Learning to learn from simulation

How can we use simulations to learn faster on hardware?

Akshara Rai

Thesis Proposal 10/16/2017

Outline

1. Introduction

2. Domain knowledge in Bayesian optimization

   DoG feature transform

   Experiments on ATRIAS

3. Learning features from data

   Neural Network feature transform

   Experiments

4. Modelling mismatch between simulation and hardware

5. Conclusions and future work


Random Point for Trial 1
sampled to initialize BO
cost=97.57

How can we learn controllers for bipedal robots?

Robotics controllers often consist of expert-designed heuristics

- Feedback laws on positions and velocities
- Desired positions and trajectories

How can we learn controllers for bipedal robots?

Robotics controllers often consist of expert-designed heuristics

- Feedback laws on positions and velocities
- Desired positions and trajectories

These heuristics can be difficult to tune in higher dimensions

- Simulation does not transfer to hardware
- Hardware experiments are expensive

How can we learn controllers for bipedal robots?

Robotics controllers often consist of expert-designed heuristics

- Feedback laws on positions and velocities
- Desired positions and trajectories

These heuristics can be difficult to tune in higher dimensions

- Simulation does not transfer to hardware
- Hardware experiments are expensive

How can we learn controllers on hardware in very few trials?

How can we learn controllers for bipedal robots?

Robotics controllers often consist of expert-designed heuristics

- Feedback laws on positions and velocities
- Desired positions and trajectories

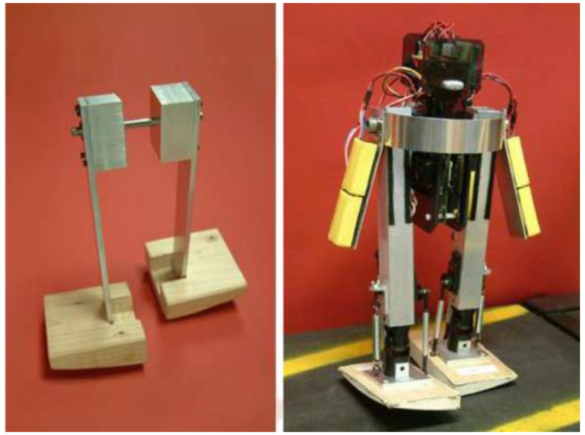
These heuristics can be difficult to tune in higher dimensions

- Simulation does not transfer to hardware
- Hardware experiments are expensive

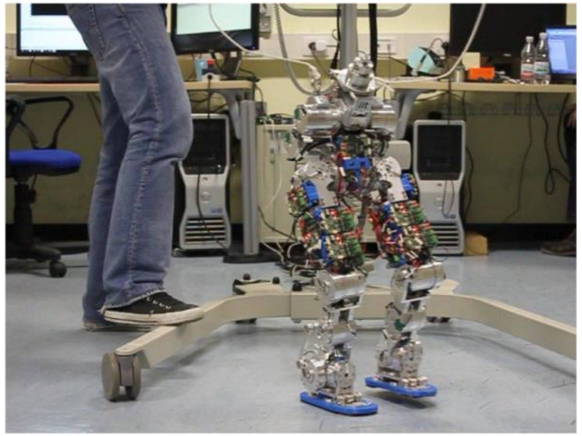
How can we learn controllers on hardware in very few trials?

We propose a two-step learning process – extract useful information from simulation and use it to learn faster on hardware

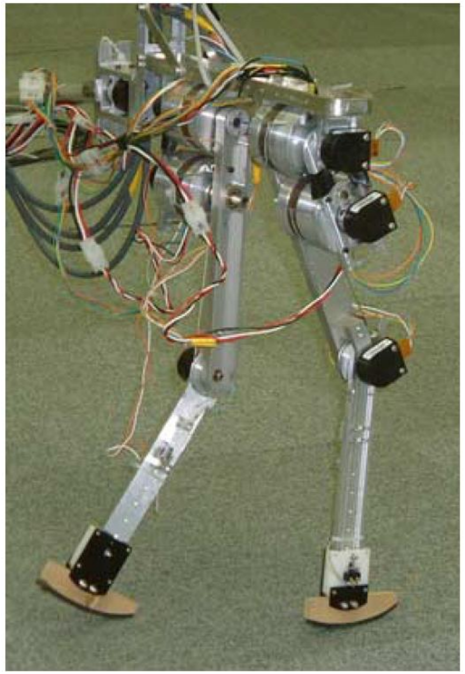
Learning for bipedal robots



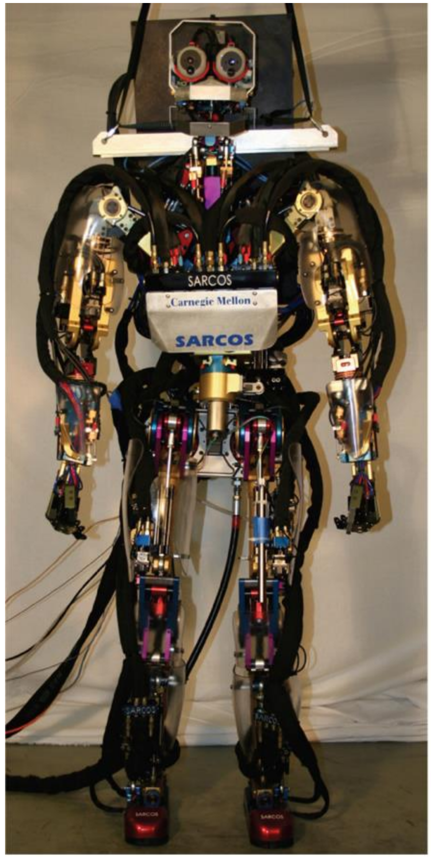
Tedrake, et al. 2004



Kormushev, et al. 2011



Morimoto, et al. 2005



Whitman, et al. 2013

Bayesian Optimization: A data-efficient optimization method

Define a cost $f(\boldsymbol{x})$, as a function of controller parameters $\boldsymbol{x}$

Bayesian Optimization: A data-efficient optimization method

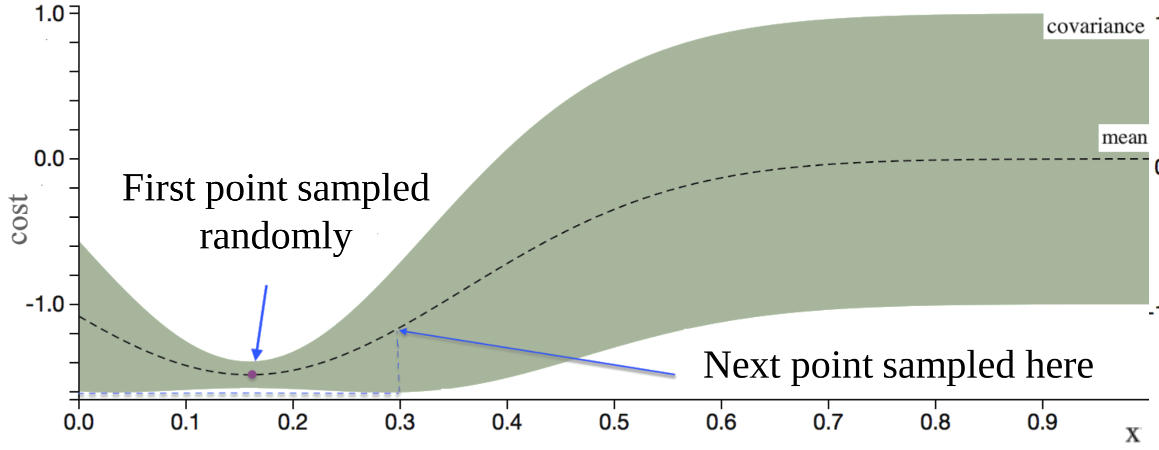
Define a cost $f(\boldsymbol{x})$, as a function of controller parameters $\boldsymbol{x}$

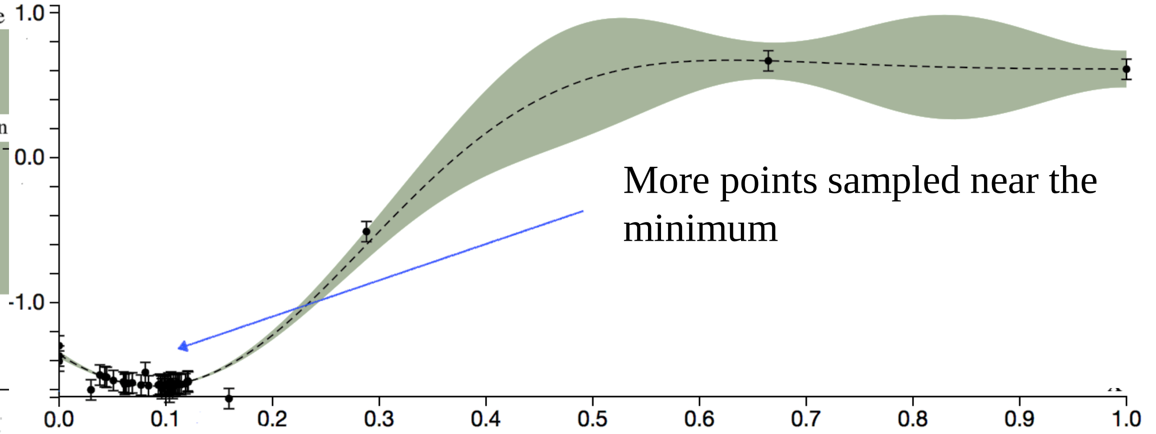
BO models what is known about $f(\boldsymbol{x})$, and uses it to sample in promising regions

Bayesian Optimization: A data-efficient optimization method

Define a cost $f(\boldsymbol{x})$, as a function of controller parameters $\boldsymbol{x}$

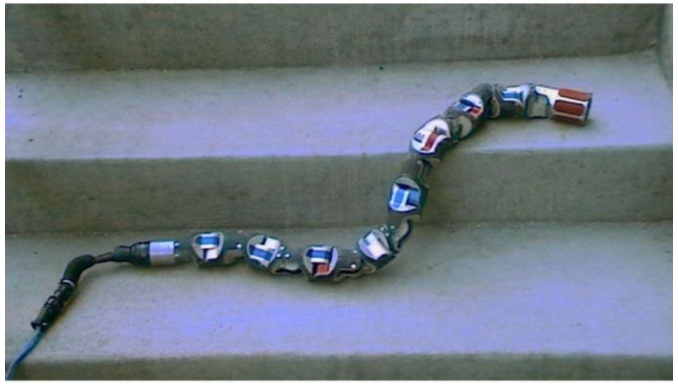
BO models what is known about $f(\boldsymbol{x})$, and uses it to sample in promising regions



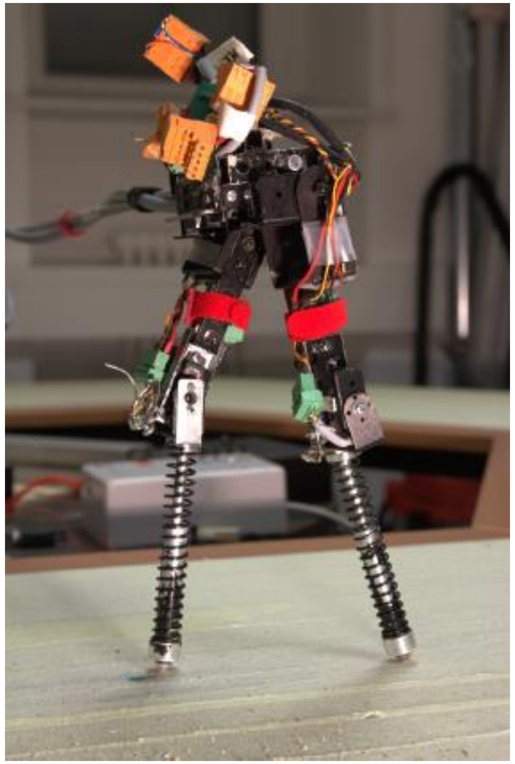


Bayesian optimization in robotics

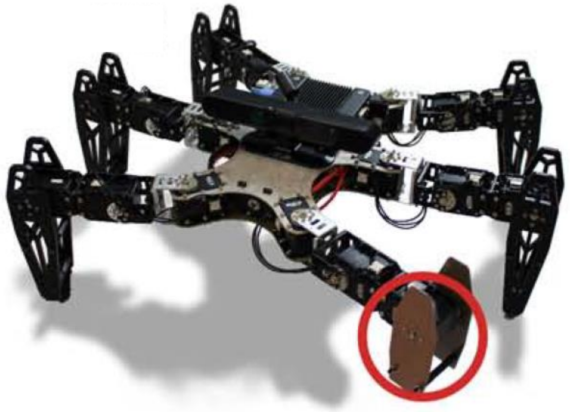
Lizotte, et al. 2007



Tesch, et al. 2011



Calandra, et al., 2015



Cully, et.al, 2015

How can we learn controllers for complex bipedal robots in less than 10 trials?

Outline

1. Introduction
    - Learning on robots
    - Bayesian optimization
2. Domain knowledge in Bayesian optimization
3. Learning features from data
    - Neural Network feature transform
    - Experiments
4. Modelling mismatch between simulation and hardware
5. Conclusions and future work

Problem structure can help in making sequential decisions

We model $f(\boldsymbol{x})$ with a Gaussian Process $f(\boldsymbol{x}) \sim GP(\underbrace{m(\boldsymbol{x})}_{\text{mean}}, \underbrace{k(\boldsymbol{x}, \boldsymbol{x}')}_{\text{covariance}})$

Problem structure can help in making sequential decisions

We model $f(\boldsymbol{x})$ with a Gaussian Process $f(\boldsymbol{x}) \sim GP(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$

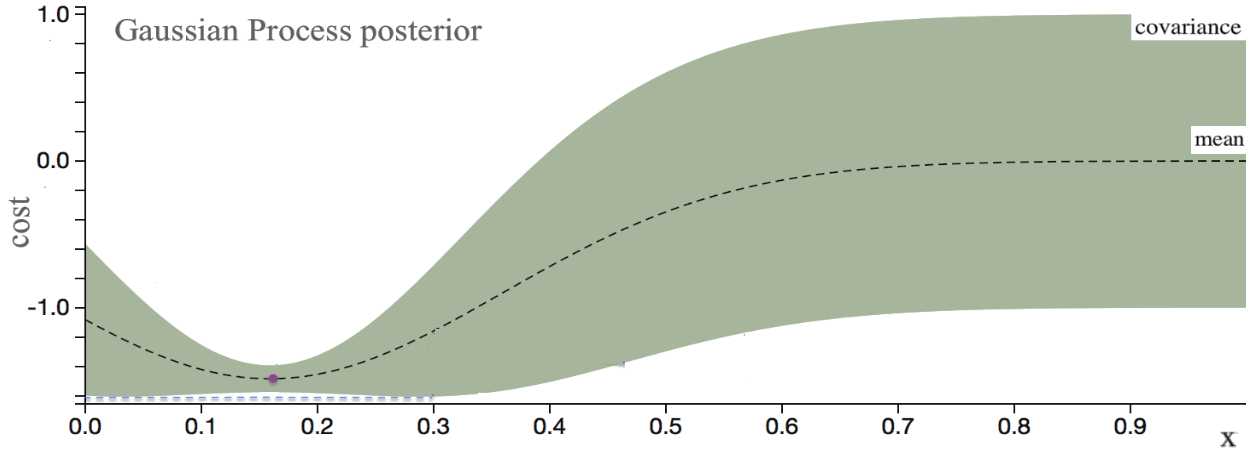
Kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ captures similarity between $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$

Problem structure can help in making sequential decisions

We model $f(\boldsymbol{x})$ with a Gaussian Process $f(\boldsymbol{x}) \sim GP(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$

Kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ captures similarity between $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$

$$k_{SE}(x_i, x_j) = \exp(-\frac{1}{2}\left|\left|x_i - x_j\right|\right|^2)$$

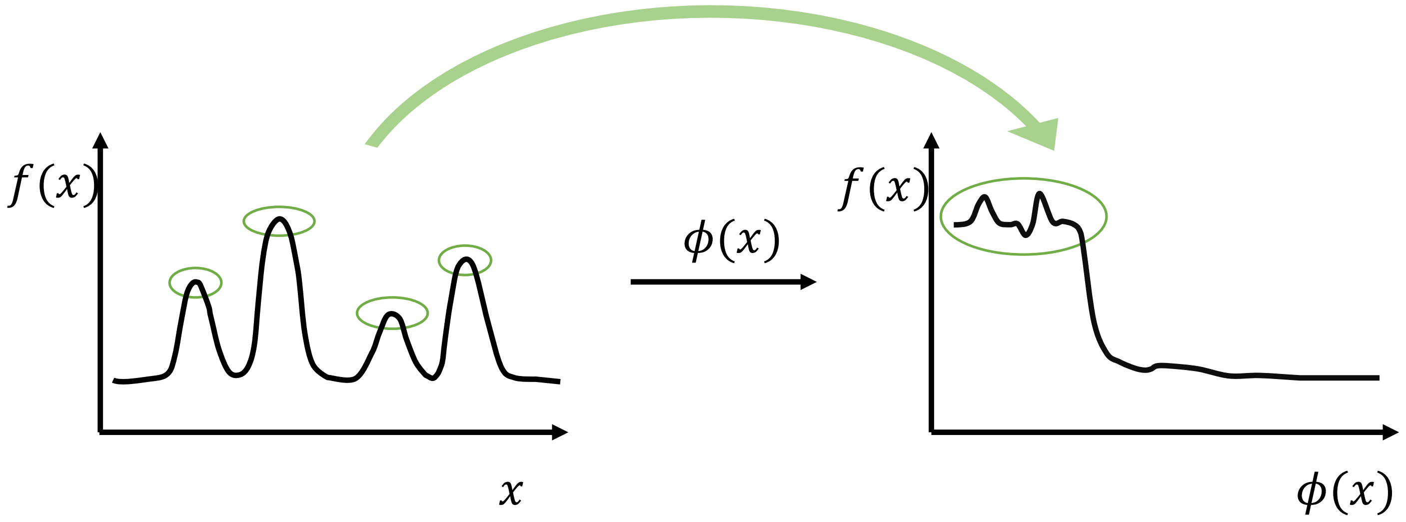


Problem structure can help in making sequential decisions

We model $f(\boldsymbol{x})$ with a Gaussian Process $f(\boldsymbol{x}) \sim GP(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$

Kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ captures similarity between $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$

Transform $\boldsymbol{x} \to \phi(\boldsymbol{x})$, $k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = k(\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j))$



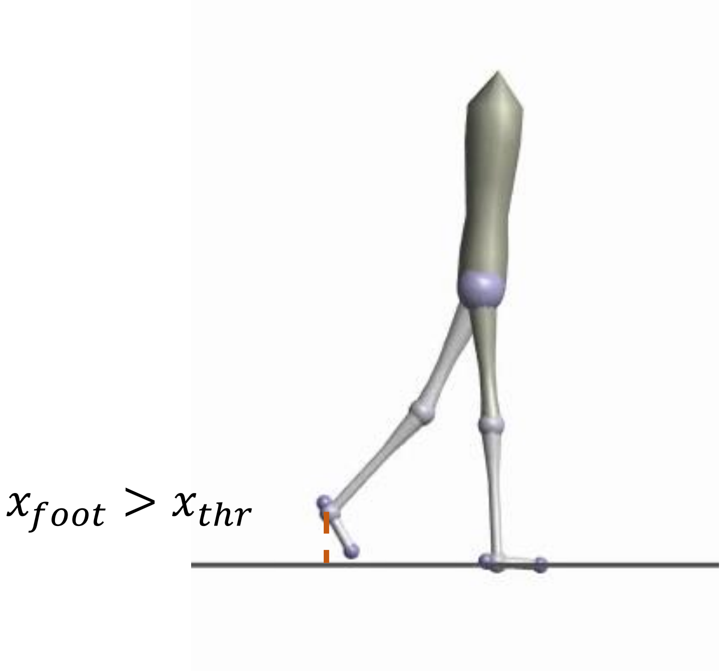
The Determinants of Gaits Feature Transform

Evaluate controllers based on basic bipedal walking metrics

The Determinants of Gaits Feature Transform

Evaluate controllers based on basic bipedal walking metrics

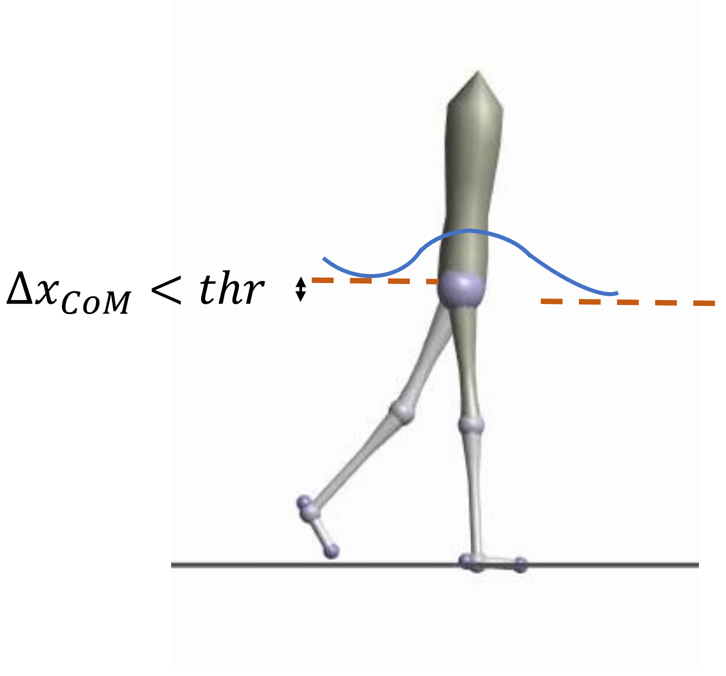
- M_1 : Is the swing leg retracted? [1/0]



The Determinants of Gaits Feature Transform

Evaluate controllers based on basic bipedal walking metrics

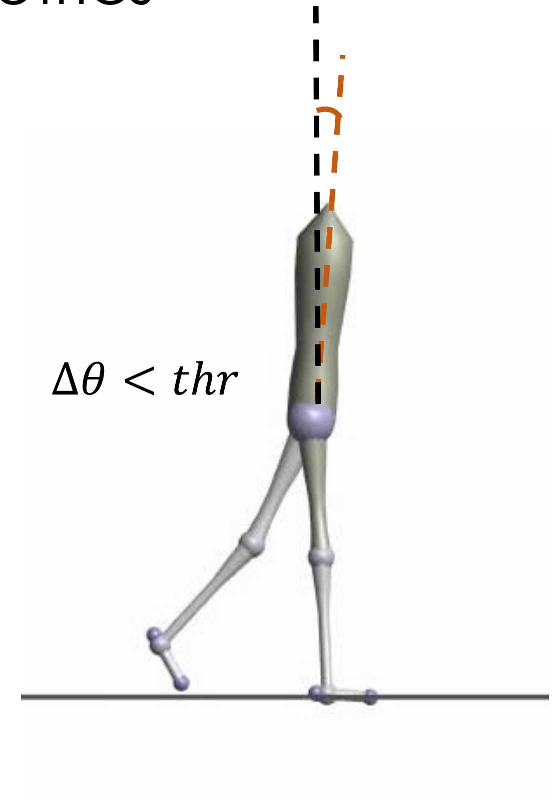
- M_1 : Is the swing leg retracted? [1/0]
- M_2 : Is the Center of Mass height changing? [1/0]



The Determinants of Gaits Feature Transform

Evaluate controllers based on basic bipedal walking metrics

- M_1 : Is the swing leg retracted? [1/0]
- M_2 : Is the Center of Mass height changing? [1/0]
- M_3 : Is the trunk lean changing? [1/0]



The Determinants of Gaits Feature Transform

Evaluate controllers based on basic bipedal walking metrics

- M_1 : Is the swing leg retracted? [1/0]
- M_2 : Is the Center of Mass height changing? [1/0]
- M_3 : Is the trunk lean changing? [1/0]
- M_4 : Average walking speed

The Determinants of Gaits Feature Transform

Evaluate controllers based on basic bipedal walking metrics

- M_1 : Is the swing leg retracted? [1/0]
- M_2 : Is the Center of Mass height changing? [1/0]
- M_3 : Is the trunk lean changing? [1/0]
- M_4 : Average walking speed

$$\phi(\boldsymbol{x}) = \sum M_i$$

The Determinants of Gaits Feature Transform

Evaluate controllers based on basic bipedal walking metrics

- M_1 : Is the swing leg retracted? [1/0]
- M_2 : Is the Center of Mass height changing? [1/0]
- M_3 : Is the trunk lean changing? [1/0]
- M_4 : Average walking speed

$$\phi(\boldsymbol{x}) = \sum M_i\,, \quad k_{DoG}\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right) = \exp\left(-\frac{1}{2}\left|\left|\phi(\boldsymbol{x_i}) - \phi\left(\boldsymbol{x_j}\right)\right|\right|^2\right)$$

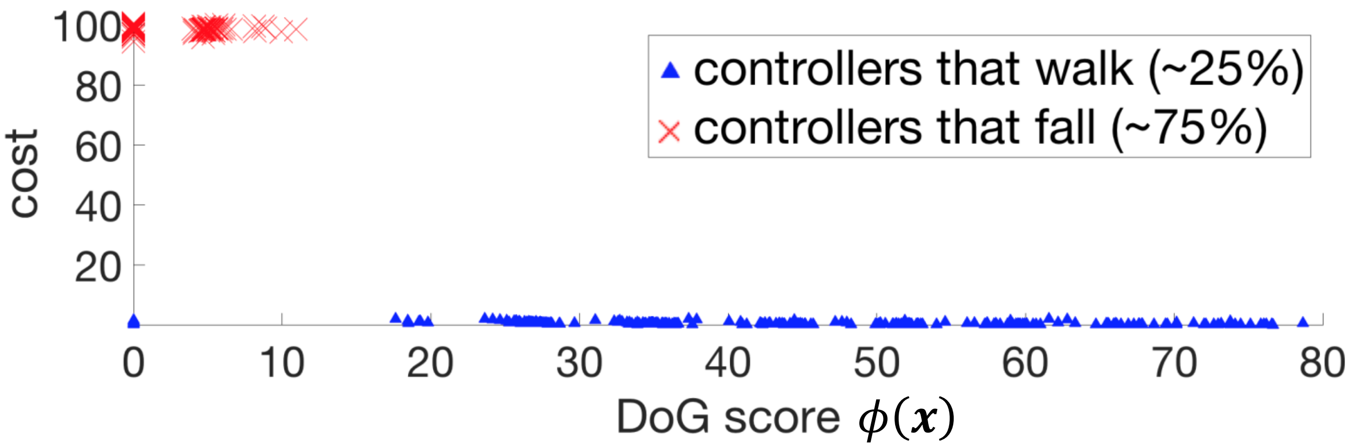
The Determinants of Gaits Feature Transform

Evaluate controllers based on basic bipedal walking metrics

- M_1 : Is the swing leg retracted? [1/0]
- M_2 : Is the Center of Mass height changing? [1/0]
- M_3 : Is the trunk lean changing? [1/0]
- M_4 : Average walking speed

$$\phi(\boldsymbol{x}) = \sum M_i$$

$$\text{cost} = \begin{cases} 100 - x_{fall}\text{ , if fall} \\ \left|\left|v - v_{tgt}\right|\right|\text{, if walk} \end{cases}$$



Evaluating DoG transform in simulation and on hardware

We evaluate our approach on the ATRIAS robot and simulation.

Evaluating DoG transform in simulation and on hardware

We evaluate our approach on the ATRIAS robot and simulation.

- Hardware
  - 5-dimensional reactively stepping controller
  - 9-dimensional reactively stepping controller
- Simulation
  - 9-dimensional reactively stepping controller
  - 50-dimensional neuromuscular controller

Evaluating DoG transform in simulation and on hardware

We evaluate our approach on the ATRIAS robot and simulation.

- Hardware
  - 5-dimensional reactively stepping controller
  - 9-dimensional reactively stepping controller
- Simulation
  - 9-dimensional reactively stepping controller
  - 50-dimensional neuromuscular controller

$$cost_{hdw} = \begin{cases} 100 - x_{fall}, \text{if fall} \\ \left|\left|v - v_{tgt}\right|\right|, \text{if walk} \end{cases}$$

$$cost_{sim} = \begin{cases} 100 - x_{fall}, \text{if fall} \\ \left|\left|v - v_{tgt}\right|\right| + c_{tr}, \text{if walk} \end{cases}$$

Evaluating DoG transform in simulation and on hardware

We evaluate our approach on the ATRIAS robot and simulation.

- Hardware
  - 5-dimensional reactively stepping controller
  - 9-dimensional reactively stepping controller
- Simulation
  - 9-dimensional reactively stepping controller
  - 50-dimensional neuromuscular controller

$$cost_{hdw} = \begin{cases} 100 - x_{fall}, \text{if fall} \\ \left|\left|v - v_{tgt}\right|\right|, \text{if walk} \end{cases}$$

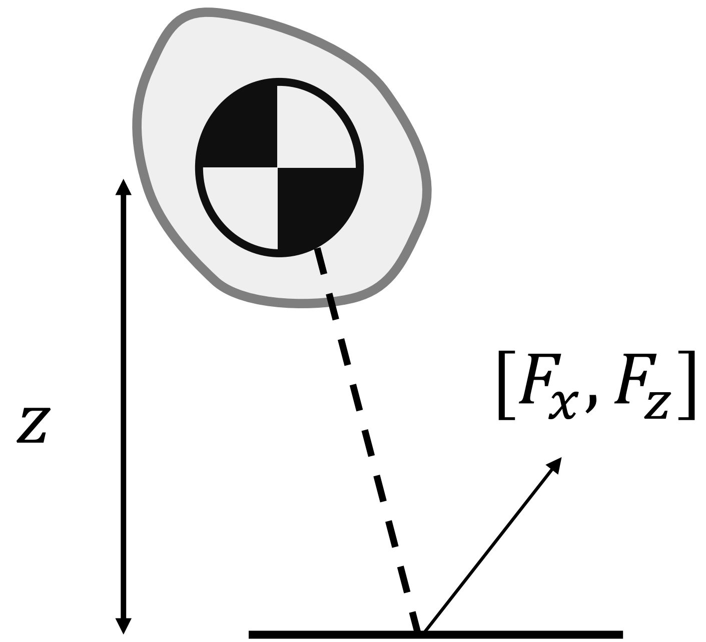
$$cost_{sim} = \begin{cases} 100 - x_{fall}, \text{if fall} \\ \left|\left|v - v_{tgt}\right|\right| + c_{tr}, \text{if walk} \end{cases}$$

We pre-compute $\phi(\boldsymbol{x})$ for a large grid of parameters by running short simulations for fast look-up when optimizing

Feedback-based Reactive stepping controller

Control the height of the Center of Mass

$$F_z = K_{pz}(z_{des} - z) + K_{dz}\,(\dot{z}_{des} - \dot{z})$$



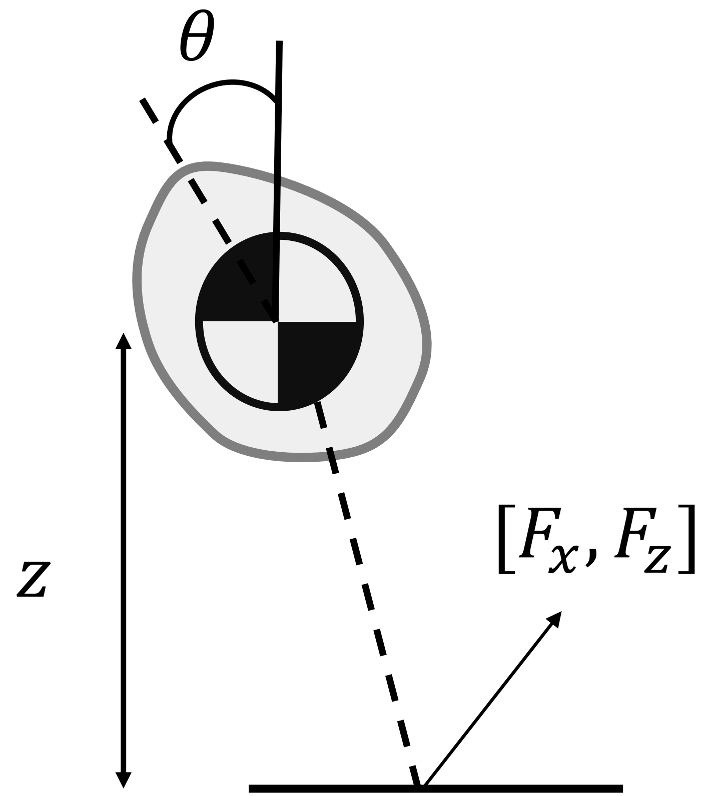
Feedback-based Reactive stepping controller

Control the height of the Center of Mass

$$F_z = K_{pz}(z_{des} - z) + K_{dz}\,(\dot{z}_{des} - \dot{z})$$

Control the torso orientation

$$F_x = K_{pt}(\theta_{des} - \theta) + K_{dt}(\dot{\theta}_{des} - \dot{\theta})$$



Feedback-based Reactive stepping controller

Control the height of the Center of Mass

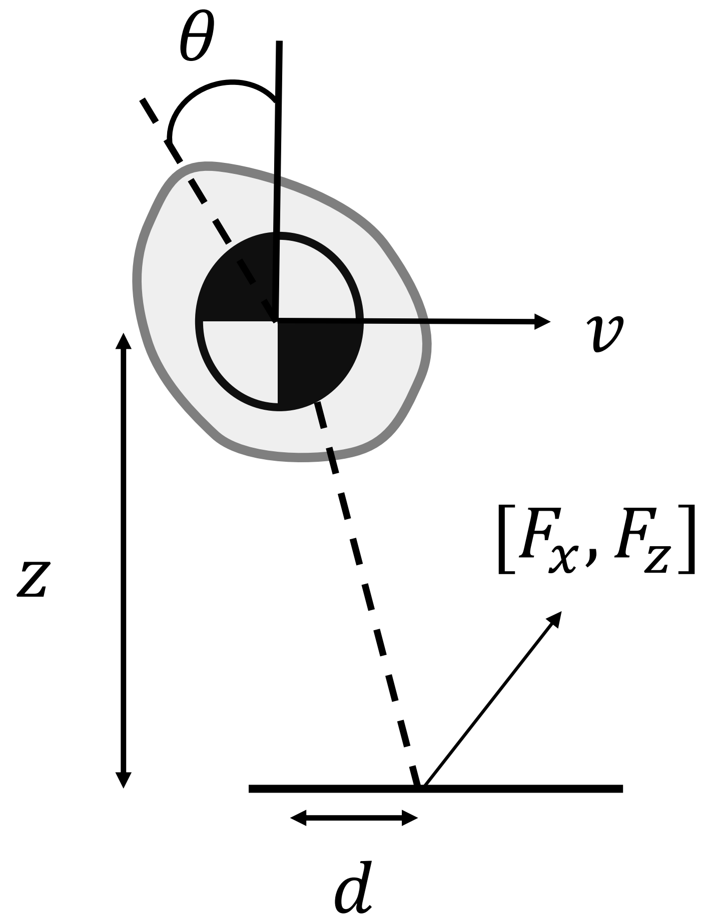
$$F_z = K_{pz}(z_{des} - z) + K_{dz}\,(\dot{z}_{des} - \dot{z})$$

Control the torso orientation

$$F_x = K_{pt}(\theta_{des} - \theta) + K_{dt}(\dot{\theta}_{des} - \dot{\theta})$$

Control swing foot placement

$$x_p = 0.5 * v * T + k_v\left(v - v_{tgt}\right) + C * d$$



5 and 9 dimensional controllers

5 dimensional controller : torso, velocity parameters

$$\boldsymbol{x} = [K_{pt}, K_{dt}, k_v, C, T]$$

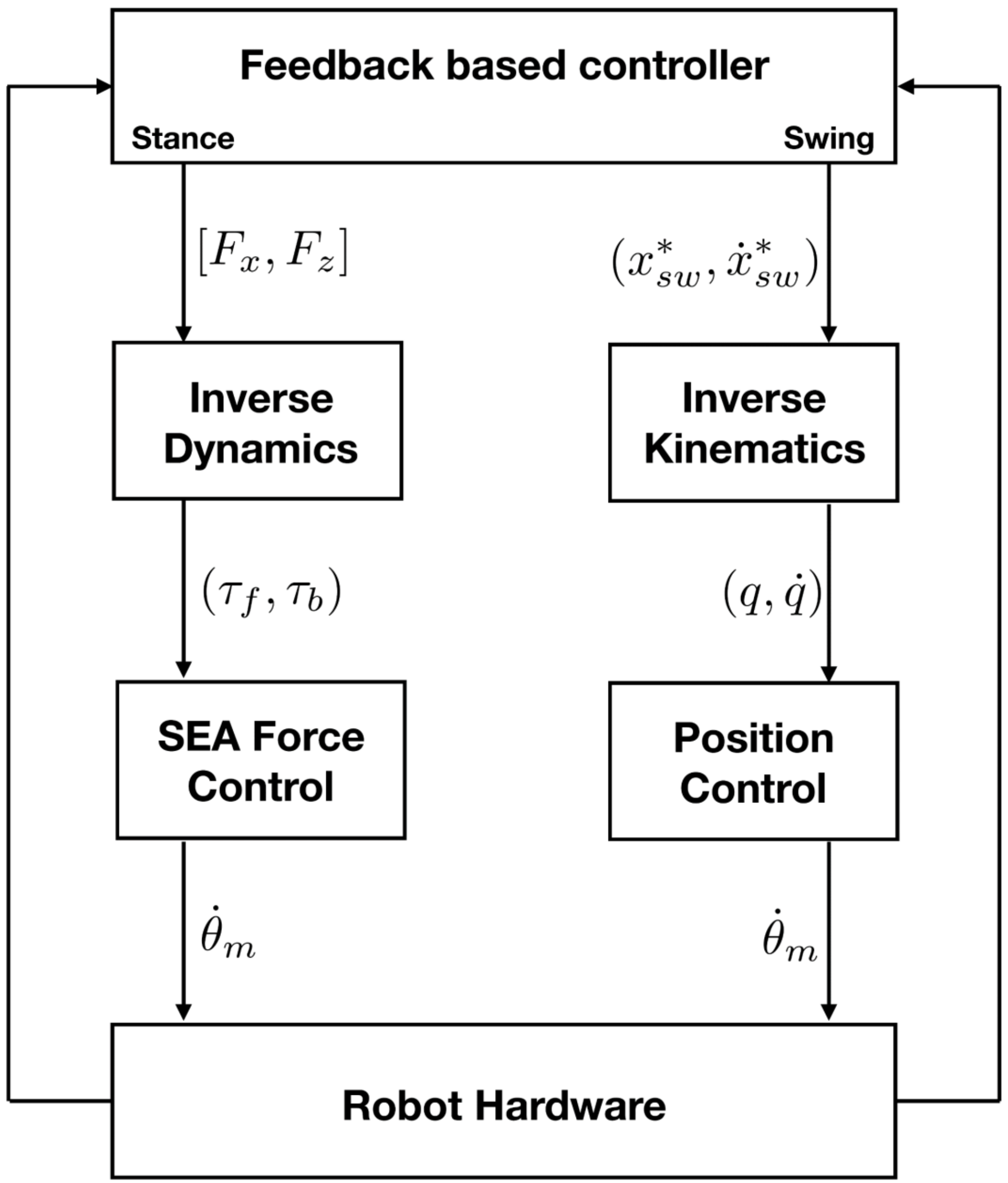
5 and 9 dimensional controllers

5 dimensional controller : torso, velocity parameters

$$\boldsymbol{x} = [K_{pt}, K_{dt}, k_v, C, T]$$

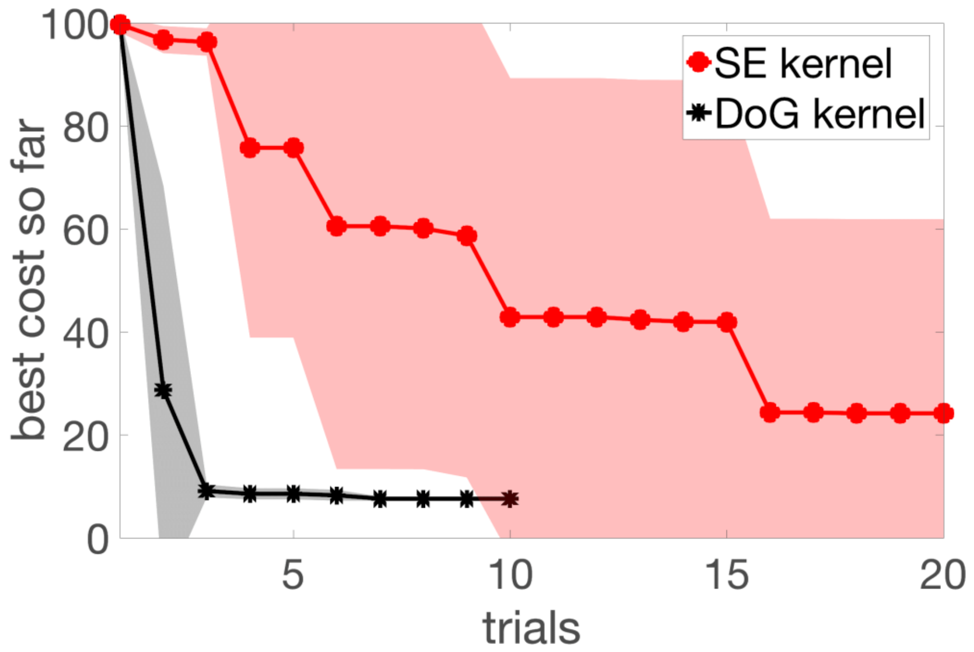
9 dimensional controller : height, torso, velocity parameters

$$\boldsymbol{x} = [K_{pz}, K_{dz}, z_{des}, K_{pt}, K_{dt}, \theta_{des}, k_v, C, T]$$



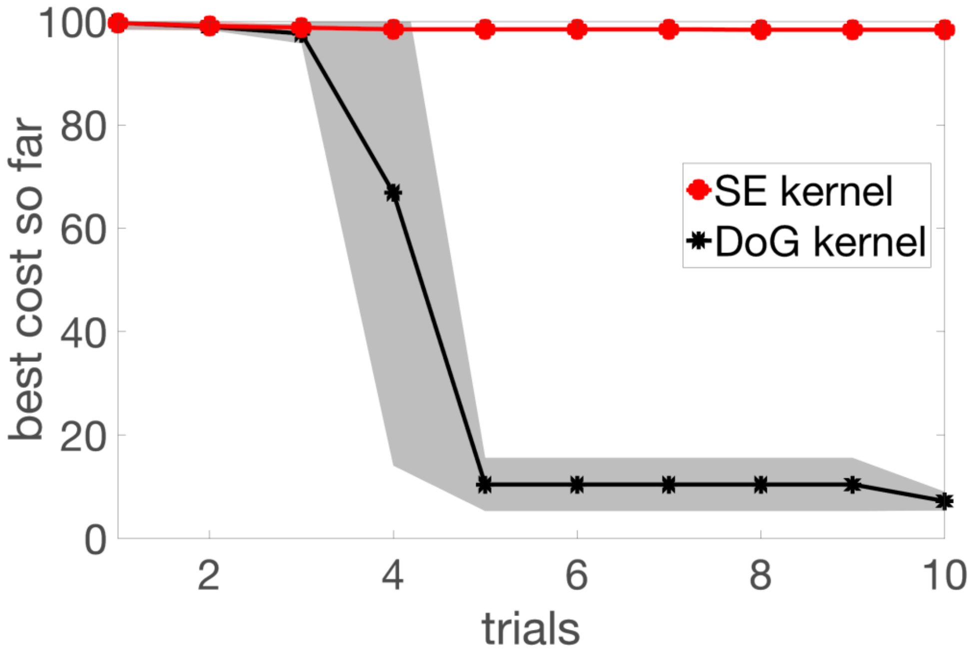
Hardware Experiments on the ATRIAS Robot

BO for 5-dimensional controller over 5 runs



Random rate of success 10%

BO for 9-dimensional controller over 3 runs



Random rate of success 3%

[1] Rai A*, Antonova R*, Song S, Martin W, Geyer H, Atkeson CG. Bayesian Optimization Using Domain Knowledge on the ATRIAS Biped. arXiv preprint arXiv:1709.06047. 2017 Sep 18.(* - equal contribution)

Bayesian Optimization for 5-dimensional controller

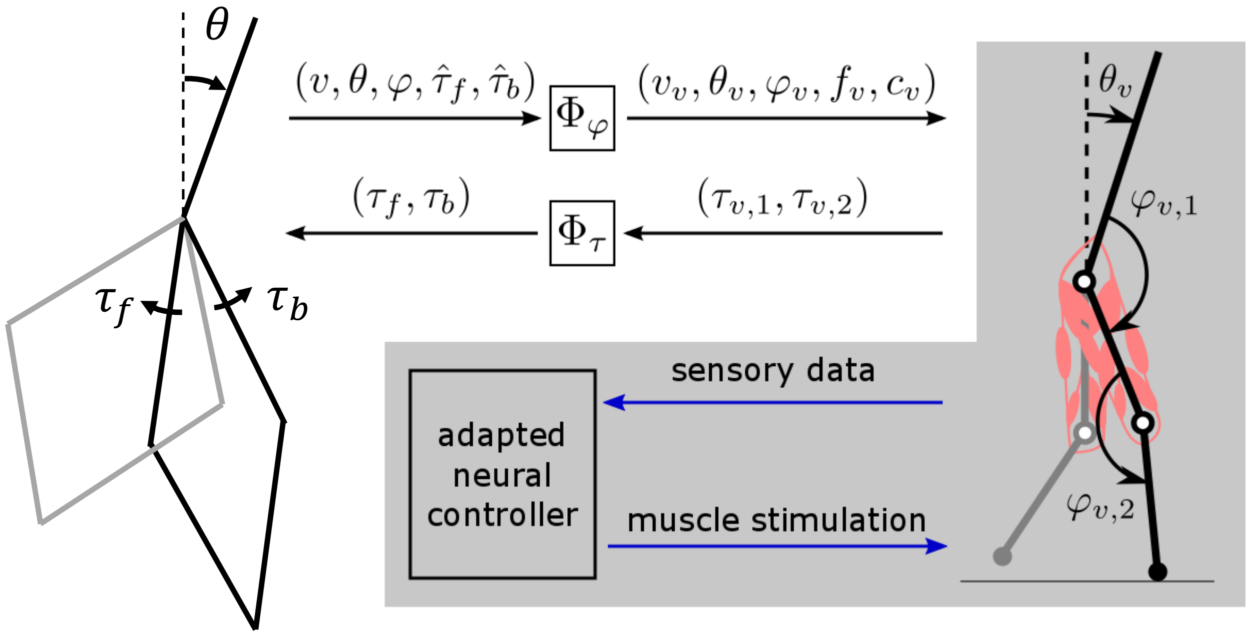
10 trials using the proposed approach

target speed profile:

0.4m/s (15 steps) – 1.0m/s (15 steps) – 0.2m/s (15 steps) – 0m/s (5 steps)

50-dimensional neuromuscular controller

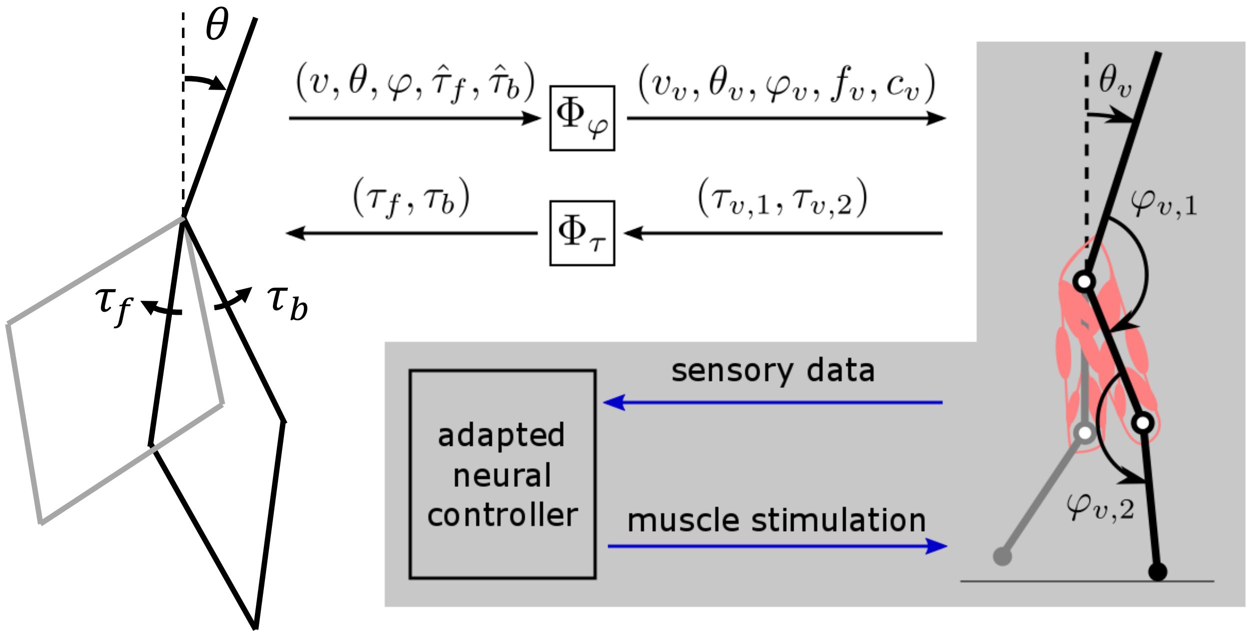
Map a human-like muscle
model to ATRIAS morphology



50-dimensional neuromuscular controller

Map a human-like muscle model to ATRIAS morphology

Emulates human spinal reflexes, can generate a wide range of behaviors

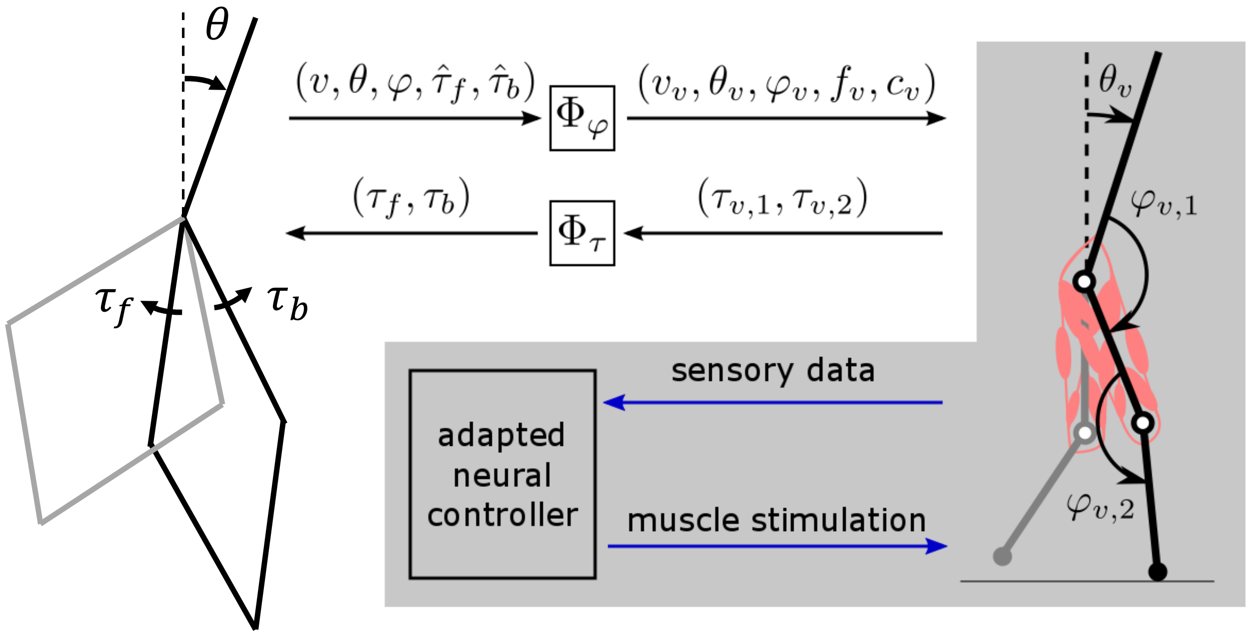


[3] Z. Batts, S. Song, H. Geyer. Toward a virtual neuromuscular control for robust walking in bipedal robots. IEEE International Conference on Intelligent Robots and Systems, Hamburg, Germany, pp. 6318-6323, 2015

50-dimensional neuromuscular controller

Map a human-like muscle model to ATRIAS morphology

Emulates human spinal reflexes, can generate a wide range of behaviors

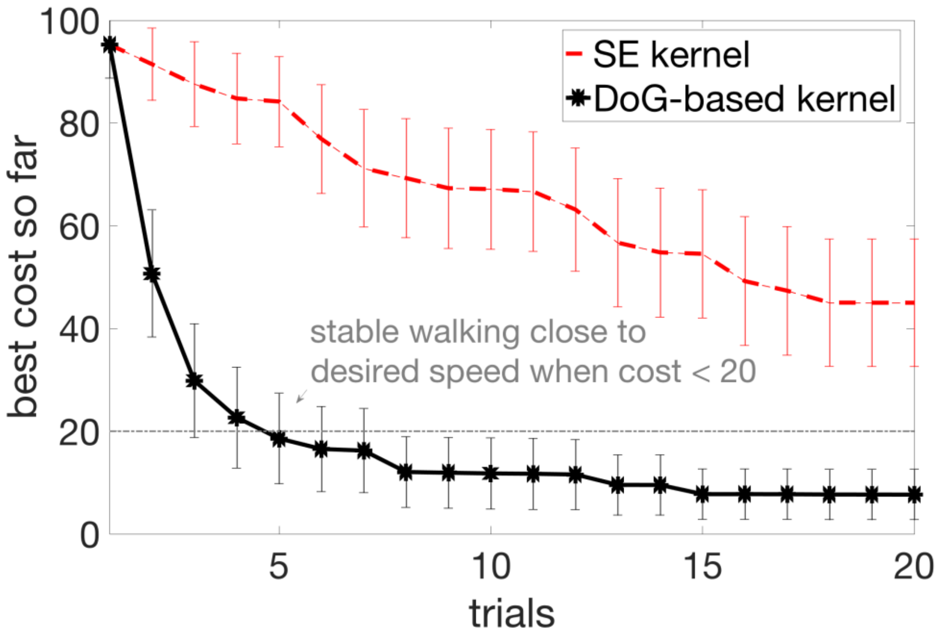


We remove some biological components, and replace them with robot counterparts

- Joint velocities, instead of muscle velocities
- No additional delay in feedback

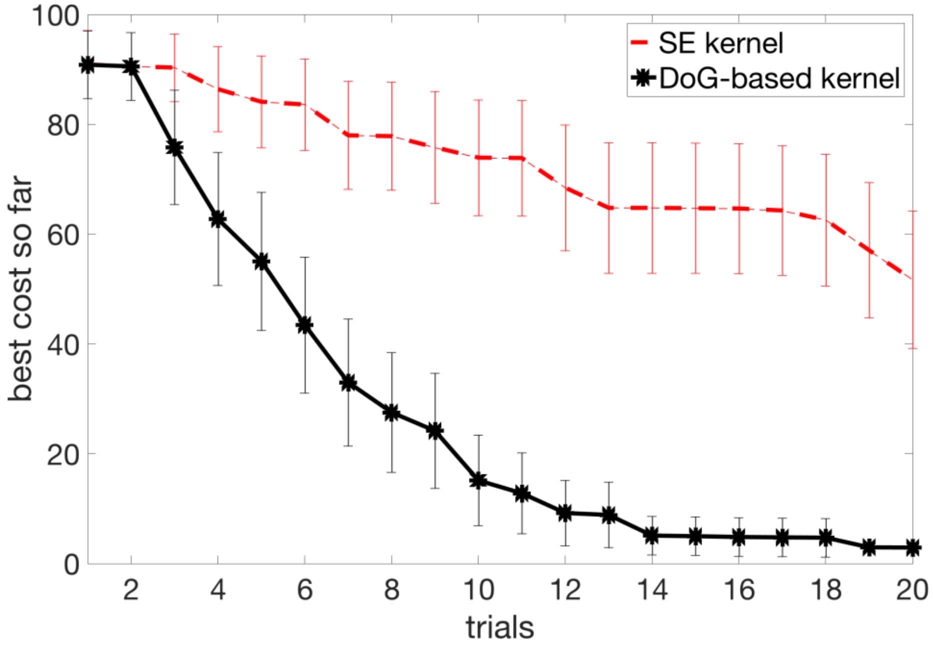
Simulation Experiments on the ATRIAS robot

BO for 9-dimensional controller over 50 runs



Random rate of success 8%

BO for 50-dimensional controller over 50 runs



Random rate of success 4%

[1] Rai A*, Antonova R*, Song S, Martin W, Geyer H, Atkeson CG. Bayesian Optimization Using Domain Knowledge on the ATRIAS Biped. arXiv preprint arXiv:1709.06047. 2017 Sep 18.(* - equal contribution)

BO with DoG transform can learn to walk in less than 20 trials for a 5, 9 and 50-dimensional controller

Outline

1. Introduction
   - Learning on robots
   - Bayesian optimization
2. Domain knowledge in Bayesian optimization
   - DoG feature transform
   - Experiments on ATRIAS
3. Learning features from data
4. Modelling mismatch between simulation and hardware
5. Conclusions and future work

How can we learn useful features directly from data?

Determinants of Gaits require a lot of domain knowledge

- Advantage: Very data-efficient, highly transparent way of optimizing
- Disadvantage: Another set of expert-designed heuristics

How can we learn useful features directly from data?

Determinants of Gaits require a lot of domain knowledge

- Advantage: Very data-efficient, highly transparent way of optimizing
- Disadvantage: Another set of expert-designed heuristics

Can we learn feature transforms without a lot of expert knowledge?

How can we learn useful features directly from data?

Determinants of Gaits require a lot of domain knowledge

- Advantage: Very data-efficient, highly transparent way of optimizing
- Disadvantage: Another set of expert-designed heuristics

Can we learn feature transforms without a lot of expert knowledge?

We train neural networks to predict properties of simulations and use it as a non-linear feature transform

$$\phi(x) = out_{NN}(x)$$

Using Neural Networks to learn useful feature transforms from data

Collect data by running short simulations

Using Neural Networks to learn useful feature transforms from data

Collect data by running short simulations

Train a network to predict the cost using simulation

Highly data-efficient, but does not generalize to other costs

Using Neural Networks to learn useful feature transforms from data

Collect data by running short simulations

Train a network to predict the cost using simulation
- Highly data-efficient, but does not generalize to other costs

Train a network to predict summaries of simulation trajectories
- CoM final position, average torso angle, average speed
- Generalizes to multiple costs

Using Neural Networks to learn useful feature transforms from data

Collect data by running short simulations

Train a network to predict the cost using simulation
- Highly data-efficient, but does not generalize to other costs

Train a network to predict summaries of simulation trajectories
- CoM final position, average torso angle, average speed
- Generalizes to multiple costs

Test on hardware or perturbed simulations

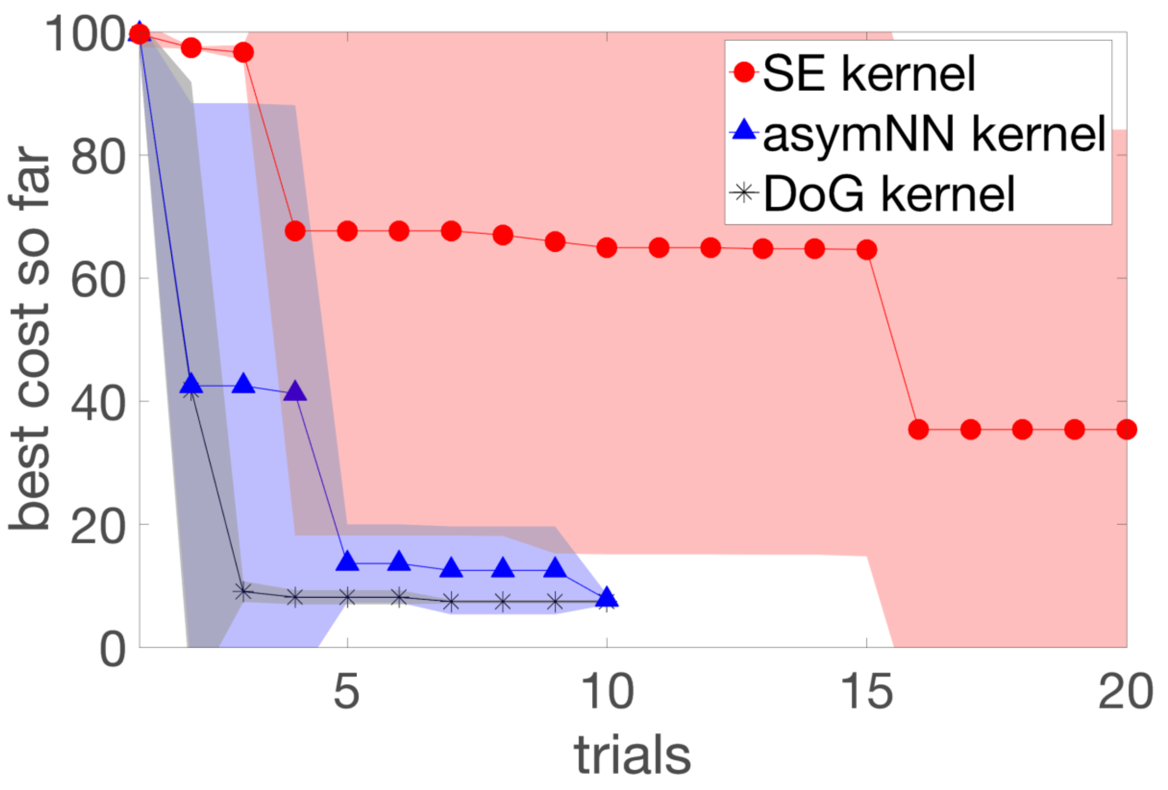
Experiments on ATRIAS robot for a 5-dimensional controller

Predicting short simulation cost

$$cost = \begin{cases} 100 - x_{fall}\text{ , if fall} \\ \left\| v - v_{tgt} \right\|\text{, if walk} \end{cases}$$

Generalizes to v_{tgt} different from that was used to generate the kernel

BO for 5-dimensional controller over 3 runs



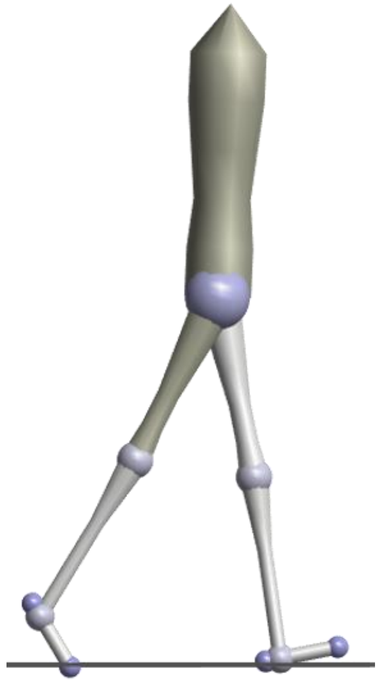
Random rate of success 10%

Simulation experiments with a 7-link biped for a 16-dimensional controller

Predicting trajectory summaries

Using a 16-dimensional neuromuscular controller on a 7-link biped

Tested on 2 different costs

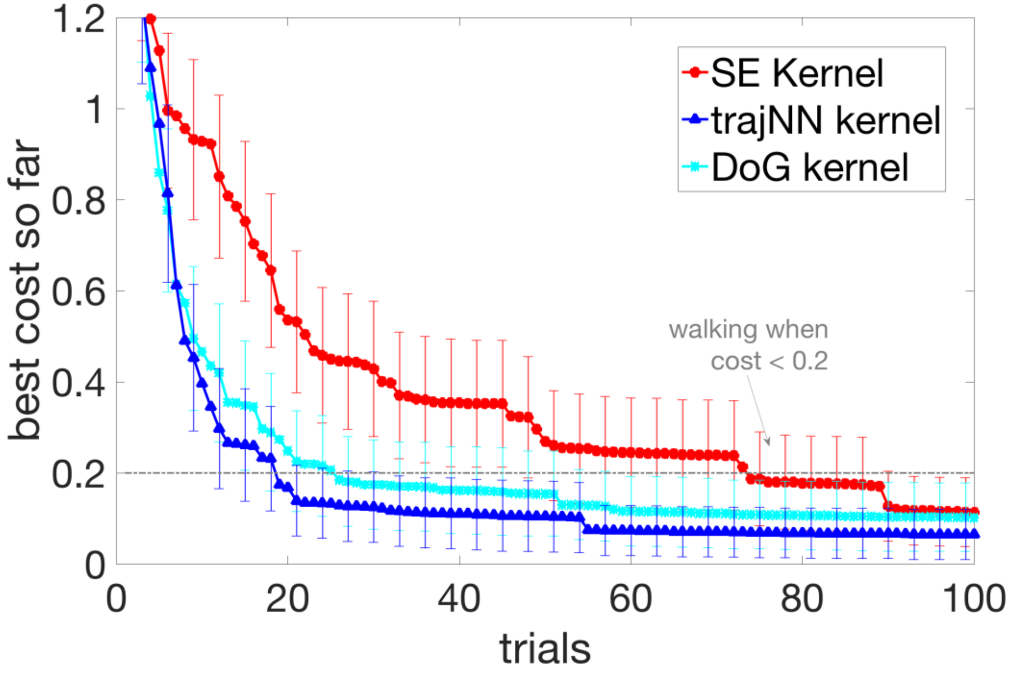


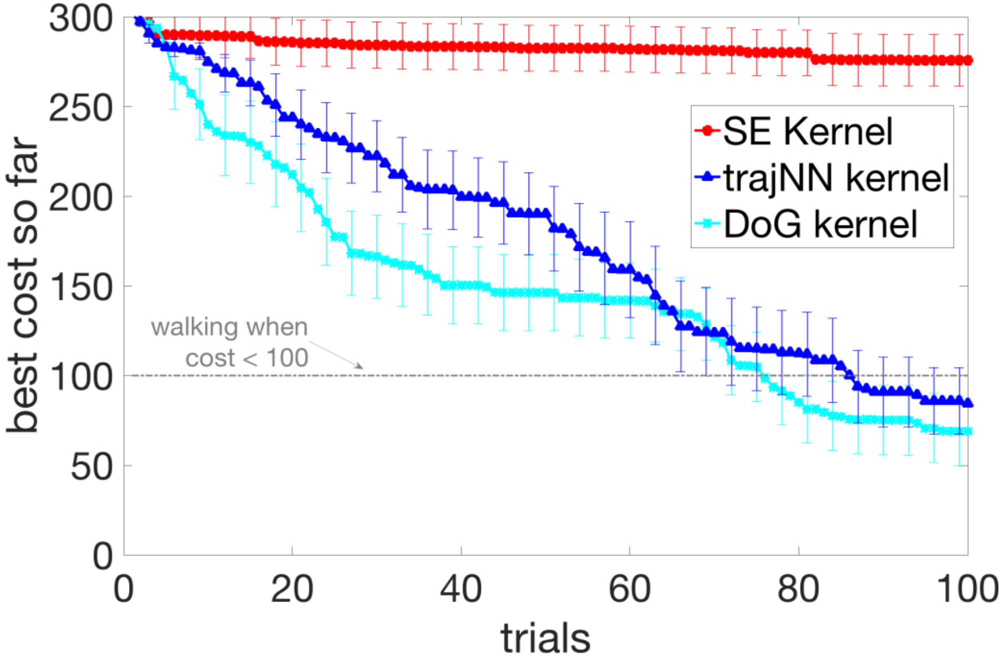
7-link biped

Simulation experiments for a 16-dimensional controller

$$cost = \frac{1}{1 + t_{final}} + \frac{0.3}{1 + x_{final}} + 0.01 \left|\left| v - v_{tgt} \right|\right|$$

$$cost = \begin{cases} 300 - x_{fall}\text{, if fall} \\ 100 \left|\left| v - v_{tgt} \right|\right| + c_{tr}\text{, if walk} \end{cases}$$





[2] Antonova R*, Rai A*, Atkeson CG. Deep Kernels for Optimizing Locomotion Controllers. arXiv preprint arXiv:1707.09062. 2017 Jul 27.(* - equal contribution)

A neural network trained to predict trajectory summaries performs competitively to hand designed feature transforms in simulation

Outline

1. Introduction
   - Learning on robots
   - Bayesian optimization

2. Domain knowledge in Bayesian optimization
   - DoG feature transform
   - Experiments on ATRIAS

3. Learning features from data
   - Neural Network feature transform
   - Experiments

4. Modelling mismatch between simulation and hardware

5. Conclusions and future work

Accounting for mismatch between simulation and hardware

What happens when the simulation does not match hardware?

Accounting for mismatch between simulation and hardware

What happens when the simulation does not match hardware?

A separate GP that models the difference between expected and observed behavior

$$g(x) = \phi_{sim}(x) - \phi_{hdw}(x)$$

$$\phi_{adj}(x) = [\phi_{sim}(x); g(x)]$$

Accounting for mismatch between simulation and hardware

What happens when the simulation does not match hardware?

A separate GP that models the difference between expected and observed behavior

$$g(x) = \phi_{sim}(x) - \phi_{hdw}(x)$$

$$\phi_{adj}(x) = [\phi_{sim}(x); g(x)]$$

New distance between points becomes

$$k_{adj}(x, x') = k\left(\phi_{adj}(x), \phi_{adj}(x')\right)$$

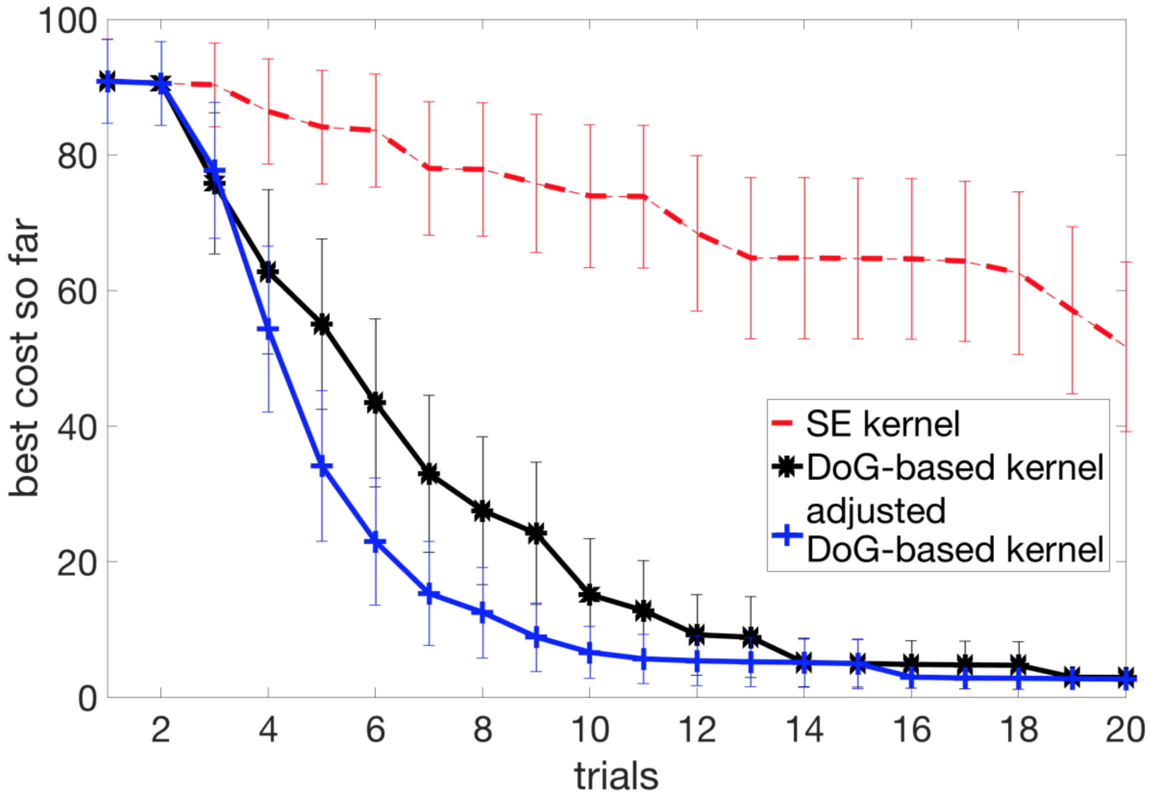
Simulation experiments with 50 dimensional controller

Controllers that walk in short simulations fall in longer simulations

Adjusted kernel with mismatch has a slight advantage over DoG-based kernel

Needs to be tested on hardware

BO for 50-dimensional controller over 50 runs



Random rate of success 4%

Outline

1. Introduction
   - Learning on robots
   - Bayesian optimization
2. Domain knowledge in Bayesian optimization
   - DoG feature transform
   - Experiments on ATRIAS
3. Learning features from data
   - Neural Network feature transform
   - Experiments
4. Modelling mismatch between simulation and hardware
5. Conclusions and future work

Conclusions

Bayesian Optimization with an informed kernel was able to learn walking policies in very few trials

Conclusions

Bayesian Optimization with an informed kernel was able to learn walking policies in very few trials

Hand-designed features and neural networks are both able to extract useful information from simulation

Conclusions

Bayesian Optimization with an informed kernel was able to learn walking policies in very few trials

Hand-designed features and neural networks are both able to extract useful information from simulation

Accounting for mismatch between short and long simulations helps sample-efficiency in simulation

Proposed work : Accounting for mismatch between hardware and simulation

Evaluate how performance of feature transforms deteriorates with increasing mismatch

- Generate kernel on unperturbed simulation, test on increasingly perturbed simulations
- Generate kernel on increasingly perturbed simulation, test on hardware

Proposed work : Accounting for mismatch between hardware and simulation

Evaluate how performance of feature transforms deteriorates with increasing mismatch

- Generate kernel on unperturbed simulation, test on increasingly perturbed simulations
- Generate kernel on increasingly perturbed simulation, test on hardware

Evaluate the mismatch adjusted kernel(s) on hardware

Proposed work : Accounting for mismatch between hardware and simulation

Evaluate how performance of feature transforms deteriorates with increasing mismatch

- Generate kernel on unperturbed simulation, test on increasingly perturbed simulations
- Generate kernel on increasingly perturbed simulation, test on hardware

Evaluate the mismatch adjusted kernel(s) on hardware

Initialize prior mismatch from simulation

- Using perturbed simulations
- Using knowledge about expected mismatch

Proposed work: 3 dimensional bipedal walking – controllers and feature transforms

Walking controllers for 3 dimensional bipedal walking

- Feedback-based reactive policy based
- 3 dimensional neuromuscular models

Proposed work: 3 dimensional bipedal walking – controllers and feature transforms

Walking controllers for 3 dimensional bipedal walking

- Feedback-based reactive policy based
- 3 dimensional neuromuscular models

Feature transforms for 3 dimensional walking

- Features that include walking features for 3D walking
- Learn features from 3D simulation

Proposed work: 3 dimensional bipedal walking – controllers and feature transforms

Walking controllers for 3 dimensional bipedal walking

- Feedback-based reactive policy based
- 3 dimensional neuromuscular models

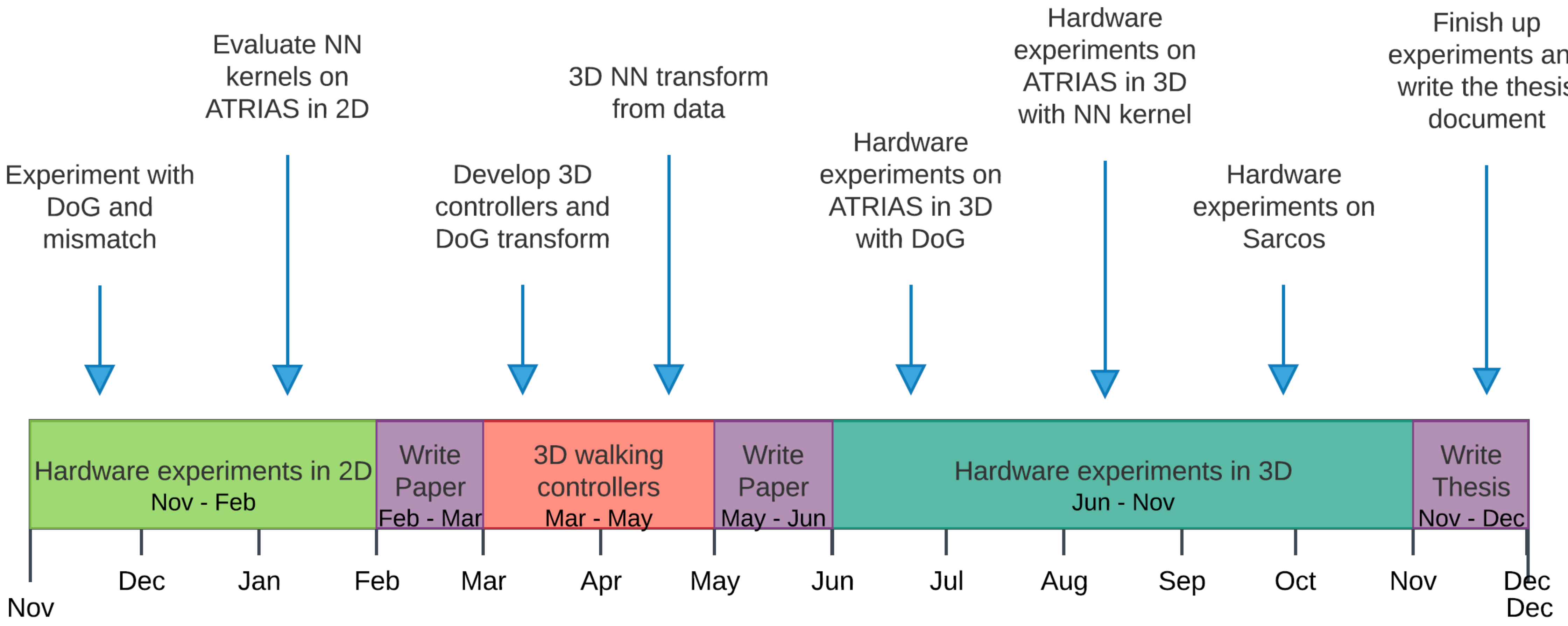
Feature transforms for 3 dimensional walking

- Features that include walking features for 3D walking
- Learn features from 3D simulation

Experiment and evaluate on hardware

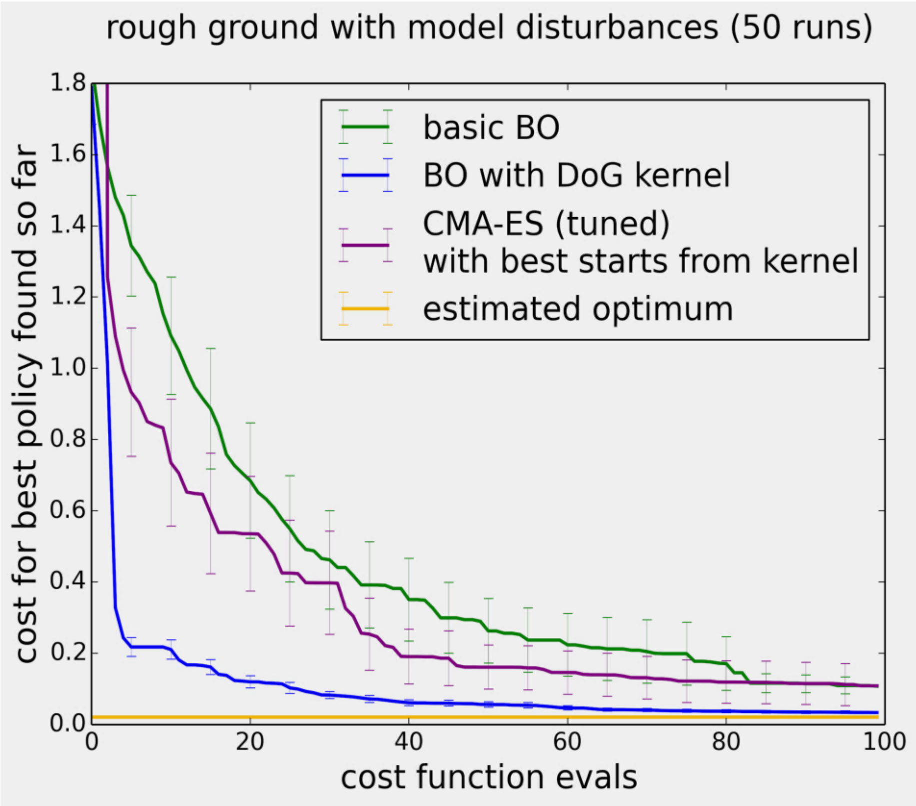
- Primarily ATRIAS
- If time permits, Sarcos

Timeline

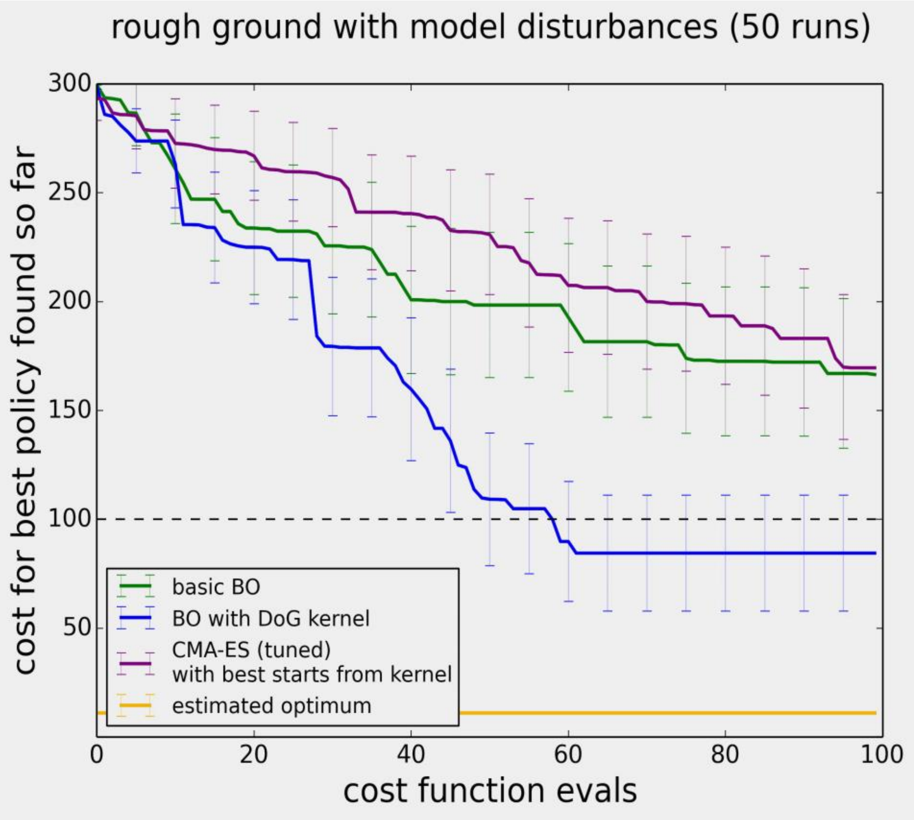

Experiment with DoG and mismatch
Evaluate NN kernels on ATRIAS in 2D
Develop 3D controllers and DoG transform
3D NN transform from data
Hardware experiments on ATRIAS in 3D with DoG
Hardware experiments on ATRIAS in 3D with NN kernel
Hardware experiments on Sarcos
Finish up experiments and write the thesis document
Hardware experiments in 2D
Nov - Feb
Write Paper
Feb - Mar
3D walking controllers
Mar - May
Write Paper
May - Jun
Hardware experiments in 3D
Jun - Nov
Write Thesis
Nov - Dec
Nov
Dec
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Dec

Thank you!

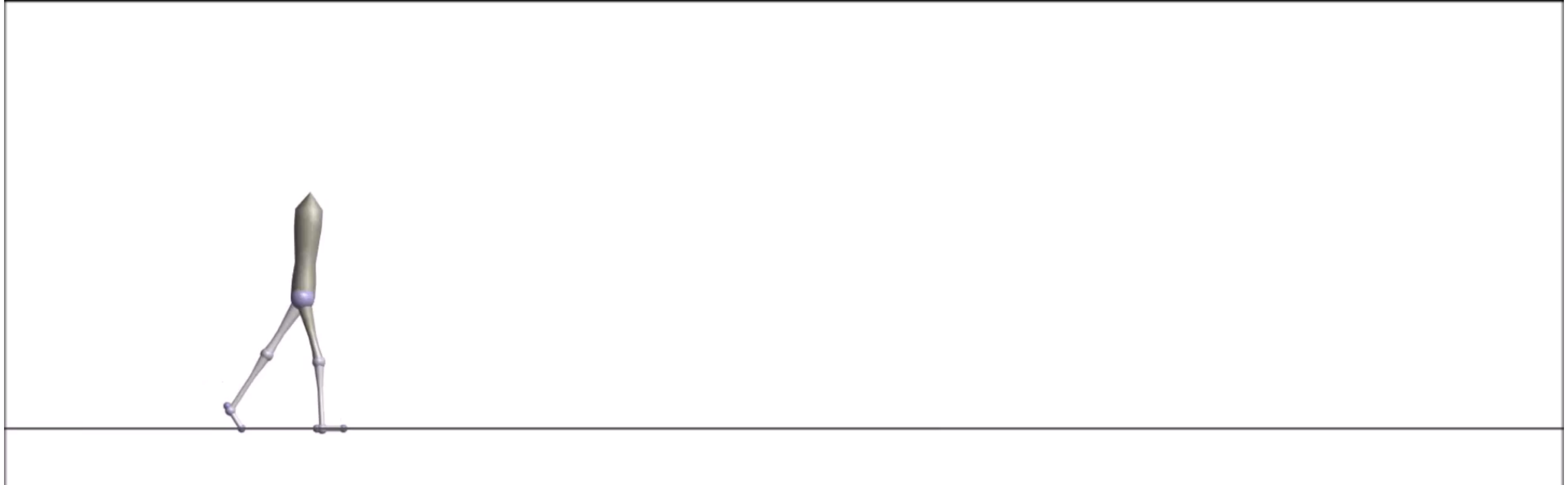
Simulation Experiments on a 7-link biped with a16-dimensional controller



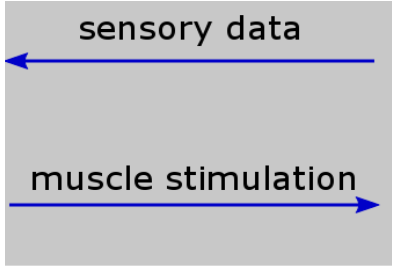
$$f(x) = \frac{1}{1 + t_{fall}} + \frac{0.3}{1 + d_{fall}} + 0.01(v - v_{tgt})$$

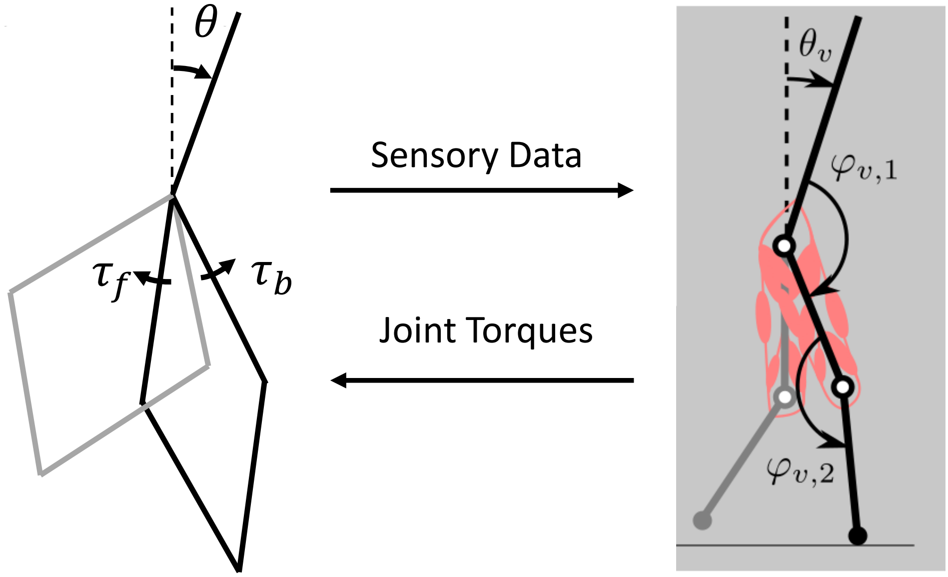


$$f(x) = \begin{cases} 300 - d_{fall}\text{, if fall} \\ 100\left\|v - v_{tgt}\right\| + c_{tr}\text{, if walk} \end{cases}$$



Walking on flat ground


sensory data
muscle stimulation


θ
τ_f
τ_b
Sensory Data
Joint Torques
θ_v
$\varphi_{v,1}$
$\varphi_{v,2}$