Quadruped Robots and Legged Locomotion

J. Zico Kolter
Computer Science Department
Stanford University

Joint work with Pieter Abbeel, Andrew Ng

Why legged robots?

Why Legged Robots?

"There is a need for vehicles that can travel in difficult terrain, where existing vehicles cannot go … Only about half of the earth's landmass is accessible to existing wheeled and tracked vehicles, whereas a much larger fraction can be reached by animals on foot."

– Marc Raibert, *Legged Robots that Balance*, 1986

Why Legged Robots?

Why Legged Robots?

… but, we aren't quite there yet with legged robots.

The Potential Versus the Reality

The Potential Versus the Reality

"… Although we take motivation from the need to travel on rough terrain, the running experiments reported here have not yet ventured beyond our very flat laboratory floor."

– Marc Raibert, *Legged Robots that Balance*, 1986

Hardware Versus Software

- Although inferior to biological animals, current legged robot *hardware* is very capable
- The challenge is designing *software* to realize this potential

The LittleDog robot, designed and built by Boston Dynamics, Inc.

The Quadruped Locomotion Task

The Quadruped Locomotion Task

- Our goal is to design a software system that enables a quadruped robot to climb over a wide variety of challenging, previously unseen terrain

The Quadruped Locomotion Task

- Our goal is to design a software system that enables a quadruped robot to climb over a wide variety of challenging, previously unseen terrain

The Quadruped Locomotion Task

- Two distinct subtasks of overall problem:

Perception

Using vision systems, build a model of the terrain in front of the robot and determine position of the robot in this model

Control

Generate a sequence of control inputs (i.e., commands to robot's joints) that move the robot over the terrain

The Quadruped Locomotion Task

- Two distinct subtasks of overall problem:

Perception

Using vision systems, build a model of the terrain in front of the robot and determine position

Use motion capture system and scanned models of terrain

Control

Generate a sequence of control inputs (i.e., commands to robot's joints) that move the robot over the terrain

Control Task

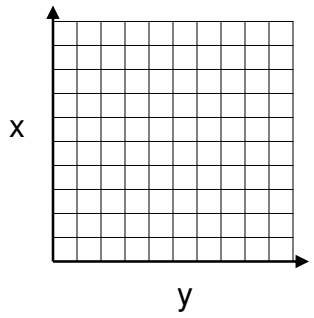
Control

Generate a sequence of control inputs (i.e., commands to robot's joints) that move the robot over the terrain

18 dimensional state space
(3-D position, 3-D orientation,
12-D joint angles)

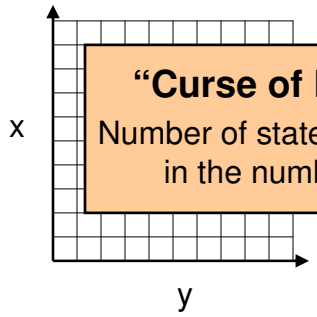
Control Task

- How do we apply dynamic programming to large, continuous state spaces?
- Simple method: discretize the state space



Control Task

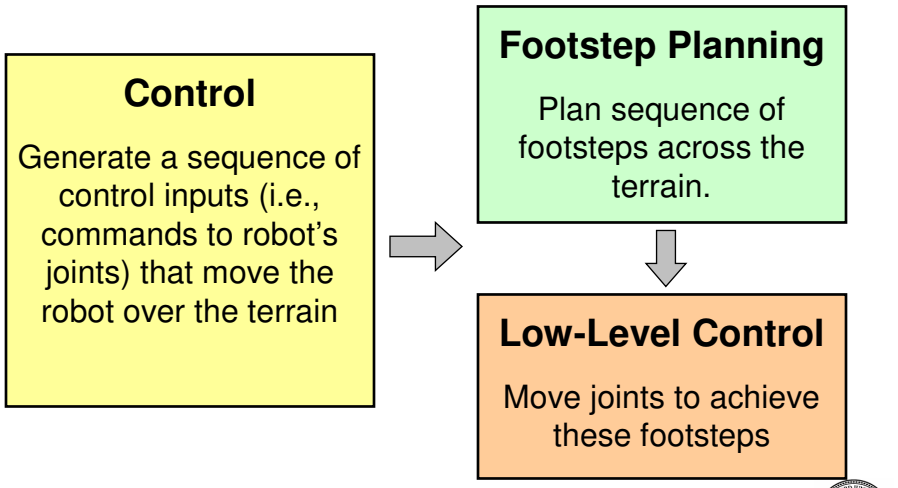
- How do we apply dynamic programming to large, continuous state spaces?
- Simple method: discretize the state space



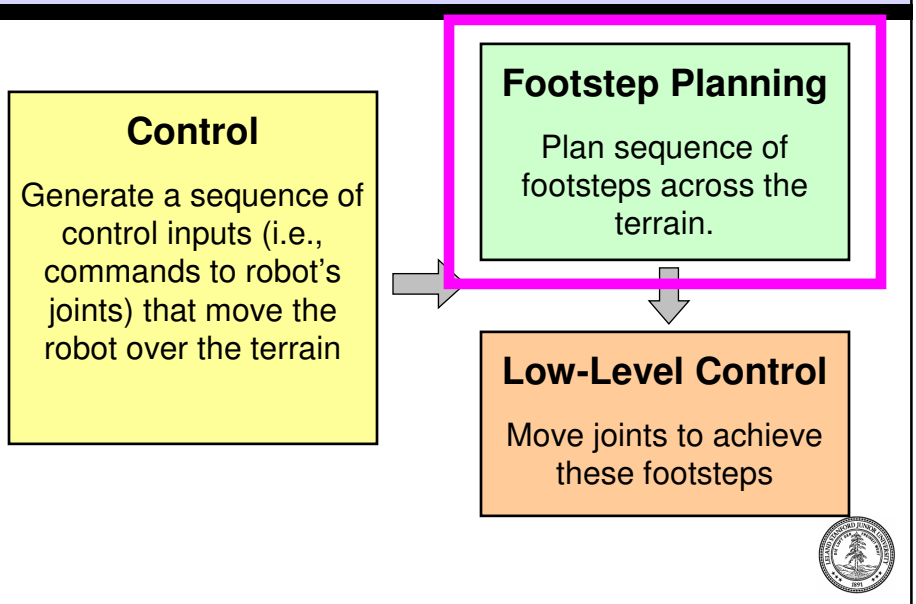
"Curse of Dimensionality"

Number of states grows exponentially in the number of dimensions

Control Task



Control Task



Footstep Planning via Value Iteration

The Footstep Planning Problem

- Given an initial position, a goal position, and a model of the terrain, plan footsteps that move the robot to the goal

The Footstep Planning Problem

Outline of approach:

Frame footstep planning problem as a Markov Decision Process, and use Value Iteration to plan footsteps

- s

MDP Review

- Markov Decision Process (MDP):

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

MDP Review

- Markov Decision Process (MDP):

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

Set of states

MDP Review

- Markov Decision Process (MDP):

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

Set of states

Set of actions

MDP Review

- Markov Decision Process (MDP):

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

Set of states

Set of actions

System dynamics

MDP Review

- Markov Decision Process (MDP):

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

Set of states

Set of actions

Discount factor

System dynamics

MDP Review

- Markov Decision Process (MDP):

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

Set of states

Set of actions

System dynamics

Discount factor

Initial state distribution

MDP Review

- Markov Decision Process (MDP):

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

Set of states

Set of actions

System dynamics

Discount factor

Initial state distribution

Reward function

State Space

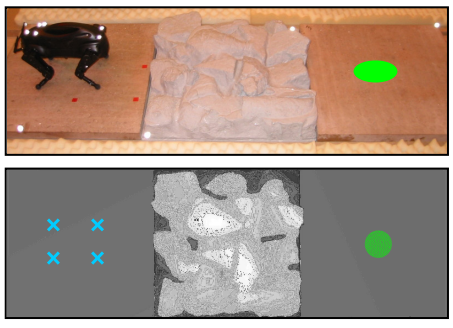
$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

Set of states

State Space

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

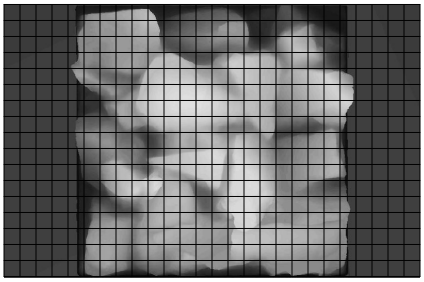
- For footstep planning, state is X-Y location of the feet on terrain



State $\in \mathbb{R}^8 =$
(front-left-x, front-left-y,
front-right-x, front-right-y,
back-left-x, back-left-y,
back-right-x, back-right-y)

State Space

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$



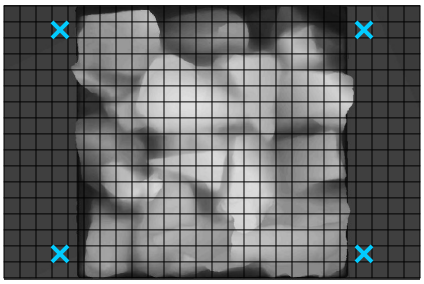
- Discretize terrain (e.g. 3cm grid squares)
- For 60cm x 60cm terrain:

$$|\mathcal{S}| = 20^8 \approx 2.5 \times 10^{10}$$

State Space

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

- But not all footstep combinations possible



State Space

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

- But not all footstep combinations possible

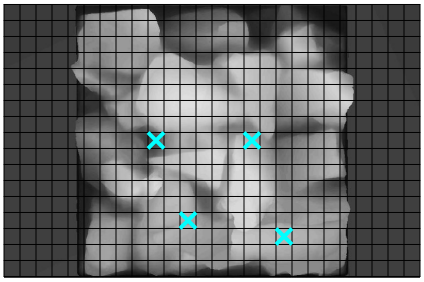
Robot Kinematics

- Problem: “Natural” robot foot state is joint positions, but we want Cartesian coordinates

- **Forward Kinematics:** convert from joint angles to 3-D coordinates of the foot
- **Inverse Kinematics:** convert from 3-D coordinates of foot to joint angles (or indicate that foot location is infeasible)

State Space

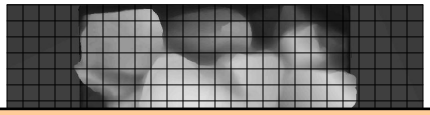
$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$



- To determine if footsteps feasible:
  - Pick location for body (e.g., center of feet)
  - Inverse kinematics to see if all feet feasible

State Space

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$



With a few additional modifications, reduces state space to ~1 million, suitable for Value Iteration

- To deter
  - Pick location for body (e.g., center of feet)
  - Inverse kinematics to see if all feet feasible

Action Space

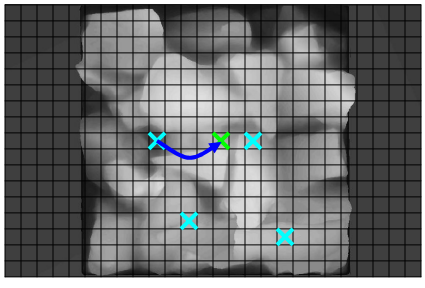
$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

Set of actions

Action Space

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

- Move one foot at a time



Action =
(foot, new-x, new-y)

- For 60cm x 60cm terrain:
$$|\mathcal{A}| = 4(20^2) = 1600$$

System Dynamics

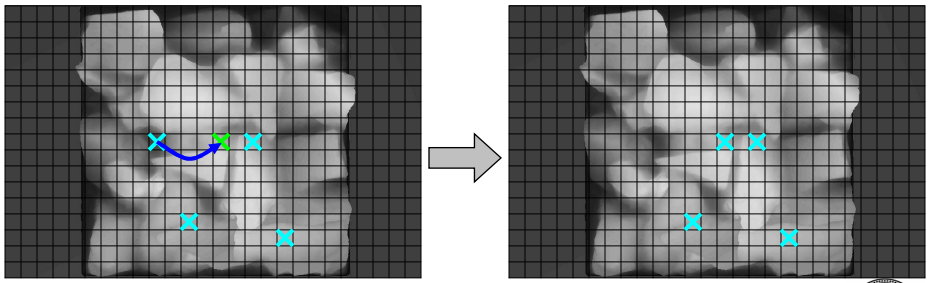
$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

System dynamics

System Dynamics

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

- If initial and next states are both feasible, then action succeeds, fails otherwise

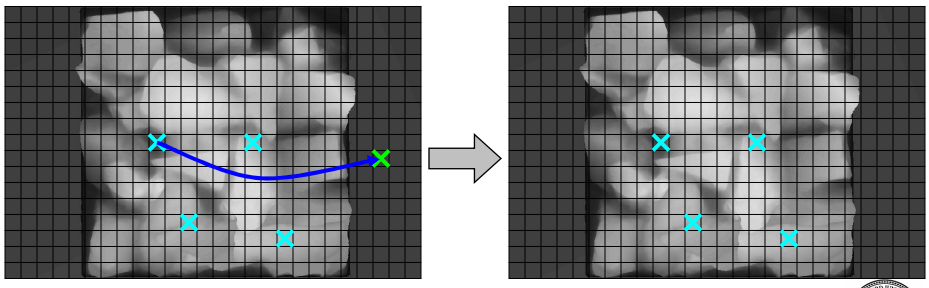


Valid Action

System Dynamics

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

- If initial and next states are both feasible, then action succeeds, fails otherwise



Invalid Action

System Dynamics

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

Discount factor

Discount Factor

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

$$\gamma = 1$$

- No discount factor, corresponds to **shortest path problem**
- Converges for non-positive reward in all states, zero reward in goal states

Initial State Distribution

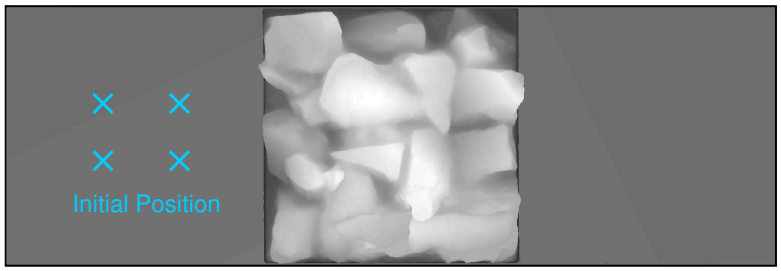
$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

Initial state distribution

Initial State Distribution

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

- Initial state distribution contains only the initial pose of the robot (no stochasticity)



Initial State Distribution

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

Reward function

Reward Function

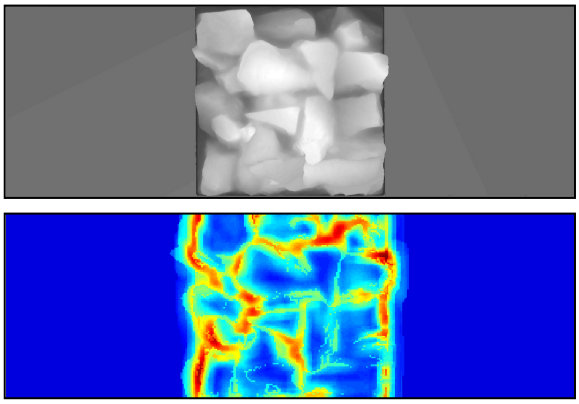
$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

- Footsteps must trade off different features
  - Slope of terrain, proximity to drop-offs, stability of robot's pose, etc.
- (Negative) reward function specifies relative weights for these features

Reward Function

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

- Example (cost for a single footstep):



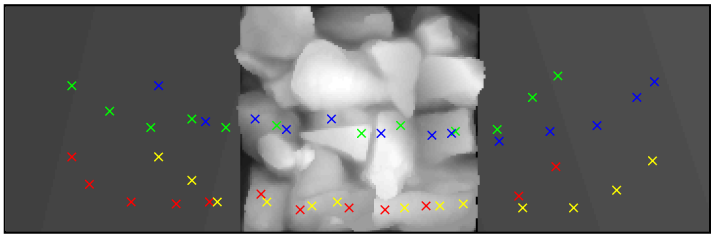
Value Iteration

- Fully defined MDP

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$$

- Run value iteration to plan footsteps

$$V(s) \leftarrow \mathcal{R}(s) + \gamma \max_a \sum_{s'} P(s'|s,a) V(s')$$



Performance

System without planned footsteps

Performance

System after planning footsteps

Another Terrain

System without planned footsteps

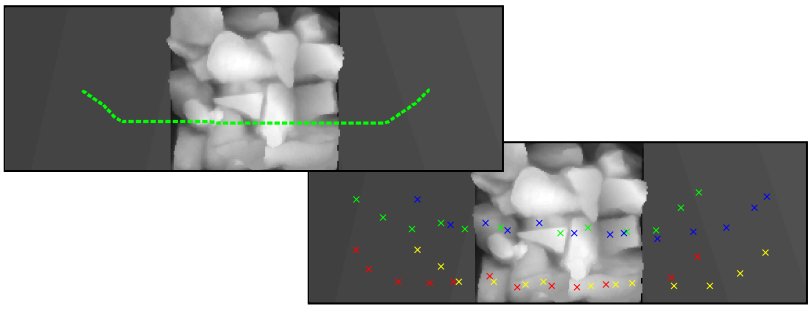
Another Terrain

System after planning footsteps

Extensions and Related Topics

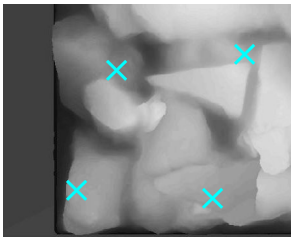
Extensions

- Problem: Number of states grows too large with more terrain, finer resolution
- Solution: Plan a general path for the body, then plan footsteps along path



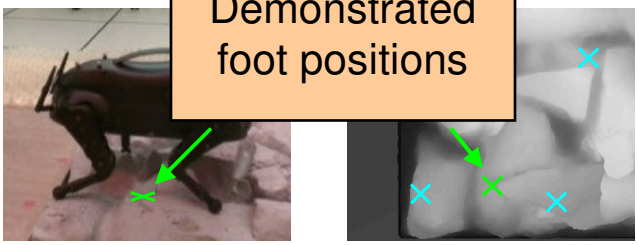
Extensions

- Problem: Reward function needs to trade off many features, hard to hand-specify
- Solution: Learn reward by *demonstrating* good footsteps ("Apprenticeship Learning")



Extensions

- Problem: Reward function needs to trade off many features, hard to hand-specify
- Solution: Learn reward by *demonstrating* good footsteps ("Apprenticeship Learning")



Control Task

Control

Generate a sequence of control inputs (i.e., commands to robot's joints) that move the robot over the terrain

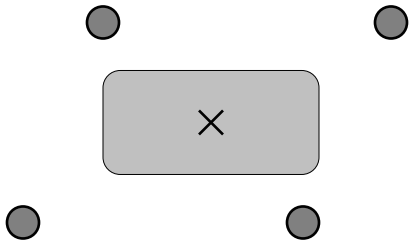
Footstep Planning

Plan sequence of footsteps across the terrain.

Low-Level Control

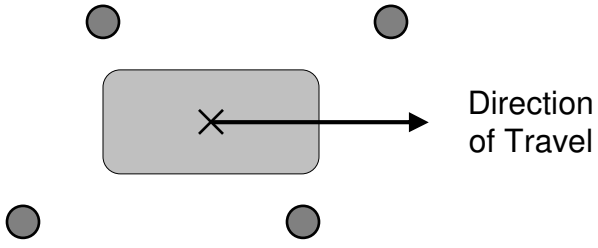
Move joints to achieve these footsteps

Low-Level Control



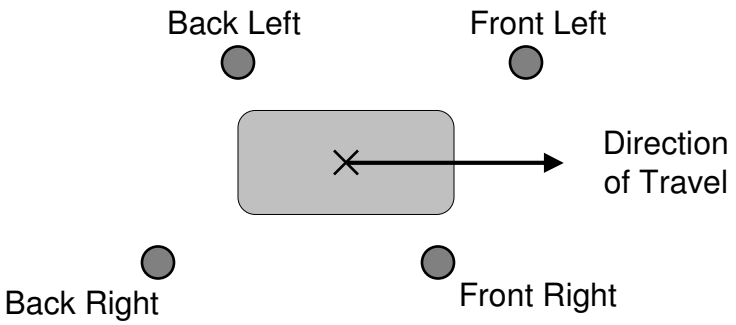
Initial setup of the robot

Low-Level Control



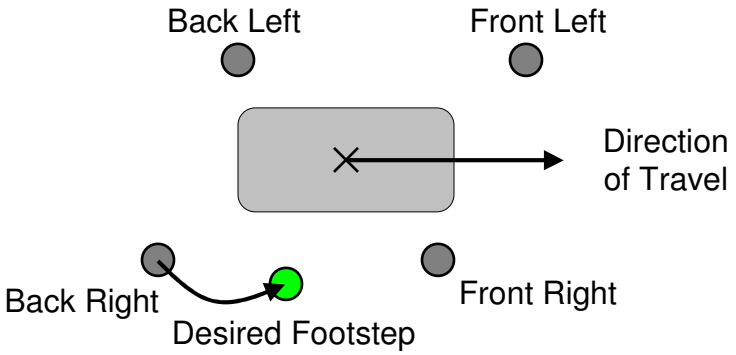
Initial setup of the robot

Low-Level Control



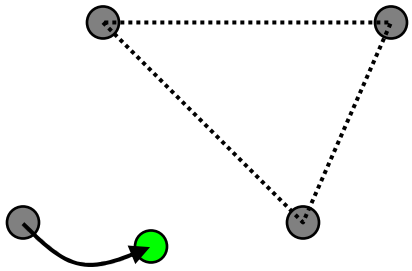
Initial setup of the robot

Low-Level Control



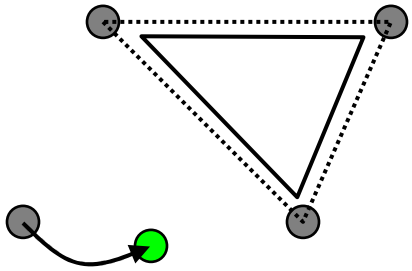
Initial setup of the robot

Low-Level Control



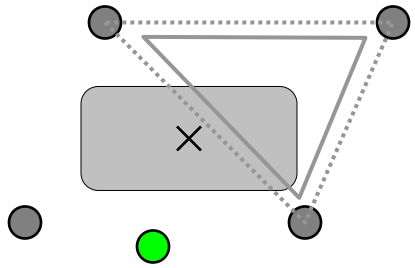
- **Supporting triangle**: If robot's center of gravity (COG) in this triangle, will not fall

Low-Level Control



- **Supporting triangle**: If robot's center of gravity (COG) in this triangle, will not fall

Low-Level Control



- First move COG into supporting triangle
- Then move foot

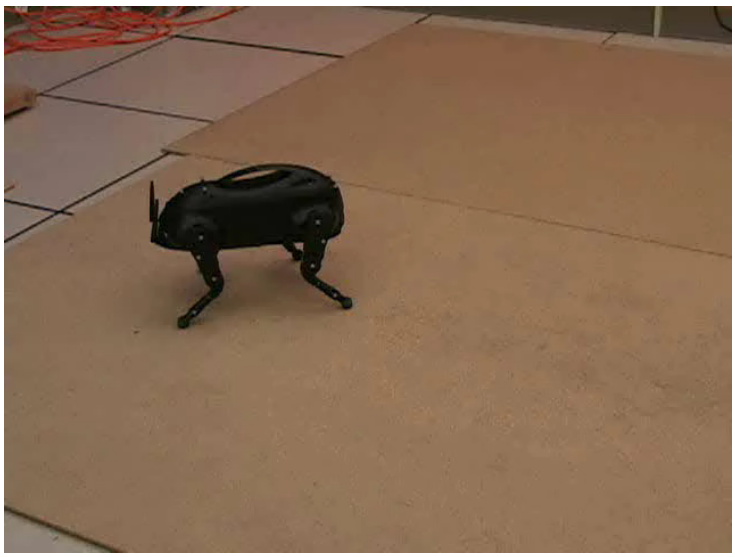
Fast Movement on Flat Ground

- Switching gears: previously focused on slow motion over challenging terrain, now looking at fast motion on flat ground
- To achieve faster speed, want to move two feet at once (trot gait)
  - Primary challenge is ***balance***: when only two feet are on the ground, robot is always falling

Learning to Balance

- Want to move robot's center of gravity to keep it as stable as possible
- But, very hard to hand-specify, a priori, a good location for the center of gravity
- ***Learning***: find a good location for the center of gravity by adjusting it in response to robot performance

Learning to Balance



References

- Kolter, Rodgers and Ng, *A Control Architecture for Quadruped Locomotion over Rough Terrain*, ICRA 2008
- Kolter, Abbeel and Ng, *Hierarchical Apprenticeship Learning with Application to Quadruped Locomotion*, NIPS 2008
- Kolter and Ng, *Learning Omnidirectional Path Following Using Dimensionality Reduction*, RSS 2007

Thank you

Papers and videos available at:
http://cs.stanford.edu/groups/littledog