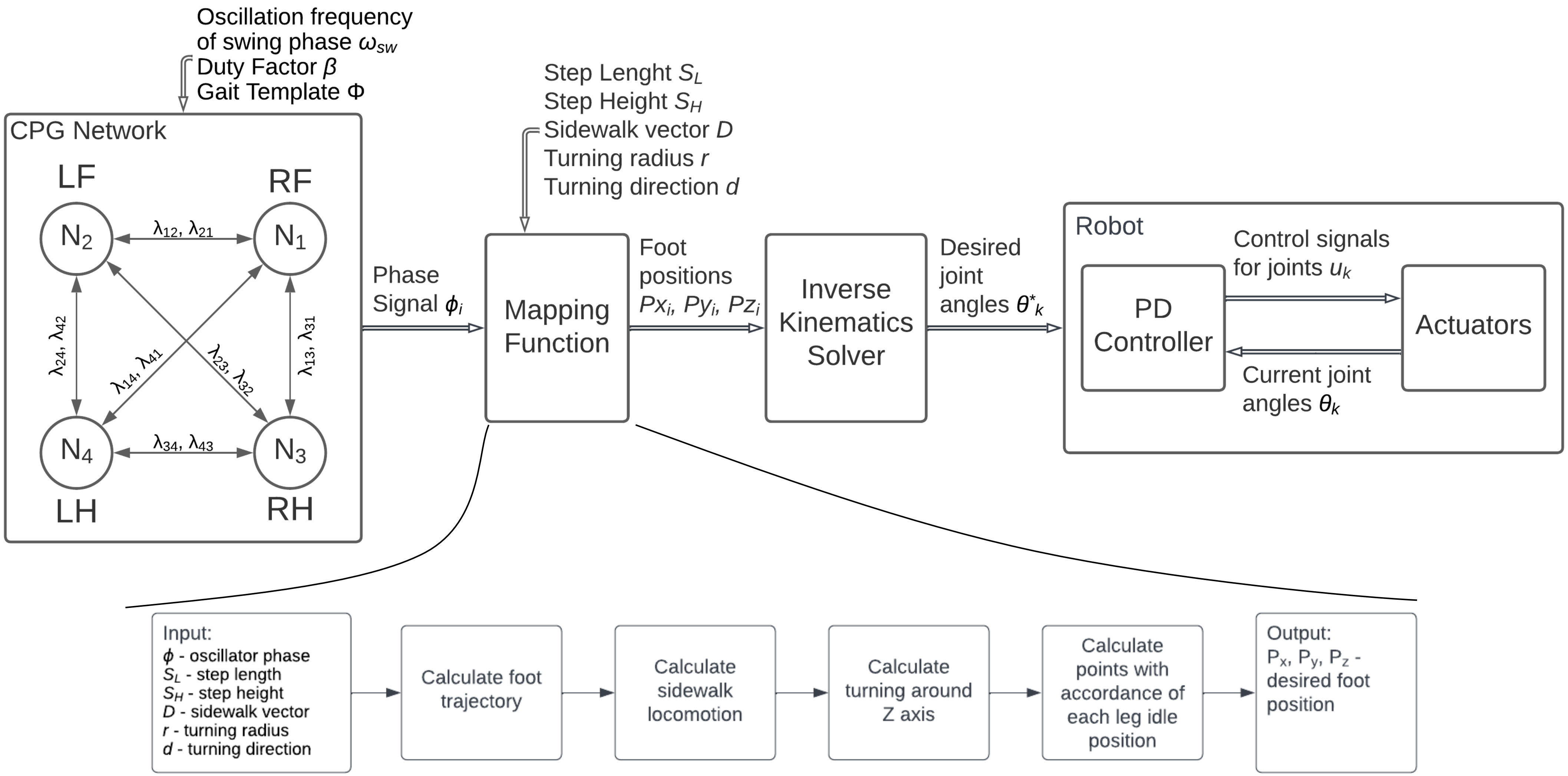
Control of Simplified Walking Robot Model using PMTG architecture

Vladimir Danilov

Block diagram of CPG-based Gait Generator

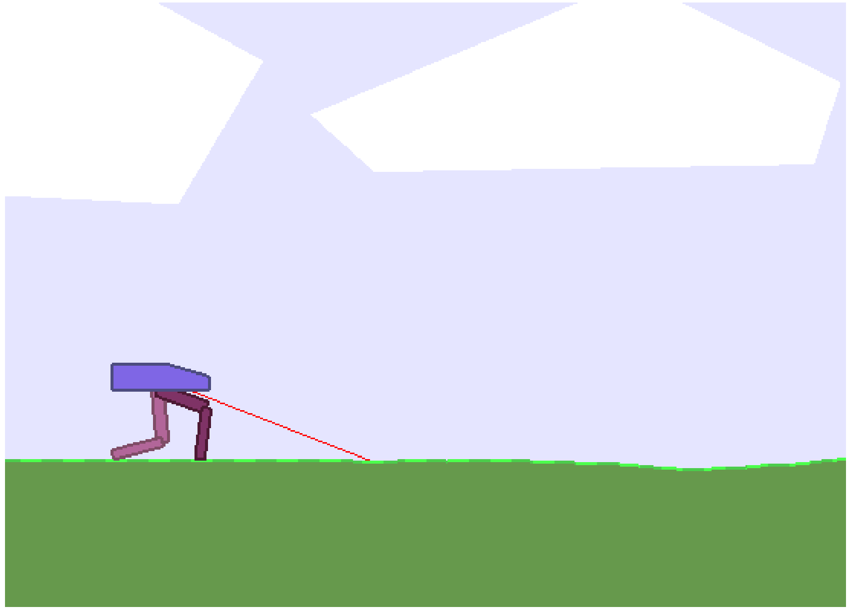


Experimental Hardware Results

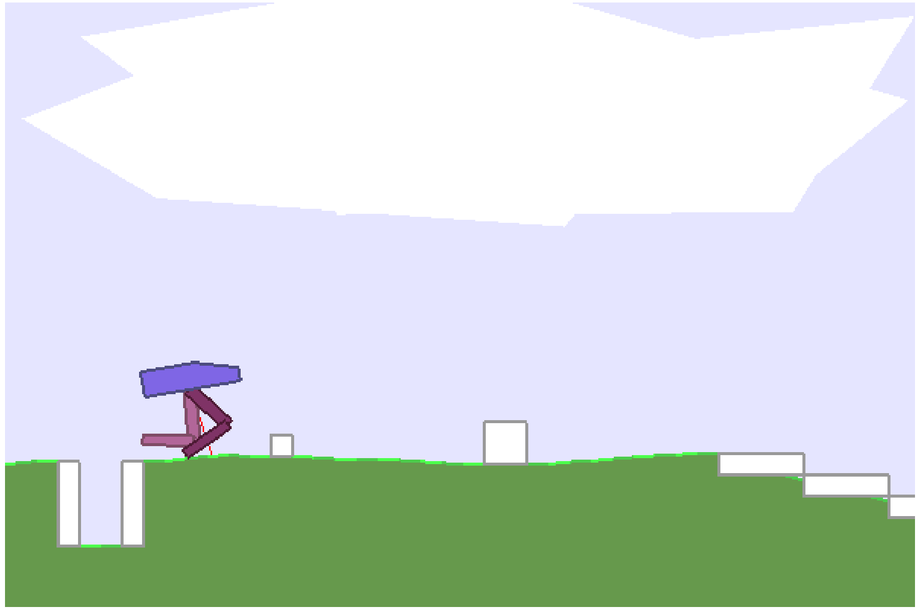
[Watch on Youtube]

Simplified Walking Robot Model

Bipedal Walker

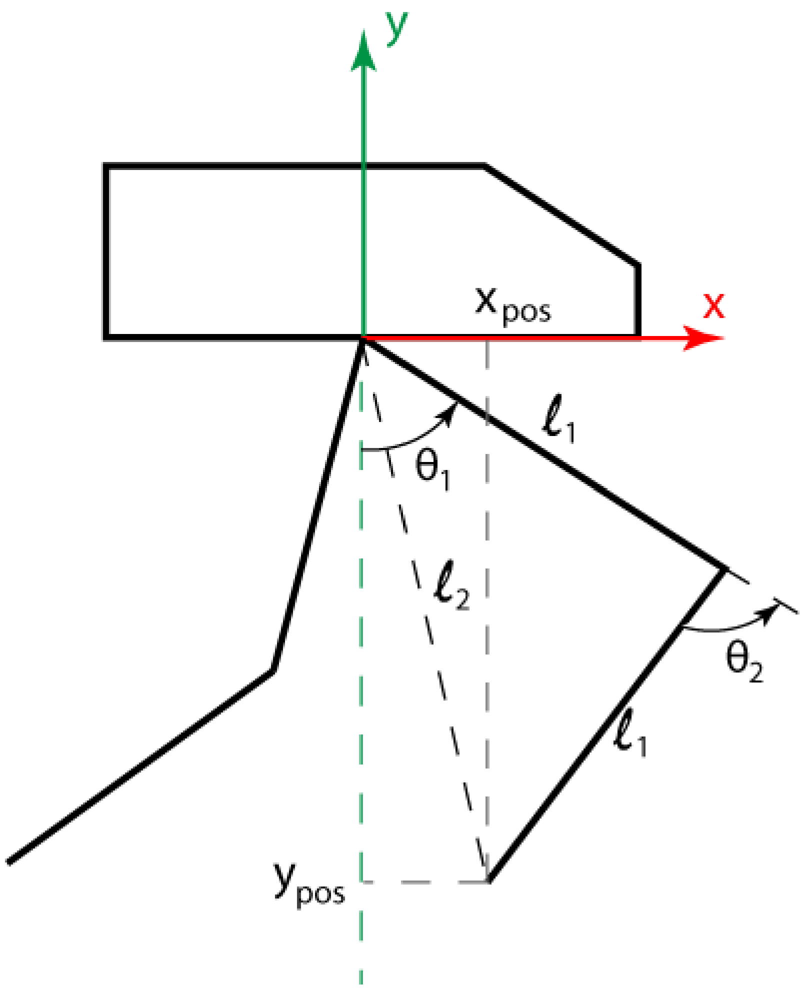


Bipedal Walker Hardcore



Starting State: Random position upright and mostly straight legs
Episode Termination: when the robot body touches ground or the robot reaches far right side of the environment
Solved Requirements: to get average reward greater than 300

Kinematics



Forward Kinematics:

$$l_2 = 2l_1^2(1 + \sin\theta_2)$$

$$x_{pos} = l_2 \sin\left(\theta_1 + \frac{\theta_2}{2}\right)$$

$$y_{pos} = \sqrt{l_2^2 - x_{pos}^2}$$

Inverse Kinematics:

$$l_2 = \sqrt{x_{pos}^2 + y_{pos}^2}$$

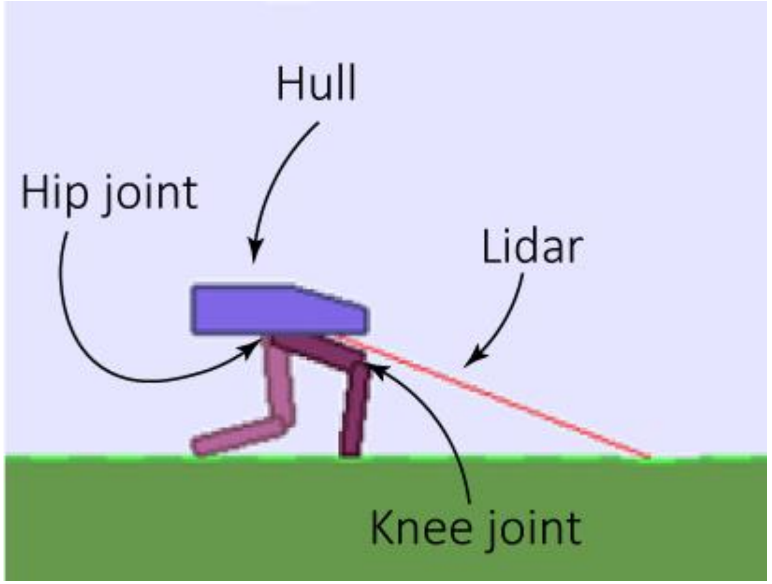
$$\theta_1 = \mathrm{atan}\frac{x_{pos}}{y_{pos}} + \mathrm{acos}\frac{l_2}{2l_1}$$

$$\theta_2 = -2\,\mathrm{acos}\frac{l_2}{2l_1}$$

Environment

Num	Observation	Min	Max
0	hull_angle	0	2π
1	hull_angularVelocity	-∞	+∞
2	vel_x	-1	+1
3	vel_y	-1	+1
4	hip_joint_1_angle	-∞	+∞
5	hip_joint_1_speed	-∞	+∞
6	knee_joint_1_angle	-∞	+∞
7	knee_joint_1_speed	-∞	+∞
8	leg_1_ground_contact_flag	0	1
9	hip_joint_2_angle	-∞	+∞
10	hip_joint_2_speed	-∞	+∞
11	knee_joint_2_angle	-∞	+∞
12	knee_joint_2_speed	-∞	+∞
13	leg_2_ground_contact_flag	0	1
14-23	10 lidar readings	-∞	+∞

Num	Action	Min	Max
0	Hip_1 (Torque / Velocity)	-1	+1
1	Knee_1 (Torque / Velocity)	-1	+1
2	Hip_2 (Torque / Velocity)	-1	+1
3	Knee_2 (Torque / Velocity)	-1	+1



Reward function:

$$r_{fw} = \frac{13}{3}\left(p_x(t) - p_x(t-1)\right) - moving\ forward\ reward$$

$$r_{hull} = -5(|\vartheta(t)| - |\vartheta(t-1)|) - hull\ deviation\ penalty$$

$$r_\tau = -0{,}028\sum_{i=1}^{12}|a_i| - torque\ penalty$$

$$r_{es} = \begin{cases} -100, & if\ p_x < 0\ \ or\ hull\ touches\ ground \\ 0, & otherwise \end{cases} - early\ stopping\ penalty$$

$$R = r_{fw} + r_{hull} + r_\tau + r_{es} - total\ reward,$$

$where:$

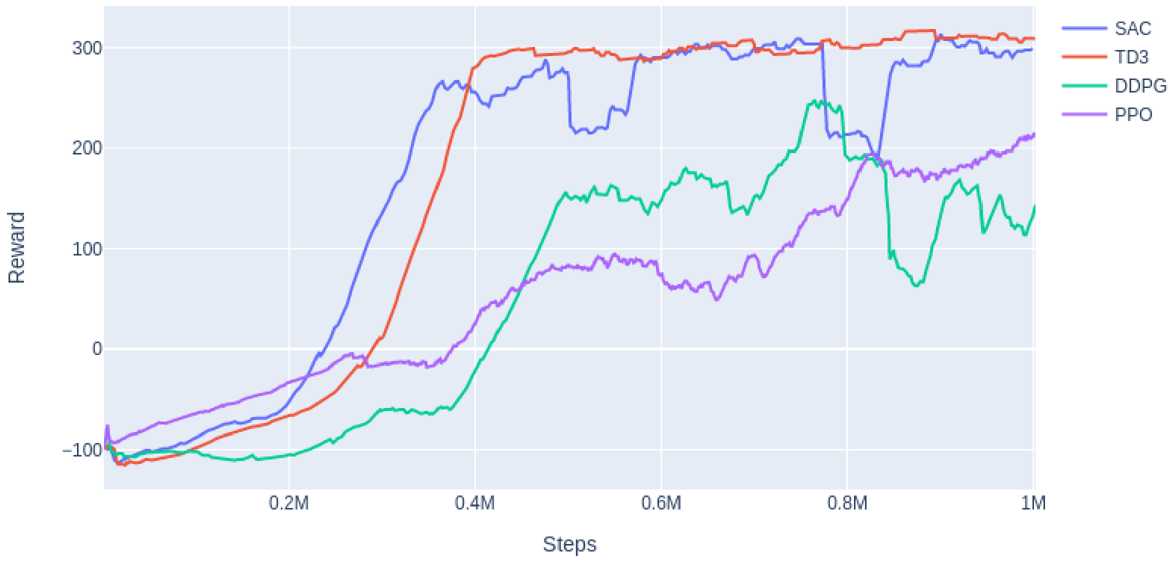
$p_x - x\ position,$

$\vartheta - hull\ angle$

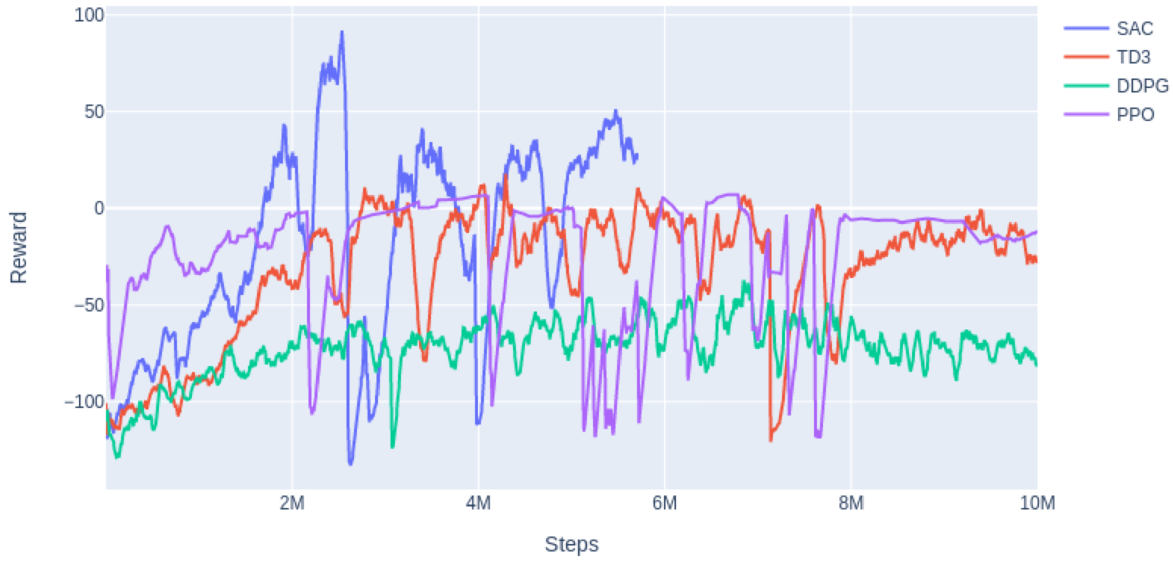
$a_i - applied\ action$

Learning with Vanilla RL Algorithms

Bipedal Walker

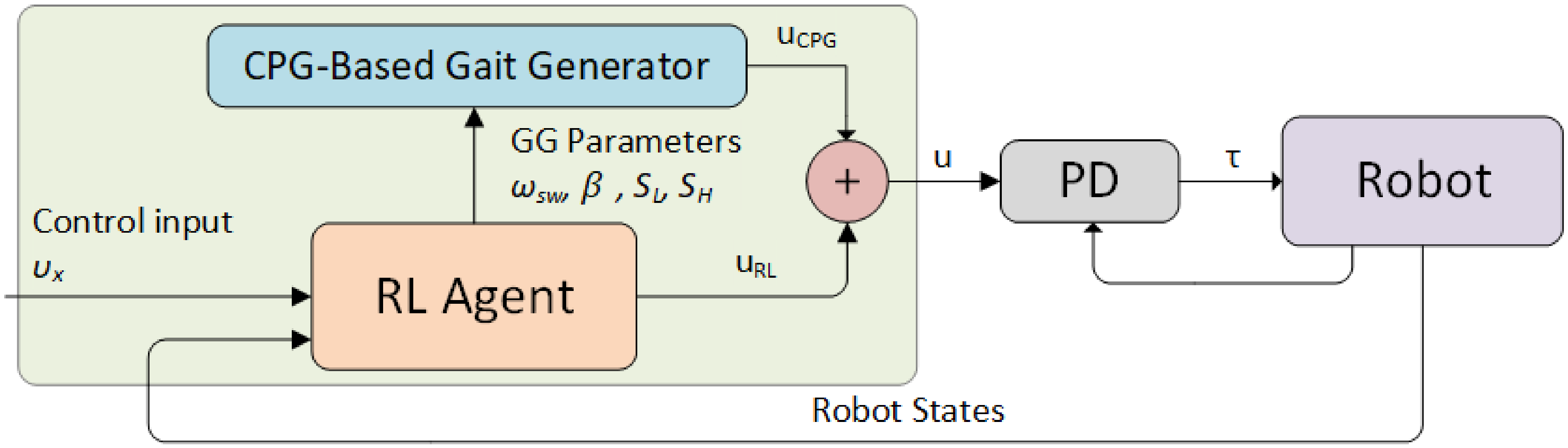


Bipedal Walker Hardcore



*all the hyperparameters were optimized by Tree-structured Parzen Estimator algorithm

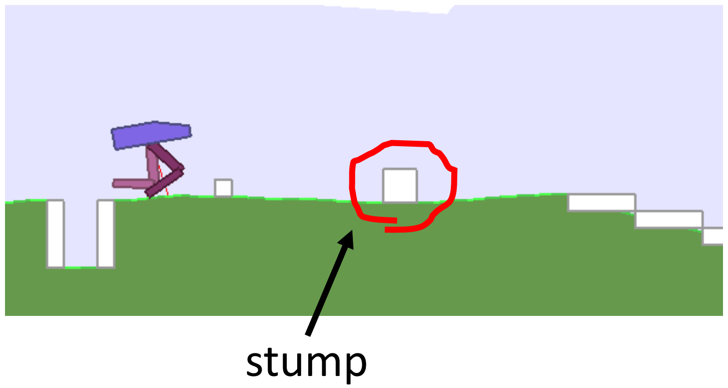
Policy Modulating Trajectory Generator (PMTG) Architecture



Supposed agents: TD3, SAC

Tricks to enhance performance:

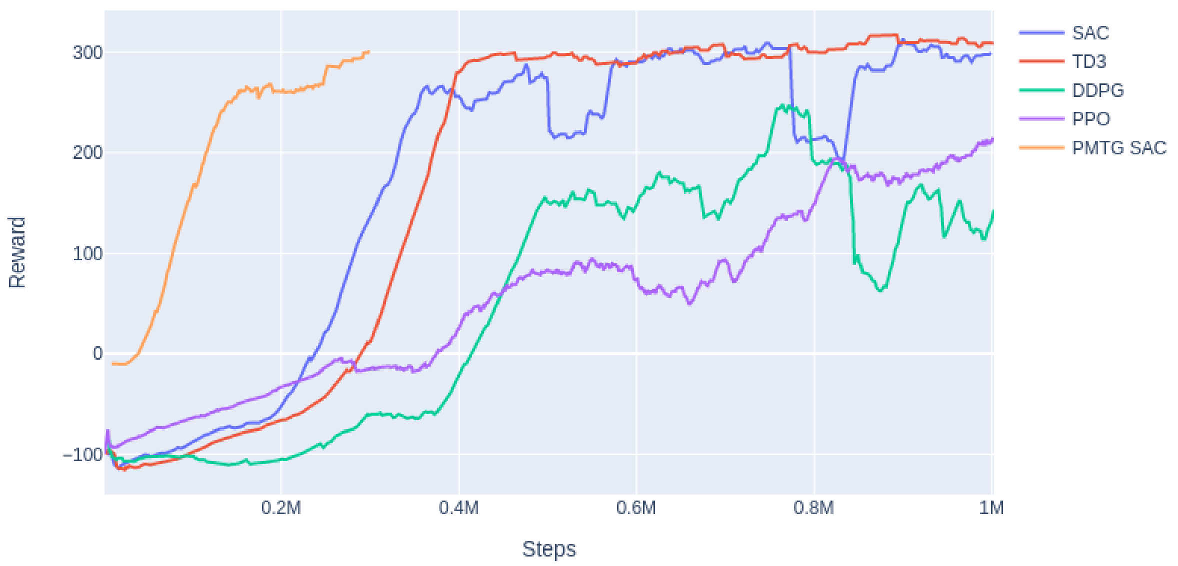
- During training the actions are repeated for three steps.
- When the agent falls, the terminal reward is clipped to zero.
- The reward is scaled by 5.
- Noises are added to the actions.
- When training, the probability of encountering a stump is increased.



Learning with PMTG

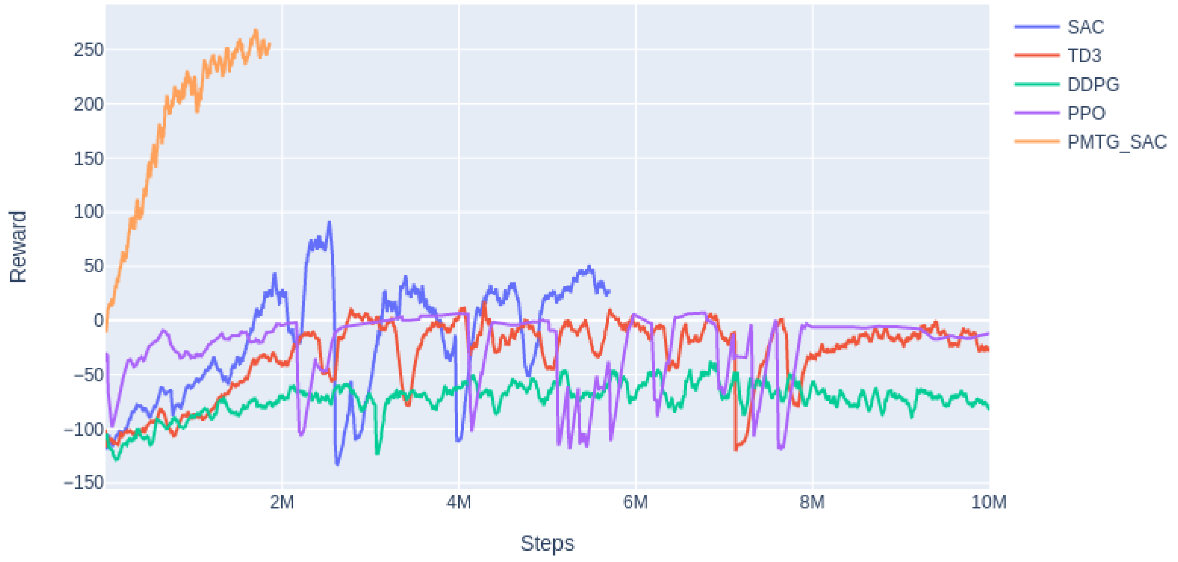
Подумать, как наглядно обозначить то, что в итог

Bipedal Walker



Learned agent 100-episode average score: 304.24
Episodes before solve: 696

Bipedal Walker Hardcore



Learned agent 100-episode average score: 302.92
Episodes before solve: 7280

OpenAi Leaderboard

Bipedal Walker

User	Episodes before solve	Write-up	Video
Nandino Cakar	474	writeup	
ZhiqingXiao	0 (use close-form preset policy)	writeup	
Nick Kaparinos	800	Write-up	gif
shnippi	925	writeup	

My learned agent 100-episode average score: 304.24
Episodes before solve: 696

Bipedal Walker Hardcore

User	Episodes before solve	100-Episode Average Score	Write-up	Video
Nick Kaparinos	15500	305.40 ± 21.35	Write-up	gif
Alister Maguire	N/A	313	Write-up	gif

My learned agent 100-episode average score: 302.92
Episodes before solve: 7280

Learned Agent Video

Bipedal Walker

Watch on Youtube

Bipedal Walker Hardcore

Watch on Youtube

Bipedal Walker Hardcore with Velocity Control

Reward function:

$$r_{fw} = \frac{4}{30}\exp\left(\frac{(v^* - v)^2}{0.2}\right) - moving\ forward\ reward$$

$$r_{hull} = -5(|\vartheta(t)| - |\vartheta(t-1)|) - hull\ deviation\ penalty$$

$$r_\tau = -0{,}056\sum_{i=1}^{12}|\tau_i| - torque\ penalty$$

$$R = r_{fw} + r_{hull} + r_\tau - total\ reward,$$

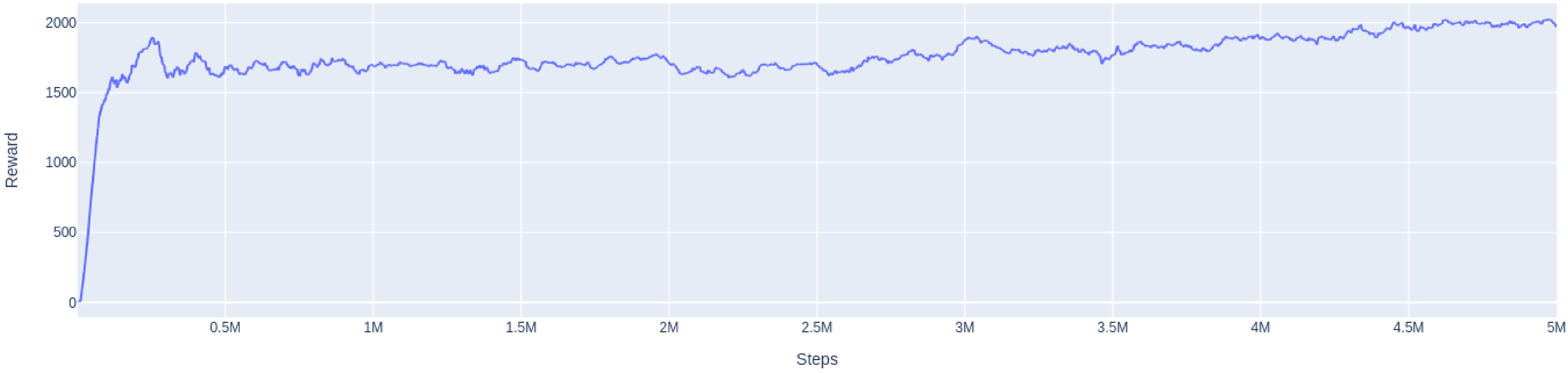
where:
p_x – *x position*,
ϑ – *hull angle*
a_i – *applied action*

Desired velocity:

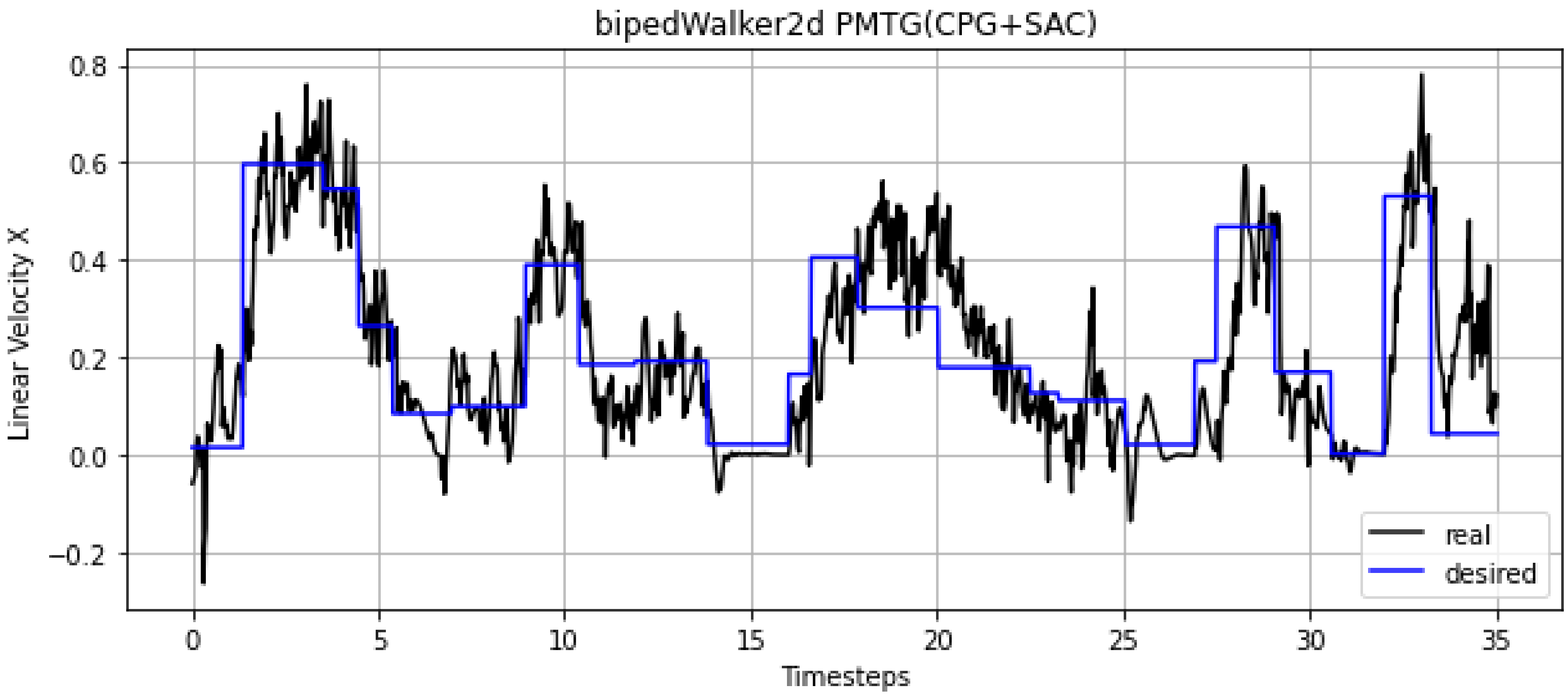
$$v^* \sim U[0, 0.6] - desired\ velocity\ distribution$$

$$t_{sw} \sim U[1, 5] - time\ before\ changing\ desired\ velocity$$

Learning Curve



Learned Agent with Velocity Control



Learned Agent with Velocity Control Video

Watch on Youtube

Thanks for Your Attention!

Control of Simplified Walking Robot Model using PMTG architecture

Vladimir Danilov