Control of the Compass Gait on Rough Terrain

Katie Byl and Russ Tedrake
Motivation

How capable can an underactuated, dynamic walking approach be on rough terrain?

- Dynamic walking:
  - Natural dynamics
  - Likely to be efficient
- But unfortunately…
  - Notoriously sensitive

Long-range goals:
- Implement on real robot
- On-line learning
Motivation

- Process toward obtaining **underactuated, dynamic walking on rough terrain**:

  1. Use minimal actuation and control strategies
    - underactuation at toe
Motivation

- Process toward obtaining underactuated, dynamic walking on rough terrain:
  1. Use minimal actuation and control strategies
     - underactuation at toe
  2. Quantify performance in stochastic environments
Motivation

- Process toward obtaining underactuated, dynamic walking on rough terrain:
  1. Use minimal actuation and control strategies
     ▪ underactuation at toe
  2. Quantify performance in stochastic environments
  3. Iterate to optimize performance
     ▪ long-living, metastable dynamics
Overview

- **Essential model for dynamic walking on rough terrain:**
  - Hip-actuated compass gait (CG) with leg inertia
  - Passive toe pivot

- **Outline:**
  - Passive walker example
  - Actuated walkers:
    - Stochastic terrain
    - Known, wrapping terrain
Overview

- Essential model for dynamic walking on rough terrain:
  - Hip-actuated compass gait (CG) with leg inertia
  - Passive toe pivot

- Outline:
  - Passive walker example
  - Actuated walkers:
    - Stochastic terrain
    - Known, wrapping terrain

Katie Byl and Russ Tedrake – Control of the Compass Gait on Rough Terrain
Passive Walker

- **Unactuated, with stochastic downhill terrain**

\[
\begin{align*}
  m &= 5 \text{ kg} \\
  m_h &= 1.5 \text{ kg} \\
  a &= 0.7 \text{ m} \\
  b &= 0.3 \text{ m}
\end{align*}
\]

The Compass Gait Walker

Walker animation: \( \gamma_{av} = 4^\circ, \sigma = 1.0^\circ \)

(Only first 5 sec will be animated...)
Passive Walker

- Constant 4° downhill slope (no noise)

Slices of the deterministic Basins of Attractors for the walkers analyzed for passive (left) and controlled (right) examples throughout.

Katie Byl and Russ Tedrake – Control of the Compass Gait on Rough Terrain
Passive Walker

- Constant 4° downhill slope (no noise)

Slice of the deterministic **Basins of Attraction** for the walker analyzed for **passive** examples throughout.

Katie Byl and Russ Tedrake – Control of the Compass Gait on Rough Terrain
Passive Walker

- **Stochastic downhill terrain, mean slope = 4°**

\[ \sigma = 0.5^\circ \]
\[ \text{mfpt} \approx 200,000 \]

\[ \sigma = 1.0^\circ \]
\[ \text{mfpt} \approx 300 \]

(*mfpt*: mean first-passage time)
Passive Walker

- **Stochastic downhill terrain, mean slope = 4°**

\[
\sigma = 0.5° \\
mfpt \approx 200,000
\]

\[
\sigma = 1.0° \\
mfpt \approx 300
\]

(mfpt: mean first-passage time)
Passive Walker

- Stochastic downhill terrain, mean slope = 4°

\[ \sigma = 0.5^\circ \]
\[ \text{mfpt} \approx 200,000 \]

\[ \sigma = 1.0^\circ \]
\[ \text{mfpt} \approx 300 \]

(mfpt : mean first-passage time)
Actuated Walker Models

- **Compass gait (CG)**
  - **Point masses** at hip \( (m_h) \) and on each leg \( (m) \)
    - \( m = m_h = 2 \text{ kg} \); \( a = b = 0.5 \text{ m} \)
  - **Passive pivot model** for “toe” of stance leg
  - **5 States:** \( \theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2, \Delta z \)
  - **Instantaneous, inelastic collisions**

- **Actuations**
  - **Torque at hip:**
    - +/- 15 N-m limit
  - **Pre-collision impulse:**
    - Constant value of 2 kg-m/s
Methodology

- **Solve iteratively to find optimal policy**
  - **Mesh** state space, using **post-collision** states
  - Define cost function to *reward continuous walking*
Methodology

- **Solve iteratively to find optimal policy**
  - Mesh state space, using post-collision states
  - Define cost function to reward continuous walking

- **Hierarchical control**
  - Low-level PD control: \( \tau = K_p (\alpha_{des} - \alpha) - K_d \dot{\alpha} \)
  - High-level, once-per-step selection of \( \alpha_{des} \)
Methodology

- **Solve iteratively to find optimal policy**
  - **Mesh** state space, using **post-collision** states
  - Define cost function to **reward continuous walking**

- **Hierarchical control**
  - Low-level PD control: \( \tau = K_p (\alpha_{des} - \alpha) - K_d \dot{\alpha} \)
  - High-level, **once-per-step** selection of \( \alpha_{des} \)

- **Additional Details**
  - Stochastic terrain, \( \Delta z \) from a Gaussian
  - Swing toe **retracts** until \( \alpha \) is within 10° of \( \alpha_{des} \)
  - PD controller is always active during step
Low-level PD Control at Hip

- PD state trajectories versus passive downhill walking

Note: While positive and negative work is done for active case, overall gait speed is only about 10% faster than passive walker.

PD control only, with no impulsive toe-off:

\[ \alpha_{\text{des}} = 35^\circ \]

Constant 4° downhill, to compare active with passive
Meshing: stochastic terrain

- **Post-collision meshing** using 4 state variables

\[
\begin{align*}
X_{m1} &= \Delta z = z_{st} - z_{sw} \\
X_{m2} &= -\Delta x = x_{sw} - x_{st} \\
X_{m3} &= \dot{\theta}_1 \\
X_{m4} &= \dot{\theta}_2
\end{align*}
\]

- Including one extra “fallen” state, there are **19,001 mesh states**

<table>
<thead>
<tr>
<th>state</th>
<th># elem’s</th>
<th>min</th>
<th>max</th>
<th>units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_{m1}$</td>
<td>19</td>
<td>-.01</td>
<td>.01</td>
<td>(m)</td>
</tr>
<tr>
<td>$X_{m2}$</td>
<td>10</td>
<td>-0.7</td>
<td>-0.16</td>
<td>(m)</td>
</tr>
<tr>
<td>$X_{m3}$</td>
<td>10</td>
<td>-2.1</td>
<td>-1.1</td>
<td>(rad/s)</td>
</tr>
<tr>
<td>$X_{m4}$</td>
<td>10</td>
<td>-1.0</td>
<td>1.5</td>
<td>(rad/s)</td>
</tr>
</tbody>
</table>

- **Action**, $\alpha_{des}$ : 15 - 40 deg (11 values)

- **Interpolation** (barycentric)
Dynamic Programming (Value Iteration)

- **Pre-compute one-step dynamics**
  - Each new state in $N$-dim space represented by $N+1$ weighted mesh nodes, each with weight $W_k$
Dynamic Programming (Value Iteration)

- **Pre-compute one-step dynamics**
  Each new state in $N$-dim space represented by
  $N+1$ weighted mesh nodes, each with weight $W_k$

- **Define one-step cost; initialize** $C_{\text{last}} = C_{\text{onestep}}$

$$C_{\text{onestep}}(i) = \begin{cases} -1, & i \notin \text{fallen} \\ 0, & i \in \text{fallen} \end{cases}$$
Dynamic Programming (Value Iteration)

- **Pre-compute one-step dynamics**
  Each new state in $N$-dim space represented by $N+1$ weighted mesh nodes, each with weight $W_k$

- **Define one-step cost; initialize** $C_{\text{last}} = C_{\text{onestep}}$

  $C_{\text{onestep}}(i) = \begin{cases} 
  -1, & i \notin \text{fallen} \\
  0, & i \in \text{fallen} 
  \end{cases}$

  One-step cost of -1 maximizes steps taken before falling.

  To maximize distance traveled, instead use: $C_{\text{onestep}}(i) = X_{m2}$
Dynamic Programming (Value Iteration)

- **Pre-compute one-step dynamics**
  Each new state in $N$-dim space represented by $N+1$ weighted mesh nodes, each with weight $W_k$

- **Define one-step cost; initialize $C_{\text{last}} = C_{\text{onestep}}$**

  $$C_{\text{onestep}}(i) = \begin{cases} -1, & i \notin \text{fallen} \\ 0, & i \in \text{fallen} \end{cases}$$

- **Iterate to minimize cost:**

  $$C_{\text{new}}(i \mid a) = \sum_{k=1}^{5} W_k \left[ \gamma \cdot C_{\text{last}}(k) + C_{\text{onestep}}(k) \right]$$

  Iterative updates:

  $$C_{\text{last}}(i) = C_{\text{new}}(i), \quad \forall i \quad \quad \quad \pi(i) = \arg \min_a C_{\text{new}}(i \mid a)$$

  One-step cost of -1 maximizes steps taken before falling.

  To maximize distance traveled, instead use: $C_{\text{onestep}}(i) = X_{m2}$

  $\gamma = 0.9$
Dynamic Programming (Value Iteration)

- **Pre-compute one-step dynamics**
  Each new state in \( N\)-dim space represented by \( N+1 \) weighted mesh nodes, each with weight \( W_k \)

- **Define one-step cost; initialize** \( C_{\text{last}} = C_{\text{onestep}} \)

\[
C_{\text{onestep}}(i) = \begin{cases} 
-1, & i \notin \text{fallen} \\
0, & i \in \text{fallen} 
\end{cases}
\]

- **Iterate to minimize cost:**

\[
C_{\text{new}}(i \mid a) = \sum_{k=1}^{5} W_k \left[ \gamma \cdot C_{\text{last}}(k) + C_{\text{onestep}}(k) \right]
\]

Iterative updates:

\[
C_{\text{last}}(i) = C_{\text{new}}(i), \quad \forall i \quad \quad \quad \pi(i) = \arg \min_a C_{\text{new}}(i \mid a)
\]

One-step cost of -1 maximizes steps taken before falling.

To maximize distance traveled, instead use: \( C_{\text{onestep}}(i) = X_m^2 \)

\( \gamma = 0.9 \)

Katie Byl and Russ Tedrake – Control of the Compass Gait on Rough Terrain
Mean first-passage time, MFPT, used to quantify stability

One-step look-ahead improves policy significantly

Control on Stochastic Terrain
Control on Stochastic Terrain

Mean first-passage time, MFPT, used to quantify stability

One-step look-ahead improves policy significantly

12,000 steps (one-step look)

76 steps (no look-ahead)
Control on Wrapping Terrain

- **For stochastic terrain:**
  - N-step look-ahead requires $4+N$ total mesh dimensions

- **Advantages of known, wrapping terrain:**
  - Allows N-step look-ahead using **only** 4 mesh dimensions (4D)
  - N steps occur in iteration algorithm, not state representation
Meshing: known, wrapping terrain

- **Post-collision meshing** using 4 state variables

\[
\begin{align*}
X_{m1} &= x_{st} \\
X_{m2} &= \Delta x = x_{st} - x_{sw} \\
X_{m3} &= \dot{\theta}_1 \\
X_{m4} &= \dot{\theta}_2
\end{align*}
\]

Including one extra “fallen” state, there are **411,601 mesh states**

<table>
<thead>
<tr>
<th>state</th>
<th># elem’s</th>
<th>min</th>
<th>max</th>
<th>units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_{m1}$</td>
<td>140</td>
<td>0</td>
<td>7</td>
<td>(m)</td>
</tr>
<tr>
<td>$X_{m2}$</td>
<td>15</td>
<td>-0.85</td>
<td>-0.15</td>
<td>(m)</td>
</tr>
<tr>
<td>$X_{m3}$</td>
<td>14</td>
<td>-3.0</td>
<td>-0.4</td>
<td>(rad/s)</td>
</tr>
<tr>
<td>$X_{m4}$</td>
<td>14</td>
<td>-0.1</td>
<td>5.1</td>
<td>(rad/s)</td>
</tr>
</tbody>
</table>

- **Action**, $\alpha_{des}$ : 10 - 40 deg (13 values)

- **Interpolation** (barycentric)
Meshing: known, wrapping terrain

- **Post-collision meshing** using 4 state variables

  \[ X_{m1} = x_{st} \]
  \[ X_{m2} = \Delta x = x_{st} - x_{sw} \]
  \[ X_{m3} = \dot{\theta}_1 \]
  \[ X_{m4} = \dot{\theta}_2 \]

  Only 1st state variable is different from stochastic modeling case

  Including one extra “fallen” state, there are **411,601 mesh states**

<table>
<thead>
<tr>
<th>state</th>
<th># elem’s</th>
<th>min</th>
<th>max</th>
<th>units</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X_{m1})</td>
<td>140</td>
<td>0</td>
<td>7</td>
<td>(m)</td>
</tr>
<tr>
<td>(X_{m2})</td>
<td>15</td>
<td>-0.85</td>
<td>-0.15</td>
<td>(m)</td>
</tr>
<tr>
<td>(X_{m3})</td>
<td>14</td>
<td>-3.0</td>
<td>-0.4</td>
<td>(rad/s)</td>
</tr>
<tr>
<td>(X_{m4})</td>
<td>14</td>
<td>-0.1</td>
<td>5.1</td>
<td>(rad/s)</td>
</tr>
</tbody>
</table>

- **Action**, \(\alpha_{des}\) : 10 - 40 deg (13 values)

- **Interpolation** (barycentric)
Results on Wrapping Terrain

- **PD with impulsive toe-off**
  - $\alpha$ is desired interleg angle

\[\alpha_{\text{min}} = 15^\circ, \; \alpha_{\text{max}} = 40^\circ\]

\[\alpha_{\text{min}} = 22^\circ, \; \alpha_{\text{max}} = 40^\circ\]

First 10 seconds of data
Results on Wrapping Terrain

- **PD with impulsive toe-off**
  - Gaps yield more pattern in footholds

\[
\alpha_{\text{min}} = 15^\circ, \ \alpha_{\text{max}} = 40^\circ
\]

\[
\alpha_{\text{min}} = 22^\circ, \ \alpha_{\text{max}} = 40^\circ
\]

First 3 seconds of data
Discussion: One-step policy

- Using heuristic cost functions on the wrapping mesh state also yields impressive results
  - Implies lengthy value iteration computation and/or exact description of terrain are not essential
- Although surprisingly good, one-step policy is inferior
  - Performance sensitive to one-step heuristic used

Animations below use only slightly different one-step heuristics…
Future Work

- Use off-line policy from simulation as basis for **on-line policy learning on real robot**
  - Direct-drive hip torque
  - Retracting toe
  - Motor encoder
  - Boom-mounted
    - Repeating terrain
  - Motion capture:
    - Leg markers
    - Terrain markers
- Maximize **expected number of steps taken**
Summary

- Compass gait model with *hip torque* and *toe impulse* can negotiate qualitatively rough terrain
Summary

- Compass gait model with hip torque and toe impulse can negotiate qualitatively rough terrain

- Apply analytical tools toward creating metastable locomotion
Summary

- Compass gait model with hip torque and toe impulse can negotiate qualitatively rough terrain

- Apply analytical tools toward creating metastable locomotion

- One-step look-ahead greatly improves performance
Summary

- Compass gait model with hip torque and toe impulse can negotiate qualitatively rough terrain

- Apply analytical tools toward creating metastable locomotion

- One-step look-ahead greatly improves performance

- What is possible if better low-level control is used?!
Summary

- Compass gait model with hip torque and toe impulse can negotiate qualitatively rough terrain
- Apply analytical tools toward creating metastable locomotion
- One-step look-ahead greatly improves performance
- What is possible if better low-level control is used?!?
- Same approach already shown to work on known, wrapping terrain: Byl and Tedrake, ICRA 2008, link to ICRA 2008 paper
- Metastable walking described further in upcoming work: Byl and Tedrake, RSS 2008, link to RSS 2008 paper
Questions?
Additional slides

- Details on eigenanalysis of discrete system
- More results on known, wrapping terrain
- Important details on interpolation method
- Fragility of impulse-only strategy
- Dynamic motion planning for a stiff robot
Eigenanalysis

- Discretized system is a Markov chain
  - Analyze corresponding transition matrix

\[
m_i = \sum_j f_{ij}m_j + 1 \rightarrow (I - f')m = 1 \rightarrow m = (I - f')^{-1}1
\]

Katie Byl and Russ Tedrake – Control of the Compass Gait on Rough Terrain
Eigenanalysis

- Discretized system is a Markov chain
  - Analyze corresponding transition matrix

$f' \equiv \text{submatrix of } f \text{ that excludes the row and column of the absorbing failure (fallen) state.}$

\[
m_i = \sum_j f_{ij} m_j + 1
\]

\[
(I - f') m = 1
\]

\[
m = (I - f')^{-1} 1
\]
Results and Discussion

- Selecting only impulse magnitude (no PD) gives fragile results
- PD-only (used in examples below) works for mild or downhill terrain

Dots (wrapping) show previous footholds
Discussion: Interpolation

- **Method of interpolating optimal action is essential**
  - Interpolating between actions oftens fails
    - Small or large may be ok, while medium step fails:

  ![Small step OK](image1)
  ![Large step OK](image2)
  ![Interpolated step NOT OK](image3)

  *Watch for occasional steps into no-go zones in the animation below!*

- **Our solution:** simulate actual dynamics one step, then select action resulting in new state with lowest cost
Control on Stochastic Terrain

- One-step heuristic (below) on random (no-wrap) terrain

- Same optimization methodology can be applied using a stochastic (e.g. Gaussian) description of terrain
One-step on wrapping terrain

- Results in continuous walking here
Motivation

- **Passive-based walking is appealing for bipeds**
  - Captures fundamental, *pendular dynamics*
  - Seems likely to be **efficient**

- **Unfortunately, passive walkers are fragile!**
  - Notoriously sensitive to initial conditions and perturbations

![Graph showing leg length and step height](image)

Leg length = 1m

0.005m drop in .34m step, or about 1°
Underactuated stiff robots

- Interested in applying same stochastic modeling to other, higher DOF robots
  - 18 DOF (12 actuated, plus 6 DOF of body) LittleDog quadruped in dynamic, underactuated gaits and motions
  - Goal to learn policies which result in better stability

See movies here:

- http://people.csail.mit.edu/katiebyl/ld/jersey_barrier/jersey_with_pacing.mov
- people.csail.mit.edu/katiebyl/ld/newdog_terrainG/terrainG_newdog_withshove.mov