Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

Task-oriented Hierarchical Control for a Quadruped Robot

Linqi Ye, Houde Liu, Xueqian Wang and Bo Yuan
Graduate School at Shenzhen
Tsinghua University
Shenzhen, Guangdong Province, China
{ye.linqi, liu.hd, wang.xq & yuanb}@sz.tsinghua.edu.cn

Bin Liang
Department of Automation
Tsinghua University
Beijing, China
bliang@tsinghua.edu.cn

***Abstract* – This study aims to build a standard framework that can be applied to accomplish different tasks for a quadruped robot. To achieve this goal, a task-oriented hierarchical control framework is proposed. The framework is composed of four layers, including the task, action, movement, and joint layer, where the action layer plays a crucial role. An action is defined as a group of movements that take the robot from an initial position/orientation to another position/orientation. All actions form the action library, which can be combined to accomplish different tasks. Based on the hierarchical control framework, two tasks including path following and ditch crossing are achieved in this paper. The advantages of the proposed hierarchical control framework are its universality and extendibility since it can be easily applied to other tasks by enriching the action library. Simulation results in V-REP are given to validate the effectiveness of the proposed method.**

Index Terms* – *Quadruped robot, hierarchical control, path following, ditch crossing.

I. INTRODUCTION

Legged robots have attracted a lot of attentions in recent years since they are considered to be more capable of traversing rough terrains than wheeled or tracked robots [1]. Legged robots can be divided into biped, quadruped, and multilegged robots, among which quadruped robot balances well between stability and complexity, making it promising to be a good assistant in many aspects such as disaster relief, field transportation, and planetary exploration.

As a kind of mobile robot, the basic task for a quadruped robot is locomotion, that is, to transfer from one place to another place. According to the walking environments, the locomotion task includes path following, ditch crossing, slope walking, stairs climbing and so on. Various methods have been proposed considering different task scenarios. The problem of following an arbitrary path is addressed in [2] using both discontinuous crab and turning gaits for quadruped robots. Path following in a cluttered environment is achieved in [3] for a real quadruped robot named TITAN-VIII by using continuous and omnidirectional crawl gait. In [4], a successive gait-transition method is proposed for a quadruped robot to realize omnidirectional static walking. In [5], omnidirectional path following is achieved with parameters optimized simultaneously for all directions of motion and turning rates. A two-phase discontinuous gait algorithm for ditch crossing and avoidance of quadruped robots with a failed leg is presented in [6]. A rhythmic motion generation and gait planning algorithm is proposed in [7] for crossing planar obstacles such as a ditch for a quadruped robot. The relationship between the morphological parameters of quadruped robots and the capability of obstacle-negotiation in the ditch crossing scenario is studied in [8] based on static stability and critical postures. Besides, slope walking is studied in [9, 10] and stairs climbing is studied in [11, 12].

A more challenging task is to transverse rocky terrain. In 2005, the Defense Advanced Research Projects Agency (DARPA) launched a three-year program named Learning Locomotion which is designed to investigate quadrupedal locomotion over rocky terrain with the Boston Dynamics "LittleDog" quadruped robot. A bunch of results have been obtained from this program by teams from different institutions. The Stanford University team [13] presents a hierarchical control architecture that enables the "LittleDog" to walk over rough terrain. The University of Southern California team [14] achieves the task by decomposing it into many sub-systems, in which the state-of-the-art learning, planning, optimization and control techniques are applied.

Although various methods have been proposed to accomplish different tasks for a quadruped robot, in this paper we aim to build a standard framework that can be applied to handle a wide variety of tasks. Motivated by [13, 14], a task-oriented hierarchical control framework is proposed. In [13], the hierarchical control includes a high-level planner that plans a set of footsteps, a low-level planner that plans trajectories for the robot's feet and center of gravity (COG), and a low-level controller that tracks these desired trajectories. In this paper we adopt a different hierarchical logic, where the framework is composed of four layers, including the task, action, movement, and joint layer. Then the task can be solved from top to bottom, step by step, which greatly reduces the complexity. The idea is very similar to the hierarchical structure in computer programming, where the system is modularized into different subfunctions and the main program calls the subfunctions to complete a sophisticated task. Based on the hierarchical control framework, two tasks including path following and ditch crossing are achieved. The advantage of the proposed hierarchical control framework is its universality and expandability. By enriching the action library, the hierarchical control framework can be easily extended to other tasks such as slope walking and stairs climbing.

*This work was supported by National Natural Science Foundation of China (No.61673239, No.61803221), Natural Science Foundation of Guangdong Province (2015A030313881) and the Basic Research Program of Shenzhen (JCYJ20160301100921349, JCYJ20160428182227081). (Corresponding author: Houde Liu)

978-1-7281-6321-5/19//$31.00 ©2019 IEEE

II. The Hierarchical Control Framework

As shown in Fig. 1, the hierarchical control framework is composed of four layers, including the task, action, movement, and joint layer.

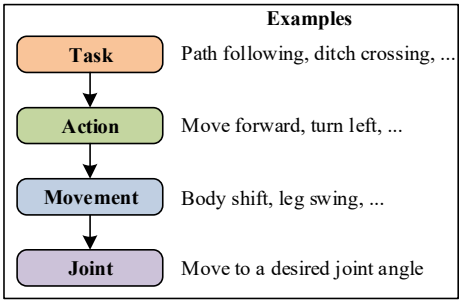


Fig. 1 The hierarchical control framework.

(1) The task layer: This is the highest layer. In this layer, the quadruped robot is treated as a mass point, and a task is usually to transfer from one place to another place. Examples of tasks are path following and ditch crossing.

(2) The action layer: This is a crucial layer. An action consists of a series of movements that take the robot from an initial position/orientation to another position/orientation, for example, move forward/backward for a certain distance, or turn left/right for a certain angle.

(3) The movement layer: This is a single movement of the body or the leg. For example, the body shift in a direction for a certain distance or pitch/roll/yaw for a certain degree. A leg swing is also a movement.

(4) The joint layer: This is the lowest layer. In this layer, all joints follow the desired joint angles so that the robot's body or leg can realize a desired movement.

With the definition of each layer, a task can then be achieved from top to bottom by designing the commands for each layer. We call this as the task-oriented hierarchical control. In this framework, the realization of a task can be divided into three steps as shown in Fig. 2.

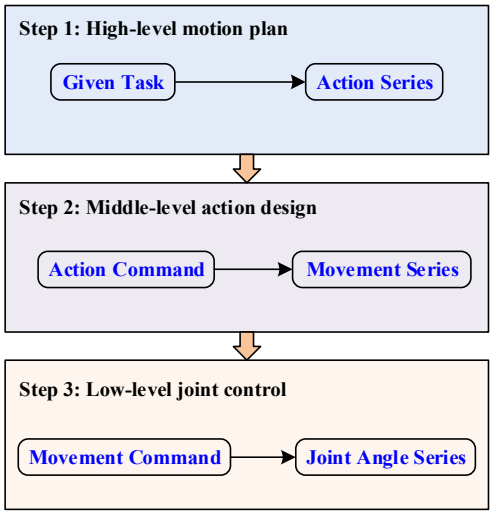


Fig. 2 Three steps of the hierarchical control framework.

In the first step, the task is decomposed into a series of actions. In this step we treat the robot as a whole and plan its motion at a macro level. In the second step, an action is decomposed into a series of movements. This step cares about the detailed movement of robot's different parts. In the third step, a movement is decomposed into a series of joint rotations which can be achieved by controlling the motors. Among the three steps, the third step is relatively easy since it is an inverse kinematics problem where many existing methods are available. Therefore, we will focus on the first two steps in the following sections.

III. Establishment of the Action Library

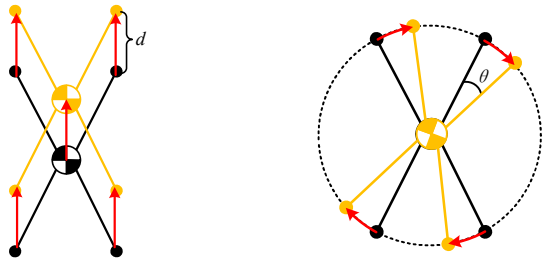
In this section, we will establish the action library which are needed to accomplish the given tasks. An action consists of a series of movements that take the robot from an initial position/orientation to another position/orientation. There are some basic requirements for an action.

(1) First of all, during the execution of an action, the robot should maintain stable. To make it simple, only static gait is considered in this paper and the center of mass (CoM) of the robot is assumed to be located below the geometric center. Therefore, the stability requirement becomes that the projection of the CoM should maintain in the support polygon.

(2) Secondly, an action should be combinable. This means that an action can be spliced with itself or other actions. It requires the final configuration of the robot in an action can be used as the initial configuration of another action.

In addition, there can be some other requirements such as the action parameters are adjustable.

When a task is given, a customized action library should be carefully built such that it has the ability to accomplish the given task. In this paper, two tasks are considered, that are, path following and ditch crossing. For these two tasks, only two basic actions are needed. As shown in Fig. 3, one is the translation action where the robot moves forward or backward for a certain distance, and the other is the rotation action where the robot rotates along its center for a certain angle. For other tasks such as slope walking and stairs climbing, more actions can be added to the action library.



(a) Translation (b) Rotation

Fig. 3 Two basic actions.

As an interface to the high-level planning subsystem, the task layer only cares about the input/output of an action, which are listed in Table I and II.

TABLE I
SPECIFICATION OF THE TRANSLATION ACTION

Action parameter	The translation distance, go forward as + and go backward as -
Initial configuration	Standard configuration
Final configuration	Standard configuration

TABLE II
SPECIFICATION OF THE ROTATION ACTION

Action parameter	The rotation angle, turn left as + and turn right as -
Initial configuration	Standard configuration
Final configuration	Standard configuration

In the middle-level action design subsystem, we need to specify the detailed movements within each action. To ensure static stability, the strategy we adopted is to let the robot switch between body shift and leg swing. During the period of body shift, the robot keeps its four feet on the ground and shifts the body to the desired position/orientation. During leg swing, the robot keeps it body still and lifts one foot, move it to reach the desired position and then put it down.

Specifically, the translation action consists of 7 movements as shown in Fig. 4. In consideration of stability, the leg swing sequence is 4-2-3-1 when going forward and 2-4-1-3 when going backward.

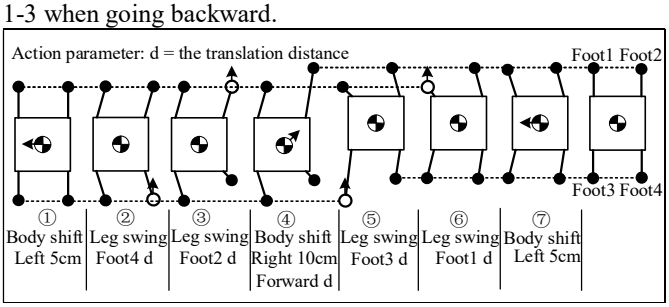


Fig. 4 Action of translation (go forward).

The rotation action consists of 9 movements as shown in Fig. 5. To obtain larger stability margin, the leg swing sequence is 1-2-4-3 when the robot makes a right turn and 2-1-3-4 when it makes a left turn.

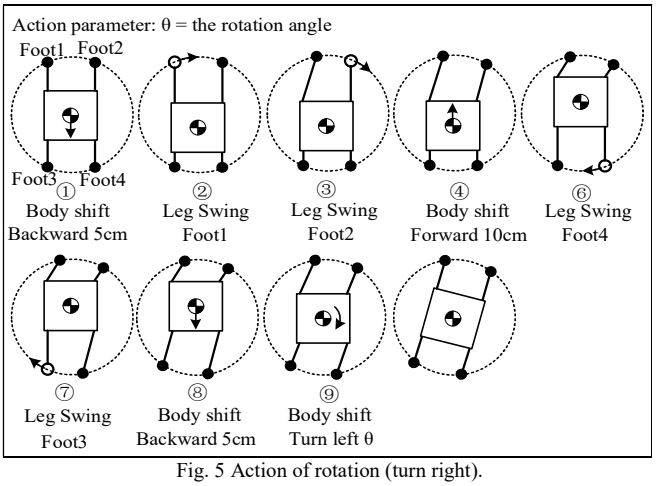


Fig. 5 Action of rotation (turn right).

During a rotation, we need to determine the next location of each foot. As shown in Fig. 6, R is half the length between two diagonal feet in the standard configuration, α is the angle between the foot diagonal line and the fore-aft centerline of the robot, and θ is the rotation angle.

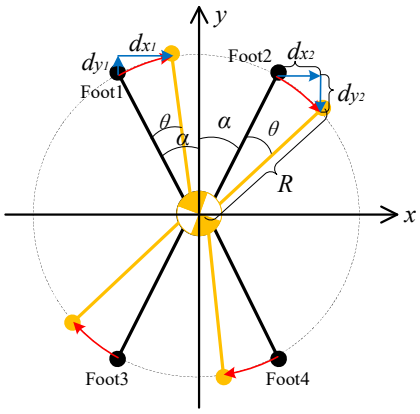


Fig. 6 Next location of each foot after a rotation.

When the robot makes a right turn of θ, the relative displacement of Foot 1 and Foot 2 can be obtained as

$$\begin{aligned} d_{x1} &= -R\left[\sin(\alpha-\theta)-\sin\alpha\right] \\ d_{y1} &= R\left[\cos(\alpha-\theta)-\cos\alpha\right] \\ d_{x2} &= R\left[\sin(\alpha+\theta)-\sin\alpha\right] \\ d_{y2} &= R\left[\cos(\alpha+\theta)-\cos\alpha\right] \end{aligned} \tag{1}$$

Similarly, for Foot 3 and Foot 4, we have

$$\begin{aligned} d_{x3} &= -R\left[\sin(\alpha+\theta)-\sin\alpha\right] \\ d_{y3} &= -R\left[\cos(\alpha+\theta)-\cos\alpha\right] \\ d_{x4} &= R\left[\sin(\alpha-\theta)-\sin\alpha\right] \\ d_{y4} &= -R\left[\cos(\alpha-\theta)-\cos\alpha\right] \end{aligned} \tag{2}$$

Then the robot can place its feet to the right position during leg swing.

III. TASK REALIZATION

Based on the action library, it is now ready to do the high-level motion planning according to different tasks. In this section we will introduce how to realize the path following and ditch crossing task, respectively.

A. Path Following

For the path following task, the center (i.e., CoM) of the quadruped robot is required to follow a predefined path on the level ground as shown in Fig. 7. Particularly, the robot is required to adjust its orientation according to its position on the path so that the robot's head is always facing to the front path.

The strategy we adopted here is similar to the "turtlesim" simulator in ROS Kinetic, where the turtle can rotate or go forward and it can move around by controlling the transition between these two actions. For path following, the key is to determine when the robot should go forward and when to make a turn. Since we have already built the action library, the

problem then becomes to determine which action to take. The detailed path following algorithm is given in Fig. 8.

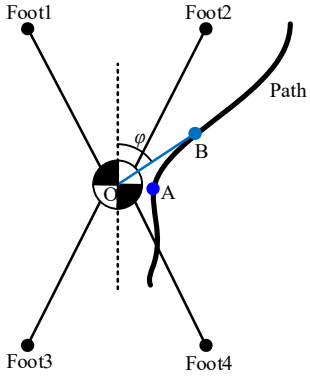


Fig. 7 Schematic diagram of path following.

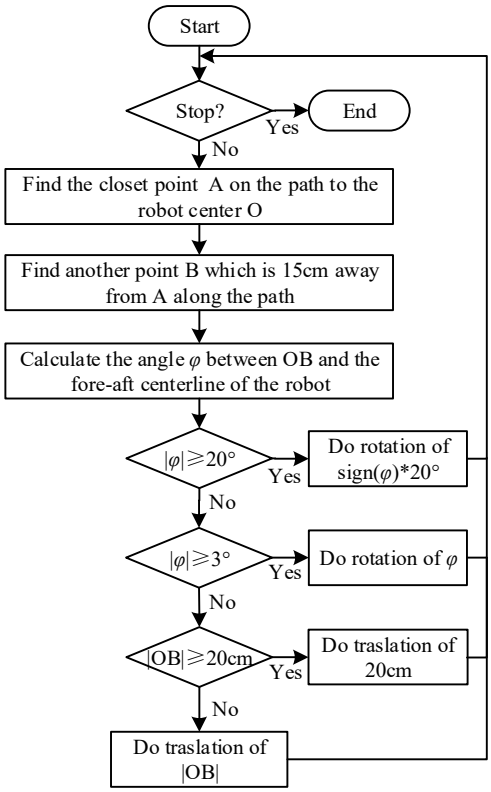


Fig. 8 The path following algorithm.

B. *Ditch Crossing*

For the ditch crossing task, the quadruped robot is required to safely cross a ditch from one side to another side on the level ground. We assume that the ditch width does not exceed the maximum step length of the quadruped robot. The basic idea of the ditch crossing strategy is: keep the body front edge parallel to the edge of the ditch, and then go forward and adjust the step length according to the location of the ditch. As shown in Fig. 9, denote the ditch width as w, the distance of the front leg to the ditch edge as d_1 and the distance of the hind leg to the ditch edge as d_2. Then the detailed ditch crossing algorithm is given in Fig. 10. With this algorithm, the robot can safely step over the ditch regardless of its initial position and orientation.

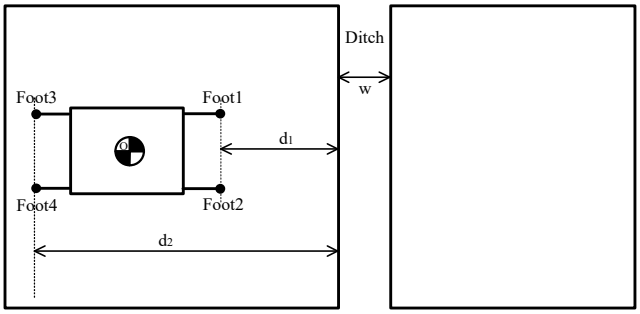


Fig. 9 Schematic diagram of ditch crossing.

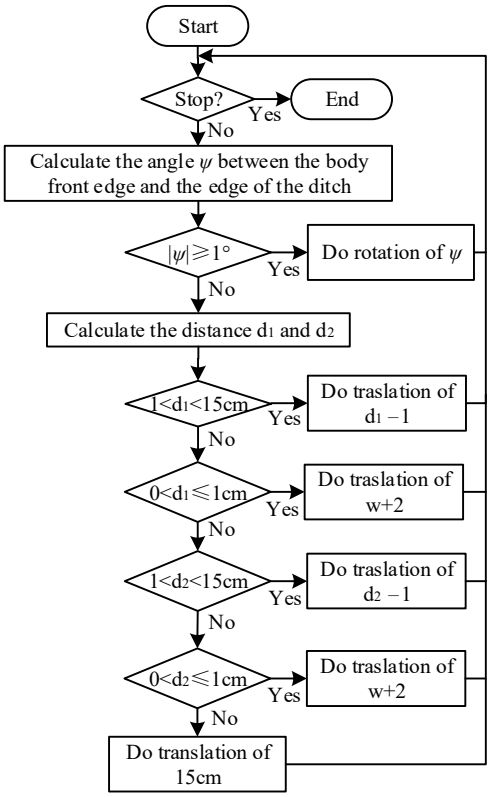


Fig. 10 The ditch crossing algorithm.

IV. Simulation Results

To validate the effectiveness of the proposed method, simulations are done in the V-REP software with the Newton physical engine. The quadruped robot model we built in V-REP is shown in Fig. 11, which is originated from our prototype quadruped robot as shown in Fig. 12.

Unlike many other quadruped robot, the leg configuration we adopted is backward/forward (outward-pointing) because: (1) To ensure that the CoM is close to the geometric center, a symmetric configuration (inward-pointing or outward-pointing) is preferred considering that the leg is relatively

heavy since one motor is placed on the knee. (2) Outward-pointing can avoid collision between the front leg and the hind leg. Other specifications of the quadruped robot model are listed in Table III.

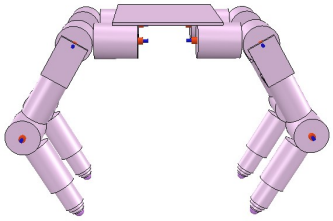


Fig. 11 The quadruped robot model in V-REP.

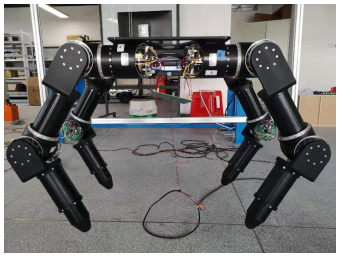


Fig. 12 The prototype quadruped robot.

TABLE III
SPECIFICATIONS OF THE QUADRUPED ROBOT MODEL

Size (L × W × H, fully stretched legs)	0.6m×0.4m×0.65m
Weight	56kg
Degrees of Freedom	12 (3 per leg, including HAA, HFE, and KFE)
Joint Torque	100 Nm for all joints

A. Path Following

To test the path following algorithm, we built the scenario as shown in Fig. 13. Two paths are considered here: one is a closed curve and the other is a pentagram.

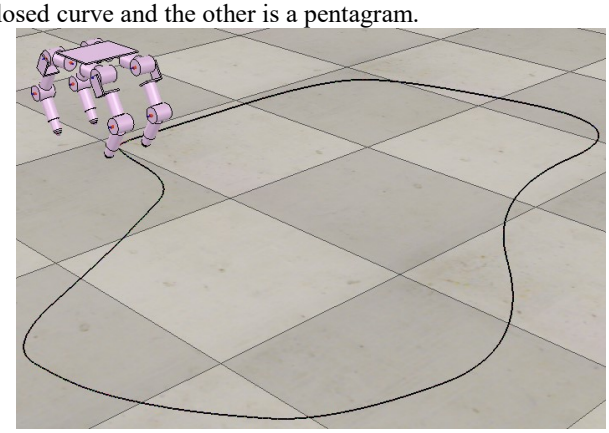


Fig. 13 Scenario of the path following task.

Fig. 14 shows the simulation results of following a closed curve. The blue line shows the trajectory of the robot center while the red dots are the position of the robot center after taking each action. It can be observed that the robot follows very well with the curve. Specifically, the robot's center is almost exactly on the path after each action, while it crossess the path back and forth during each action. Fig. 15 shows the simulation results of following a pentagram path. The robot still follows very well. Particularly, it is noticed that the red dots are dense at the sharp corners. This is because that the robot do rotation for several times at the corner to adjust its orientation so that the robot can face to the path.

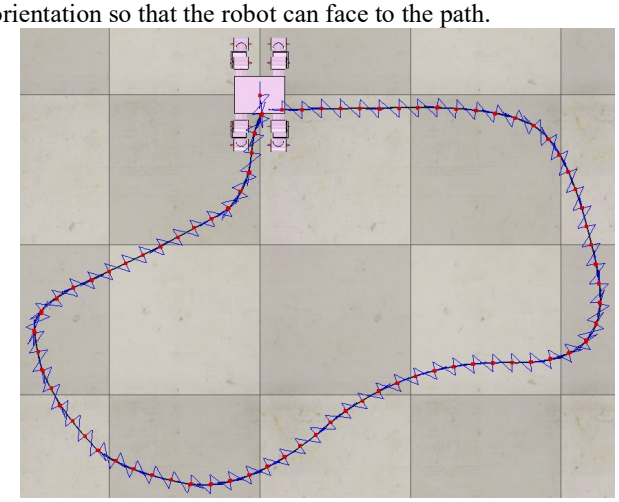


Fig. 14 Simulation results of following a closed curve. The blue line shows the trajectory of the robot center while the red dots are the position of the robot center after taking each action.

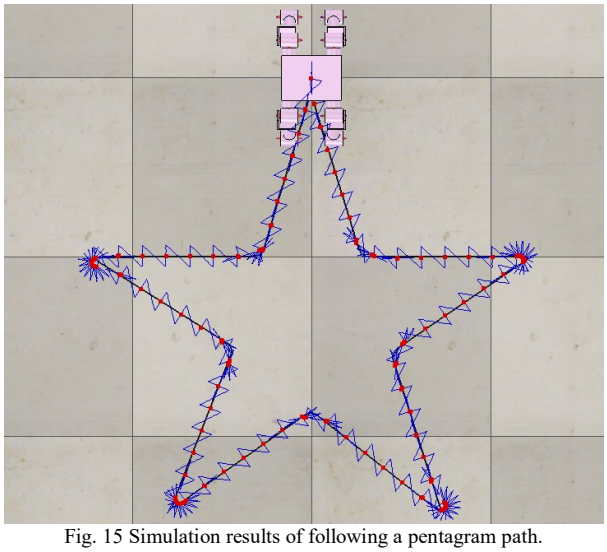


Fig. 15 Simulation results of following a pentagram path.

B. Ditch Crossing

To test the ditch crossing algorithm, we built the scenario as shown in Fig. 16. The ditch width is 20cm, and the initial orientation of the robot's body is not aligned with the ditch.

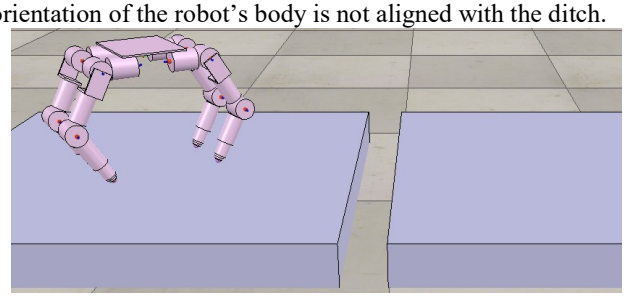


Fig. 16 Scenario of the ditch crossing task.

The simulation result is shown in Fig. 17. The red dots represent the footprints of the front legs and the blue dots for the hind legs. It can be seen that all the footprints avoids the ditch very well which indicates that the robot successfully crosses the ditch. The footprints show that the robot is adjusting its orientation at the beginning to make its body front edge be parallel with the ditch edge. After its orientation is ready, the robot starts to approach the ditch. Particularly, it can be noticed that the robot reduces its step length when the front legs are close to the ditch.

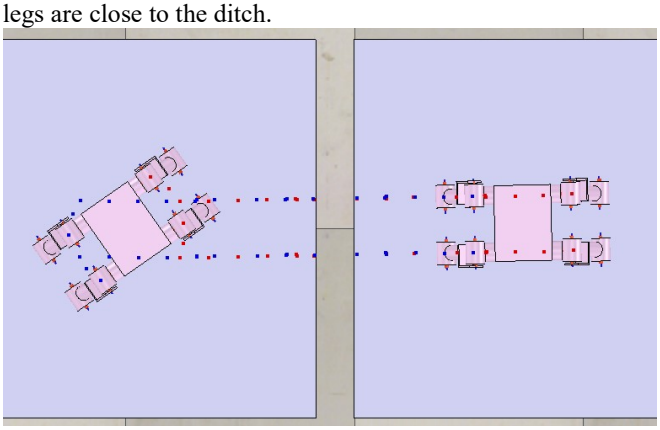


Fig. 17 Simulation results of ditch crossing. The red dots represent the footprints of the front legs and the blue dots for the hind legs.

To further verify that the ditch crossing ability is irrelevant with the robot's initial position/orientation, 50 MonteCarlo runs are taken with the robot's initial orientation selected randomly within [0, 360°] and its initial position selected randomly within a 40cm × 40cm region (the black square in Fig. 18). The simulation results are given in Fig. 18. Still the red dots represent the footprints of the front legs and the blue dots for the hind legs. It can be seen that the robot successfully crosses the ditch for all the 50 trials, which verifies the robustness of the ditch crossing algorithm.

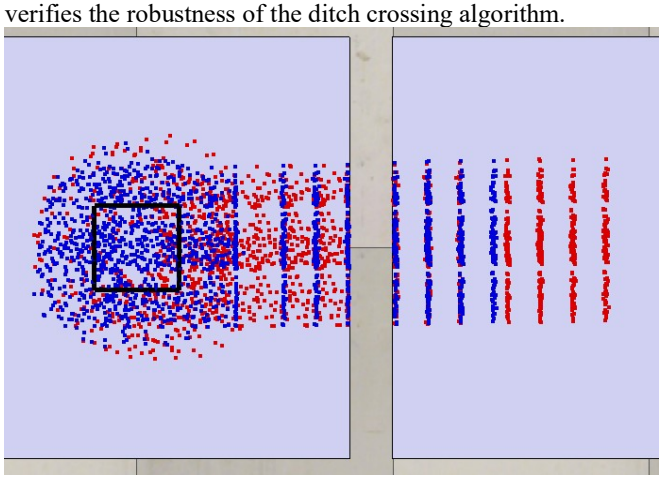


Fig. 18 Monte Carlo simulation results of ditch crossing. The robot's initial orientation is selected randomly within [0, 360°] and its initial position is selected randomly within the black square.

V. Conclusions

This paper proposes a task-oriented hierarchical control framework for a quadruped robot. There are four layers in this framework, including the task, action, movement, and joint layer. The action is the core in this framework, which consists of a series of movements that take the robot from an initial position/orientation to another position/orientation. Different actions can be combined to achieve a given task. Two tasks are considered in this paper, including path following and ditch crossing. A customized action library is built and the high-level control algorithms are presented. This framework takes into account both universality and extendibility, which gives it the potential to accomplish a wide variety of tasks. In the future, more complicated tasks such as slope walking and stairs climbing will be considered and experiments will be done with our practical robot platform.

References

[1] X. Chen, Z. Yu, W. Zhang, Y. Zheng, and Q. Huang, A. Ming, "Bioinspired Control of Walking with Toe-off, Heel-strike and Disturbance Rejection for a biped robot," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 10, pp. 7962-7971, 2017.

[2] P. G. De Santos, E. Garcia, and J. Estremera, *Quadrupedal locomotion: an introduction to the control of four-legged robots*. Springer Science & Business Media, 2007.

[3] X. Chen, K. Watanabe, K. Kiguchi, and K. Izumi, "Path tracking based on closed-loop control for a quadruped robot in a cluttered environment," *Journal of Dynamic Systems, Measurement, and Control*, vol. 124, no. 2, pp. 272-280, 2002.

[4] S. Ma, T. Tomiyama, and H. Wada, "Omnidirectional static walking of a quadruped robot," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 152-161, 2005.

[5] J. Z. Kolter and A. Y. Ng, "Learning omnidirectional path following using dimensionality reduction," *Robotics: Science and Systems*, 2007.

[6] J. M. Yang, "Two-phase discontinuous gaits for quadruped walking machines with a failed leg," *Robotics and Autonomous Systems*, vol. 56, no. 9, pp. 728-737, 2008.

[7] X. Wu, X. Shao, and W. Wang, "Gait planning of crossing planar obstacles for a quadruped robot," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2013, pp. 692-697.

[8] Y. Gao, V. Barasuol, D. G. Caldwell, and C. Semini, "Study on the morphological parameters of quadruped robot designs considering ditch traversability," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2016, pp. 283-288.

[9] M. Focchi, A. Del Prete, I. Havoutis, R. Featherstone, D. G. Caldwell, and C. Semini, "High-slope terrain locomotion for torque-controlled quadruped robots," *Autonomous Robots*, vol. 41, no. 1, pp. 259-272, 2017.

[10] C. Yu, L. Zhou, H. Qian, and Y. Xu, "Posture Correction of Quadruped Robot for Adaptive Slope Walking," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2018, pp. 1220-1225.

[11] C. Liu, Q. Chen, and D. Wang, "CPG-inspired workspace trajectory generation and adaptive locomotion control for quadruped robots," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 3, pp. 867-880, 2011.

[12] D. Pongas, M. Mistry, and S. Schaal, "A robust quadruped walking gait for traversing rough terrain," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 1474-1479.

[13] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "A control architecture for quadruped locomotion over rough terrain," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 811-818.

[14] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Fast, robust quadruped locomotion over challenging terrain," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 2665-2670.