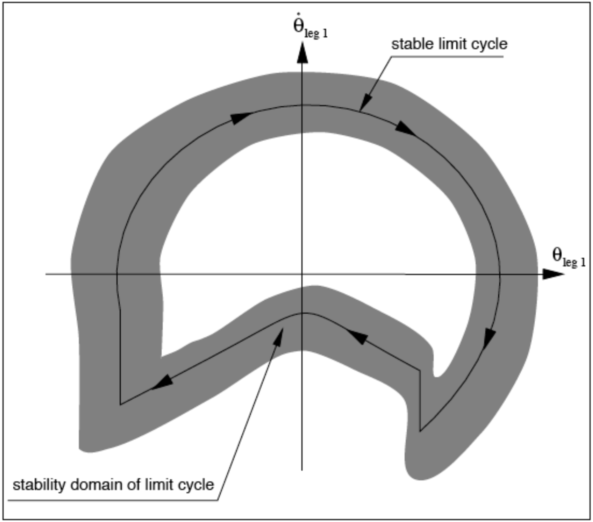
[CLICK IMAGE TO PLAY MOVIE]

- Torque only at the hip.
- No foot scuffing.
- Impulsive, Inelastic Collisions
- Instantaneous transfer of support

Compass Gait: The nominal (passive) limit cycle



Compass Gait: The nominal (passive) limit cycle



Goswami, 1996

Control for the Compass Gait

- A good model for control experiments:

Control for the Compass Gait

- A good model for control experiments:
  - Captures challenge of hybrid limit cycle stability
  - Avoids unnecessary complexity
  - **We don't yet have satisfying control solutions**

Control for the Compass Gait

- A good model for control experiments:
  - Captures challenge of hybrid limit cycle stability
  - Avoids unnecessary complexity
  - **We don't yet have satisfying control solutions**
- Good previous work on stabilizing the nominal limit cycle:
  - Local linear or nonlinear feedback increases basin (Westervelt, Shiriaev, Ruina, Goswami,..)
  - "Global" methods like dynamic progamming suffer from discretization, and don't scale (Byl,Morimoto,...).

Control for the Compass Gait

- A good model for control experiments:
  - Captures challenge of hybrid limit cycle stability
  - Avoids unnecessary complexity
  - **We don't yet have satisfying control solutions**
- Good previous work on stabilizing the nominal limit cycle:
  - Local linear or nonlinear feedback increases basin (Westervelt, Shiriaev, Ruina, Goswami,..)
  - "Global" methods like dynamic progamming suffer from discretization, and don't scale (Byl,Morimoto,...).
- but I want more...

Goals for Control

- Goal: Systematically design a feedback controller such that every point in a (bounded subset of) state space that *can* be driven to the goal *will* be driven to the goal.

Goals for Control

- Goal: Systematically design a feedback controller such that every point in a (bounded subset of) state space that *can* be driven to the goal *will* be driven to the goal.
  - Non-trivial because of actuator limits and nonlinear dynamic (underactuation) constraints

Goals for Control

- Goal: Systematically design a feedback controller such that every point in a (bounded subset of) state space that *can* be driven to the goal *will* be driven to the goal.
  - Non-trivial because of actuator limits and nonlinear dynamic (underactuation) constraints
- Observation: Trajectory optimization and trajectory stabilization work very well (locally)

Goals for Control

- Goal: Systematically design a feedback controller such that every point in a (bounded subset of) state space that *can* be driven to the goal *will* be driven to the goal.
  - Non-trivial because of actuator limits and nonlinear dynamic (underactuation) constraints

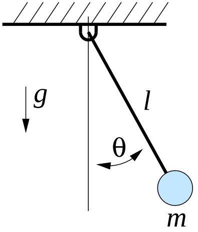
- Observation: Trajectory optimization and trajectory stabilization work very well (locally)
- Possible solution: Trajectory libraries
  - Chris Atkeson has been arguing this for years
  - Can we find a "minimal" set of trajectories that cover the space?

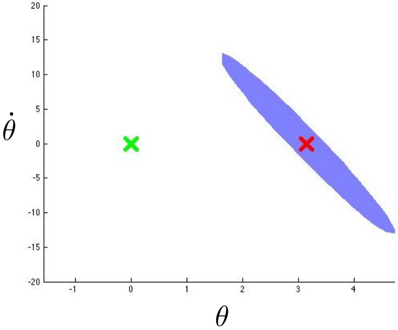
Estimating basins of attraction

- New tools from systems theory can estimate basins of attraction for linear feedback using convex optimization.

Estimating basins of attraction

- New tools from systems theory can estimate basins of attraction for linear feedback using convex optimization.
- Pendulum Example:





Sums-of-Squares (SOS) Optimization

- Given polynomial, $p(x)$, with unknown coefficients, c, verify uniform positive definiteness:

$$\exists c \forall x \quad p(x) \geq 0.$$

Sums-of-Squares (SOS) Optimization

- Given polynomial, $p(x)$, with unknown coefficients, c, verify uniform positive definiteness:

$$\exists c \forall x \quad p(x) \geq 0.$$

- Feasibility set is convex $\longrightarrow$ convex optimization.

Sums-of-Squares (SOS) Optimization

- Given polynomial, $p(x)$, with unknown coefficients, c, verify uniform positive definiteness:

$$\exists c \forall x \quad p(x) \geq 0.$$

- Feasibility set is convex $\longrightarrow$ convex optimization.
- Can also handle equality constraints, and/or optimize a linear objective

Polynomial Lyapunov functions

- Pablo Parrilo popularized SOS tools for control verification.
- Example: Given a polynomial dynamical system:

$$\dot{x} = \sum_{i=0}^{N} \alpha_i x^i,$$

can search for coefficients of a polynomial Lyapunov function, $V(x)$, such that $\dot{V}(x) \leq 0$.

"Certificates" for LQR Design

- Given $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$
- Linearize around operating point to obtain

$$\dot{\bar{\mathbf{x}}} \approx \mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\bar{\mathbf{u}}.$$

- LQR design gives:

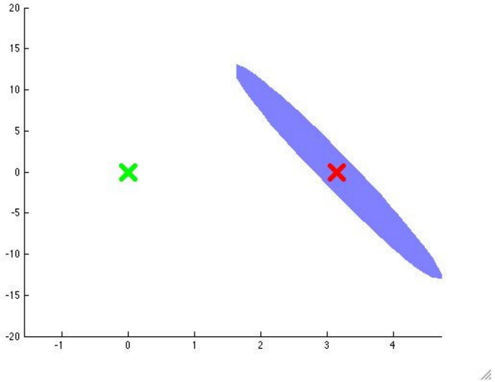
$$\bar{\mathbf{u}} = -\mathbf{K}\bar{\mathbf{x}}, \quad J(\mathbf{x}) \approx \bar{\mathbf{x}}^T\mathbf{S}\bar{\mathbf{x}},$$

  where $J(\mathbf{x})$ is the approximate *cost-to-go*.
- Approximate $\mathbf{f}$ with higher-order Taylor expansion.
- Use SOS tools to find largest scalar ρ for which

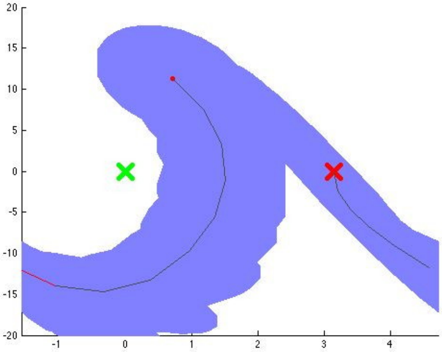
$$\forall \mathbf{x} \text{ with } J(\mathbf{x}) \leq \rho, \quad \frac{d}{dt}J(\mathbf{x}) \leq 0.$$

- Also works for LQR trajectory stabilization (time-varying)

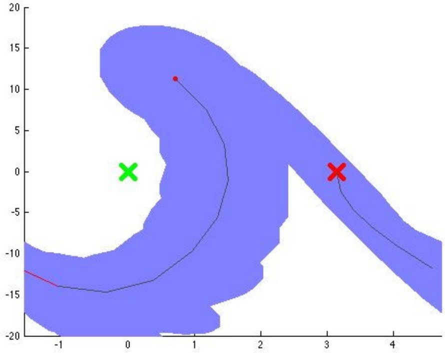
Pendulum "Funnels"

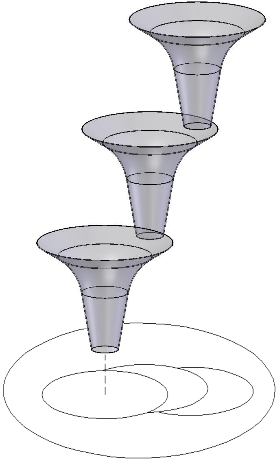


Pendulum “Funnels”



Pendulum "Funnels"





Erdmann,Mason,Koditschek

Randomized Feedback Motion Planning

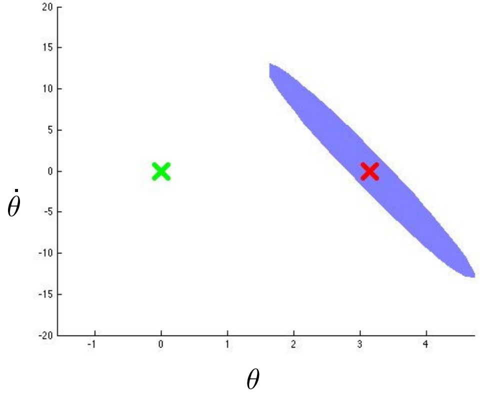
- Planning funnels are based on trajectories.
  - Represented compactly by matrix **S** and scalar ρ.
  - Conservative in almost every way.

Randomized Feedback Motion Planning

- Planning funnels are based on trajectories.
  - Represented compactly by matrix **S** and scalar ρ.
  - Conservative in almost every way.
- Combine funnels with randomized motion planning
  - Rapidly-exploring randomized trees (RRTs)
  - Probabilistic Roadmaps (PRMs)

The "LQR-Tree" Algorithm

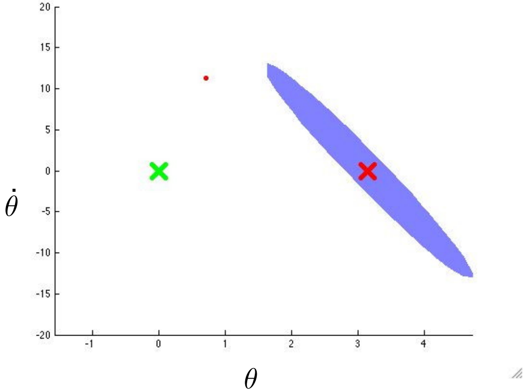
Grow a stabilizing tree backwards from the goal:



The "LQR-Tree" Algorithm

Grow a stabilizing tree backwards from the goal:

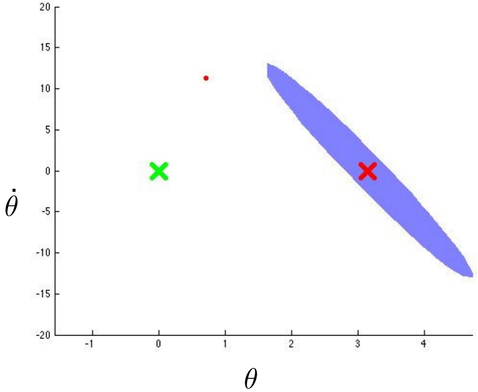
1. Choose a sample randomly from state space



The "LQR-Tree" Algorithm

Grow a stabilizing tree backwards from the goal:

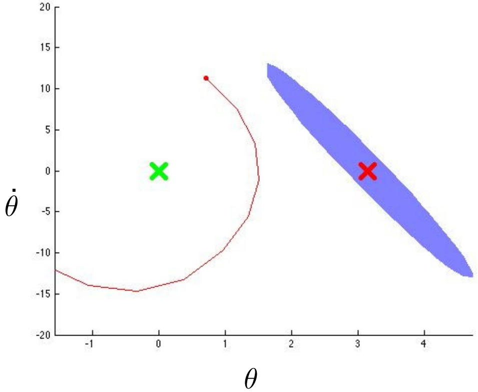
1. Choose a sample randomly from state space
2. Find closest leaf in the tree (via LQR metric)



The "LQR-Tree" Algorithm

Grow a stabilizing tree backwards from the goal:

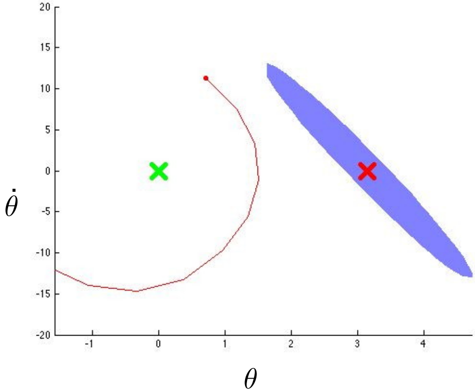
1. Choose a sample randomly from state space
2. Find closest leaf in the tree (via LQR metric)
3. Grow the tree towards the sample



The "LQR-Tree" Algorithm

Grow a stabilizing tree backwards from the goal:

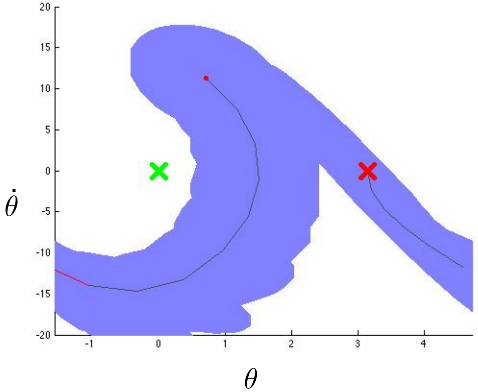
1. Choose a sample randomly from state space
2. Find closest leaf in the tree (via LQR metric)
3. Grow the tree towards the sample
   - If connection fails, discard sample.



The "LQR-Tree" Algorithm

Grow a stabilizing tree backwards from the goal:

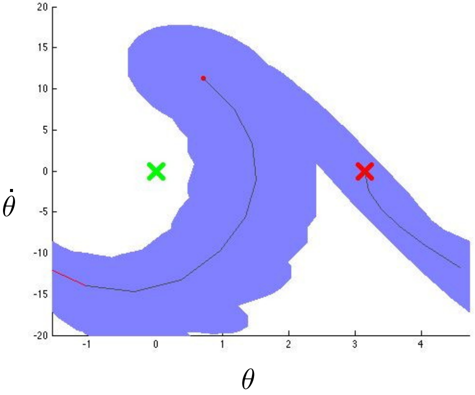
1. Choose a sample randomly from state space
2. Find closest leaf in the tree (via LQR metric)
3. Grow the tree towards the sample
   - If connection fails, discard sample.
4. Compute LQR stabilizing controller and Lyapunov 'certificates' for new leaf.



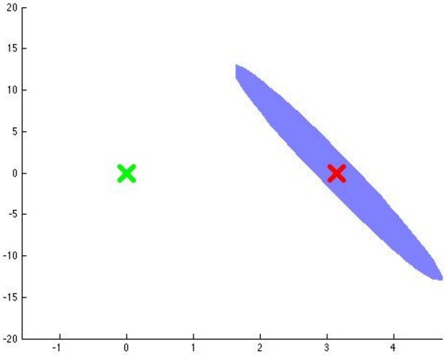
The “LQR-Tree” Algorithm

Grow a stabilizing tree backwards from the goal:

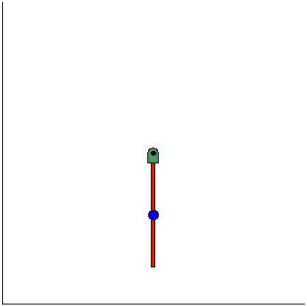
1. Choose a sample randomly from state space
2. Find closest leaf in the tree (via LQR metric)
3. Grow the tree towards the sample
   - If connection fails, discard sample.
4. Compute LQR stabilizing controller and Lyapunov ‘certificates’ for new leaf.
5. Repeat



Simple Pendulum Example



[CLICK IMAGE TO PLAY MOVIE]



[CLICK IMAGE TO PLAY MOVIE]

Properties of the algorithm

Properties of the algorithm

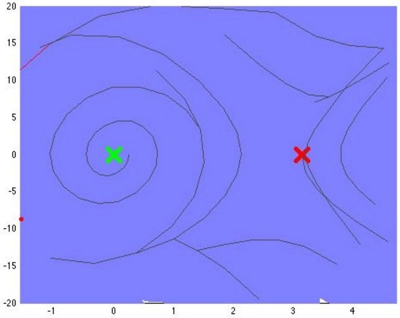
- Probabilistically covers reachable space with stabilizing controller (under mild assumptions)

Properties of the algorithm

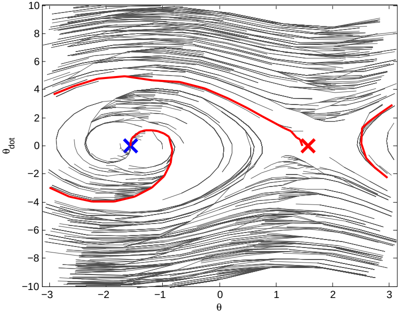
- Probabilistically covers reachable space with stabilizing controller (under mild assumptions)
- Efficient in number of nodes; each node requires computation

Properties of the algorithm

- Probabilistically covers reachable space with stabilizing controller (under mild assumptions)
- Efficient in number of nodes; each node requires computation

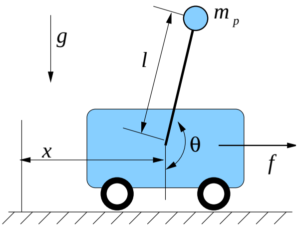


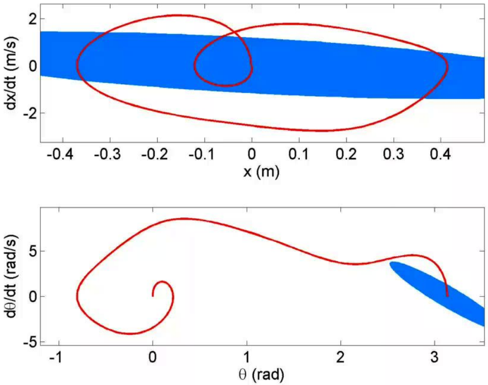
LQR-Trees



RRT w/ Euclidean metric

Certificates for the Cart-Pole system

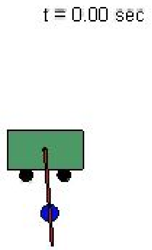




[CLICK IMAGE TO PLAY MOVIE]

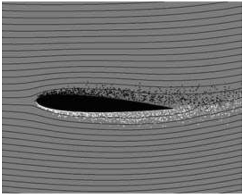
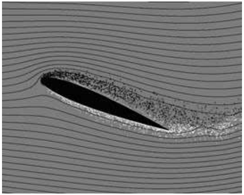
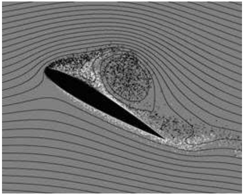
by Philipp Reist

LQR-Trees for the Cart-Pole system

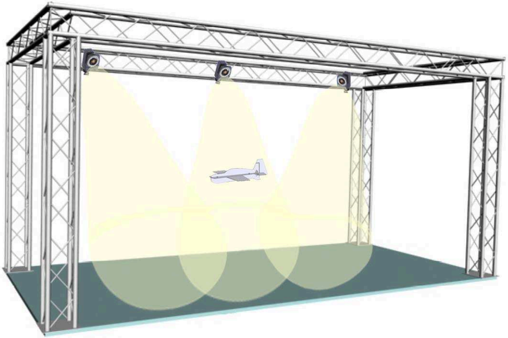


[CLICK IMAGE TO PLAY MOVIE]

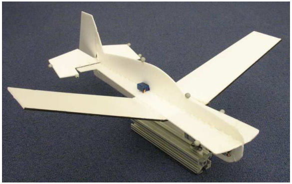
The “Perching” Problem


Experiment Design



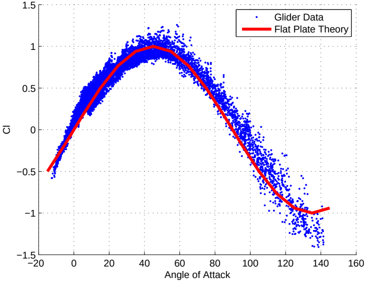
- Glider (no propellor)
- Flat wings
- Dihedral (passive roll stability)
- Offboard sensing and control



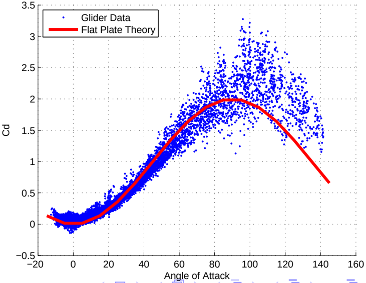
System Identification

- Nonlinear rigid-body vehicle model
- Linear (w/ delay) actuator model
- Real flight data (no wind tunnel)
  - Very high angle-of-attack regimes
  - Relatively small number of physics-based basis functions
  - Vortex shedding

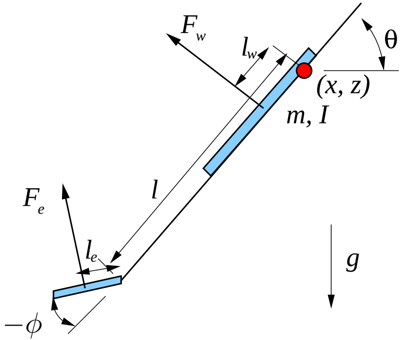
Lift Coefficient



Drag Coefficient



A dynamic model



- Planar dynamics
- Aerodynamics fit from data
- State: $\mathbf{x} = [x, y, \theta, \phi, \dot{x}, \dot{y}, \dot{\theta}]$
- Only actuator is the elevator angle, $\mathbf{u} = \dot{\phi}$

Glider Perching

- Enters motion capture @ 6 m/s.
- Perch is < 3.5 m away.
- Entire trajectory @ 1 second.

Requires Separation!

[CLICK IMAGE TO PLAY MOVIE]

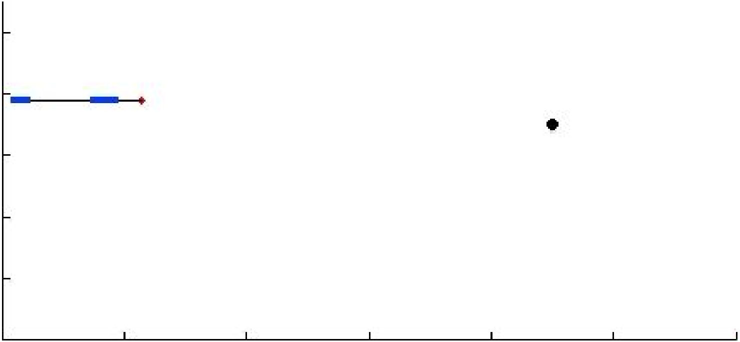
Glider Perching

- Enters motion capture @ 6 m/s.
- Perch is < 3.5 m away.
- Entire trajectory @ 1 second.

Requires Separation!

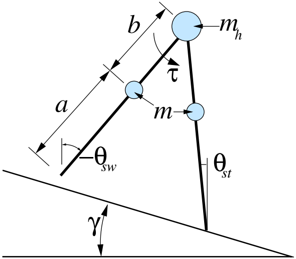
[CLICK IMAGE TO PLAY MOVIE]

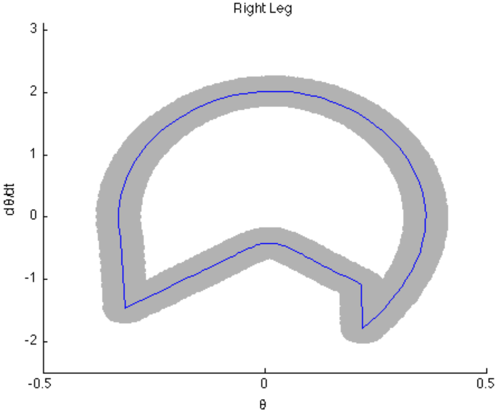
Preliminary results: Trees for Perching



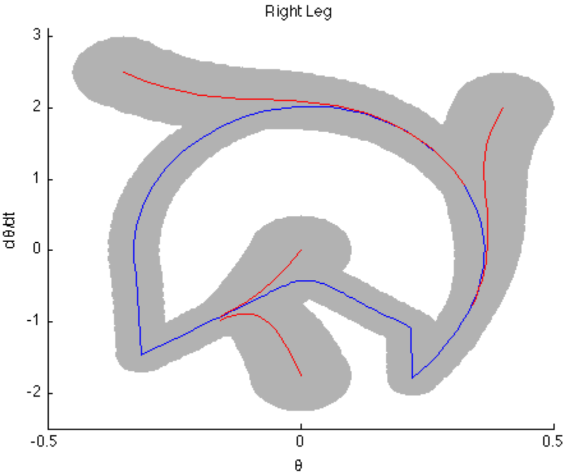
[CLICK IMAGE TO PLAY MOVIE]

Cartoon: LQR-Trees for bipedal walking





Cartoon: LQR-trees for bipedal walking



Extensions to the basic algorithm

- Can replace LQR with favorite trajectory stabilizer

Extensions to the basic algorithm

- Can replace LQR with favorite trajectory stabilizer
  - Stochastic basins requires more thought, but not impossible

Extensions to the basic algorithm

- Can replace LQR with favorite trajectory stabilizer
  - Stochastic basins requires more thought, but not impossible
  - LQR is convenient because it yields cost-to-go

Extensions to the basic algorithm

- Can replace LQR with favorite trajectory stabilizer
  - Stochastic basins requires more thought, but not impossible
  - LQR is convenient because it yields cost-to-go

- Certificates for more complicated problems (e.g., actuator constraints)

Extensions to the basic algorithm

- Can replace LQR with favorite trajectory stabilizer
  - Stochastic basins requires more thought, but not impossible
  - LQR is convenient because it yields cost-to-go

- Certificates for more complicated problems (e.g., actuator constraints)
  - Can replace some pieces of SOS with direct simulation without losing coverage

Extensions to the basic algorithm

- Can replace LQR with favorite trajectory stabilizer
  - Stochastic basins requires more thought, but not impossible
  - LQR is convenient because it yields cost-to-go
- Certificates for more complicated problems (e.g., actuator constraints)
  - Can replace some pieces of SOS with direct simulation without losing coverage
- Multi-query algorithms.
  - Backwards tree is big stabilizing web of trajectories.
  - Reuse funnel computation when goal changes.

Extensions to the basic algorithm

- Can replace LQR with favorite trajectory stabilizer
  - Stochastic basins requires more thought, but not impossible
  - LQR is convenient because it yields cost-to-go
- Certificates for more complicated problems (e.g., actuator constraints)
  - Can replace some pieces of SOS with direct simulation without losing coverage
- Multi-query algorithms.
  - Backwards tree is big stabilizing web of trajectories.
  - Reuse funnel computation when goal changes.
- Tentative: Combine with policy-gradient methods to adjust to model errors

Scaling

Scaling

- Randomized sampling scales

Scaling

- Randomized sampling scales
- Trajectory optimization scales

Scaling

- Randomized sampling scales
- Trajectory optimization scales
- LQR design scales

Scaling

- Randomized sampling scales
- Trajectory optimization scales
- LQR design scales
- Polynomial expansion for certificates is expensive

Scaling

- Randomized sampling scales
- Trajectory optimization scales
- LQR design scales
- Polynomial expansion for certificates is expensive
- In high dimensions, cover only relevant portion of state space

Scaling

- Randomized sampling scales
- Trajectory optimization scales
- LQR design scales
- Polynomial expansion for certificates is expensive
- In high dimensions, cover only relevant portion of state space
  - Can add to the tree during execution (multi-query)

Scaling

- Randomized sampling scales
- Trajectory optimization scales
- LQR design scales
- Polynomial expansion for certificates is expensive
- In high dimensions, cover only relevant portion of state space
  - Can add to the tree during execution (multi-query)

- Currently demonstrated in 5 dimensions (same as dynamic programming) with virtually no optimization

Scaling

- Randomized sampling scales
- Trajectory optimization scales
- LQR design scales
- Polynomial expansion for certificates is expensive
- In high dimensions, cover only relevant portion of state space
  - Can add to the tree during execution (multi-query)

- Currently demonstrated in 5 dimensions (same as dynamic programming) with virtually no optimization
- Goal is @ 10 dimensions. Time will tell.

Summary and Conclusions

Summary and Conclusions

- Trajectory libraries are a good way to systematically design nonlinear controllers using linear control.

Summary and Conclusions

- Trajectory libraries are a good way to systematically design nonlinear controllers using linear control.
- It pays to reason about the funnels as you plan:
  - Efficient - thanks to new tools from verification
  - Sparseness - relatively few trajectories required
  - Stronger guarantees - "probabilistic feedback coverage"