Development and Evaluation of a learned Biped Walking Trajectory based on 3D Points on the NAO Robotic System

Zichong Lu

Hamburg University of Technology
Institute of Control Systems

01.02.2019

TUHH
Hamburg University of Technology

ICS
Institute of Control Systems

Contents

Introduction

The NAO humanoid robot

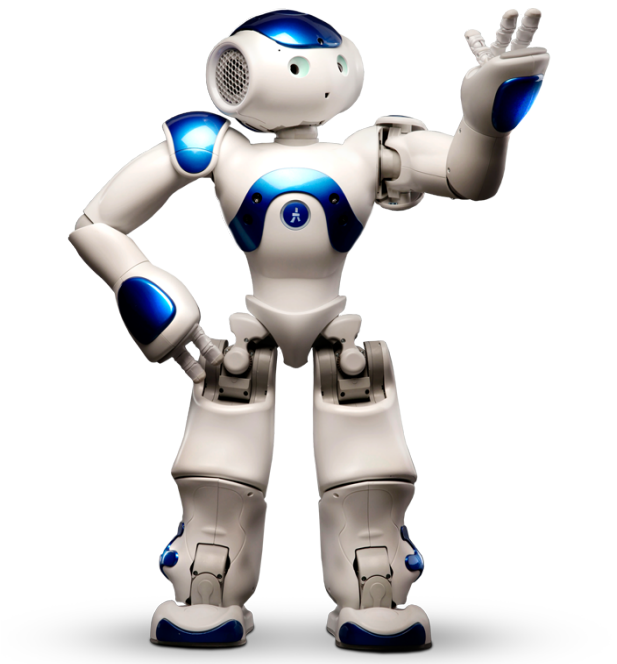


Figure: NAO[1]

[1] SoftBank Robotics. https://www.softbankrobotics.com/us/nao.

Gait generation

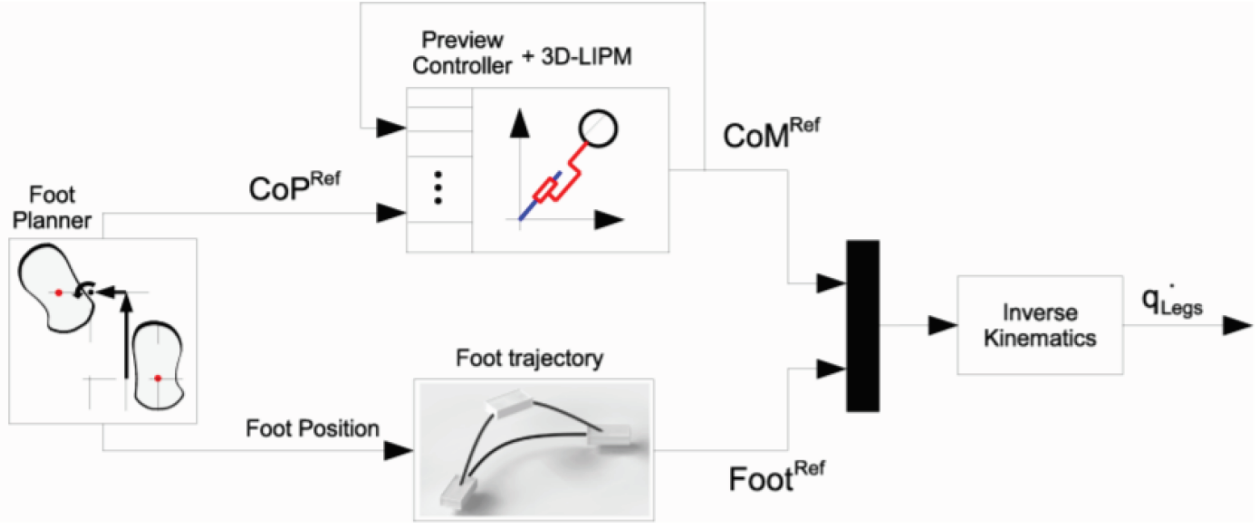


Figure: Process of gait generation[2]

[2] David Gouaillier, Cyrille Collette, and Chris Kilner. "Omni-directional closed-loop walk for NAO". In: *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on Humanoid Robots*. IEEE. 2010, pp. 448–454.

Motivation

- To compare existing biped walking models
- To replace the biped walking model
- To perform self-improvement with reinforcement learning

Simulation environment

A biped multibody system

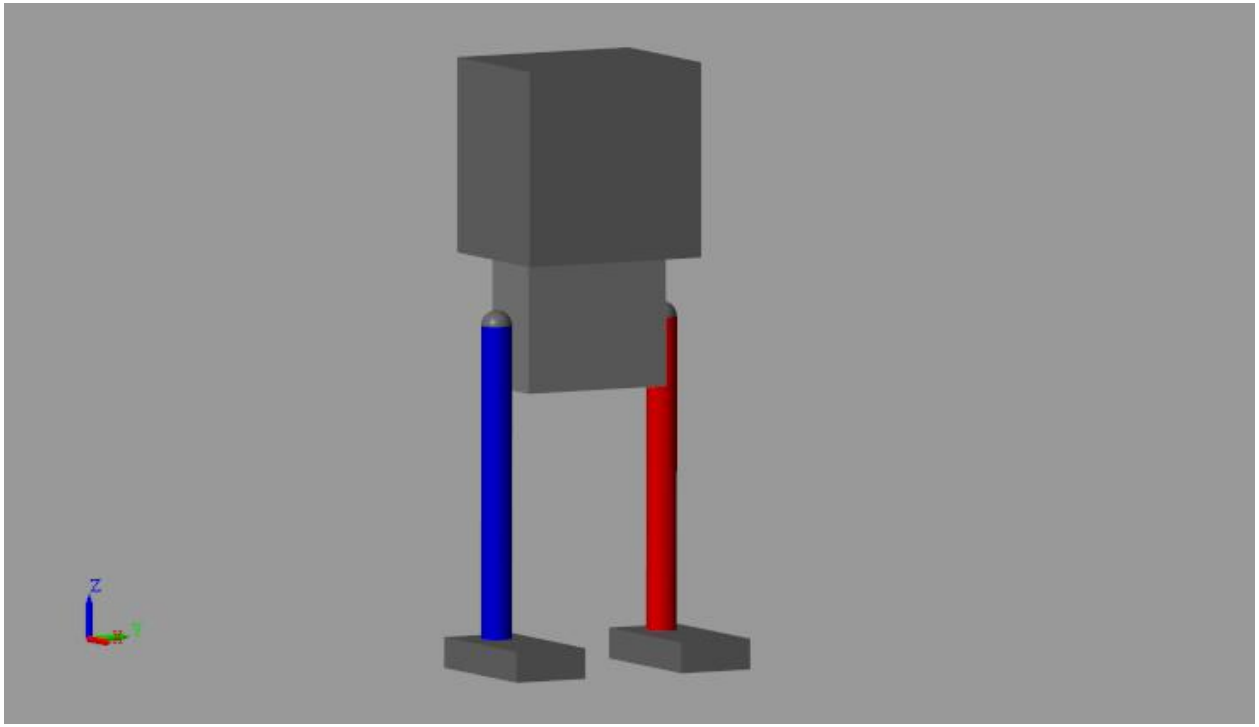


Figure: A biped multibody system

Simulation environment

Foot-to-ground forces

- Pressure: a spring system[3]

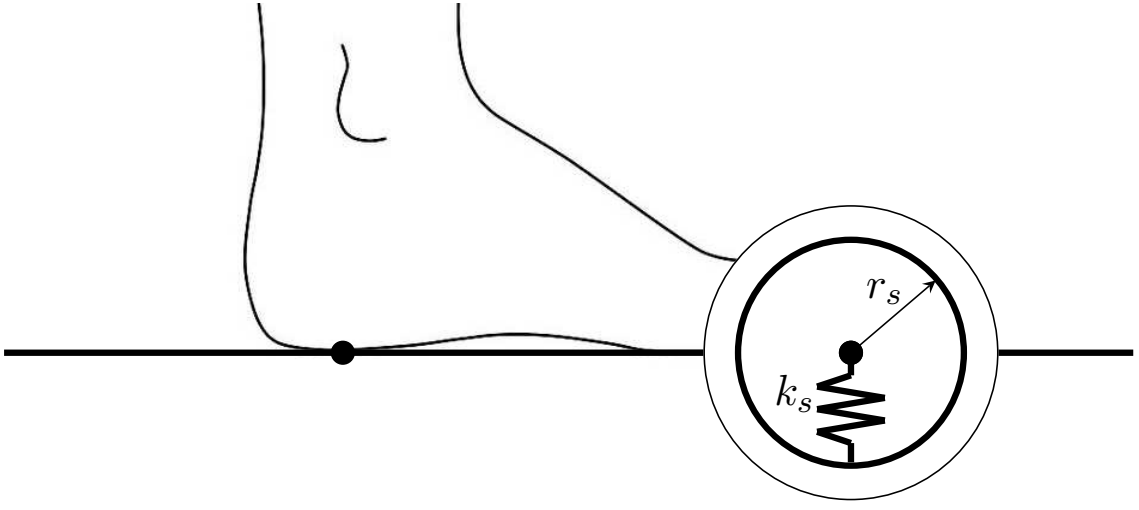


Figure: Foot pressure model

$$k_s = \frac{m_r \cdot g}{r_s}$$

[3] Steve Miller. *Simscape Multibody Contact Forces Library*. 2018. URL: https://www.mathworks.com/matlabcentral/fileexchange/47417-

Simulation environment

Foot-to-ground forces

- Pressure: a spring system
- Friction: to determine μ_s andμ_k by tuning
  - Walking speed
  - Elapse time
  - $\mu_s = 1000$ and $\mu_k = 1000$

Walking with models

Previous works

- A zero-moment point (ZMP) previewing gait[3]
- A 3D linear inverted pendulum model (LIPM) gait[4]
- Walk2014 from UNSW[5]

[3] Shuuji Kajita et al. "Biped walking pattern generation by using preview control of zero-moment point". In: *ICRA*. Vol. 3. 2003, pp. 1620–1626.

[4] Colin Graf and Thomas Röfer. "A Closed-loop 3D-LIPM Gait for the RoboCup Standard Platform League Humanoid". In: *Proceedings of the Fifth Workshop on Humanoid Soccer Robots in conjunction with the 2010 IEEE-RAS International Conference on Humanoid Robots*. Ed. by Enrico Pagello et al. Nashville, TN, USA, 2010.

[5] Bernhard Hengst. *runswift walk2014 report robocup standard platform league*. Tech. rep. Technical report, The University of New South Wales, 2014.

A ZMP previewing gait

The zero-moment point

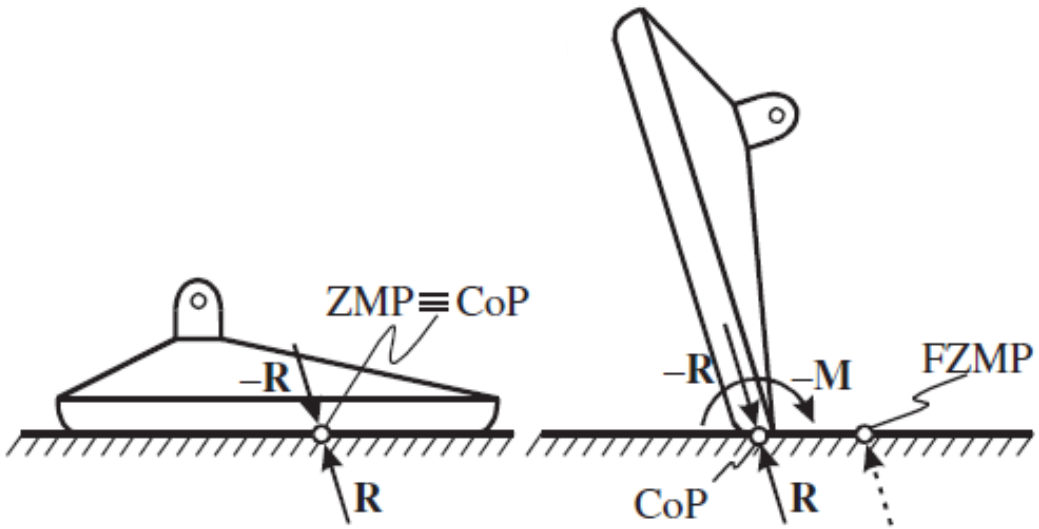


Figure: The zero-moment point[6]

[6] Miomir Vukobratović and Branislav Borovac. "Zero-moment point - thirty five years of its life". In: *International journal of humanoid robotics* 1.01 (2004), pp. 157–173.

A ZMP previewing gait

Cart-table model (1)

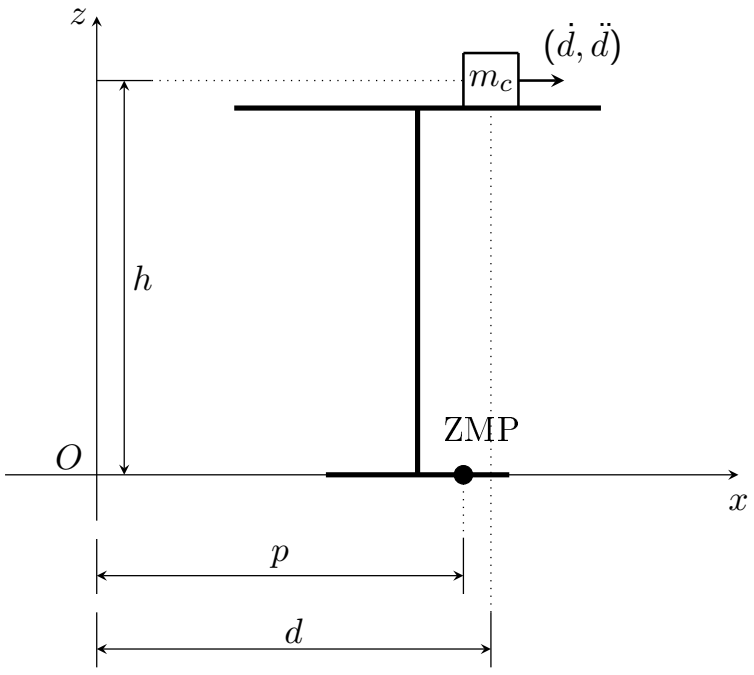


Figure: Cart-table model

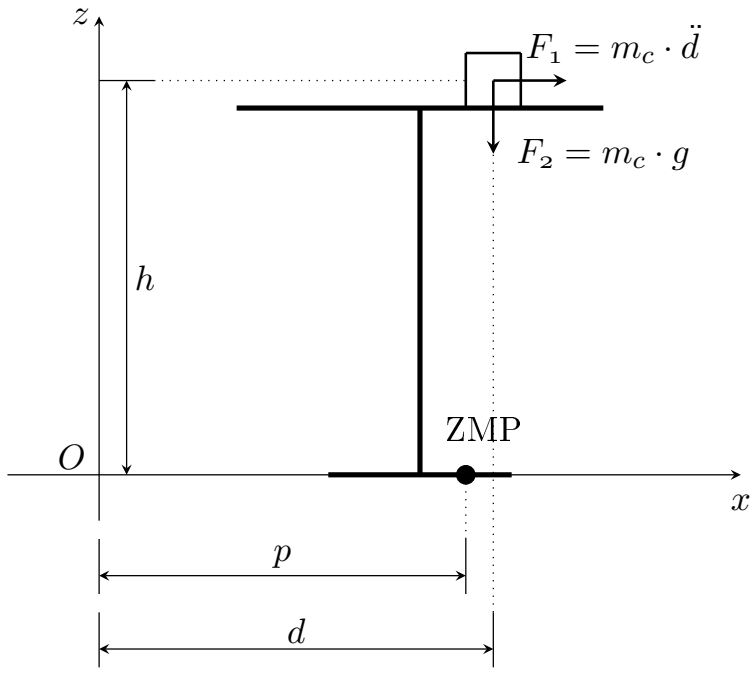


Figure: Force diagram

$$m_c \cdot \ddot{d} \cdot (d - p) - m_c \cdot g \cdot h = 0$$

A ZMP previewing gait

Cart-table model (2)

- State vector: $\begin{bmatrix} d & \dot{d} & \ddot{d} \end{bmatrix}^T$
- Input: jerk
- Output: position of ZMP
- Continuous-time state-space representation:

$$\begin{bmatrix} \dot{d} \\ \ddot{d} \\ \dddot{d} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \dot{d} \\ \ddot{d} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u$$

$$p = \begin{bmatrix} 1 & 0 & -\frac{h}{g} \end{bmatrix} \begin{bmatrix} d \\ \dot{d} \\ \ddot{d} \end{bmatrix}$$

A ZMP previewing gait

Cart-table model (2)

- State vector: $\begin{bmatrix} d(k) & v(k) & a(k) \end{bmatrix}^T$
- Input: jerk $j(k)$
- Output: position of ZMP $p(k)$
- Discrete-time state-space representation (sampling time T_s):

$$\begin{bmatrix} v(k+1) \\ a(k+1) \\ j(k+1) \end{bmatrix} = \begin{bmatrix} 1 & T_s & \frac{T_s^2}{2} \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d(k) \\ v(k) \\ a(k) \end{bmatrix} + \begin{bmatrix} \frac{T_s^3}{6} \\ \frac{T_s^2}{2} \\ T_s \end{bmatrix} u(k)$$

$$p(k) = \begin{bmatrix} 1 & 0 & -\frac{h}{g} \end{bmatrix} \begin{bmatrix} d(k) \\ v(k) \\ a(k) \end{bmatrix}$$

A ZMP previewing gait

Preview controller (1)

- Why preview controller?
- The optimal preview controller:

$$u(k) = G_e \sum_{i=0}^{k} e(k) - G_x x(k) - \sum_{l=1}^{N} G_r(l) r(k+l)$$

- The cost function:

$$J = \sum_{i=k}^{\infty} \left[Q_e e^2(i) + \Delta \boldsymbol{x}^T(i) Q_x \Delta \boldsymbol{x}(i) + R \Delta u^2(i) \right]$$

A ZMP previewing gait

Preview controller (2)

- Limitations:

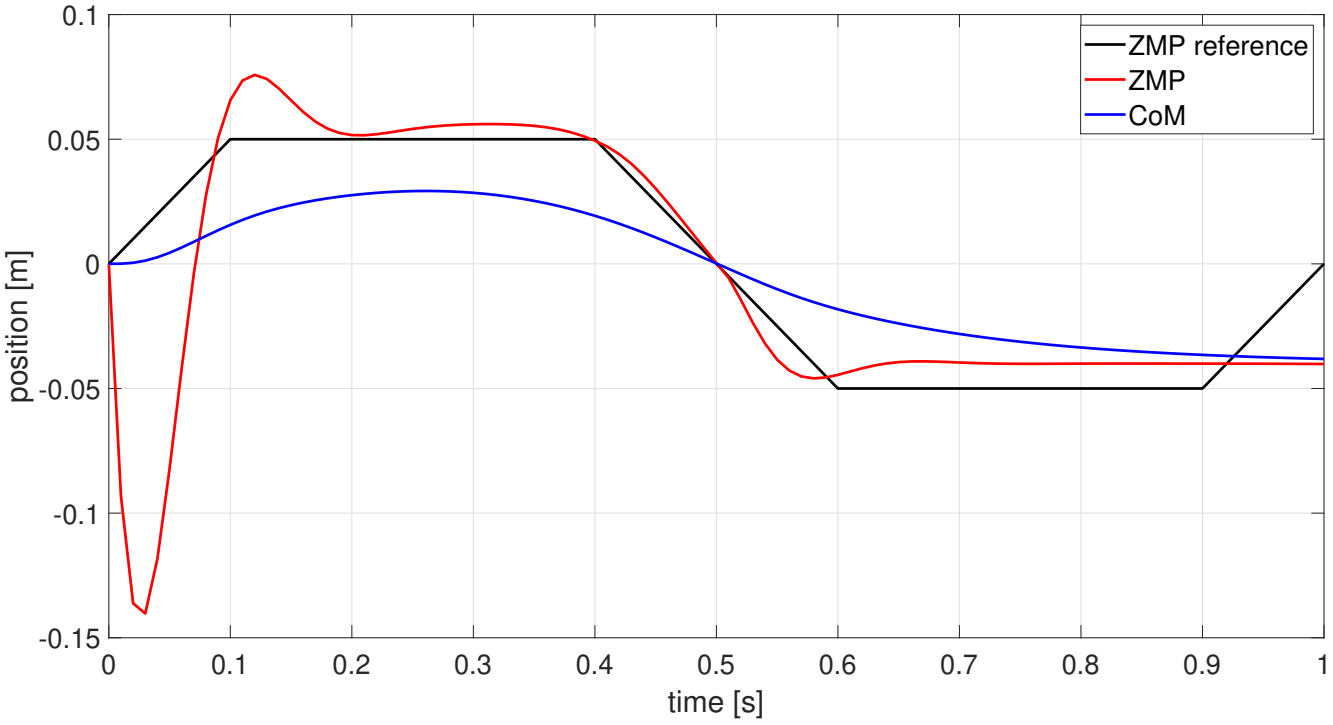


Figure: Limitations when using preview controller

A ZMP previewing gait

Preview controller (3)

- First strategy: to reserve some space, $Q_e = 50500$, $Q_x = 0$, $R = 0.59$ and $N = 35$

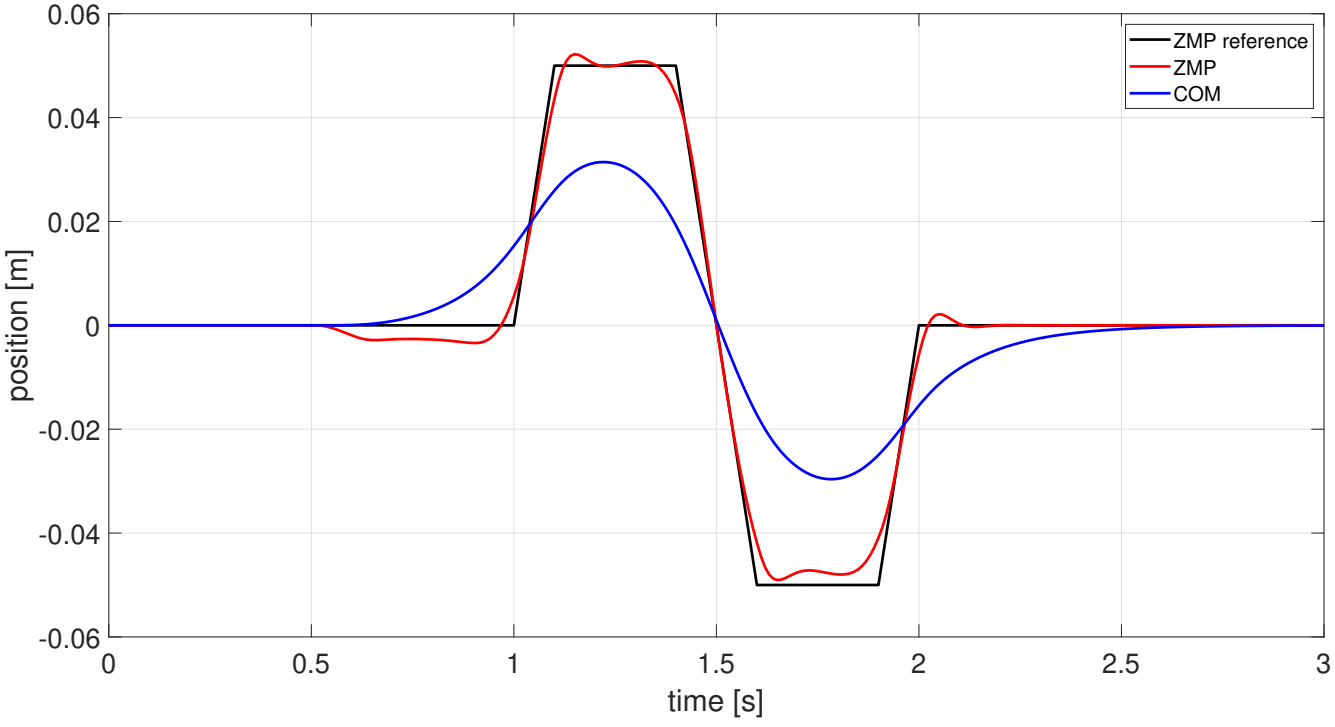


Figure: Reserving space for the preview controller

A ZMP previewing gait

Preview controller (3)

- First strategy: to reserve some space
- Second strategy: to plan ZMP reference in time horizon much longer than simulation time

A 3D LIPM gait

3D LIPM

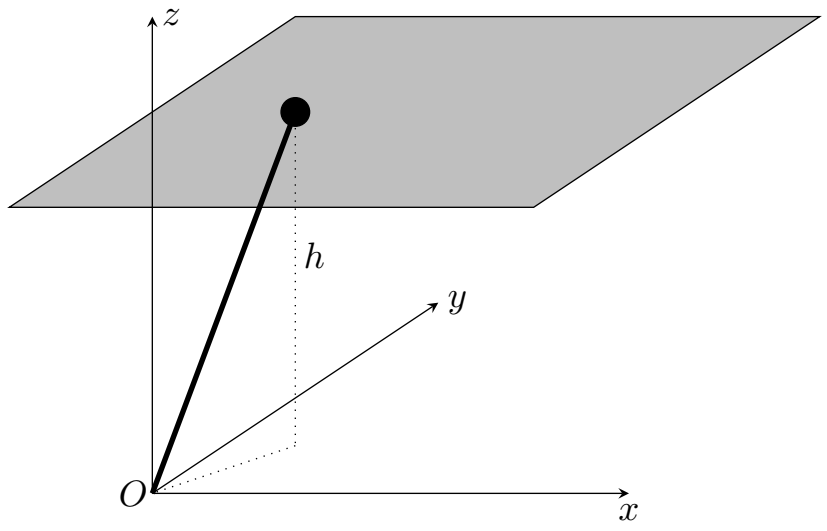


Figure: 3D inverted pendulum

$$x(t) = x_0 \cdot \cosh(k_c \cdot t) + \frac{\dot{x}_0 \cdot \sinh(k_c \cdot t)}{k_c} \quad , \; k_c = \sqrt{\frac{g}{h}}$$

$$\dot{x}(t) = k_c \cdot x_0 \cdot \sinh(k_c \cdot t) + \dot{x}_0 \cdot \cosh(k_c \cdot t)$$

A 3D LIPM gait

Calculation of step duration (1)

- Calculate the step duration iteratively
- Some time points:
  - t_e: end point of the current single support phase
  - t_b: start point of the next single support phase
  - t_0: inflection point of pendulum motion, $\dot{x}_0 = 0$

A 3D LIPM gait

Calculation of step duration (1)

- Calculate the step duration iteratively
- Some time points: t_e, t_b and t_0
- Some coordinate systems:

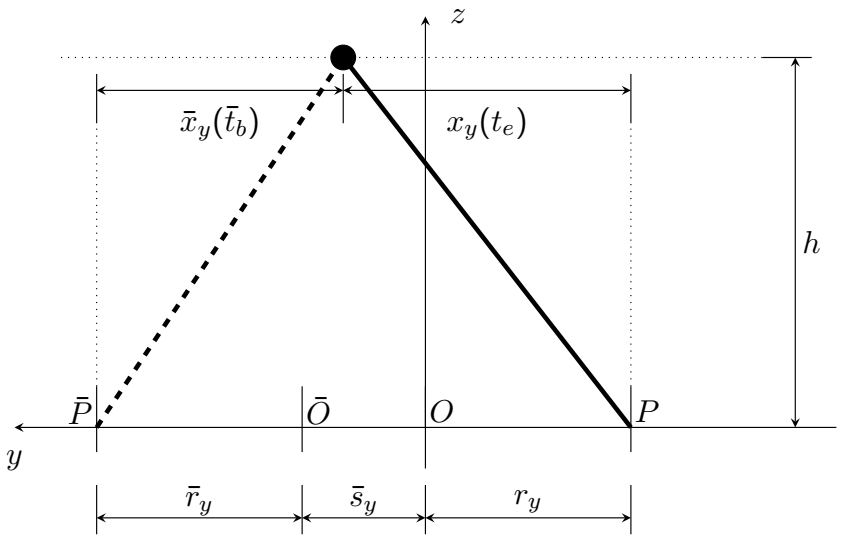


Figure: Two facing inverted pendulum on the $y-z$ plane

$$\bar{r}_y + \bar{s}_y + r_y = \bar{x}_y(\bar{t}_b) + x_y(t_e)$$
$$\dot{x}_y(t_e) = \dot{\bar{x}}_y(\bar{t}_b)$$

A 3D LIPM gait

Calculation of step duration (2)

Algorithm 1 Computing the step duration

1: $t_e \leftarrow$ Initial guess
2: **repeat**
3: $\quad \bar{t}_b \leftarrow \frac{1}{k_c} \cdot \operatorname{arcsinh}\left(\frac{x_{0_y} \cdot \sinh(k_c \cdot t_e)}{\bar{x}_{0_y}}\right)$
4: $\quad \dot{x}_y(t_e) \leftarrow k_c \cdot x_{0_y} \cdot \sinh(k_c \cdot t_e)$
5: $\quad \Delta t_e \leftarrow \frac{\bar{r}_y + \bar{s}_y + r_y - (\bar{x}_y(\bar{t}_b) + x_y(t_e))}{2 \cdot \dot{x}_y(t_e)}$
6: $\quad t_e \leftarrow t_e + \Delta t_e$
7: **until** $|\Delta t_e| < \varepsilon$
8: $\bar{t}_b \leftarrow \frac{1}{k_c} \cdot \operatorname{arcsinh}\left(\frac{x_{0_y} \cdot \sinh(k_c \cdot t_e)}{\bar{x}_{0_y}}\right)$

Walk2014

- One of the most popular gait nowadays in SPL
- Inverted-pendulum-like dynamics
- Not model-based but measurement-based

Walk2014

- One of the most popular gait nowadays in SPL
- Inverted-pendulum-like dynamics
- Not model-based but measurement-based

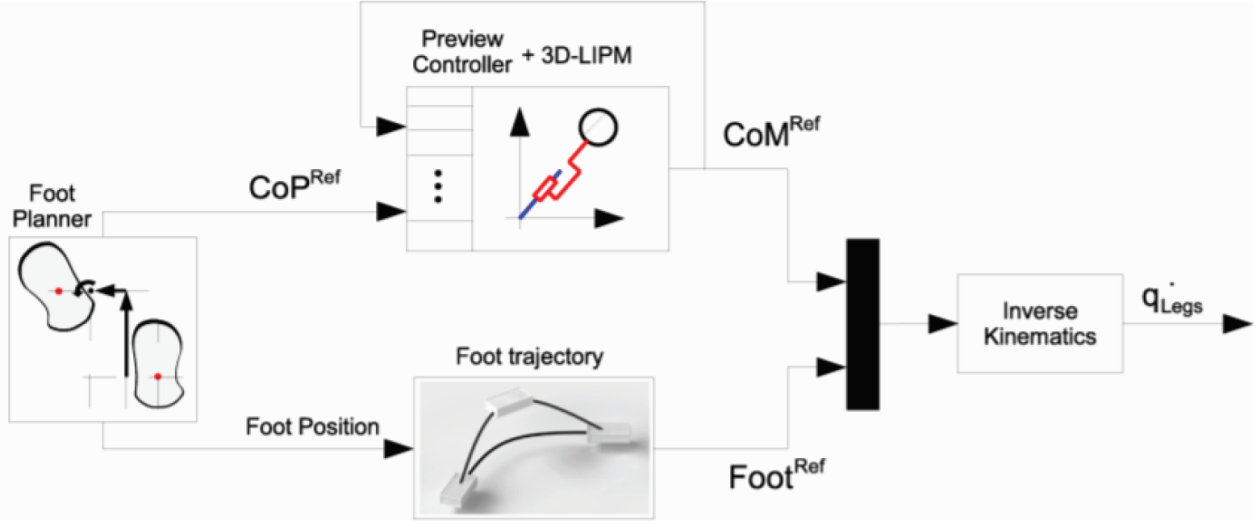


Figure: Process of gait generation

Walk2014

- One of the most popular gait nowadays in SPL
- Inverted-pendulum-like dynamics
- Not model-based but measurement-based

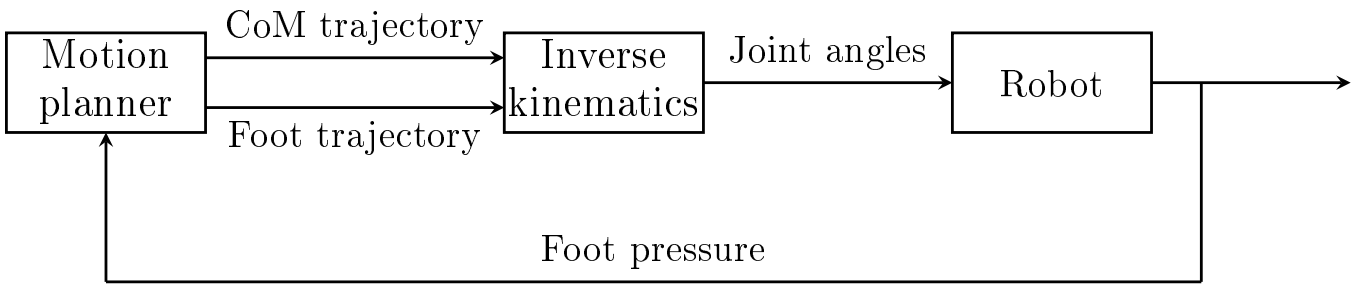


Figure: Process of gait generation for Walk2014

Walk2014

Measurement-based control

- Logic control:

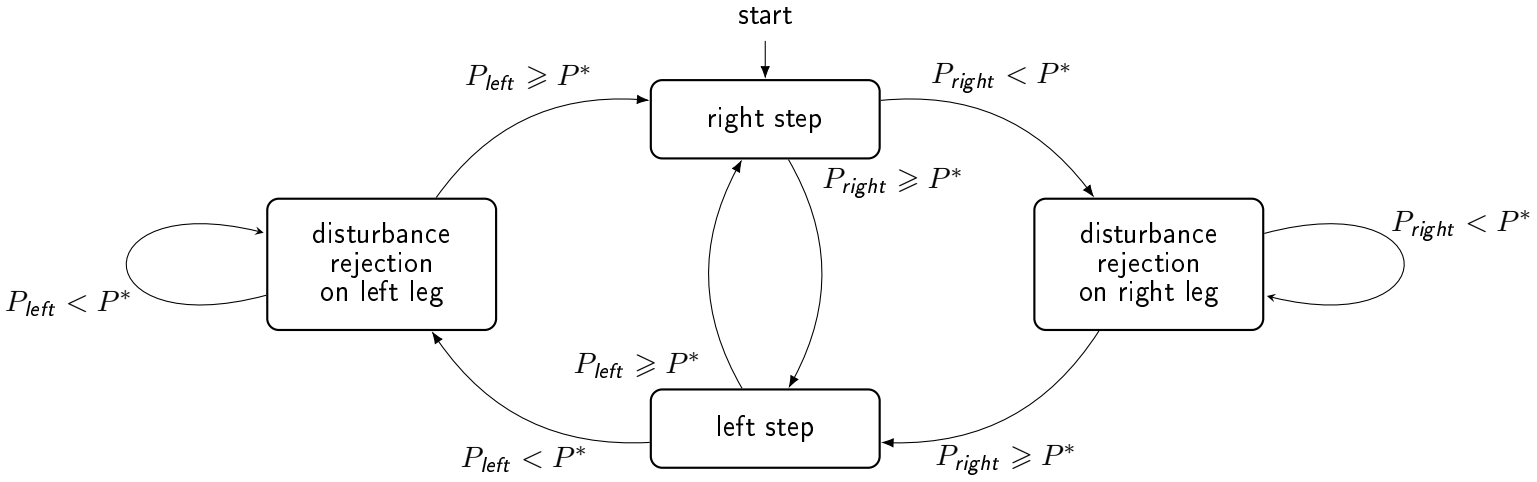


Figure: Logic control of Walk2014

- Pressure filtering:

$$G(s) = \frac{\omega_c}{s + \omega_c}$$

Walk2014

Inverted-pendulum-like dynamics

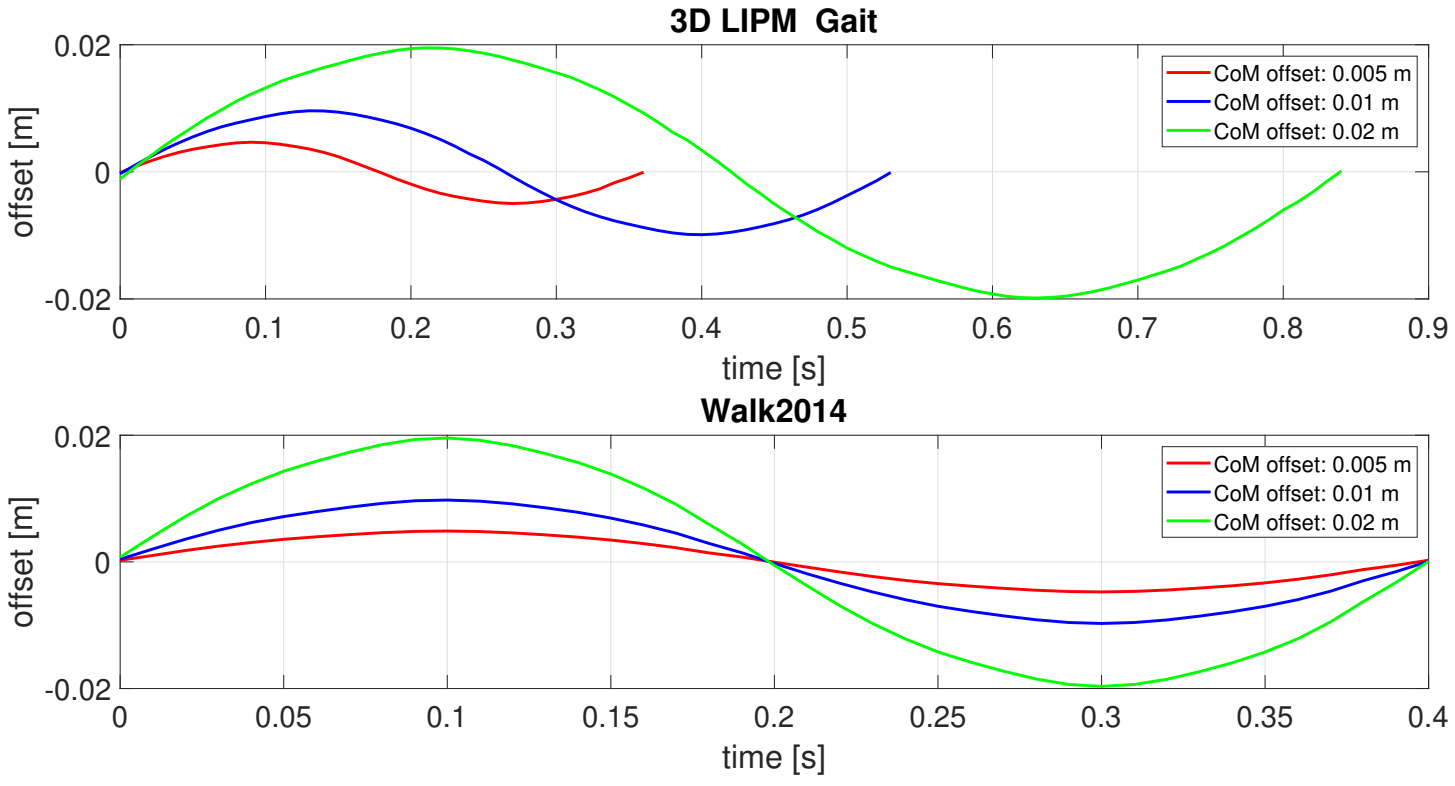


Figure: CoM swinging offset vs. step duration

Simulation results and conclusion

Performance indices

- Walking speed
- Elapse time
- CoM swinging offset
- Ability of disturbance (pushing) rejection

Simulation results and conclusion

Walking speed

Table: Average forward walking speed of different gaits

Gait	Speed [cm/s]
ZMP Previewing Gait	25.73
3D LIPM Gait	32.50
Walk2014	32.77

Table: Average sideward walking speed of different gaits

Gait	Speed [cm/s]
ZMP Previewing Gait	7.30
3D LIPM Gait	9.92
Walk2014	10.43

Simulation results and conclusion

Elapse time

Table: Elapse time for simulating different gaits

Gait	Time [s]
ZMP Previewing Gait	84.29
3D LIPM Gait	59.84
Walk2014	49.52

Simulation results and conclusion

CoM swinging offset

Table: CoM swinging offset of different gaits (step duration 0.2 s)

Gait	Offset [m]
ZMP Previewing Gait	5.8×10^{-3}
3D LIPM Gait	6.6×10^{-3}
Walk2014	flexible

Simulation results and conclusion

Disturbance rejection

Table: Time for disturbance (pushing) rejection of different gaits

Gait	Time [s]
ZMP Previewing Gait	slow because of calculation
3D LIPM Gait	1.01
Walk2014	0.93

Walking with movement primitives

- Preparation of movement primitives for walking
- Dynamic movement primitives (DMPs)[7]
- Robot execution
- Improvement with reinforcement learning

[7] Stefan Schaal. "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics". In: *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

Preparation of movement primitives

The Kinect sensor

Figure: The Kinect sensor[8]

[8] URL: https://en.wikipedia.org/wiki/Kinect.

Preparation of movement primitives

The Kinect sensor

Preparation of movement primitives

Joint angle sequences

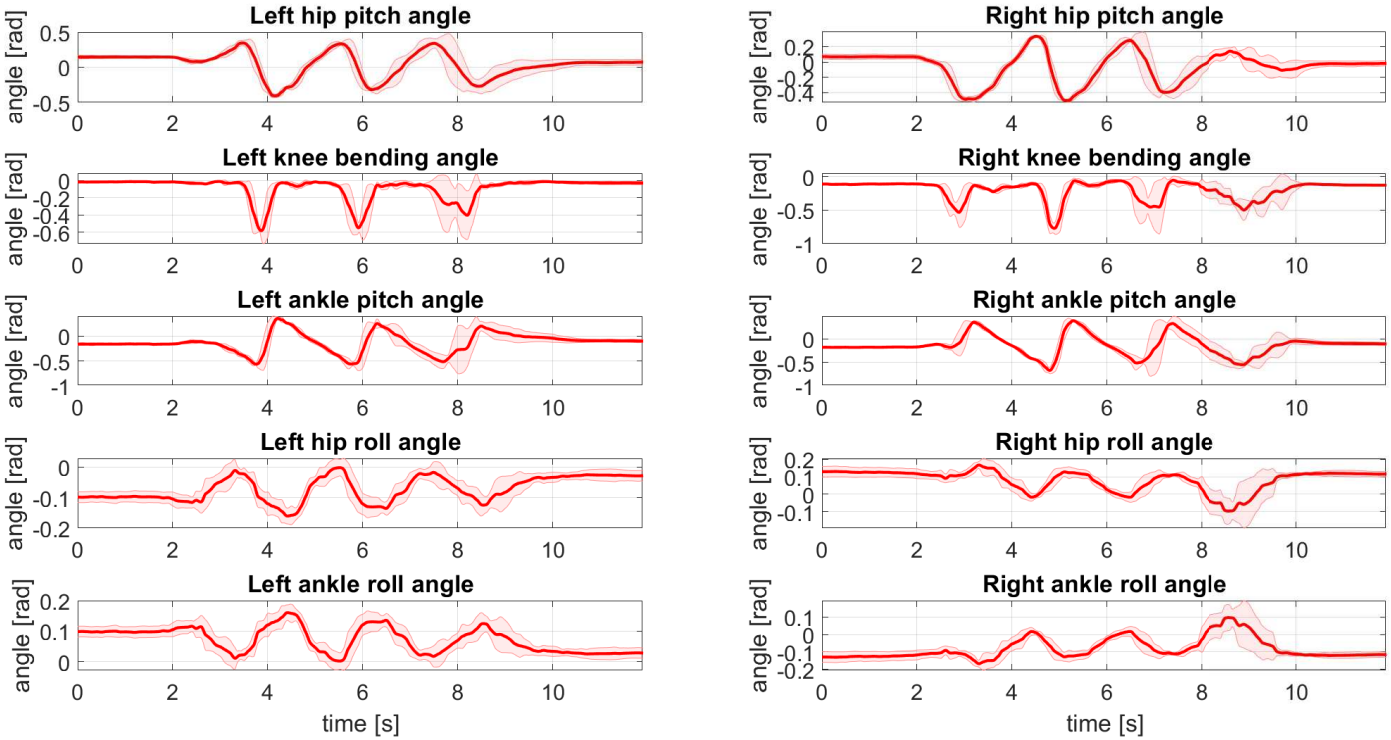


Figure: Joint angle sequences of a straight walking process

Preparation of movement primitives

Joint angle sequences

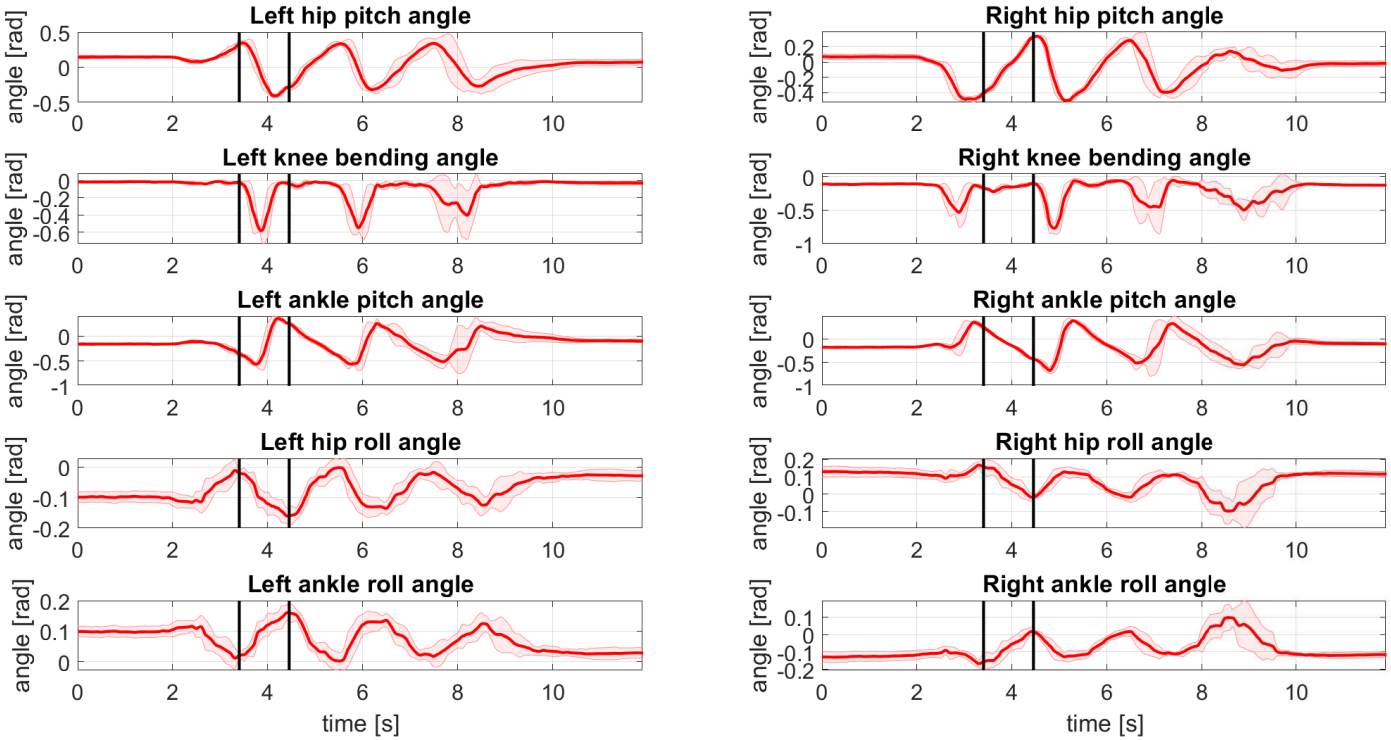


Figure: Joint angle sequences of a straight walking process

Preparation of movement primitives

Joint angle sequences

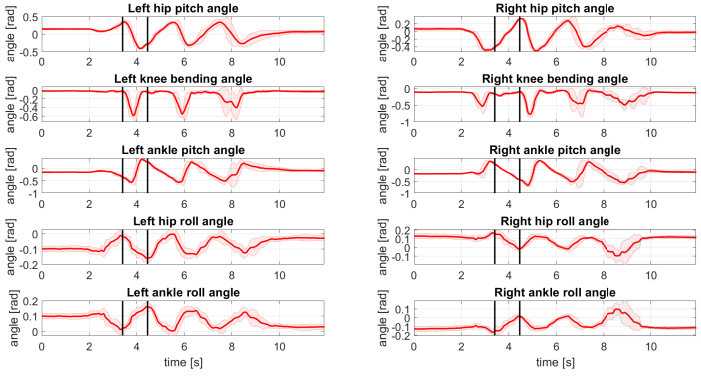


Figure: Joint angle sequences of a straight walking process

Dynamic movement primitives

Basics

- A canonical system: to remove explicit time dependence

$$\dot{x}(t) = -\alpha_x \cdot \frac{1}{\tau} \cdot x(t)$$

  - Faster execution: $0 < \tau < 1$
  - Slower execution: $\tau > 1$

- A second-order system: critically damped, $\alpha_y = 4 \cdot \beta_y$

$$\tau^2 \ddot{y}(t) = \alpha_y \left(\beta_y \left(y_g - y(t)\right) - \tau y(t)\right) + f(t)$$

- A forcing term:

$$f(x) = \frac{\sum_{i=1}^{N_{bf}} w_i \Psi_i(x)}{\sum_{i=1}^{N_{bf}} \Psi_i(x)} x(y_g - y_0)$$

Dynamic movement primitives

Imitation learning (1)

- Movement primitive from human: $\begin{bmatrix} \boldsymbol{y}_D & \dot{\boldsymbol{y}}_D & \ddot{\boldsymbol{y}}_D \end{bmatrix}^T$
- The weights:

$$w_i = \frac{\boldsymbol{s}^T \Gamma_i \boldsymbol{f}_G}{\boldsymbol{s}^T \Gamma_i \boldsymbol{s}}$$

$$\boldsymbol{s} = \begin{bmatrix} x_1 \left(y_{D_T} - y_{D_1}\right) \\ \vdots \\ x_T \left(y_{D_T} - y_{D_1}\right) \end{bmatrix}, \Gamma_i = \begin{bmatrix} \Psi_{i_1} & & 0 \\ & \ddots & \\ 0 & & \Psi_{i_T} \end{bmatrix}, \boldsymbol{f}_G = \begin{bmatrix} f_{G_1} \\ \vdots \\ f_{G_T} \end{bmatrix}$$

$$\boldsymbol{f}_G = \ddot{\boldsymbol{y}}_D - \alpha_y \left(\beta_y (y_{D_T} - \boldsymbol{y}_D) - \dot{\boldsymbol{y}}_D\right)$$

Dynamic movement primitives

Imitation learning (1)

- The weights:

$$w_i = \frac{\boldsymbol{s}^T \Gamma_i \boldsymbol{f}_G}{\boldsymbol{s}^T \Gamma_i \boldsymbol{s}}$$

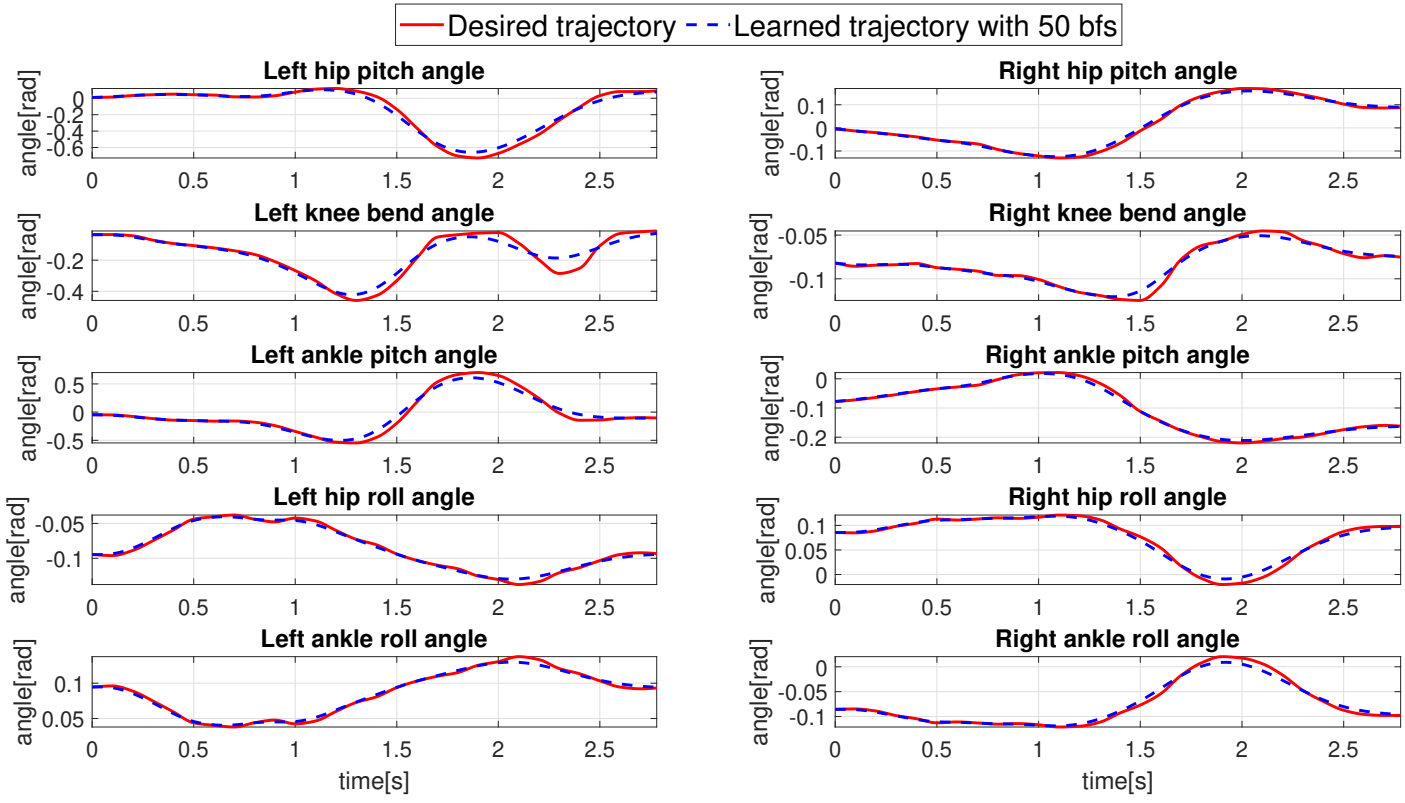
- The regeneration process:

Algorithm 2 Regenerating a given curve

Initializing: $y_{R_1} = y_{D_1}$, $\dot{y}_{R_1} = 0$, $\ddot{y}_{R_1} = 0$
for $j = 2$ to T **do**
  $\ddot{y}_{R_{j-1}} = \frac{1}{\tau^2}\left(\alpha_y\left(\beta_y(y_{D_T} - y_{R_{j-1}}) - \tau\dot{y}_{R_{j-1}}\right) - f(x_{j-1})\right)$
  $\dot{y}_{R_j} = \dot{y}_{R_{j-1}} + \ddot{y}_{R_{j-1}} \cdot \Delta t$
  $y_{R_j} = y_{R_{j-1}} + \dot{y}_{R_{j-1}} \cdot \Delta t$
end for

Dynamic movement primitives

Imitation learning (2)



Dynamic movement primitives

Temporal scaling

- Temporal scaling factor: τ
  - Faster execution: $0 < \tau < 1$
  - Slower execution: $\tau > 1$
- Theoretical bound: instability comes from the explicit Euler method

$$\frac{\alpha_y \cdot \Delta t}{4} < \tau$$

- Physical execution bound: achieved by trial and error

$$0.5 \leqslant \tau$$

Dynamic movement primitives

Primitives concatenation (1)

Reproduction and concatenation of movement primitives

Algorithm 3 Reproducing and concatenating movements

1: From the second primitive on: initializing y_{R_1} with the end value of the last primitive and $\dot{y}_{R_1} = 0$, $\ddot{y}_{R_1} = 0$
2: **for** $j = 2$ to T **do**
3: $\quad \ddot{y}_{R_{j-1}} = \frac{1}{\tau^2}\left(\alpha_y\left(\beta_y(y_{D_T} - y_{R_{j-1}}) - \tau\dot{y}_{R_{j-1}}\right) - f(x_{j-1})\right)$
4: $\quad \dot{y}_{R_j} = \dot{y}_{R_{j-1}} + \ddot{y}_{R_{j-1}} \cdot \Delta t$
5: $\quad y_{R_j} = y_{R_{j-1}} + \dot{y}_{R_{j-1}} \cdot \Delta t$
6: **end for**

Dynamic movement primitives

Primitives concatenation (2)

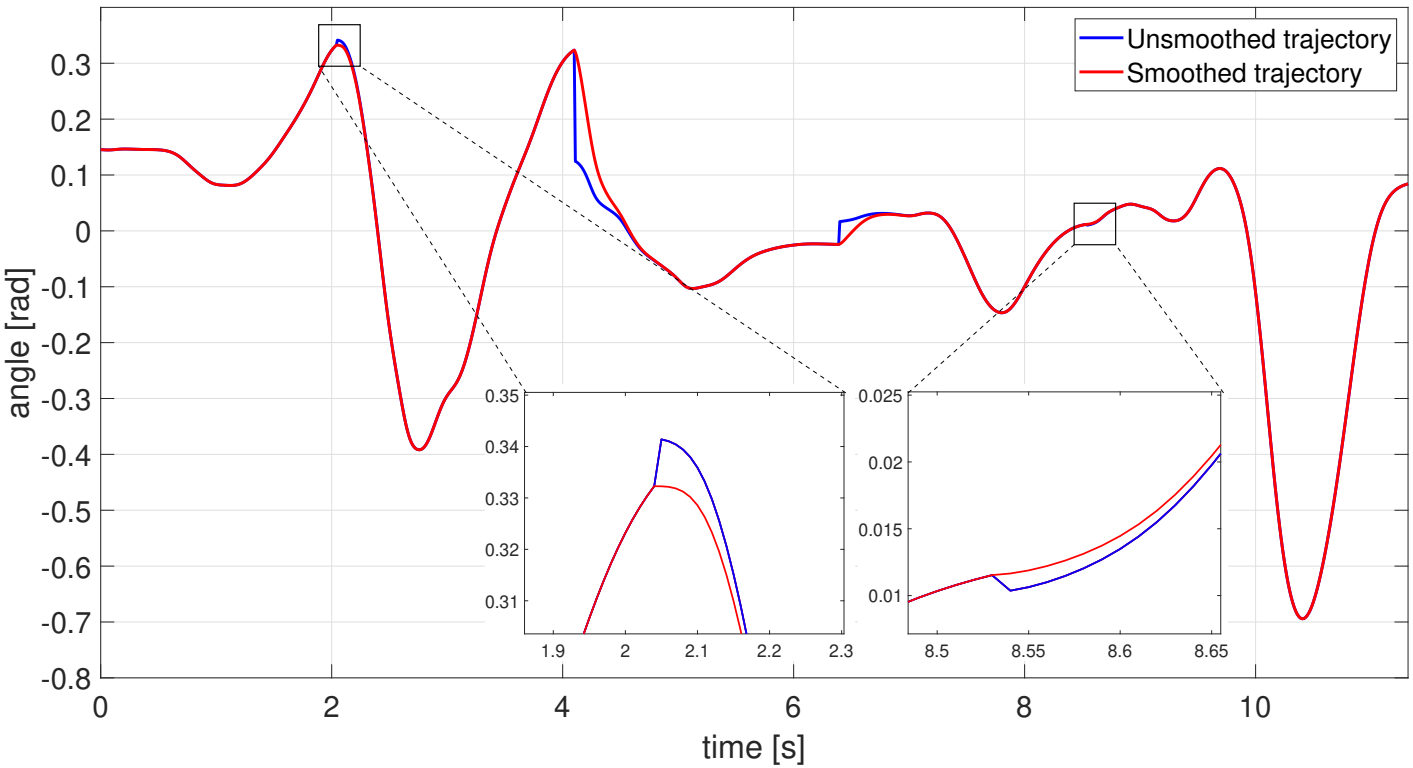


Figure: Regeneration and concatenation of several movement primitives

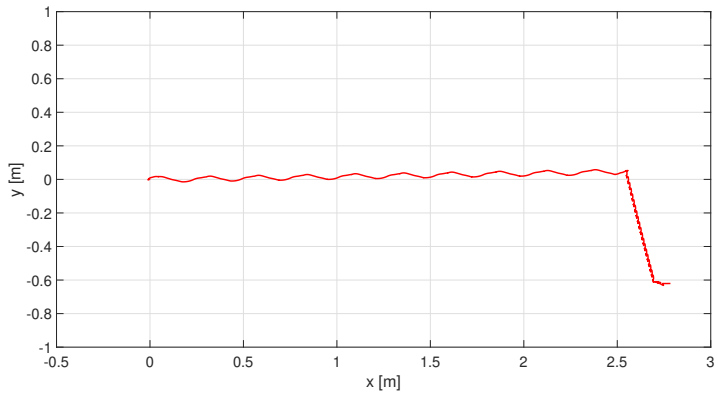
Robot execution

Animation and simulation results

- Model: Walk2014
- A scenario: 10 straight walking steps, 20 sideward walking steps and a kick

Robot execution

Animation and simulation results



CoM trajectory (top view)

Primitive improvement

Reinforcement learning

- Reinforcement learning: at time step t
  - Environment state: $s_t \in \mathcal{S}$
  - Action: $a_t \in \mathcal{A}$
  - Policy: $\pi(s_t)$
  - Reward: $R_{t+1} \in \mathcal{R}$ or R
- Policy learning by Weighting Exploration with the Returns (PoWER)[8]

[8] Jens Kober and Jan Peters. "Learning motor primitives for robotics". In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE. 2009, pp. 2112–2118.

Primitive improvement

Reward functions

- Stability: $R_{stability} = \begin{cases} 0 & \text{if stable} \\ -10000 & \text{otherwise} \end{cases}$
- Joint angles: $R_{joint} = \begin{cases} -10000 & \text{if beyond limits} \\ 0 & \text{otherwise} \end{cases}$
- Speed: $R_{sideward} = k_{sideward} \cdot d_{sideward}$
- Elimination of undesired movements:

$$R_{x_{offset}} = \begin{cases} 0 & \text{if } |x_f - x_0| \leqslant 0.01 \\ -k_{x_{offset}} \cdot |x_f - x_0| & \text{otherwise} \end{cases}$$

$$R_{y_{offset}} = \begin{cases} 0 & \text{if } |y_f - y_0| \leqslant 0.01 \\ -k_{y_{offset}} \cdot |y_f - y_0| & \text{otherwise} \end{cases}$$

Primitive improvement

PoWER

- At k^{th} rollout, perform a single rollout with stochastic exploration on weights: $w'_k = w_k + \phi$, $\phi \sim \mathcal{N}\left(\mu, \sigma^2\right)$
- Evaluate this rollout with reward functions
- Store best rollouts in an importance sampler with capacity M
- After k^{th} rollout, update the weights: $w_{k+1} = w_k + \frac{\sum_{i=1}^{M} \phi_i \cdot R_i}{\sum_{i=1}^{M} R_i}$
- Until $w_{k+1} \approx w_k$

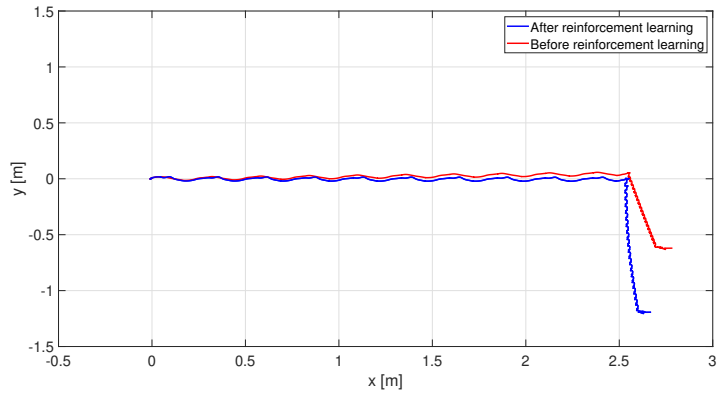
Primitive improvement

Animation and simulation results

- The same scenario: 10 straight walking steps, 20 sideward walking step and a kick

Primitive improvement

Animation and simulation results



CoM trajectories (top view)

Conclusion and future work

Conclusion and future work

- Best biped walking model: Walk2014
- The walking model can be partially replaced by primitives

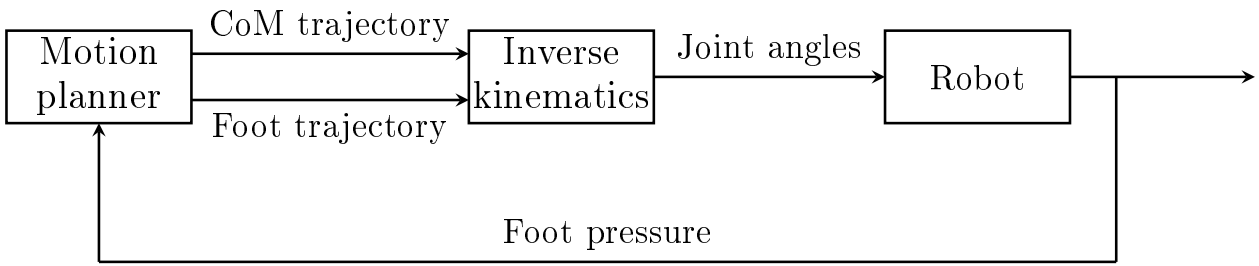


Figure: Process of gait generation for Walk2014

The End

Thank you very much for your attention!

Simulation data of friction coefficient

Table: Walking distance in 10 s of a simple gait with different friction coefficients

	$\mu_k = 1^*$	$\mu_k = 10$	$\mu_k = 100$	$\mu_k = 1000$	$\mu_k = 10000$
$\mu_s = 1^{**}$	0.250***	0.465	0.522	0.530	0.530
$\mu_s = 10$	0.448	0.975	1.088	1.090	1.090
$\mu_s = 100$	0.468	1.297	1.299	1.299	1.299
$\mu_s = 1000$	0.473	1.326	1.330	1.331	1.331
$\mu_s = 10000$	0.394	1.330	1.330	1.331	1.331

*The kinetic friction coefficient μ_k is dimensionless.

**The static friction coefficient μ_s is dimensionless.

***The measured data in this table are length values with unit [m].

Simulation data of friction coefficient

Table: Elapse time of a simple gait with different friction coefficients

	$\mu_k = 1^{*}$	$\mu_k = 10$	$\mu_k = 100$	$\mu_k = 1000$	$\mu_k = 10000$
$\mu_s = 1^{**}$	23.39***	24.80	26.59	35.36	51.10
$\mu_s = 10$	24.20	23.31	26.19	27.73	30.67
$\mu_s = 100$	24.76	25.59	25.03	26.14	27.55
$\mu_s = 1000$	26.60	27.01	26.28	26.98	28.15
$\mu_s = 10000$	29.93	29.09	27.77	28.94	29.27

*The kinetic friction coefficient μ_k is dimensionless.

**The static friction coefficient μ_s is dimensionless.

***The measured data in this table are time values with unit [s].

Explanation of coordinate system of 3D LIPM

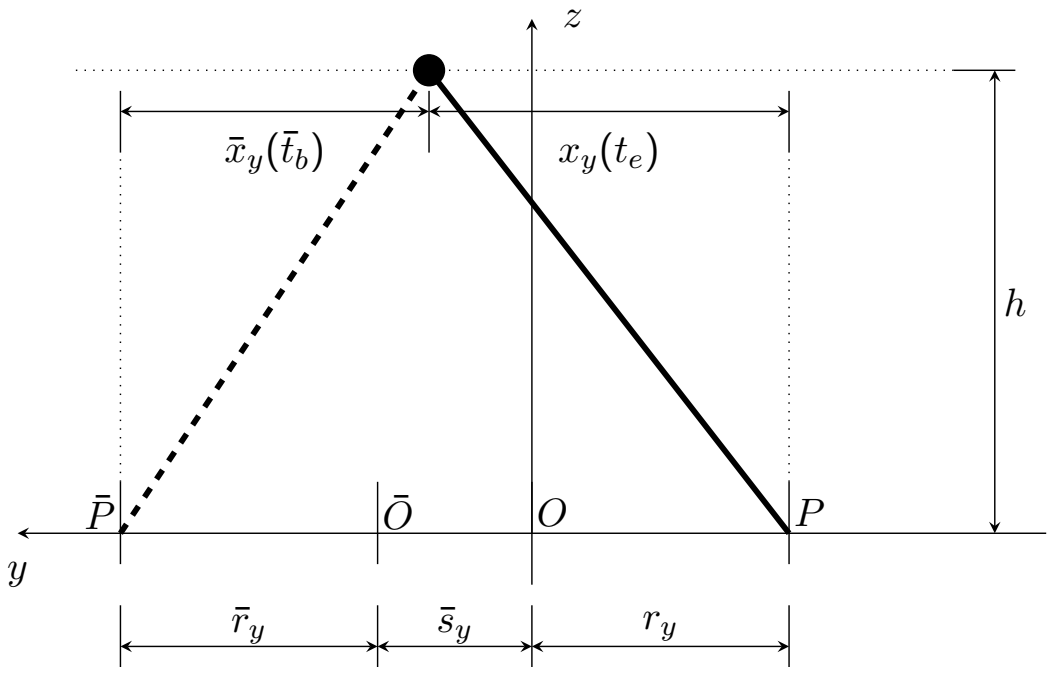


Figure: Two facing inverted pendulum on the $y - z$ plane

$$\bar{r}_y + \bar{s}_y + r_y = \bar{x}_y(\bar{t}_b) + x_y(t_e)$$
$$\dot{x}_y(t_e) = \dot{\bar{x}}_y(\bar{t}_b)$$

Forward and sideward trajectories of three gaits

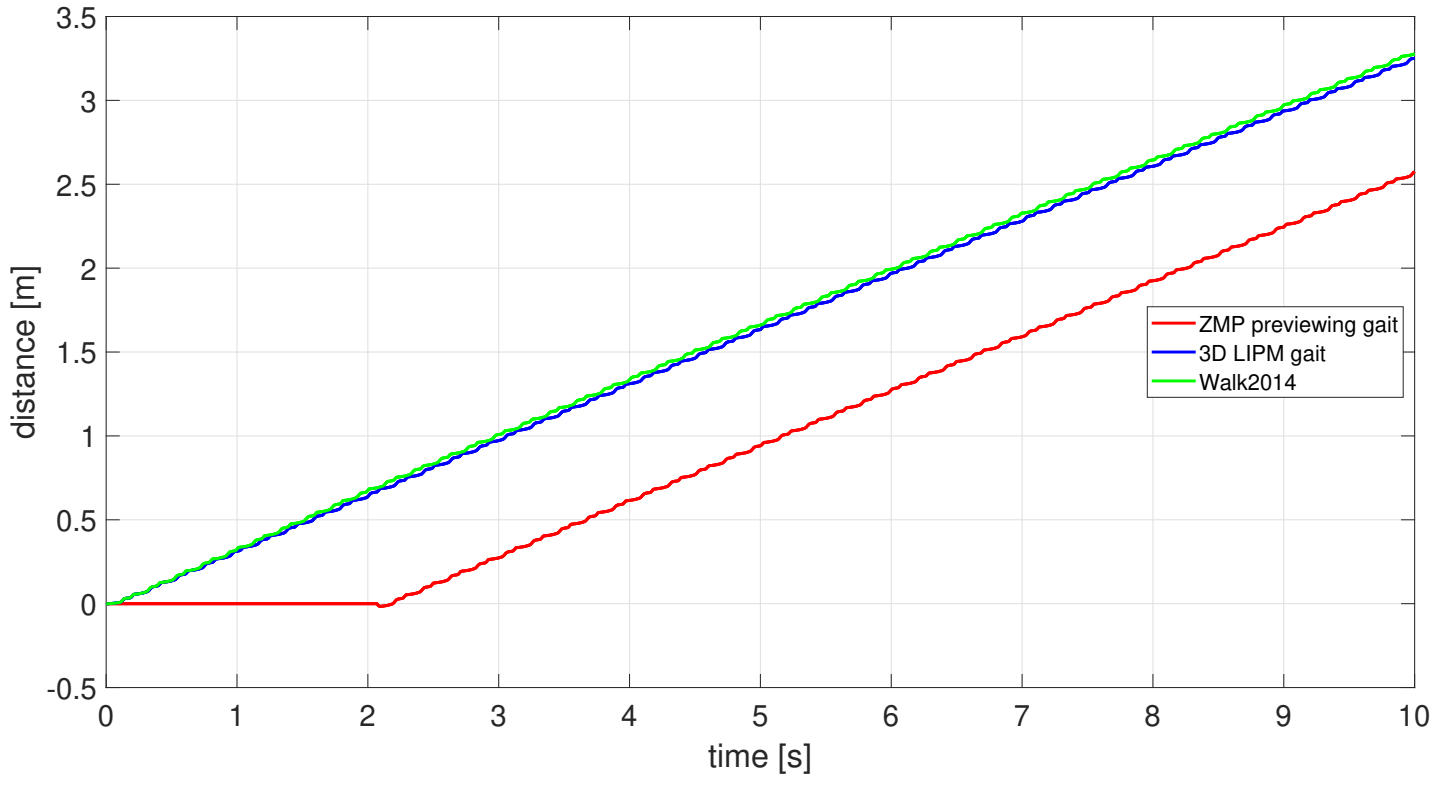


Figure: CoM trajectory of three gaits during forward walking

Forward and sideward trajectories of three gaits

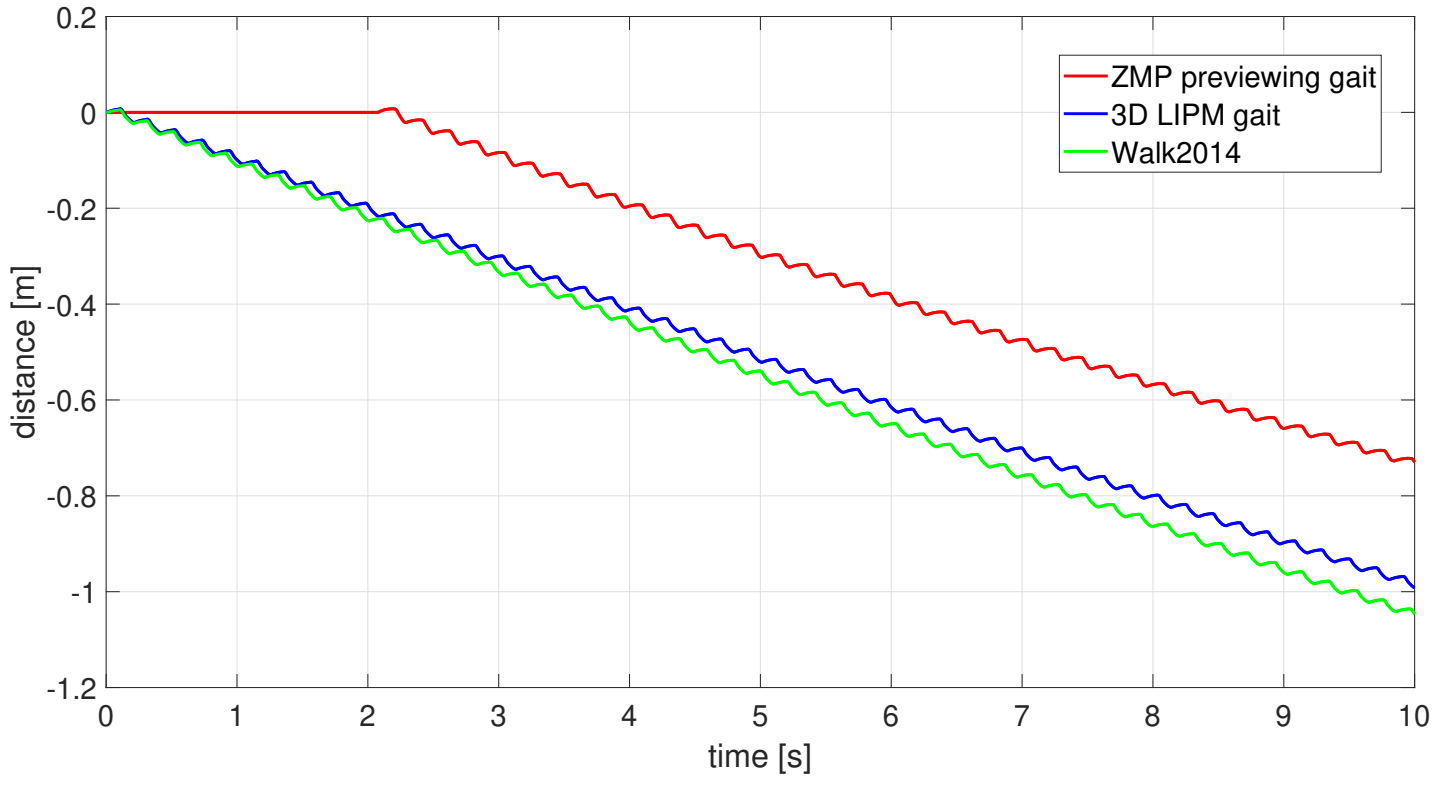
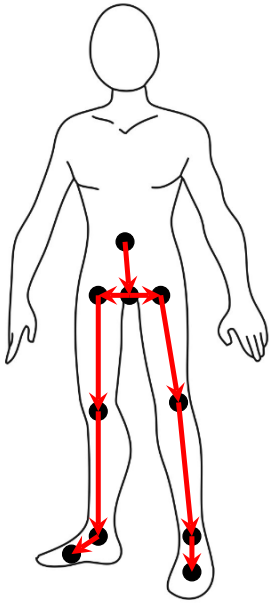


Figure: CoM trajectory of three gaits during sideward walking

Mapping from 3D points to joint angles

- Direction vectors:



$$\boldsymbol{v}_{knee2ankle} = \begin{bmatrix} x_{ankle} - x_{knee} \\ y_{ankle} - y_{knee} \\ z_{ankle} - z_{knee} \end{bmatrix}$$

- Joint angle calculation: Euclidean dot product formula

$$\cos\theta = \frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_2}{\|\boldsymbol{v}_1\|\|\boldsymbol{v}_2\|}$$

Imitation learning

Calculating the weights

- A given curve: $\begin{bmatrix} \boldsymbol{y}_D & \dot{\boldsymbol{y}}_D & \ddot{\boldsymbol{y}}_D \end{bmatrix}^T$
- In finite time horizon: $[1, 2, \cdots, T]$
- A goal forcing term:

$$\boldsymbol{f}_G = \ddot{\boldsymbol{y}}_D - \alpha_y \left(\beta_y (y_{D_T} - \boldsymbol{y}_D) - \dot{\boldsymbol{y}}_D\right)$$

Imitation learning

Calculating the weights

- A given curve: $\begin{bmatrix} \boldsymbol{y}_D & \dot{\boldsymbol{y}}_D & \ddot{\boldsymbol{y}}_D \end{bmatrix}^T$
- In finite time horizon: $[1, 2, \cdots, T]$
- A goal forcing term: $\boldsymbol{f}_G$
- Locally Weighted Regression (LWR):

$$w_i = \frac{\boldsymbol{s}^T \Gamma_i \boldsymbol{f}_G}{\boldsymbol{s}^T \Gamma_i \boldsymbol{s}}$$

Imitation learning

Calculating the weights

- A given curve: $\begin{bmatrix} \boldsymbol{y}_D & \dot{\boldsymbol{y}}_D & \ddot{\boldsymbol{y}}_D \end{bmatrix}^T$
- In finite time horizon: $[1, 2, \cdots, T]$
- A goal forcing term: $\boldsymbol{f}_G$
- Locally Weighted Regression (LWR): w_i
- The cost function for each Gaussian function Ψ_i

$$J_i = \sum_{j=1}^{T} \Psi_i(j) \left[f_G(j) - w_i \left(y_{D_T} - y_D(j) \right) x(j) \right]^2$$

Imitation learning

Calculating the weights

- A given curve: $\begin{bmatrix} \boldsymbol{y}_D & \dot{\boldsymbol{y}}_D & \ddot{\boldsymbol{y}}_D \end{bmatrix}^T$
- In finite time horizon: $[1, 2, \cdots, T]$
- A goal forcing term: $\boldsymbol{f}_G$
- Locally Weighted Regression (LWR): $w_i = \frac{\boldsymbol{s}^T \Gamma_i \boldsymbol{f}_G}{\boldsymbol{s}^T \Gamma_i \boldsymbol{s}}$

$$\boldsymbol{s} = \begin{bmatrix} x_1 \left(y_{D_T} - y_{D_1}\right) \\ \vdots \\ x_T \left(y_{D_T} - y_{D_1}\right) \end{bmatrix}, \Gamma_i = \begin{bmatrix} \Psi_{i_1} & & 0 \\ & \ddots & \\ 0 & & \Psi_{i_T} \end{bmatrix}, \boldsymbol{f}_G = \begin{bmatrix} f_{G_1} \\ \vdots \\ f_{G_T} \end{bmatrix}$$

Imitation learning

Calculating the forcing term

- The calculated weights:

$$w_i = \frac{\boldsymbol{s}^T \Gamma_i \boldsymbol{f}_G}{\boldsymbol{s}^T \Gamma_i \boldsymbol{s}}$$

- The forcing term:

$$f(x) = \frac{\sum_{i=1}^{N_{bf}} w_i \Psi_i(x)}{\sum_{i=1}^{N_{bf}} \Psi_i(x)} x(y_g - y_0)$$

Ψ_i: Gaussian function $\Psi_i(x) = e^{-h_i(x-c_i)^2}$

w_i: weight of Gaussian function Ψ_i

N_{bf}: number of Gaussian functions

x: the canonical system

$(y_g - y_0)$: a spatial scaling term

Imitation learning

Regenerating a given curve

- A curve to be regenerated: $\begin{bmatrix} \boldsymbol{y}_R & \dot{\boldsymbol{y}}_R & \ddot{\boldsymbol{y}}_R \end{bmatrix}^T$
- Regeneration of a given curve: explicit Euler method with step size Δt

Algorithm 2 Regenerating a given curve

Initializing: $y_{R_1} = y_{D_1}$, $\dot{y}_{R_1} = 0$, $\ddot{y}_{R_1} = 0$
for $j = 2$ to T **do**
  $\ddot{y}_{R_{j-1}} = \frac{1}{\tau^2}\left(\alpha_y\left(\beta_y(y_{D_T} - y_{R_{j-1}}) - \tau\dot{y}_{R_{j-1}}\right) - f(x_{j-1})\right)$
  $\dot{y}_{R_j} = \dot{y}_{R_{j-1}} + \ddot{y}_{R_{j-1}} \cdot \Delta t$
  $y_{R_j} = y_{R_{j-1}} + \dot{y}_{R_{j-1}} \cdot \Delta t$
end for

Temporal scaling and stability analysis

- Temporal scaling factor: τ
  - Faster execution: $0 < \tau < 1$
  - Slower execution: $\tau > 1$
- Instability comes from the explicit Euler method:

$$\ddot{y}_{j+1} = \frac{1}{\tau^2}\alpha_y\left(\beta_y(y_g - y_j) - \tau\dot{y}_j\right)$$
$$\dot{y}_{j+1} = \dot{y}_j + \ddot{y}_j \cdot \Delta t$$
$$y_{j+1} = y_j + \dot{y}_j \cdot \Delta t$$

$$\begin{bmatrix} y_{j+1} \\ \dot{y}_{j+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ -\frac{\alpha_y \cdot \beta_y \cdot \Delta t}{\tau^2} & 1 - \frac{\alpha_y \cdot \Delta t}{\tau} \end{bmatrix} \begin{bmatrix} y_j \\ \dot{y}_j \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{\alpha_y \cdot \beta_y \cdot \Delta t}{\tau^2} \cdot y_g \end{bmatrix}$$

- Theoretical bound:

$$\frac{\alpha_y \cdot \Delta t}{4} < \tau$$