ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

RSL Robotic Systems Lab
Prof. Dr. Marco Hutter

15 Credit Project

Recovery controller for ANYmal on various terrains

Autumn Term 2019

Supervised by:
Joonho Lee
Takahiro Miki

Author:
Kyoungyeon Choi

Contents

Symbols

Symbols

${}_I e_{x,y,z}$	Inertial frame basis vector
${}_I e_{x,y,z}$	Body frame basis vector
e_g	Gravity vector
${}_B v_{IB}$	Base linear velocity in body frame
${}_B \omega_{IB}$	Base angular velocity in body frame
ψ	Joint positions
$\dot{\psi}$	Joint velocities
h	Base height
${}_B n$	Terrain normal vector in body frame
${}_B \hat{n}$	Terrain normal observation in body frame

Chapter 1

Introduction

Legged robots have promising capabilities of traversing and performing tasks in challenging environments. However, unexpected disturbances can easily lead the robot to fall in such harsh environments. Therefore, it is essential to recover from a fall in various terrains and stabilize itself. There are existing model-based control methods for recovery, but they are based on analytical models [1] and often require predefined contact points [2] that are challenging to optimize. Even on simple flat terrain environments, it is almost impossible to model all the contact sequences and the interactions with the ground.

An alternative method is a data-driven approach like reinforcement-learning. The robot automatically learns to optimize the given reward function interacting with the environments. To deal with the complexity of the task, we use model-free Deep Reinforcement Learning to obtain a robust recovery controller on various terrains. Model-free Deep Reinforcement Learning requires massive data, and it is dangerous to train the robot in the physical world. Therefore, we trained the policy in simulated environments and transferred it to real-systems. Lee et al. [3] successfully applied a deep reinforcement learning framework to train a recovery policy on flat terrain. In this work, we focus on the sloped terrain ranging up to 36 degrees.

The contributions of this project are 1) Design of a recovery controller on sloped terrain. We train the recovery policy using PPO(Proximal Policy Optimization) on sloped terrain with randomized properties(slope, friction coefficient). 2) Terrain-normal vector estimation. For stable behaviors, we find that terrain-normal observation is crucial during recovery on sloped terrain. Therefore, we trained a terrain-normal estimator using a deep neural network. 3) Design of a standing-up controller on sloped terrain. After recovery, the robot has to stand up to walk and perform various tasks. With different reward functions from the recovery controller, we train standing up policy. 4) Finite state machine. We designed a finite state machine that enables hybrid control of the two previous policies (recovery, standing up) by utilizing transition rules based on state errors. The policy is tested on the quadrupedal robot ANYmal in both simulated and physical environments.

Chapter 2

Method

This section describes the details of the control pipelines and implementations of each part.

2.1 Recovery Controller

The objective of the recovery task is to recover from arbitrary falling configurations and sit down with a stable position on sloped terrain. To come up with a robust policy, we randomize the body mass, terrain friction coefficient($1.0 \sim 2.0$), terrain slope($0 \sim 36$ degrees), and observation noise. The policy was trained with PPO+GAE [4] , [5]. Observation and reward are normalized using running mean and variance values(see Appendix A.1). The recovery task MDP consists of 69 dimension observation space and 12 dimension action space. The key cost term is the orientation cost. It encourages the robot base z-axis to align with the terrain normal direction.

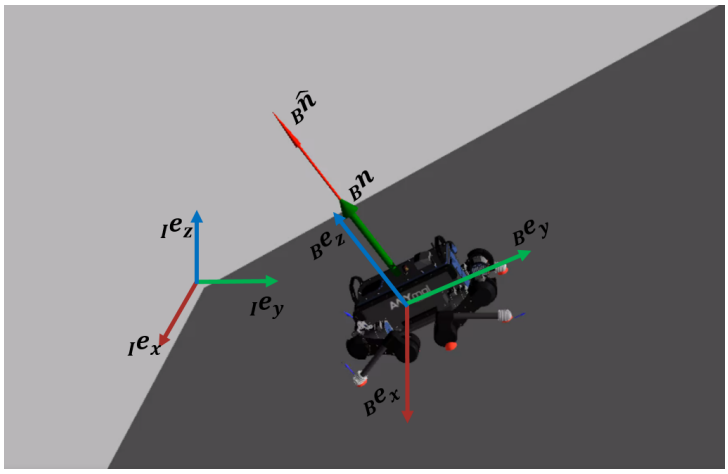


Figure 2.1: Sloped terrain environment

2.1.1 Observation and action

Observation space of the recovery task is defined in table 2.1. Base velocity and height states are excluded from observation space because they have drift issues in fall configurations.

Table 2.1: Observation space

Symbol	Observation
e_g	Gravity vector
${}_B\omega_{IB}$	Base angular velocity in body frame
ψ	Joint positions
$\dot{\psi}_{t_k}, \dot{\psi}_{t_k-0.01}, \dot{\psi}_{t_k-0.02}$	History of joint velocities
a_{k-1}	Previous action
${}_Bn$	Terrain normal vector in body frame

Observation at time step $t = t_k$ can be expressed as 69 dimension tuple.
$O_{t_k} = < e_g, {}_B\omega_{IB}, \psi, \dot{\psi}_{t_k}, \dot{\psi}_{t_k-0.01}, \dot{\psi}_{t_k-0.02}, a_{k-1}, {}_Bn >$
$e_g \in \mathbb{R}^3, {}_B\omega_{IB} \in \mathbb{R}^3, \psi \in \mathbb{R}^{12}, (\dot{\psi}_{t_k}, \dot{\psi}_{t_k-0.01}, \dot{\psi}_{t_k-0.02}) \in \mathbb{R}^{36}, a_{k-1} \in \mathbb{R}^{12}, {}_Bn \in \mathbb{R}^3$
All states except for terrain normal vector can be directly observed from sensor inputs. Action at time step t a_t is defined as $\psi_d = \psi_t + ka_t$ $(k = 0.1)$, where $\psi_d \in \mathbb{R}^{12}$ is desired joint positions and $\psi_t \in \mathbb{R}^{12}$ is current joint positions.

2.1.2 Cost function

Cost terms are summarized in table 2.2.

Table 2.2: Cost terms

Cost terms	
Orientation	$c_o = ({}_Bn \cdot {}_Be_z - 1)^2$
Joint position	$c_{jp} = K_\alpha(\psi_d - \psi_t)$
Joint velocity	$c_v = \sum_{i=1}^{12} \|min(\dot{\psi}_i, 20)\|^2, \quad if \ \|\dot{\psi}_i\| > 8(rad/s)$
Joint acceleration	$c_a = \sum_{i=1}^{12} \|\ddot{\psi}_i\|^2$
Internal contact	$c_{cin} = \mid I_{c,in} \mid$
Action difference	$c_{ad} = \sum_{i=1}^{12} \|a_{t-1} - a_t\|^2$
Body impulse	$c_{bi} = \sum_{n \in I_c \setminus I_{c,f}} \|i_{c,n}\| / (\|I_c\| - \|I_{c,f}\|)$
Torque	$c_\tau = \sum_{i=1}^{12} \|\tau\|^2$
${}_Be_z$	base z-axis in body frame
$K_\alpha(x)$	kernel function $-1/(e^{\alpha x} + e^{-\alpha x})$
I_c	index set of the contact points
$I_{c,f}$	index set of the foot contact points
$I_{c,in}$	index set of the self-collision points
$i_{c,n}$	impulse of the nth contact

The cost function is defined as $k_o c_o + k_{jp} c_{jp} + k_v c_v + k_a c_a + k_{cin} c_{cin} + k_{ad} c_{ad} + k_{bi} c_{bi} + k_\tau c_\tau$. The main cost term is orientation cost and joint position cost. Target joint states of the recovery task are joint positions of the stable sitting configuration and zero joint velocities. The remaining cost makes the motion smooth and safe, thereby enabling the policy to be deployed on real environments.

2.2 Normal Estimator

As described in the previous section, the robot has to observe the terrain normal vector ${}_B n$. However, it is not directly observable in the real-world situation and therefore must be estimated. To this end, we trained a neural network that outputs the terrain normal vector from observable inputs. Figure 2.2 shows an overview of the control pipeline.

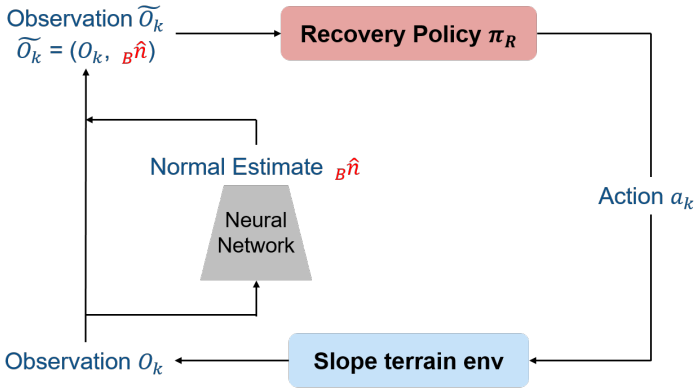


Figure 2.2: Control loop for recovery on sloped terrain environment

2.2.1 Input and output

The input of the normal estimator network is 15 dimension tuple $< e_g, \psi >$. We ruled out all the states that have drift issues and that are noisy: base linear/angular velocity, joint velocities. The network outputs the spherical representation angles of the normal vector instead of cartesian output because it has two degrees of freedom with unit length. The output representation is depicted in figure 2.3. But Zhou et al. [6] shows that angular representation is a discontinuous mapping and argues that discontinuous representation can be more difficult for neural networks to approximate, so we represent the output as cosines and sines of the angle $<$ $\cos {}_B\theta_n, \sin {}_B\theta_n, \cos {}_B\phi_n, \sin {}_B\phi_n >$to make the mapping continuous. Also, we tried to feed action and observation history into the network and compared the test errors of the output. The following result is discussed in section 3.2.

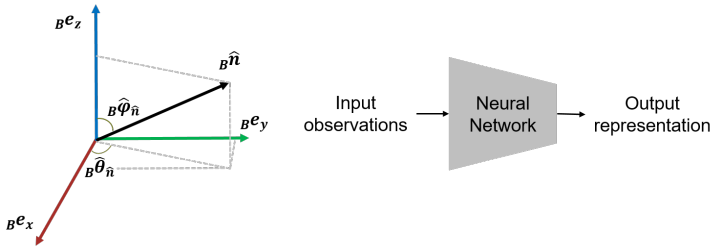


Figure 2.3: Network outputs spherical angle ${}_B\theta_n, {}_B\phi_n$ of the normal vector ${}_B n$

2.3 Standing up Controller

The purpose of the standing up controller is to stand up from near sitting configurations on sloped environments. It was trained in a similar manner to the recovery

case but with different cost terms and observation space.

2.3.1 Observation and action

Observation at time step $t = t_k$ can be expressed as 69 dimension tuple.
$O_{t_k} = < e_g, {}_B v_{IB}, {}_B\omega_{IB}, \psi, \dot{\psi}_{t_k}, \dot{\psi}_{t_k-0.01}, \dot{\psi}_{t_k-0.02}, a_{k-1} >$
In the standing up task, Base linear velocity ${}_B v_{IB} \in \mathbb{R}^3$ is added and terrain normal vector ${}_B n \in \mathbb{R}^3$ is excluded from the observation space compared to the recovery one. Action space is similar to that of the recovery task.

2.3.2 Cost function

The main cost term in the standing up policy is height, joint position, and foot slip cost. Let the cost term be $c_h = K_\alpha(h - 0.4), c_{fs} = \sum\|v_{f,i}\|$, where $v_{f,i}$ represents the velocity of the foot which is in contact with the ground. The cost function is defined as $k_h c_h + k_{jp} c_{jp} + k_{fs} c_{fs} + k_o c_o + k_v c_v + k_a c_a + k_{cin} c_{cin} + k_{ad} c_{ad} + k_{bi} c_{bi} + k_\tau c_\tau$. Although height state is not observed, it can be inferred from joint positions using forward kinematics, assuming that all the feet are in contact with the ground which is true while standing up. Foot slipping motion is penalized to maintain stable position while not slipping. Joint target position is the nominal standing configuration that maintains stable posture on sloped terrain.

2.4 Finite State Machine

In this section, we demonstrate the design of the finite state machine that recovers from a fall and stands up on sloped terrain. Recovery and standing up policy can be regarded as states of the finite state machine, and state transition rule is formed through trial and error. The differences between desired states and current states are defined as $\psi_{err} = \frac{1}{12}\sum_{i=1}^{12}|\psi_{t,i} - \psi_{d,i}|, \dot{\psi}_{err} = \frac{1}{12}\sum_{i=1}^{12}|\dot{\psi}_{t,i} - \dot{\psi}_{d,i}|, n_{err} = \cos^{-1}({}_B n \cdot {}_B e_z)$, where $\psi_{err}, \dot{\psi}_{err}$ are the errors of the joint positions and velocities, and n_{err} is the angle difference between the terrain normal vector and the base z-axis vector. Transition between policy occurs if these state errors satisfies certain conditions. The behavior transition structure is described in figure 2.5.

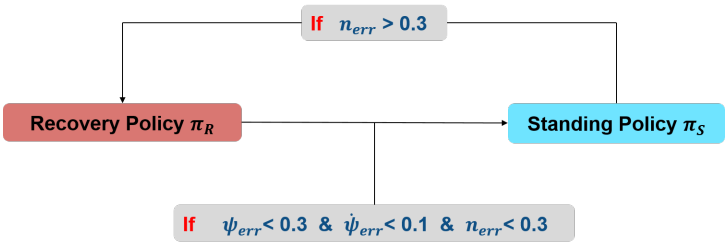


Figure 2.4: Finite state machine

Chapter 3

Results

The results of the experiment are discussed in this section.

3.1 Recovery Controller

We applied the trained policy on simulation environments. First, we tested the policy trained on sloped terrain without observing the terrain normal vector. It successfully recovers in some cases, but the behavior is unstable and rarely converges to a sitting configuration in most cases. One of the failure cases on 36 degrees slope is shown in Figure 3.1. The orientation cost is shaped to promote the base z-axis to align with the gravity vector because the terrain normal is not observable. Even if the robot is in the same joint states, the action must differ depending on the slope as the dynamics change. To train the policy conditioned on various slope terrains, ANYmal has to observe the terrain normal vector. In fact, ANYmal can observe complete states in simulated environments and be trained with ground-truth normal observation. It turns out that the resulting motion of the policy with normal observation is much more natural and desirable. However, the normal vector is not observable in a real-world environment. To resolve this problem, we estimate the terrain normal direction from observable states.

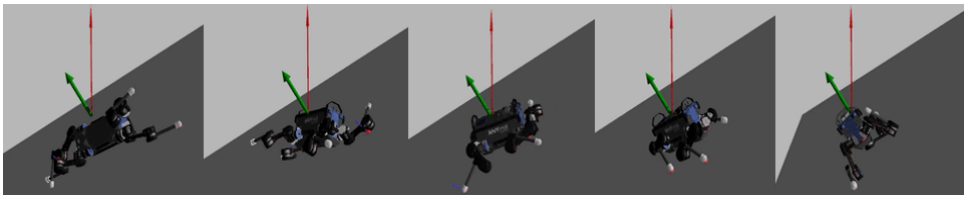


Figure 3.1: Recovery without terrain normal observation

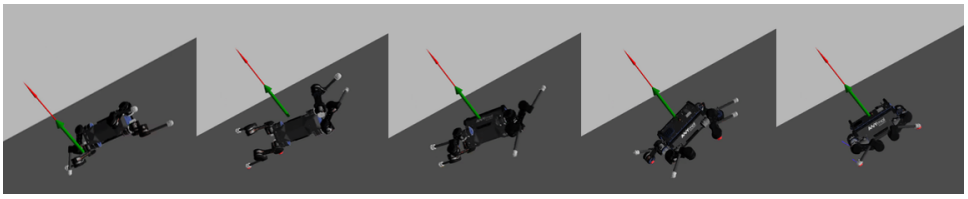


Figure 3.2: Recovery with terrain normal observation

3.2 Normal Estimator

We trained a neural network estimator in a supervised way to predict the normal. 1 million time steps of data was collected from the recovery policy in simulation.

3.2.1 Normal estimation error

The network is a three-layered feed-forward neural network with 128 units and ReLu activation. It outputs a spherical representation of the normal vector expressed in the body frame every time step. Applying cosines and sines to the output representation resulted in a smoother estimation than the raw angle output(see Appendix A.2 for plots). Also, we examine the effect of history input. Past observation and action pairs $< O_{k-4}, a_{k-4}, O_{k-3}, a_{k-3}, ..., O_k >$ are given as input and trained with the same network to compare the error. Figure 3.3 shows that the history of observations does not lead to a reduction in error.

Table 3.1: Estimation error

	mean(deg)	std(deg)
no history	6.806	8.17
history input	7.544	8.69

Although we did not analyze this issue deeply in this work, we think that this is because past observations provide little information about the current normal. Past observation inputs should undergo two stages to estimate the current normal state: estimation of normal expressed in past body frame and transformation into the current body frame. Since the estimation is inaccurate when the feet are not in contact with the ground, the prediction is noisy in the early steps. Thus normal estimation error in most cases decreases as time goes by. Even if we assume that past observations accurately estimate the normal in the corresponding time step, the network has to learn rotational relations between time steps to output the normal vector in the current body frame, which is complex to represent.

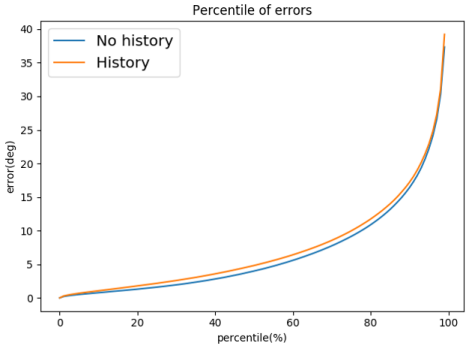


Figure 3.3: Estimation error percentile

3.2.2 Policy comparison

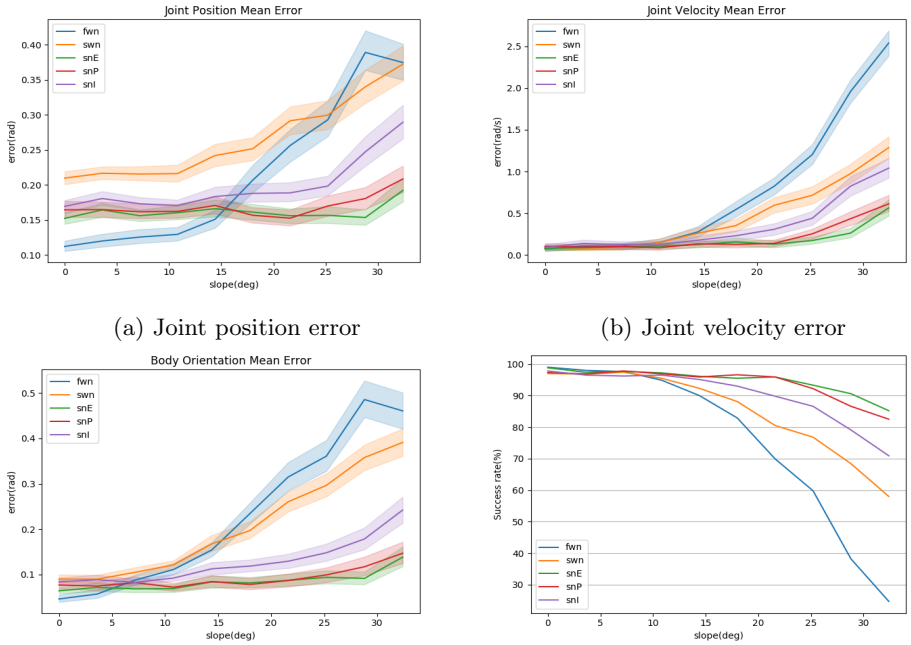
We compare the policies trained with different observations. In particular, five different policies are compared: flat without normal(fwn), slope without normal(swn) slope normal Explicit(snE), slope normal Predicted(snP), slope normal

Implicit(snI). The trained and tested environments in each case are shown in table 3.2. The column represents the normal observation type with which the robot is trained. Implicit observation of the normal means that the normal is given only in the reward function during training but not included in the observation space.

Table 3.2: Policy comparison

	No normal	Normal(truth)	Normal(estimate)
fwn	Train(flat)/Test(slope)		
swn	Train(slope)/Test(slope)		
snE		Train(slope)/Test(slope)	
snP		Train(slope)	Test(slope)
snI	Test(slope)	Train(slope)	

We trained each policy in equal conditions, and the state error was measured. Joint position, velocity and orientation error in the last time step are calculated to compare the performance: $\psi_{err} = \frac{1}{12}\sum_{i=1}^{12}|\psi_{t,i} - \psi_{d,i}|$, $\dot{\psi}_{err} = \frac{1}{12}\sum_{i=1}^{12}|\dot{\psi}_{t,i} - \dot{\psi}_{d,i}|$, $n_{err} = \cos^{-1}({}_B n \cdot {}_B e_z)$, respectively. Figure 3.4 shows the error of each state as the slope increases.



(a) Joint position error

(b) Joint velocity error

(c) Body orientation error

(d) Success rate

Figure 3.4: Error comparison of 5 different policies

We can observe from the graph that explicit normal observation leads to the smallest error in every state. The key error that determines the failure of the task is the velocity error because only after all the joint positions and the orientation are close to the desired state the velocity goes to the desired zero steady states. We define the success of the task as $\psi_{err} < 0.6(rad), \dot{\psi}_{err} < 0.1(rad/s), n_{err} < 0.6(rad)$. With this definition, the success rate is shown in Figure 3.4.d. It implies that it is imperative to observe the normal explicitly to successfully recover and stabilize on a sloped terrain environment. The recovery behavior with predicted normal

observation is depicted in Figure 3.5(see more in Appendix A.2). The red arrow indicates the estimated normal direction, whereas the green one indicates the true normal direction.

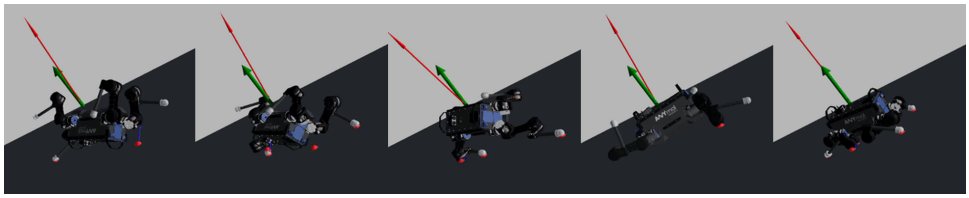


Figure 3.5: Policy with normal estimation

3.3 Standing up Controller

Standing up policy was trained and tested on sloped terrains. In inclined environments, high base height makes it difficult to balance itself, and thus we tuned the desired joint position and base height until it does not fall over. Height, joint position, slip cost terms are essential for this task, and they are composed hierarchically. To be specific, the joint reward is given only when the height is above a certain value to avoid sub-optimal behavior. Moreover, sampling target position states as an initial state boosted exploration because it is unlikely to explore such states during the early training stage. The policy was tested on slopes with randomized initial states. The standing movement is shown in Figure 3.6.

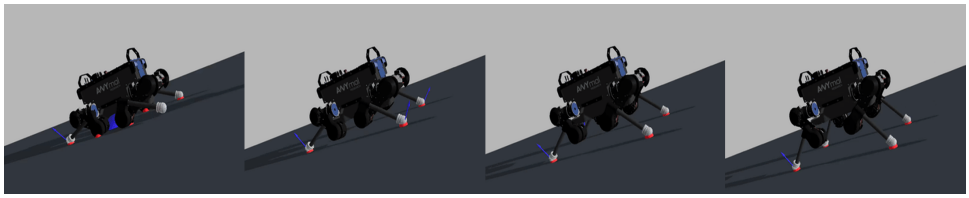


Figure 3.6: Standing up

3.4 Finite State Machine

We developed a finite state machine that enables the transition between multiple tasks. Transition rule based on state error is considered, and the threshold value was determined by trial and error: $\psi_{err} < 0.3$, $\dot{\psi}_{err} < 0.1$, $n_{err} < 0.3$. If the error is below such values, the recovery policy is switched to the standing up policy. Although it is possible to train a task selector by reinforcement learning, this approach is more simple and easy to implement. Figure 3.6 describes the sequence of motions(see more in Appendix A.3). We also tested the policy on stairs to examine its robustness to unseen environments. Appendix A.4 shows that the policy is directly transferable to stair environments.

Figure 3.7: Finite state machine

3.5 Deployment

The policy was deployed on ANYmal in real-world environment. Hwangbo et al. [7] trained a actuator network to model the actuator dynamics and enabled sim-to-real transfer of the policy. We utilized this method by attaching a trained actuator network to the policy and directly deployed the trained policy on the real robot. Figure 3.8 shows the recovery and standing maneuver on flat and sloped terrain.

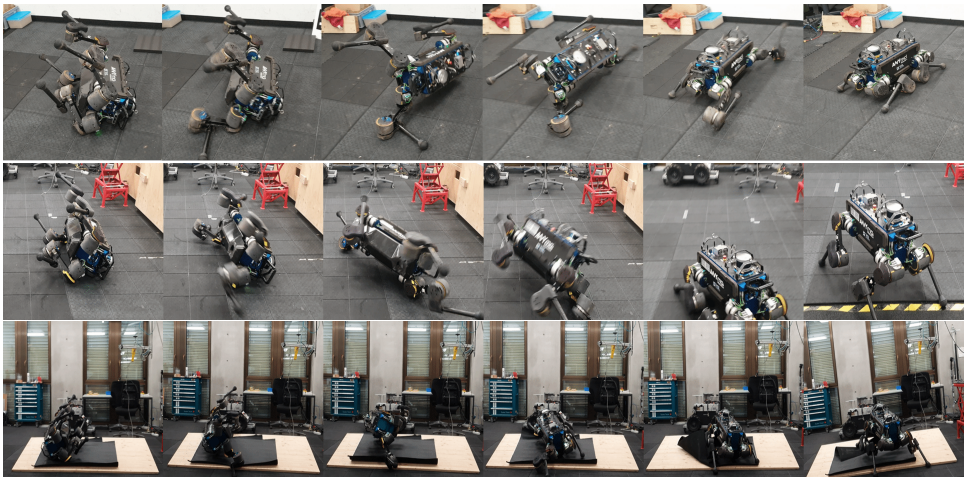


Figure 3.8: Deployment on ANYmal

Bibliography

[1] I. Mordatch, E. Todorov, and Z. Popovic, "Discovery of complex behaviors through contact-invariant optimization," *ACM Trans. Graph.*, vol. 31, pp. 43:1–43:8, 2012.

[2] D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 3, pp. 2261–2268, 2018.

[3] J. Lee, J. Hwangbo, and M. Hutter, "Robust recovery controller for a quadrupedal robot using deep reinforcement learning," *ArXiv*, vol. abs/1901.07517, 2019.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017.

[5] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *CoRR*, vol. abs/1506.02438, 2015.

[6] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *CVPR*, 2018.

[7] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, 2019.

Appendix A

Figures & Setup

All the policies presented in the figure is tested on 36 degrees sloped terrain with the friction coefficient of 1.2.

A.1 Experimental setup

Table A.1: Hyperparameters for PPO

Hyperparameter	Value
Timesteps per iteration	16000
Discount factor (γ)	0.993
GAE discount (λ)	0.95
Value network LR	0.001
Value network num. epochs	10
Policy network hidden	[128, 128]
Value network hidden layers	[128, 128]
Number of minibatches	1
Policy LR	0.001
Policy epochs	10
Entropy coefficient	0.005
Clipping coefficient	0.2

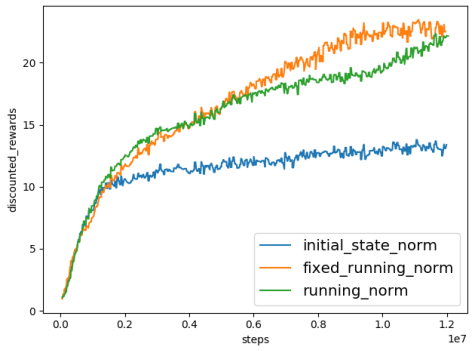


Figure A.1: Reward with observation normalization. Policy with running mean normalization outperforms the policy normalized with nominal values

Table A.2: Cost coefficients($\Delta t = 0.0025$)

Cost coefficient	Recovery	Standing up
Orientation (k_o)	$7.0\Delta t$	$6.0\Delta t$
Joint position (k_{jp})	$6.0\Delta t$	$3.0\Delta t$
Joint velocity (k_v)	$0.025\Delta t$	$0.05\Delta t$
Joint acceleration (k_a)	5e-$2\Delta t$	5e-$5\Delta t$
Internal contact (k_{cin})	$7.0\Delta t$	$3.0\Delta t$
Action difference (k_{ad})	$1.5\Delta t$	$1.5\Delta t$
Body impulse (k_{bi})	$4.0\Delta t$	$1.0\Delta t$
Torque (k_τ)	2e-$5\Delta t$	2e-$5\Delta t$
Foot slip (k_{fs})	-	$15.0\Delta t$
Height (k_h)	-	$30.0\Delta t$

Joint position cost $c_{jp} = K_\alpha(\psi_d - \psi_t), \alpha = 7.0$
Height cost $c_h = K_\alpha(h - 0.4), \alpha = 10.0$

A.2 Normal estimation figures

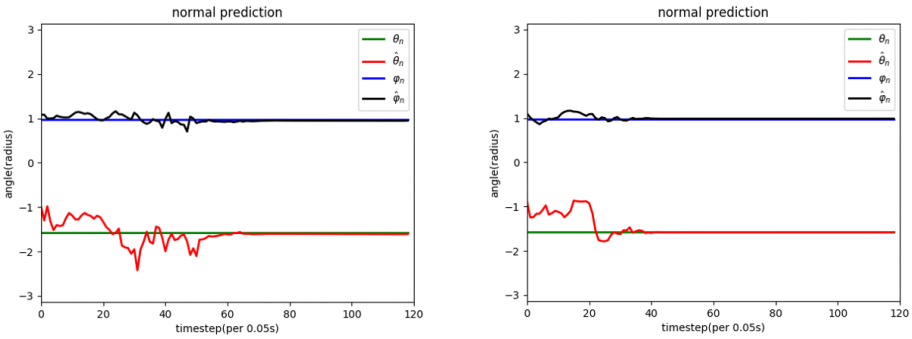


Figure A.2: Normal estimation error of different representation (left: raw angle output, right: cosine and sine output)

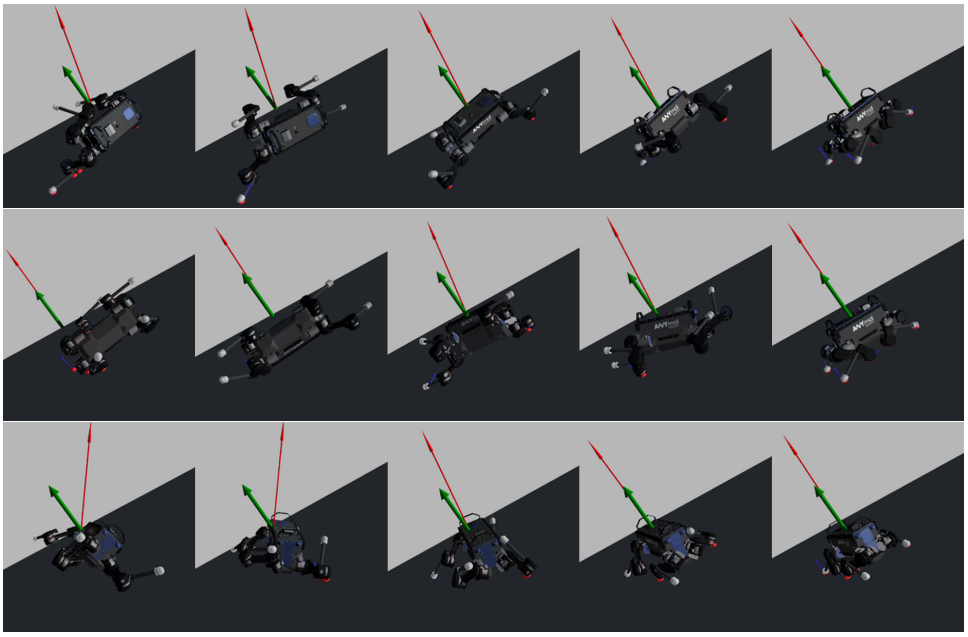


Figure A.3: Policy with normal estimation

A.3 Finite state machine figures

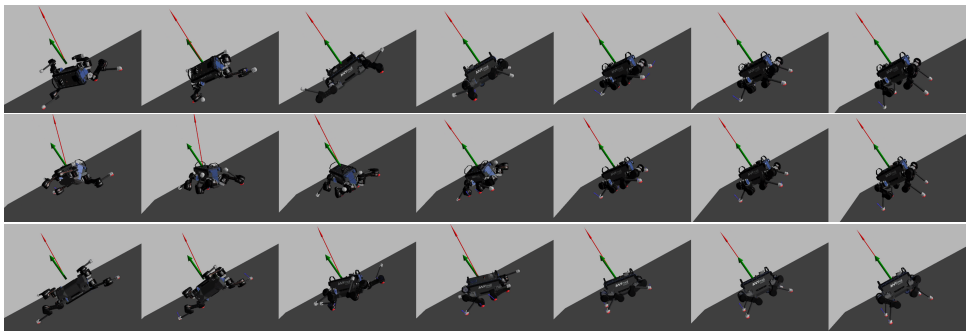


Figure A.4: Finite state machine result

A.4 Policy on stairs

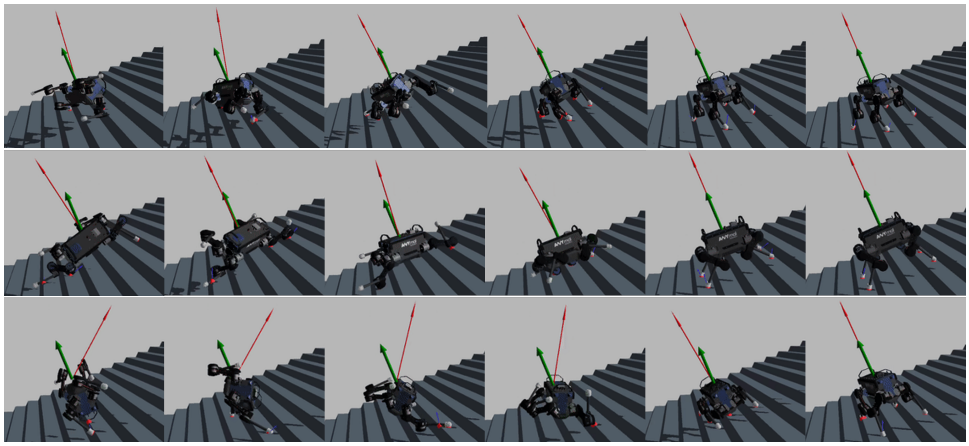


Figure A.5: Policy with normal estimation