Model-Free Cooperative Step Climbing for Under-actuated Legged Millirobots

Guangzhao Yang

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-43
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-43.html

May 15, 2019

Acknowledgement

The author would like to thank the members of the Biomimetic Millisystems Laboratory for their support, especially Carlos Casarez for his expertise in VelociRoACH platform, Anusha Nagabandi for helping me understanding different types of algorithms in reinforcement learning, and Professor Ronald S. Fearing for all the insightful discussions and guidance.

Model-Free Cooperative Step Climbing for Under-actuated Legged Millirobots

by Guangzhao Yang

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee

Professor Ronald S. Fearing
Research Advisor

(Date)

★ ★ ★ ★ ★ ★ ★

Professor Grace X. Gu
Second Reader

(Date)

Abstract

Legged millirobots have a wide range of real world applications due to their small size, high mobility, and low manufacturing costs. They are able to go through narrow passages and tunnels which are inaccessible to the larger robots, traverse in complex environments, and have higher potential to climb over obstacles of their sizes compared to wheeled robots. This technical report presents the simulation results of two VelociRoACH hexapedal legged robots collaboratively climbing over a step which cannot be accomplished by one such robot. Two magnets and one tether are added to form and release the connections between two robots in order to help with the robot alignment and provide a tether assist force. When the robots are in each other's vicinity, magnetic connections will be formed. After the front robot successfully climbs over the obstacle, all connections will be released so that the front robot can continue moving forward. Multiple reward functions are provided during different stages of the training process in order to learn the primitives for step climbing task such as going forward, keeping aligned with each other, pushing and climbing. The experiments demonstrate that after training with different reward functions in the desirable order, the robots are able to climb up the steps with small height variations using the same learned policy. Since a robot can overcome an obstacle taller than itself with the help of other robots, given enough number of such legged millirobots, potentially they are able to climb over any random obstacle by first climbing on top of others. The experiment videos can be found at https://drive.google.com/drive/folders/1XRE5br1szYWwkqelqulygDXDQNhJAILl?usp=sharing. The code can be found at https://github.com/philip324/roach_step_climbing and https://github.com/philip324/rllab-philip.

Contents

1 Introduction

Legged millirobots have a great potential in assisting human rescue teams in disaster scenarios because of the size and mobility. Although millirobots cannot move heavy objects like the large-scale robots, they are able to navigate through narrow spaces in collapsed buildings, helping the rescue team locate survivors and provide optimal rescue strategies. Comparing to wheeled robots, the legged robots of the same size are more capable of traversing through complex environment, especially the environments with a lot of obstacles. Another advantage of the millirobots is that they can be produced in large quantities due to the low manufacturing cost. Deploying a team of legged millirobots potentially allows them to cooperate with each other in order to overcome larger obstacles. All of these advantages makes them some of the most mission-capable small-scale robots available

While having all these advantages, legged millirobots are hard to control because they are under-actuated. The robots in the simulation are modeled based on the VelociRoACH, a Robotic Autonomous Crawling Hexapod (RoACH) experimental platform designed for high velocity running through its elegant minimal hardware design [1]. There are only two motors on board to control all 6 legs, so we cannot control each individual leg. Moreover, modeling the dynamics of the under-actuated legged millirobots is exceedingly difficult due to complicated ground contact physics when moving dynamically on complex terrains. Therefore, climbing over a step that is higher than the robot itself poses a great challenge for an individual robot.

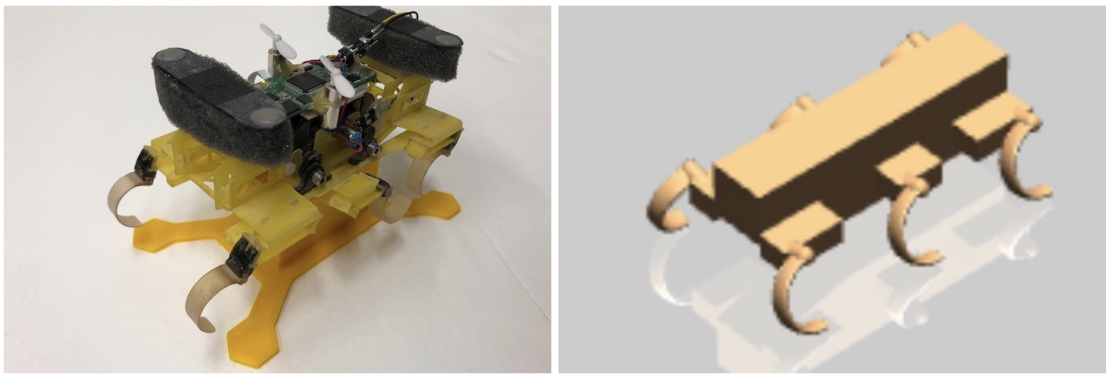

Figure 1.1: Left: VelociRoACH in real world. Right: VelociRoACH in MuJoCo simulation

The contribution of this report is to present an approach for controlling under-actuated legged millirobots in the MuJoCo simulation [2] such that they can overcome a step obstacle cooperatively. In order to make the controller more robust, we establish 3 simple connections between the millirobots. Two magnetic connections are crucial for aligning the robots, while the tether connection provides an extra force to help the front robot pitch up against the step. Moreover, the report also demonstrates that by using different reward functions in a desired order during the learning process, we are able to learn the necessary primitives intrinsically for the cooperative step climbing task. These reward functions describe different general goals at different stages, and therefore enable the system to learn different policies.

The structure of this technical report is as follows: Chapter 2 reviews some recent literature on robots that overcome obstacles cooperatively with different climbing mechanisms. In chapter 3, we discuss the simulation model/environment in which the experiments are conducted, and the model-free learning algorithm we use to learn a combined policy for both robots in order to climb up the step cooperatively. Moreover, we describe how we modify the reward function during the learning process in order to learn the primitives for the step climbing task. Chapter 4 presents a primitive analysis on each of the three cooperative climbing primitives when performed independently: aligning, pitching up, and climbing. In chapter 5, we discuss some limitations and drawbacks of this method, and things that we can improve in the future.

2 Related Work

Controlling Legged Millirobots: The VelociRoACH [1] is an under-actuated legged millirobot that is controlled by two motors and can run at a high speed. Many of the prior closed-loop steering control methods simplify the system dynamics and estimate each leg of the robot as a spring loaded inverted pendulum (SLIP) model [3]. In order to deal with the complex dynamics of the system, prior work have achieved success by using a model-based reinforcement learning method with Model Predictive Control (MPC) [4], which is expressive and sample efficient. However, model-based methods have more assumptions and approximations comparing to model-free methods, and does not generalize well for other tasks. Since we want the millirobots to be able to climb over different obstacles, we choose a policy gradient method [5] and conduct our experiments in the simulation with low sampling cost.

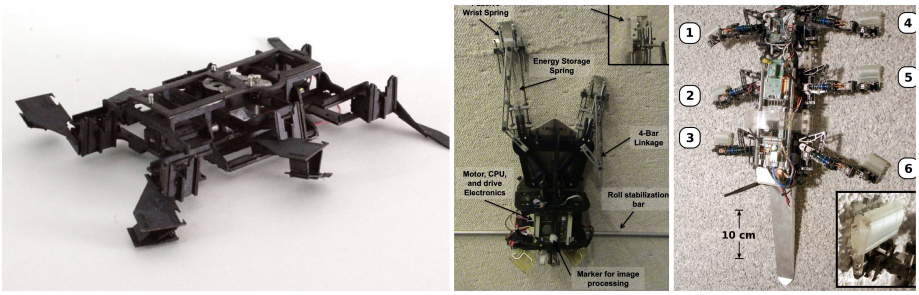

Figure 2.1: Robots use claws or spines to climb the wall. Left: CLASH [6]. Middle: RiSE [7]. Right: DynoClimber [8].

Climbing Obstacles: Climbing obstacles is hard because there is a huge variation in obstacle types. Many robots have specialized climbing mechanisms such as claws and spines [6, 7, 8], which works well on rough surfaces but not on smooth surfaces. Other robots use gecko-inspired adhesives [9, 10], which works better on smooth surfaces than on rough surfaces. These robots are able to climb vertical walls successfully, but the climbing mechanisms do not generalize. Vertigo uses two 360-degree tiltable propellers that produce thrust angled both upwards and against the wall surface in order to climb the wall [11]. The propellers allow the robot to climb the wall regardless of the types of wall surface. However, propellers consume a lot of energy and take up too much space on the robot, and therefore it is not implementable on millirobots.

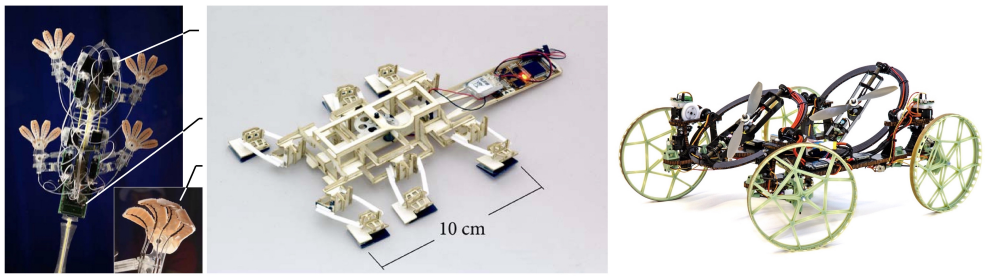

Figure 2.2: Robots use gecko-inspired adhesives or propellers to climb the wall. Left: Stickybot [9]. Middle: CLASH [10]. Right: VertiGo [11].

Robot Cooperation: Prior work on robot cooperation includes tether cooperation between robots. The Dante II robot is able to continuously adjust the tether tension in order to maintain its stability while traversing steep terrain [12]. Miki et al. proposed a novel cooperative system between an Unmanned Aerial Vehicle (UAV) and an Unmanned Ground Vehicle (UGV) [13]. These systems assume that one of the robots is already at the top of an obstacle. However, this assumption may not always be valid under certain circumstances. For instance, when a team of robots are searching in a collapsed building. The space is too small for a UAV to fly without collision, and the robots have to be able to climb over an obstacle without an anchor at the top of the obstacle. Casarez et al. used two legged millirobots with a magnetically detachable tether connection to successfully get both robots to the top of an obstacle without a top anchor [14]. The cooperative climbing task is separated into 5 specific primitives, which are hand-specified. However, the experiments are conducted in a low-friction channel in order to restrict the yaw of the robots. Instead, our work uses model-free method to learn a cooperative policy such that it can control the yaw to stabilize the robots. Moreover, the learned policy allows the robots to climb over different types of obstacles (slopes and steps) with slightly different variations (angles and heights).

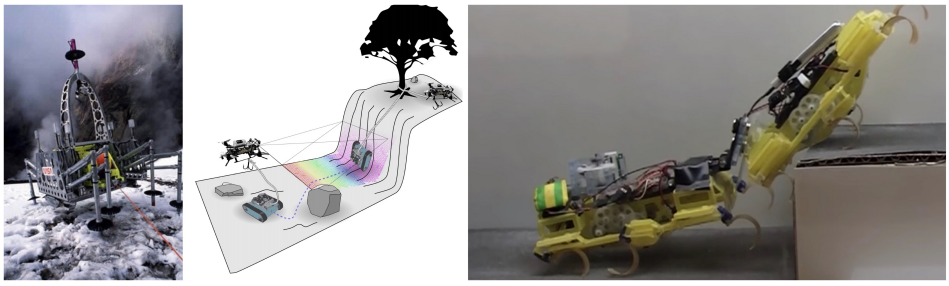

Figure 2.3: Robots use tether to climb the wall. Left: Dante II [12]. Middle: UAV assists UGV to climb a cliff [13]. Right: VelociRoACH [14].

3 Model-Free Cooperative Step Climbing

In this work, we propose a method that allows small, dynamic legged millirobots to climb over an obstacle higher than themselves cooperatively in MuJoCo simulation. In this section, we describe the simulation environment in which the experiments are conducted, how we use Trust Region Policy Optimization (TRPO) to learn a combined policy for both robots, and the reward function modification during the learning process in order to learn different primitives for the step climbing task.

3.1 Experiment Setup

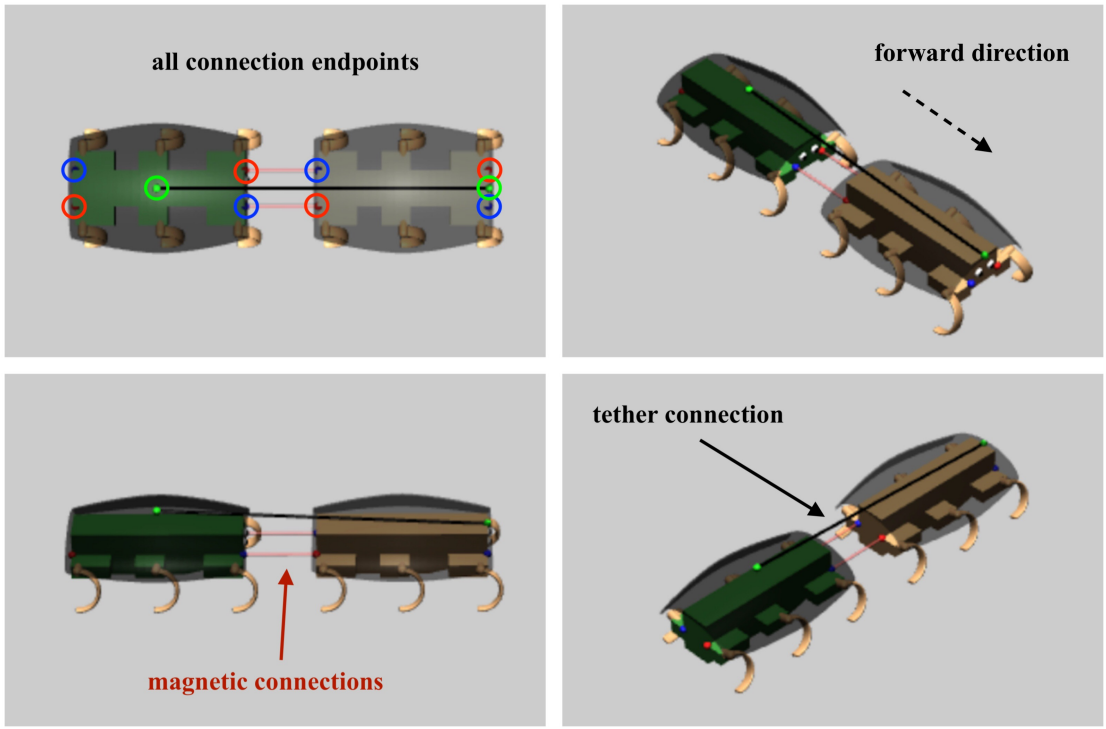

Figure 3.1: The yellow robot is the front robot, and the green robot is the back robot. The tether is indicated by a black line, and two magnetic connections are indicated by two red lines. Each magnetic connection is between a red dot and a blue dot, which represent the two polarities of a magnet.

In order to make the transfer from simulation to real world easier in the future, we model the robot in the simulation based on the VelociRoACH millirobot with C-shaped legs. The dimension of the robot is 10cm by 6cm by 4.5cm (length by width by height), and the mass of the robot is 51.3g. Moreover, we adjust the stiffness in leg joints such that the simulation model will have a relative leg stiffness ($k_{\text{rel,ind}}$) of 9.74, corresponding with the animal data [15]. Finally, a shell is added to each robot in order to simplify the contact between two robots. The shell is almost weightless, and therefore does not change the robot's center of mass.

In the simulation, we use a step with a coefficient of friction of 1 and height of 5cm as the final obstacle to overcome. The obstacle is at half a meter away from the front robot. Each robot has 6 legs and two independent motors, each of which controls three legs on the same side of the robot. Moreover, there are three extra connections between two robots to make the climbing easier. The robots are connected by a tether which can be tightened and help the front robot to pitch up against the step. Two magnetic connections will be formed when the robots are close to each other in order to help the robots stay aligned and stabilize the yaw of the front robot when climbing the obstacle. Figure 3.1 shows the details of all three connections.

3.2 Learning Policies

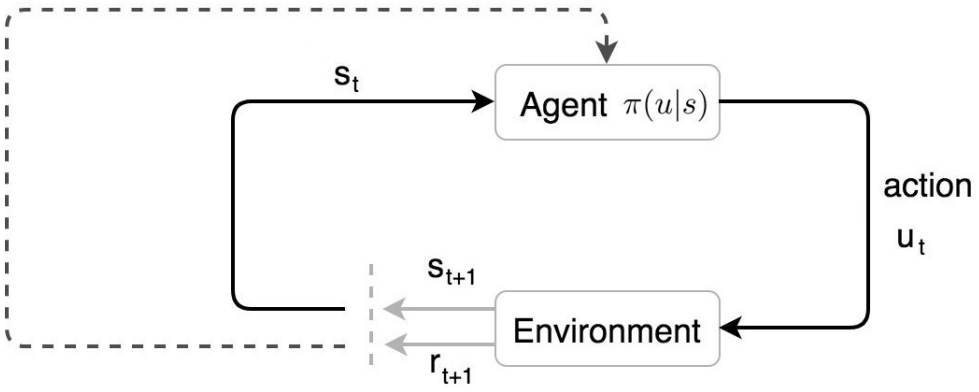

Figure 3.2: Diagram illustrating policy gradient. Instead of using line search, TRPO uses trust region as the optimization method, and is able to guarantee a policy improvement as long as we optimize the local approximation within a trusted region. s_t, u_t and r_t represent the state, the action and the reward at time t. $\pi(u|s)$ is the distribution of actions given the current state.

TRPO is implemented to directly optimize the step climbing policy with guaranteed monotonic improvement. Since this is a centralized multi-robot system, the state of the system consists of the state of the obstacle and both robots. We define the state of each robot to be $s = [x, y, z, v_x, v_y, v_z, \phi_r, \phi_p, \phi_y, \alpha_L, \alpha_R]^T$

where (x, y, z) represent the center of mass, (v_x, v_y, v_z) represent the linear velocity, (ϕ_r, ϕ_p, ϕ_y) represent the Euler angles which describes the pose, and (α_L, α_R) represent the motor positions of the robot.

We define the action of the system to be the leg velocity of the robots and the forces of the connections: $u = [u_{\text{front\_left}}, u_{\text{front\_right}}, u_{\text{back\_left}}, u_{\text{back\_right}}, u_{\text{tether}}, u_{\text{mag1}}, u_{\text{mag2}}]^T$. The action u_t is selected based on the current optimal policy $\pi_\theta(u_t|s_t)$, which is a stochastic distribution of action u_t given state s_t during training time, and a deterministic distribution during test time. θ is the parameter which represents the weights in the neural network we use, and θ will be updated in order to maximize the reward function r_t.

The size of the neural network's fully-connected hidden layers is 64 by 64, and we choose such size because it works well on other legged robots such as Ant_v2. We set batch size to be 5000, step size to be 0.05 seconds, and discount to be 0.995. We want each episode to take 10 seconds in simulation time. Since the time step is 5 milliseconds, the path length is 2000. However, we do not want to update the action every 5 millisecond because the controller cannot run that fast in the real world. Instead, we update the action every 0.1 second, which means that we need to repeat the same action 20 times before updating to a new action. Therefore, the actual path length we use is 100. Finally, we initialize our policy as a Gaussian Multilayer Perceptron (MLP) policy. For each of the following task in the next section, we initialize the learning with the trained policy from the previous task, and keep training the policy until it plateaus.

3.3 Reward Function Modification

3.3.1 Flat Ground

Instead of directly training the robots to climb over an obstacle, we first train them to walk forward in an environment with no obstacles. Since we want both robots to first learn how to walk, we do not need any connections between them. Therefore, the last three actions are set to zeros in this stage. Without loss of generality, we assume that the obstacle is located at the positive side of x-axis, and we define the positive direction of x as the forward direction. The first reward function we use is

$$R_1 = r_{\text{srvl}} + (v_{x,1} + v_{x,2}) - 0.5 \cdot [\text{abs}(v_{y,1}) + \text{abs}(v_{y,2})] = r_1$$

where r_{srvl} is the survival reward, $v_{x,1}$, $v_{y,1}$, $v_{x,2}$ and $v_{y,2}$ are the x and y velocities of the front and back robots respectively. r_{srvl} is set to 0.05, and it is obtainable as long as the simulation is still running. The goal here is to encourage both robots to walk in the positive direction of x, and try to make them walk as straight as possible by minimizing the y direction velocity. Since velocities are the major part of this reward function, the actual initial locations of the robots do not matter. Therefore, it doesn't matter where the robots are initialized. Although both robots are able to walk straight forward at the end of the training,

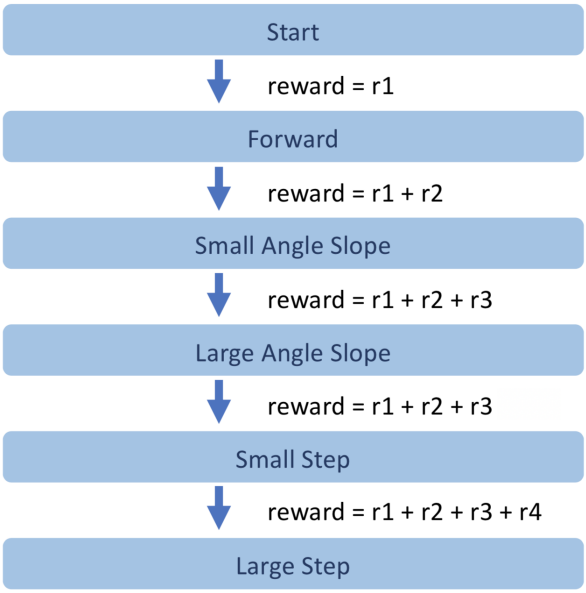

Figure 3.3: Flow chart illustrating the learning process

their trajectories are independent of each other. In order for the robots to get aligned, we update the reward function as follows:

$$R_2 = r_1 + r_{\text{align}} - 0.5 \cdot \text{abs}(y_2 - y_1) = r_1 + r_2$$

where r_{align} represents the alignment reward, and y_1 and y_2 represent the front and back robots' y positions. r_{align} is a binary reward which can be acquired if the back robot is behind the front robot and the distance between two robots is less than 3cm. When the these two conditions are met, two magnetic connections will be formed in order to pull the robots together and keep them aligned.

During training, we initialize the front robot at the origin, and the back robot at (-0.3, 0). Both robots have a yaw of zero degree. We also add small random noises to the initial positions and poses of both robots to avoid over-fitting. Since the reward function is designed for the robots to learn how to align themselves before they start to climb the obstacle, no obstacle is needed at this stage.

3.3.2 Slope

We train the policy with the reward function R_2 on both flat ground and slope with small angle such that the robots are able to walk without slipping. When the robots start to slide and are not able to climb up the slope, we add a yaw penalty to the reward function:

$$R_3 = r_1 + r_2 - p \cdot \text{abs}(\phi_{y,1}) = r_1 + r_2 + r_3$$

where $\phi_{y,1}$ is the yaw of the front robot, and p is the penalty weight. We want the front robot to learn how to stabilize its yaw and not fall on its side when trying to climb up the slope. The penalty weight p is crucial because we don't want to penalize the yaw of the front robot too much so that the front robot thinks the best strategy is to not move at all.

3.3.3 Step

After the front robot is able to balance on a relatively steep slope (30 degrees), we use the same reward function to train the robots to climb a small step. The step size should be small enough (less than 2cm) such that the front robot is able to climb over easily. Then, we first increase the height of the step to 3cm and change the reward function to

$$R_4 = r_1 + r_2 + r_3 + z_{\text{front},1} + z_{\text{rear},1} = r_1 + r_2 + r_3 + r_4$$

where $z_{\text{front},1}$ and $z_{\text{rear},1}$ are the z components of the front end and rear end of the front robot. While training, we gradually increase the height of the step up to 5cm. We soon realize that it is hard for the front robot to keep its yaw stable as the step height increases. Thus, we add two low-friction glass walls on each side of the robots to form a channel to help stabilizing the yaw.

In order to climb over the step, the front robot needs to learn to pitch itself up against the step. A tether connection will be formed when the front robot touches the step in order to help it pitch up against the step. Next, the back robot needs to push the front robot so that it can reach the top of the step. Finally, we release all connections once the hind legs of the front robot are close enough to the edge of the step because we don't want these connections keep pulling and stop the front robot from climbing up the step.

After the front the robot successfully overcomes the step with height of 5cm in a narrow low-friction channel, we keep training the policy while moving the glass walls further apart from each other. Eventually, the width of the channel is 20cm, which is two times the length of the robot.

4 Results

4.1 Cooperative Step Climbing Primitives

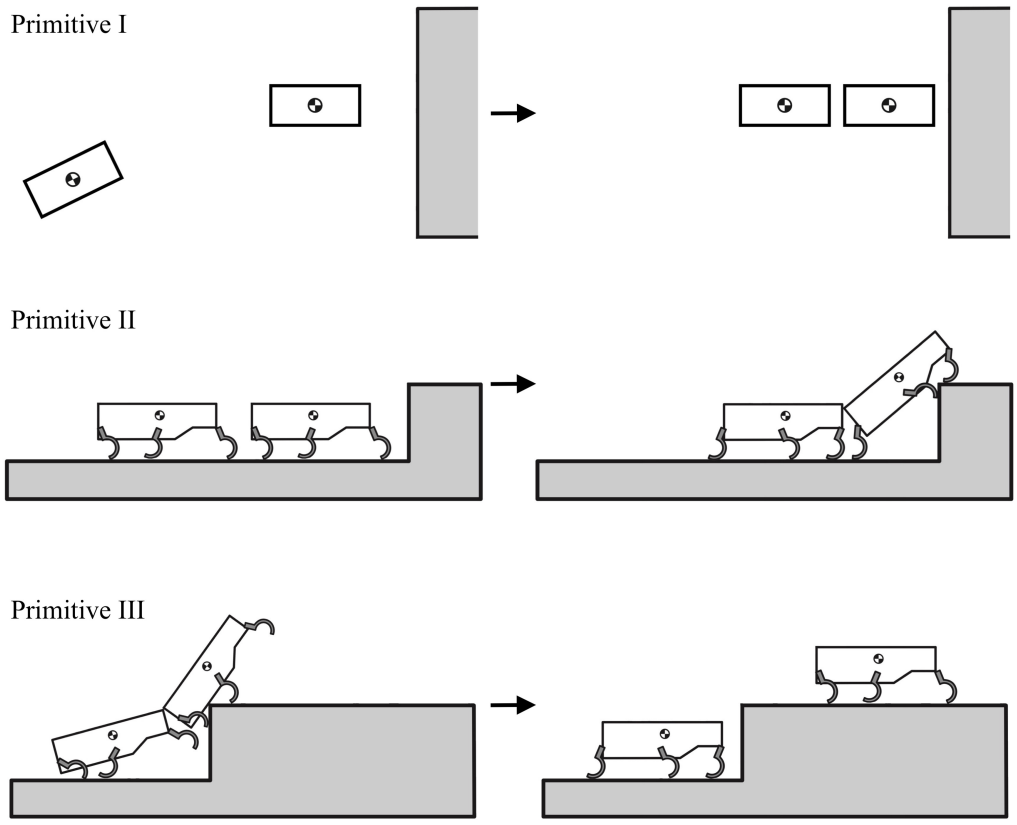

Figure 4.1: Diagram illustrating primitives I, II, and III of cooperative step climbing.

As shown in Figure 4.1, we separate the cooperative step climbing task into the following three primitives: (I) two robots align themselves in front of the obstacle; (II) the front robot pitches up against the step; (III) the back robot pushes the front robot up the step. We conduct experiments for each primitive independently in order to find the success rates of different initial conditions and the state transition probabilities.

4.1.1 Primitive I: Aligning

In primitive I, we want to see how well the policy is in terms of aligning the two robots. At the beginning of primitive I, the front robot is initialized at the origin and its yaw $\phi_{y,1}$ is set to zero. The back robot is initialized at some location (x_2, y_2) behind the front robot and its yaw $\phi_{y,2}$ is set to $\left[\text{mod}(\tan^{-1}(\frac{y}{x}), 2\pi) - \pi\right]$ such that it points towards the front robot. Moreover, we want to add small random noises to the position and pose of the back robot. The initial position of the back robot is $(x_2 + r\cos(\beta), y_2 + r\sin(\beta))$, where r is uniformly distributed between 0 and 1cm, and β is uniformly distributed between 0 and 2π. The initial pose of the back robot is $\left[\text{mod}(\tan^{-1}(\frac{y}{x}), 2\pi) - \pi + \gamma\right]$, where γ is uniformly distributed between $\pm 15 \cdot \frac{\pi}{180}$ rad. Figure 4.3 shows the probabilities of success when the back robot is initiated at different locations. Each point represents the location the back robot is initiated, and the lines represents the bounds of initial yaw of the back robot. We run at least 100 trials for each initial position.

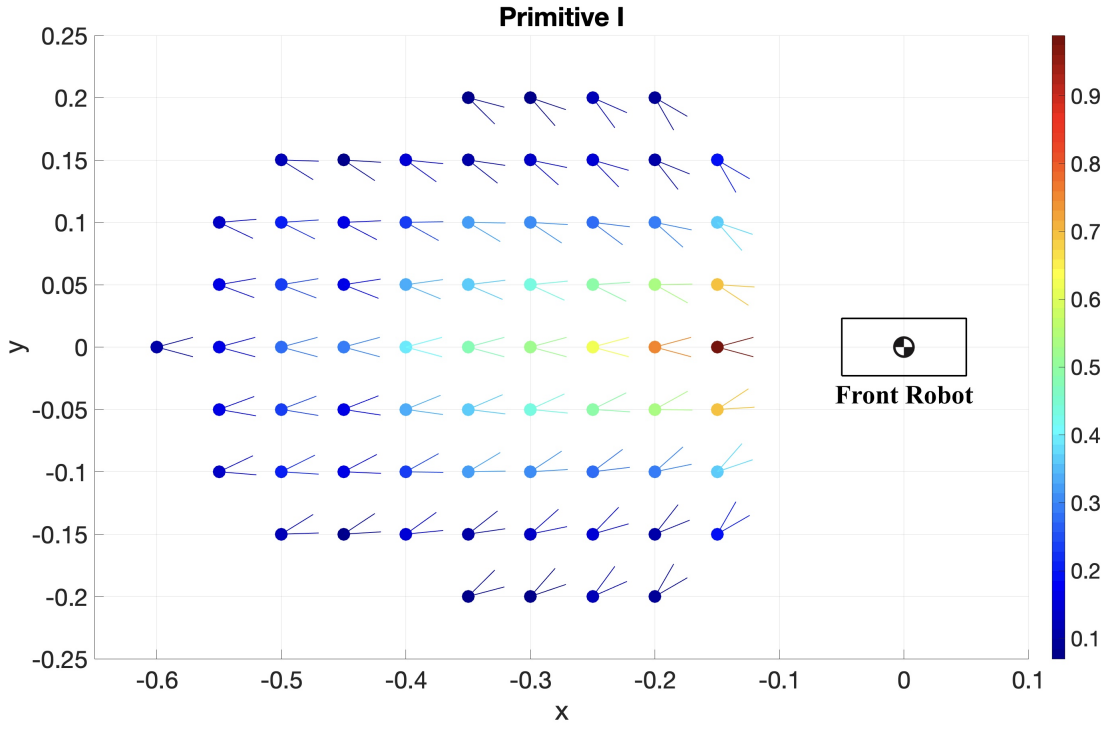

Figure 4.2: Primitive I: probability of success when the back robot is initiated at different locations.

Since the magnetic connections will pull two robots together and keep them aligned if two robots are close to each other, we claim that the goal of primitive I is achieved if the magnets are active. As shown in Figure 4.3, the point (-0.15, 0) has the highest probability of success since it is the closest position to the front

Figure 4.3: The magnets attract and pull two robots together in the simulation.

robot. As the initial position gets further away, the probability of success drops. This is because both robots are trying to maximize the reward. Even though the front robot tries to slow down in order for the back robot to catch up and obtain the alignment reward r_{align}, it won't completely stop since it will lose the forward velocity reward $v_{x,1}$. Moreover, the robots have trouble aligning themselves when the back robot is away from y-axis because we train the policy with the back robot having only small random variation in y direction. Therefore, the policy have not seen these states during training. If the y deviation is large, the back robot may catch up with the front robot and pass the front robot.

4.1.2 Primitive II: Pitching Up

In primitive II, the robots start from an aligned position with both magnets active, and approach the step in a line formation. Once the front robot detects the step, the tether connection will be established to help the front robot to pitch up against the step. The tether may not be necessary for climbing the step because the front robot can also pitch up by using the frictional force. However, the tether certainly helps speeding up the climbing process, especially when climbing larger obstacles.

The experiment result in Figure 4.5 shows that it is pretty reliable to get the front legs of the front robot on top of the step once they are aligned. In order to get more accurate transition probabilities, we run the experiments for 2000 trials. The three connections between the robots are the main reasons why we are able to get a success rate that is close to 100%. The fact that the result is almost deterministic implies that there isn't much learning involved in this primitive. From the reward function's perspective, we see that the tether connection guarantees the reward $z_{\text{front},1}$, and the magnetic connections make sure that the yaw penalty is small since the motions in primitive II are not as dynamic as those in primitive III.

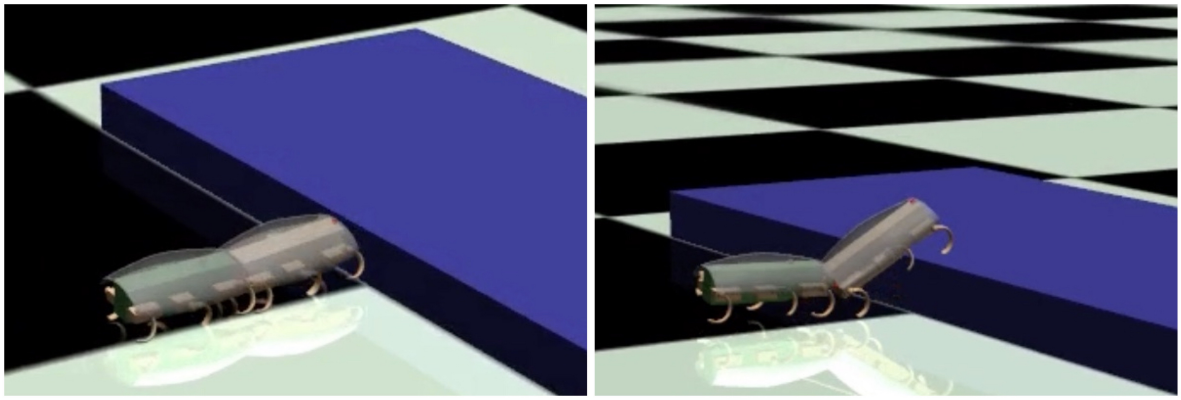

Figure 4.4: Picture showing the start and end states of primitive II in simulation.

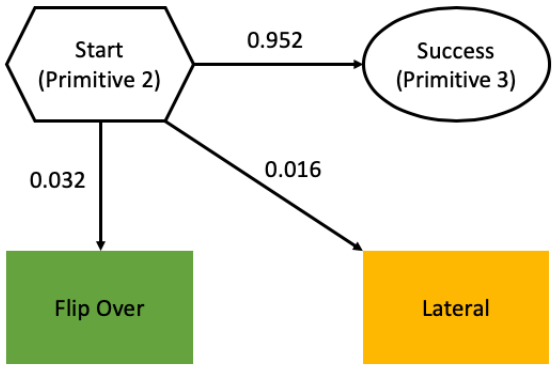

Figure 4.5: Primitive II transition probabilities for step size equal to 5cm. "Start" represents the start state of primitive II, namely that two robots are aligned in front of the step. "Success" represents the end state of primitive II (or start state of primitive III), namely that the front robot successfully pitches up against the step. "Flip Over" and "Lateral" represent the failure states in which the front robot is either flipped over or walking along the y-axis.

4.1.3 Primitive III: Climbing

In primitive III, the robots start from the state in which the front robot has pitched up against the step and all three connections are active. The goal state is that the front robot successfully climbs up the step. The motions in this primitive are more dynamic, and thus the robots have a hard time keeping the yaw of the front robot stable. Despite the three connections we have, the policy has to learn how to stabilize the yaw while maximizing the rewards. As a result, the failure rate is much higher than the failure rate of primitive II. Figure 4.7 below shows that the probability of flipping over is much larger than "Aligned" and "Lateral" states.

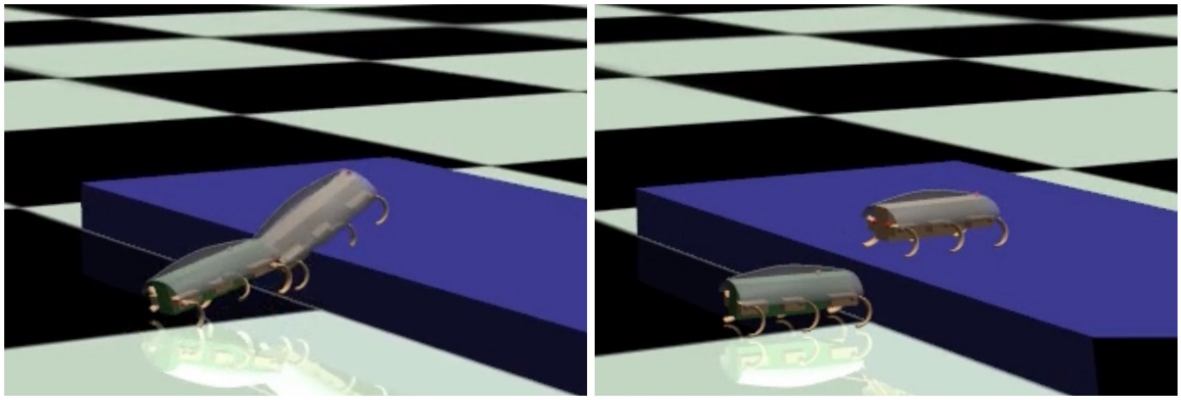

Figure 4.6: Picture showing the start and end states of primitive III in simulation.

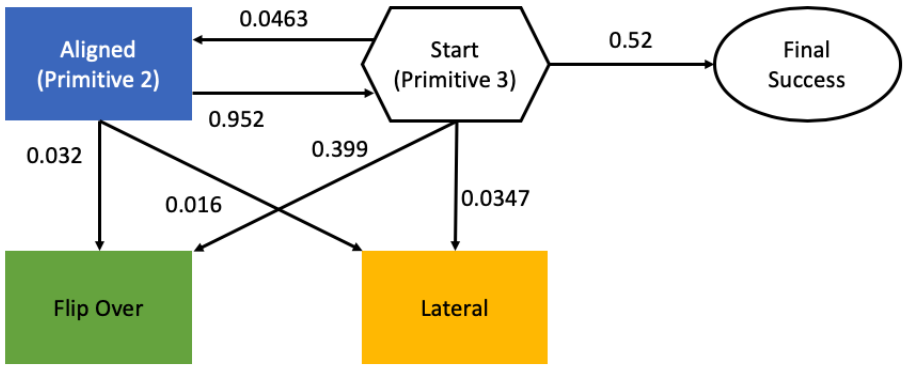

Figure 4.7: Primitive III transition probabilities for step size equal to 5cm. "Start" represents the start state of primitive III. "Success" represents the end state of primitive III, namely that the front robot successfully reaches the top of the step. "Aligned" represents the start state of primitive II.

4.2 Slope vs Step

We compare the performances when the robots try to overcome slopes with different angles and steps with different heights. The angles of the slopes are 20, 30, 40 and 50 degrees respectively, and the tops of the slopes are all 5cm above the ground. The heights of the steps are 3cm, 4cm, 5cm and 6cm respectively.

As shown in Figure 4.9, when the slope angle is 20 degrees, the robots have no trouble climbing up the slope and reach the top of the obstacle. Since the robots are able to walk without slipping on this 20-degree slope, we do not have a "Pitched-up" state. The only observed failure state is that the front robot walks along the y-axis ("Lateral" state). As the angle of the slope increases, it becomes harder for the front robot

to climb up the slope, and a new failure state "Flip Over" appears.

A slope with angle of 90 degrees is a step, and we stop increasing the slope angle after 50 degrees because the front robot starts to have a much harder time to pitch up. To a certain extent, it is harder for the robots to climb over a slope with a large angle (greater than 40 degrees) than a step with the same height because if the obstacle is a step, the back robot can push the front robot right at the edge of the step. On the other hand, if the obstacle is a slope with a large angle, part of the back robot is on the slope and it keeps sliding down. Therefore, the back robot is further away from the top edge of the slope and is not able to push as hard.

Figure 4.10 shows the experiment results when the step height is equal to 3cm, 4cm, 5cm and 6cm. Since the height of the robot in the simulation is about 4.5cm, the first two steps are lower than the robot, and the last two steps are higher than the robot. Unlike the experiments with a slope as the obstacle, one major difference here is that the probability of ending up in "Flip Over" state is much higher. This is because the robots have a harder time to keep the yaw of the front robot stable when climbing a step than a slope. A slope can provide an extra frictional force to help stabling the yaw if the front robot starts to fall to one side while a step cannot. Therefore, it is more likely for the robots to end up in "Lateral" state when climbing the slope, and in "Flip Over" state when climbing the step.

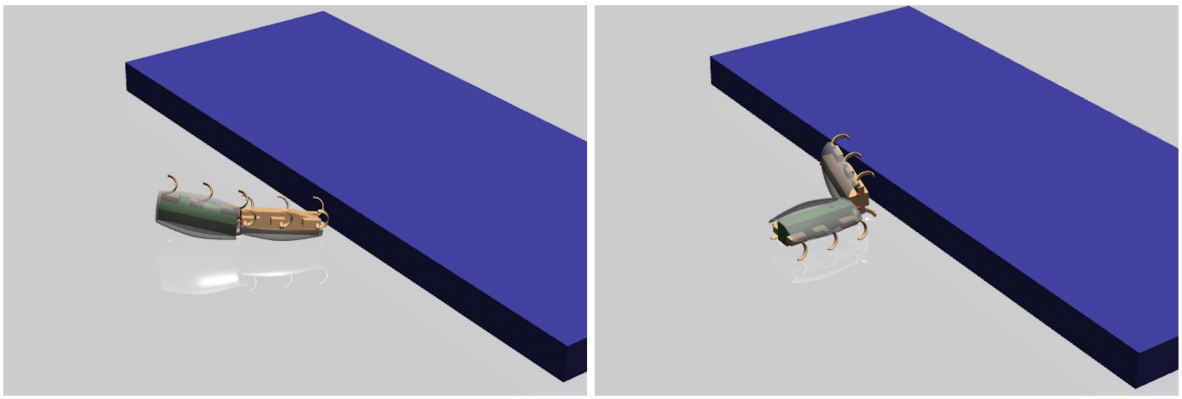

Figure 4.8: Two failure states. Left: "Flip Over". Right: "Lateral".

Another observation is that there is a huge drop in the transition probability from "Start" to "Success" (from 86.5% to 44%) when the slope's angle increases from 30 to 40 degrees. One reason is that we have trained the policy on slopes up to only 30 degrees before switching to step, and a different climbing strategy may be needed in order for the robots to climb over a 40-degree slope. Therefore, this is an obstacle that the

robots have never seen before, and the learned policy does not generalize well.

The transition probability is summarized in table 4.1. Finally, we collect 50 successful trials, and conclude that the average time of success is 13.823 seconds, and the standard deviation is 5.047 seconds.

From State	To State	Slope (5cm tall)				Step			
		20 degree	30 degree	40 degree	50 degree	3cm	4cm	5cm	6cm
Start	Flip Over	-	-	-	0.007	-	0.022	0.032	0.0585
Start	Lateral	0.014	0.029	0.023	0.034	0.0045	-	0.016	-
Start	Pitched Up	-	0.971	0.977	0.959	0.9955	0.978	0.952	0.9415
Start	Success	0.986	-	-	-	-	-	-	-
Pitched Up	Start	-	0.0193	0.066	0.2831	0.0571	0.124	0.0463	0.0098
Pitched Up	Flip Over	-	0.0482	0.211	0.1456	0.0056	0.044	0.399	0.5759
Pitched Up	Lateral	-	0.0675	0.283	0.2063	0.0123	0.072	0.0347	0.0443
Pitched Up	Success	-	0.865	0.44	0.365	0.925	0.76	0.52	0.37

Table 4.1: Summary of Figure 4.6 and Figure 4.7.

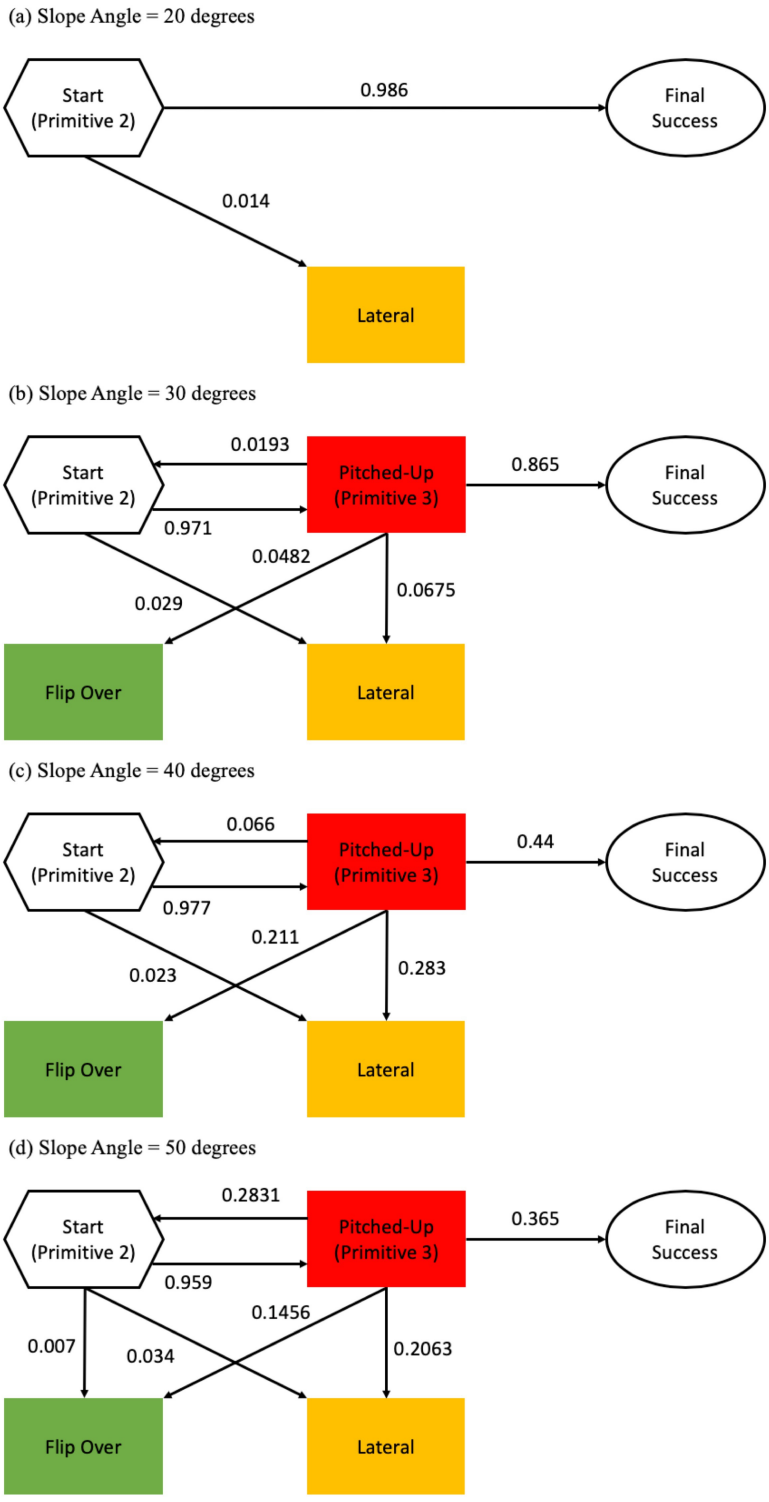

Figure 4.9: (a)-(d) show the state transition probability of the robots trying to overcome slopes with angles equal to 20, 30, 40 and 50 degrees.

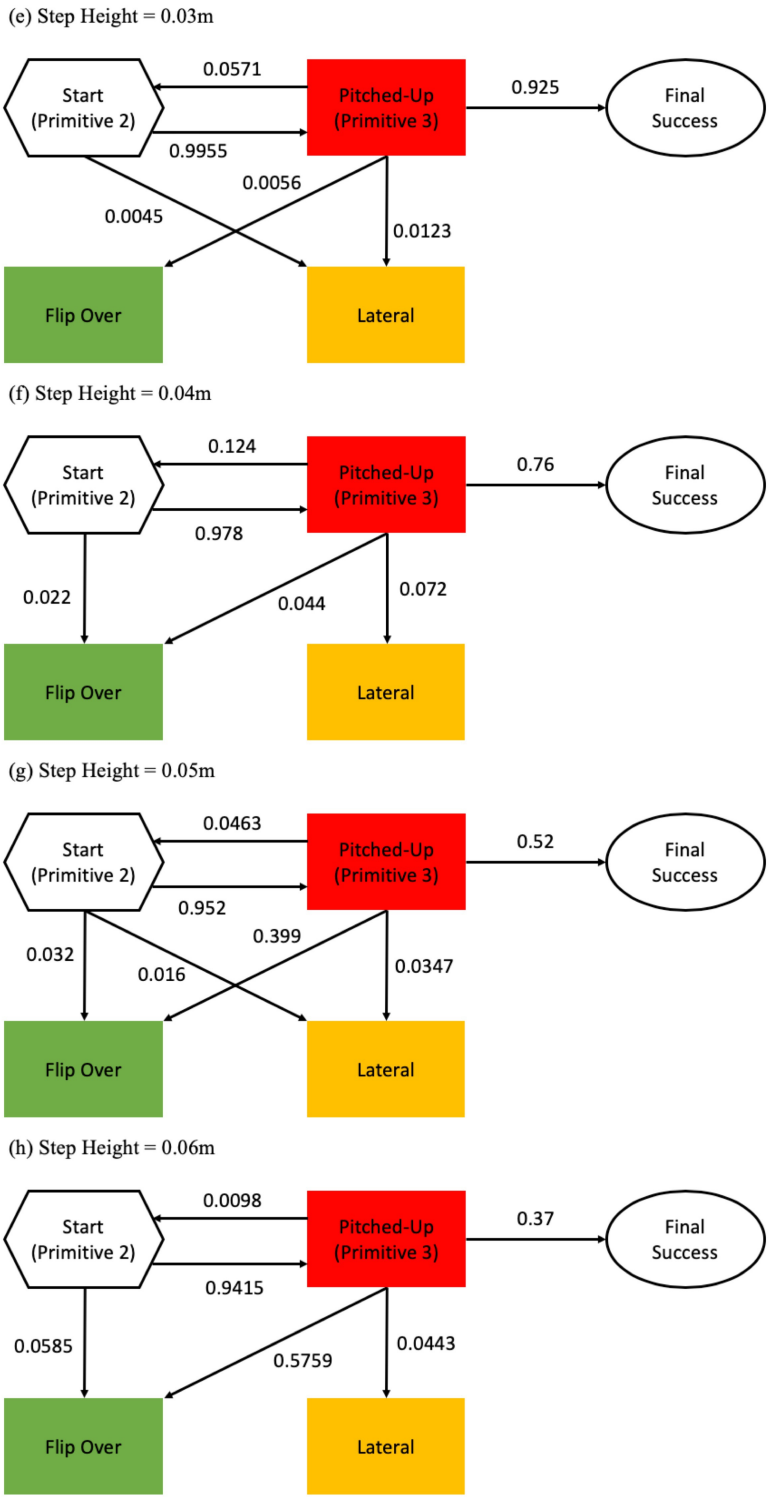

Figure 4.10: (e)-(h) show the state transition probability of the robots trying to overcome steps with heights equal to 3cm, 4cm, 5cm and 6cm.

5 Discussion

In conclusion, we have demonstrated a model-free learning method to control under-actuated legged millirobots to perform a specific cooperation task which allows one of the robots to overcome an obstacle taller than the robot itself. Using different reward functions in a desired order to train the policy, our approach is able to intrinsically learn the primitives which are necessary for this cooperative step climbing task. For a step with height equal to 5cm, the result shows that each primitive has a success rate of at least 50% if the back robot starts at a location within three body lengths behind the front robot.

One limitation of our method is that we have three connections between the robots. While we can form and release the magnetic connections by using electromagnets and controlling electric currents going through the coil, we cannot do so with the tether connection because tether's physical existence. In order for the robots to move independently before and after the cooperation, the back robot needs to attach the tether on the front robot and retrieve it afterwards. This attachment mechanism can be hard to implement on a millirobot like VelociRoACH.

Another limitation is that we cannot use model-free learning method if we want to transfer from simulation to real world, which is one direction for future work. We want to be able to transfer the results to VelociRoACH millirobot. However, we are not able to collect as fast or as many samples in the real world as in the simulation. Therefore, we need to develop a model-based learning method to learn the dynamics of the system. Furthermore, we want to design the tether and magnetic connections on the real robot such that either robot can be the front or the back robot because we don't want to assign a specific role to each robot.

Another interesting line of future work includes using more robots to overcome the obstacles. One advantage of having more robots is that we may find different climbing strategies for the same obstacle. Moreover, more robots can provide more stability, and therefore we may be able to remove the tether and even magnetic connections. However, when the number of robots increases, we can no longer use a centralized control method because the state space and action space will become too large to search. If we use a decentralized control method, there are other issues such as how the robots communicate and cooperate with each other without knowing other robots' full states.

6 Acknowledgements

The author would like to thank the members of the Biomimetic Millisystems Laboratory for their support, especially Carlos Casarez for his expertise in VelociRoACH platform, Anusha Nagabandi for helping me understanding different types of algorithms in reinforcement learning, and Professor Ronald S. Fearing for all the insightful discussions and guidance.

Bibliography

[1] D. W. Haldane, K. C. Peterson, F. L. Garcia Bermudez, and R. S. Fearing, "Animal-inspired design and aerodynamic stabilization of a hexapedal millirobot," *IEEE International Conference on Robotics and Automation*, pp. 3279–3286, May 2013.

[2] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

[3] D. W. Haldane and R. S. Fearing, "Roll oscillation modulated turning in dynamic millirobots," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4569–4575, May 2014.

[4] A. Nagabandi, G. Yang, T. Asmar, R. Pandya, G. Kahn, S. Levine, and R. S. Fearing, "Learning image-conditioned dynamics models for control of underactuated legged millirobots," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4606–4613, 2018.

[5] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *International Conference on Machine Learning (ICML)*, 2015.

[6] P. Birkmeyer, A. G. Gillies, and R. S. Fearing, "Clash: Climbing vertical loose cloth," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5087–5093, Sep 2011.

[7] G. A. Lynch, J. E. Clark, P.-C. Lin, and D. E. Koditschek, "A bioinspired dynamical vertical climbing robot," *The International Journal of Robotics Research*, vol. 31, no. 8, pp. 974–996, 2012.

[8] M. J. Spenko, G. C. Haynes, J. A. Saunders, M. R. Cutkosky, A. A. Rizzi, R. J. Full, and D. E. Koditschek, "Biologically inspired climbing with a hexapedal robot," *Journal of Field Robotics*, vol. 25, no. 4-5, pp. 223–242, 2008.

[9] S. Kim, M. Spenko, S. Trujillo, B. Heyneman, D. Santos, and M. R. Cutkosky, "Smooth vertical surface climbing with directional adhesion," *IEEE Transactions on Robotics*, vol. 24, pp. 65–74, Feb 2008.

[10] P. Birkmeyer, A. G. Gillies, and R. S. Fearing, "Dynamic climbing of near-vertical smooth surfaces," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 286–292, Oct 2012.

[11] P. Beardsley, R. Siegwart, M. Arigoni, M. Bischoff, S. Fuhrer, D. Krummenacher, D. Mammolo, and R. Simpson, "Vertigo – a wall-climbing robot including ground-wall transition," Dec 2015. https://la.disneyresearch.com/publication/vertigo/.

[12] J. E. Bares and D. S. Wettergreen, "Dante ii: Technical description, results, and lessons learned," *The International Journal of Robotics Research*, vol. 18, no. 7, pp. 621–649, 1999.

[13] T. Miki, P. Khrapchenkov, and K. Hori, "UAV/UGV autonomous cooperation: UAV assists UGV to climb a cliff by attaching a tether," *CoRR*, vol. abs/1903.04898, 2019.

[14] C. Casarez and R. S. Fearing, "Step climbing cooperation primitives for legged robots with a reversible connection," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3791–3798, May 2016.

[15] R. Blickhan and R. J. Full, "Similarity in multilegged locomotion: Bouncing like a monopode," *Journal of Comparative Physiology A*, vol. 173, pp. 509–517, Nov 1993.