Proceedings of the 2022 IEEE
International Conference on Robotics and Biomimetics
December 5-9, 2022, Xishuangbanna, China

CapPlanner: Adaptable to Various Topology and Locomotion Capability for Hexapod Robots

Changda Tian[1] and Yue Gao[2*]

***Abstract*— Hexapod robots are good at traversing on complex terrains, yet its capability is challenging to define. The robot's traverse ability varies due to its structure, topology, and locomotion controller. The existing motion planner rarely considers the robot's traverse ability, causing higher failure risk when it gives motion commands that do not match the robot's capability. In this paper, we present CapPlanner, a hierarchical motion control and planning system which can do long-range locomotion control and planning according to the learned traverse capability of the robot in different topologies. It consists of two layers, the bottom-level controller computes the trajectory of the body and the feet according to the terrain, local target and current feets' positions. Besides, it controls the motors to track the calculated trajectory. The top-level controller learns the traverse ability of the robot with its bottom-level controller by simulating locomotion tasks on various terrains and in different topologies. Hence our CapPlanner can guide the robot to reach a long-term destination with a much higher success rate. In the experiment, we test CapPlanner in simulation and on our real hexapod robot, Qingzhui. The results show that CapPlanner is able to accomplish long distance and tough terrain locomotion planning for hexapod robot.**

I. Introduction

Locomotion control of a hexapod robot is a complicated task, since it only relies on limited feet to contact complex terrain, while maintaining body balance and surmounting obstacles to reach its destination. Hierarchical control strategy is widely used in legged robot locomotion [1], [2]. The bottom level controller (BLC) controls the robot walking stably in complex terrain, while the top level controller (TLC) guides the robot to avoid obstacles and reach its long-range destination.

Recently, many effective methods have emerged in the field of motion control of legged robots, which play the role of BLC. Whole-body control (WBC) [3] is one of them. It can stabilize locomotion and design the ground reaction force to control the robot to desired state. MIT's Cheetah robot has gained fruitful research results in recent years under this scheme [4]. On MIT's Cheetah3, Di Carlo *et al.* used whole-body control to model the quadruped robot's dynamics and converts it to an MPC problem [5]. Then the MPC problem can be reformulated to a QP problem and solved by a QP optimizer. Similar works [6]–[8] with WBC on legged robot also perform well. Trajectory optimization (TO) is another widely used motion control method for legged robot, which uses the optimizer to find reasonable

This work is supported by the National Key Research and Development Program of China (Grant No. 2021YFF0307900), National Natural Science Foundation of China (Grant No. 61903247), and Shanghai Municipal Science and Technology Major Project (Grant No. 2021SHZDZX0102).

[1]Changda Tian is with Department of Automation, Shanghai Jiao Tong University, Shanghai, P.R. China, deepfluency@sjtu.edu.cn

[2]Yue Gao is with MoE Key Lab of Artificial Intelligence and AI Institute, Shanghai Jiao Tong University, Shanghai, P.R. China, yuegao@sjtu.edu.cn

* Corresponding author

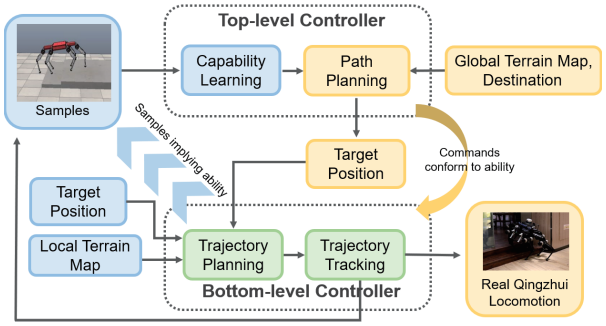


Fig. 1. Overall framework of CapPlanner. TLC learns the traverse ability of BLC-robot system and give commands that conform to BLC-robot system. The blue elements run in simulation, the yellow elements run in reality, and the green elements run in both simulation and reality.

feet trajectories according to robot model and environment constraints. Winkler *et al.* [9] presented a phase-based end-effector optimization method for legged walking robot to find a feasible trajectory of feet and body to across the terrain obstacles, which creates free gaits as well. However, WBC and TO have limitation that complex terrain will significantly increase model and optimization complexity. In this situation, learning based motion controllers are growing into a popular trend to the BLC of legged robot. In this approach, a simulation is often required to train the robot to walk and avoid obstacles. Hwangbo *et al.* [10] trained a network policy in simulation and transfered it to ANYmal robot. And Lee *et al.* [11] used a reinforcement learning method to train a proprioceptive locomotion controller in a simulation. Learning based method has stronger adaptability to different environments, however, it is hard to bridge the gap between simulation and reality. Besides, the training process is time-consuming and difficult to design.

The robot with a suitable BLC can successfully reach a local target, meanwhile, a TLC is still necessary for long-range planning. There are many classic path planning methods based on search and sampling. A^* algorithm [12] and its modifications such as $Theta^*$ [13] and jump point search A^* (JPS A^*) [14] are search based, they use heuristic searching on the shortest path [15]. Search based planning are resolution-complete but computationally expensive for complex environment. Sampling based planning approaches such as Probabilistic Road Map (PRM) [16] based methods and Rapidly-exploring Random Tree (RRT) [17] based methods. RPM methods randomly sprinkle points to find the target, then connect points to get the path. Rapidly-exploring Random Tree (RRT) methods and their modifications such as RRT* [18], [19], BIT* [20] search forward to the target step by step by generating random points. They can solve path planning problems in high-dimensional space and complex constraints. However, none of these methods consider the traverse capability of legged robots. As a result, how to design the TLC of legged robots which considers the ability of the system consists of the BLC and robot is worth

978-1-6654-8109-0/22/$31.00 © 2022 IEEE

studying.

In this paper, we propose CapPlanner for legged robot. It is a long distance locomotion planner and controller that is adaptable to topology and capability of robot. It has a hierarchical architecture consists of BLC and TLC. The function of BLC is to find feasible trajectories of feet and the torso according to local terrain, current topology and the local destination, then it controls the robot to track these trajectories. TLC learns the capability of the BLC with different topology, after which it uses learned model with value iteration method to obtain a global guidance trajectory. The contribution of CapPlanner mainly divided into the following three aspects:

- TLC of CapPlanner learns the traverse capability of BLC-robot system and gives path guidance from a global perspective. This inter-cooperative hierarchical controller design relieves the disadvantages of existing legged robots that are weak in global motion planning and local obstacle avoidance and gives robots the right to freely choose BLC that suits it.
- CapPlanner takes the robot's topology into consideration in locomotion control, which will affect the traverse capability of legged robot in different terrains. TLC learns the ability of BLC-robot system to leap over tough terrains and avoid obstacles in different topologies, after which it allows the robot to walk in the most suitable and safest topology based on its local terrain and task. This topology varying strategy effectively improves the stability and obstacle avoidance of legged robots.
- CapPlanner applies a sim-to-real framework. A precise robot model is created in a physical simulation together with it BLC. The TLC learns the locomotion capability of BLC-robot system in the simulation which is safe and efficient. We mend the gap between sim and real by applying position control both in sim and real. And the learned policy is applied to real robot which functions well in real robot experiment.

II. Methodology of CapPlanner

A. Non-linear Programming Based BLC

Inspired by the TO method proposed by Winkler *et al.* [9], we exploit an NLP based TO motion control strategy for our hexapod robot. In this control method, the feet and body trajectory are solved by formulating a nonlinear feasibility problem with constraints consist of robot kinematics and dynamics model, contact schedule, terrain heightmap, stance force, and start-end position constraints.

For a walking robot in global coordinate system at time t, let the center of mass (CoM) position be $\mathbf{r}(t)$, and the azimuth of the robot be $\boldsymbol{\theta}(t)$. And we define the positions of the i_{th} foot in body coordinate system as $\mathbf{p}_i$. Then, we use a bounding box around the nominal foothold position $\bar{\mathbf{p}}_i$ to represent the kinematics constraint. The size of the bounding box is decided after considering the geometry structure, the joint space limit and the interference among the legs.

$$\mathbf{p}_i \in B_i(\mathbf{r}, \boldsymbol{\theta}) \Leftrightarrow |R(\boldsymbol{\theta}(t))[\mathbf{p}_i(t) - \mathbf{r}(t)] - \bar{\mathbf{p}}_i| \leq \mathbf{b} \quad (1)$$

where $R(\boldsymbol{\theta}(t))$ is the rotation matrix from global frame to robot body frame, $\mathbf{b}$ is a 3d vector which represents the size of the bounding box.

Since our Qingzhui robot applies a parallel mechanism [21], the motors that control the leg joints are all concentrated in the upper body, which means the mass of the legs is much more lighter than the robot's torso mass, single rigid body dynamics is suitable. At time t, for each ground reaction force $\mathbf{f}_i(t) \in \mathbb{R}^3$, the torque from CoM to the i_{th} foot is $\mathbf{r}(t) - \mathbf{p}_i(t)$, the dynamics model of the robot in global coordinate system is given by according to [22]

$$\begin{aligned} \ddot{\mathbf{r}}(t) &= \frac{\sum_{i=1}^{n} \mathbf{f}_i(t)}{m} - \mathbf{g} \\ \mathbf{I}\dot{\boldsymbol{\omega}}(t) + \boldsymbol{\omega}(t) \times \mathbf{I}\boldsymbol{\omega}(t) &= \sum_{i=1}^{n} \mathbf{f}_i(t) \times (\mathbf{r}(t) - \mathbf{p}_i(t)) \end{aligned} \quad (2)$$

in Eq(2) , m is the robot mass, $\mathbf{g}$ is the gravity acceleration, and n is the number of legs. $\boldsymbol{\omega}(t)$ is the angular velocity of the robot, which can be derived from body azimuth Euler angles $\boldsymbol{\theta}(t)$ and its derivative $\dot{\boldsymbol{\theta}}(t)$ by the method in [23]. $\mathbf{I} \in \mathbb{R}^{3\times3}$ is the rotational inertia of our robot in nominal standing posture.

For contact schedule constraint, 3-3 gait type for six-legged robot has 2 phase, swing and stance, and 2 group of legs, each consists of three non-adjacent legs. We denote the time segment set in which foot i is in stance phase as C_i, the phase duration of leg group i in the j_{th} phase as $\Delta T_{i,j}$, and the maximum number of steps leg i can take in the total time T as $n_{s,i}$. When foot i is contact with the ground, its position should not change; when foot i is in swing phase, its reaction force should be zero. Total duration constraint is implied in the contact schedule constraint, which means the total duration's sum should equal to the total time of a locomotion task.

$$\mathbf{p}_i(t \in C_i) = const.\ \mathbf{f}_i(t \notin C_i) = 0.\ \sum_{j=1}^{2n_{s,i}} \Delta T_{i,j} = T \quad (3)$$

For terrain constraint, assume the heightmap is $h_{map}(x, y)$, and let $\mathbf{p}_i^{xy}$ be a 2D vector which means the position of foot i in world frame's $x - y$ plane, p_i^z be the projection of foot i on z axis. The following equations define terrain constraint for foot i in swing phase and foot i in stance phase.

$$p_i^z(t \notin C_i) > h_{map}(\mathbf{p}_i^{xy}).\ \ p_i^z(t \in C_i) = h_{map}(\mathbf{p}_i^{xy}) \quad (4)$$

For foot i in stance phase, we can decompose the reaction force on the foot-end into 3 components along 3 axes, let their value be f_x, f_y and f_z. For f_z, we use f_{min} to enforce the foot is indeed contacting with the ground. And f_{max} is the maximum value that the leg can tolerate vertically. Friction is another constraint which is in $x - y$ plane. Let the friction coefficient between foot and the terrain be μ. The foot-end in stance phase should not slide on the ground. The stance force constraint can be described as:

$$\begin{aligned} f_{min} \leq & f_z \leq f_{max} \\ -\mu f_z \leq \pm f_x \leq \mu f_z, & \quad -\mu f_z \leq \pm f_y \leq \mu f_z \end{aligned} \quad (5)$$

Finally, the start and target position constraint:

$$\begin{aligned} [\mathbf{r}, \boldsymbol{\theta}]\,(t=0) = [\mathbf{r}_0, \boldsymbol{\theta}_0].\ \ \mathbf{p}_i(t=0) = \mathbf{p}_i(0) \\ \mathbf{r}(t=T) = \mathbf{r}_{goal} \end{aligned} \quad (6)$$

We formulate a nonlinear feasibility problem to find the trajectories of feet and torso with defined constraints (Eq.1-6). The problem can be solved with the help of TOWR [24].

Fig. 2. Different Topologies' Influence in Traversibility. When walking forward and going up stairs, bend-inside leg posture is safer than bend-outside. When getting over higher stairs, farther nominal footholds' positions is easier.

B. Trajectory Tracking

When BLC has the trajectories of the feet and body in Cartesian world frame which originated at current CoM's vertical projection on $x-y$ plane, we can use inverse kinematics to get a joint position sequence for each motor. Let $\mathbf{q}_i$ be the joint position vector of leg i, we have $\mathbf{q}_i = Invk(\mathbf{p}_i, \mathbf{r}, \boldsymbol{\theta})$ where $Invk$ is the inverse kinematics function in world frame. Then, BLC uses a PD controller to track the joint position sequence. Let the motor torque vector of leg i in the n_{th} point of the trajectory be $\boldsymbol{\tau}_{i,n}$, and $\boldsymbol{v}_{i,n}$ be the rate of $\mathbf{q}_i$ which can be calculate from the difference between $\mathbf{q}_{i,n}$ and $\mathbf{q}_{i,n+1}$, we have

$$\begin{aligned} \boldsymbol{v}_{i,n} &= (\mathbf{q}_{i,n+1} - \mathbf{q}_{i,n})/\Delta T \\ \boldsymbol{\tau}_{i,n} &= K_p(\mathbf{q}_{i,n} - \mathbf{q}'_{i,n}) + K_d(\boldsymbol{v}_{i,n} - \boldsymbol{v}'_{i,n}) + \boldsymbol{\tau}_i^{ext} \end{aligned} \tag{7}$$

where K_p and K_d are PD controller factors. $\mathbf{q}'_{i,n}$ and $\boldsymbol{v}'_{i,n}$ are real joint position and velocity measured by the encoders on the robot motors.$\boldsymbol{\tau}_i^{ext}$ is compensate torque due to gravity and Coriolis force for the leg.

C. Bridging the Gap between Sim and Real

Since our BLC and TLC are trained in simulation, modeling in simulation causes a gap between sim and real. Torque control policy trained in sim can hardly achieve the same good result in real robot control. Unlike WBC and learning based control methods which focus on the calculation of joint torques, we focus on the trajectories of feet and torso. Thus, as long as accurate kinematics modeling is performed for the robot in the simulation, the joints' positions, which are the same both in sim and real, can be obtained using the inverse kinematics method. Then we can use II-B to trace the trajectories both in sim and real but with different control parameters.

D. Definition of Robot Topology

Different robot postures and nominal footholds distribution will influence the traverse capability of the legged robot. Take our Qingzhui six-legged robot for example, in Fig. 2, the left 2 pictures show that in bend-outside leg posture, the robot's calf is easier to collide with the terrain. While in bend-inside leg posture, the situation is improved significantly. And the right 2 pictures show that longer distance between front and hind nominal footholds' positions provides stronger stability since it has larger contact supporting polygon.

We define that legged robots have different topologies if their bending direction of legs or nominal footholds distribution are different. In the BLC of CapPlanner, we consider 6 different topologies shown in Fig.3

E. TLC that Learns the Capability of BLC-robot System

BLC of CapPlanner leads the robot reach its target in local terrain, however, on tough terrain, the NLP optimizer can not give long-way feasible trajectory in time due to computational complexity. To solve this problem, we designed a

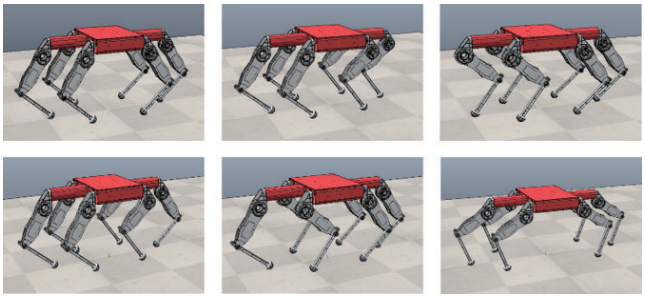


Fig. 3. The Topologies of Qingzhui Robot. The topologies from left to right are Nominal, Hind-Bend, Front-Bend, Hind-1-Bend, Front-1-Bend,Split-Up

TLC that can lead the BLC to its destination by doing global trajectory guidance. Unlike other path planning method, CapPlanner's TLC first learns the traverse capability of the haxapod robot in different topologies when the robot walking through various terrains in simulation under BLC's control. The learned capability can be represented by a transition probability model which gives successful BLC mission probability of a certain state. Then, the TLC uses the learned transition probability model and value iteration method to obtain a global guidance trajectory for the robot and decompose the global trajectory into segments that meet the capability of BLC-robot system.

1) Learning Based Transition Probability Model: The transition probability model describes the successful rate for the robot using a certain action to reach the local target in a certain topology. To simplify the action space, we let a grid world cover the global terrain map, then we define the action as walking to an adjacent grid (we set the target in the middle of the grid) in a certain topology. So the action space is walking to gird 1-8 in Fig.4 with 6 topologies in Fig.3. Whether the robot can successfully walk into an adjacent grid is related to its local terrain, initial feet positions and its body topology. Therefore, we generating samples in simulation where TLC records current terrain information by discretely sample the local heightmap in body frame. To reduce the complexity of TLC the training process, we divide the surrounding 8 grids into 2 groups, as is shown in Fig 4, grid 1,3,6,8 form the diagonal group, grid 2,4,5,7 form the orthogonal group. If we limit the heading direction of the robot be multiples of $45°$, the transition between any of the two groups is self-turning in place. We only need to pick one gird from each group to generate samples in simulation and train a CNN for each group to predict the success rate of BLC mission to that grid in a certain circumstance. As a result, a CNN to predict the success rate of *spin in place* which can be the link between actions is also needed. So, there are 3 CNNs for each topology that need to be trained:

- $O(M_O, p)$ for the orthogonal group, where M_O is the terrain information for the orthogonal group, p is the initial feet positions. $O(M_O, p)$ represents the successful rate for BLC-robot system walking to an adjacent grid in the orthogonal group.
- $D(M_D, p)$ for the diagonal group, where M_D is the terrain information for the diagonal group. $D(M_D, p)$ represents the successful rate for BLC-robot system walking to an adjacent grid in the diagonal group.
- $S(M_S, d)$ for self-turning in place, where M_S is the terrain information for self-turning in place. Here we assume the feet positions in body frame are the same

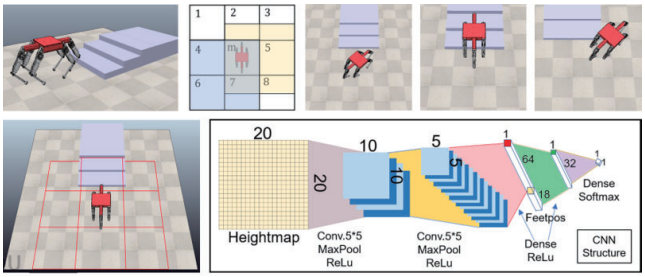


Fig. 4. TLC Local Grids and Ability Network Structure. When generating locomotion samples in simulation, local terrain information and feet positions are recorded as network input. The actions are going to adjacent grid 1-8. For each topology, we train 3 CNNs for orthogonal, diagonal grids and turn $45°$ in space. These CNNs can act as transition probability model in value iteration global path planning.

before and after self-turning. $S(M_S, d)$ represents the successful rate for BLC-robot system rotate $45°$ around z axis in world frame, and $d \in \{0, 1\}$ is the turning direction. Since turning $45°$ is a bridge between the robot's current heading direction and the desired grid of BLC's mission. For example, as shown in Fig 4, the robot is heading to grid 2, if the target grid is 5, the transition probability is

$$T = S(M_S, 1)S(M'_S, 1)O(M_O, p) \tag{8}$$

For a certain target grid, the terrain information can also be reduced. In Fig 4, if the robot's target grid is 5, it is enough to use the terrain information of light yellow region as M_O. And it is the same for diagonal group, if the robot's target grid is 6, the light blue region can also serve as M_D, since the other terrain information is useless for this BLC mission. For M_S, since the self-turning mission only needs the terrain information in grid m, we adopt it as M_S. To ensure the same network input shape, we increase the density of sampling in grid m. The structure of CNNs is shown in Fig.4. The input includes two heads. The terrain information are matrices of 20×20 corresponding to reduced heightmaps. The other head is initial feet position and adding one entry of direction for turning. We use two convolutional layers with four 5×5 kernels with appropriate zero-padding to extract terrain features. Both convolution are followed with maxpooling and ReLu. Then, the flattened features go through dense layer and concatenated with the other head of input. Together, these data go through 2 dense layer and get a rate ranged 0-1 indicating the successful rate of given state. After generating plenty of samples of BLC tasks, whether they are successful or failed, they can be well used to train CNNs. When the training is over, we will get 18 different CNNs for 6 topologies, which serves as the transition probability model with a 6×8 dimensions' action space.

2) Global path planning by value iteration with transition probability model: With the learned transition probability model, we can use the value iteration (VI) method [25] of reinforcement learning to do global path planning for our hexapod robot considering its traverse capability. When a robot is looking for a target in a grid world, its searching process can be modeled as a discrete Markov decision process (MDP) [26]. We use the grid which the robot is in as the state, denoted by s. Therefore, when the robot moves, the state of the robot will change, which conforms to Markov property. The action a indicates in which topology and to which adjacent grids the robot walks. The terrain

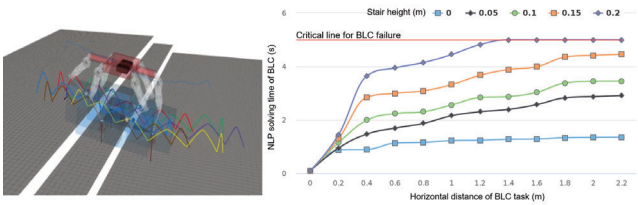


Fig. 5. Feet and body trajectory given by BLC. The figure shows the NLP solving time varies in different task distance and terrain difficulty.

information will affect the policy of the robot. For example, when the robot faces a wall, the robot cannot walk into the adjacent grid blocked by the wall, which means this action will fail. At the same time, the robot will receive a negative reward. When the robot reaches its destination, the robot will receive a large positive reward. In this way, we can calculate the utility of each grid, which is the accumulated value of the reward. Let the utility of a state (which grid the robot is in) be $V(s)$, the expected utility of taking some action in a certain state be $Q(s, a)$, the n_{th} iteration of them are denoted as $V_n(s)$,$Q_n(s, a)$, and the policy for a state be $\pi(s)$. In VI algorithm, for utility and Q value, we have:

$$\begin{aligned} V_{n+1}(s) &= \max_a Q_n(s, a) \\ Q_n(s, a) &= \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V_n(s')] \end{aligned} \tag{9}$$

where $T(s, a, s')$ is learned transition probability in II-E.1, which means the successful rate of robot from state s, through action a, get to state s'. And $R(s, a, s')$ the reward of robot in state s taking action a to state s'. γ is a discount factor, because the grids closer to the destination should have a greater utility value. By introducing an discount factor, the utility of different grids can be clearly differentiated according to the distance of the grid from the destination. The utility V_n in value iteration algorithm will converge to its optimal value V^* as $n \to \infty$, and we can obtain the optimal policy for each grid through policy extraction method:

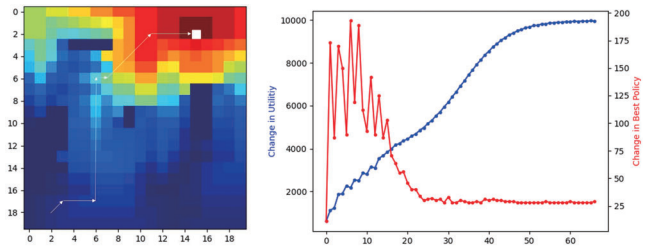
$$\pi^*(s) = \arg\max_a Q_\infty(s, a) \tag{10}$$

In TLC VI global trajectory planning part, the most important component is transition probability $T(s, a, s')$. In our TLC, $T(s, a, s')$ is not a constant probability kernel which is used in ordinary VI method, our $T(s, a, s')$ is obtained from the 18 CNNs we have trained in II-E.1. As a result, our TLC VI trajectory planning is dynamic and fit the current terrain and the traverse capability of our hexapod robot.

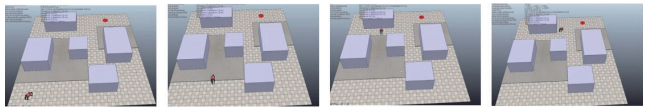
III. Experiments and Results

A. Non-linear Programming Based BLC

Given the local terrain information, initial feet' positions and local target position, BLC can obtain the trajectory of feet and body by solving a nonlinear feasibility problem described in II-A. To examine BLC's performance, we did experiments about the BLC on an Inter Core i7/2.8GHz 12-core laptop. The scene is placing two stairs of 0.4m's width at 0.5m in front of Qingzhui. We change the height of stairs and the target distance, and let BLC do its task. In order to ensure the efficiency of BLC, we set an upper limit of 5s solving time for it, exceeding which the BLC task will be regarded as failure immediately. The BLC result and solving time of different situation are shown in Fig.5. We



(a) Utility map of VI result, extracted guidance trajectory and VI process



(b) Long-range locomotion of Qingzhui in simulation

Fig. 6. Complete locomotion process with CapPlanner in simulation. Different colors in the utility map of VI result represent the utilities of different grids, the warmer, the higher. After around 60 iterations, VI converges, and we can extract guidance trajectory for BLC. With CapPlanner, our Qingzhui hexapod robot successfully reached its long-range goal.

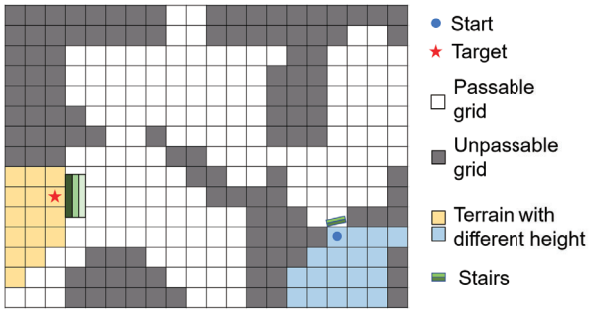
can see that whether BLC can successfully solve its task is relevant to the target distance and terrain complexity. The more complex the terrain is, the smaller range that BLC can successfully solve.

B. TLC that Learns the Capability of BLC-robot System

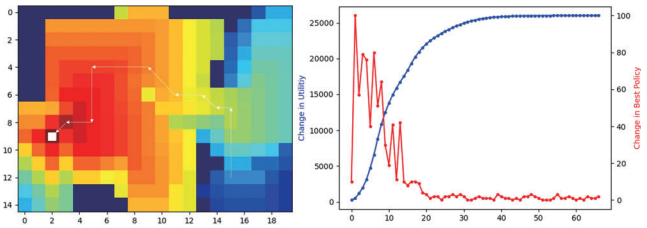
In simulation, we create 700 different scenes with regular terrains including flat ground; slops with different gradient; stairs with different number, length, height, and width; gaps with different depth, length, width, number and shape. And 300 different scenes with irregular terrains such as sine and cosine shaped terrains, random roughness terrains and terrains with random obstacles. Then we set 72 (360/5) different initial heading directions of our robot in these scenes, so we get 72000 samples to train each of our 3 CNNs described in II-E.1. For the input of terrain information, as shown in Fig 4, we sample the terrain heightmap of 9 local grids of $1mm$ around the robot, the sample interval for the orthogonal and diagonal groups is 10cm, and 5cm for self-turning. Therefore, M_O, M_D and M_S are all 20×20, matching the input format required by CNNs. Then, we simulate the samples of BLC tasks in 6 robot topologies mentioned above, and save terrain information, initial feet positions and the results of BLC task for our ability network training. We use MSE-loss in our CNNs' training. After training the 18 CNNs, we have obtained the transition probability model that can be used in value iteration for global trajectory planning. Our CNN can give prediction of success rate in a certain situation in 0.013 ms, so that our transition probability model can give TLC the success rate of each action of a grid in no more than 1.248 ms, which is fast enough for VI.

Since the transition probability model is learned, the TLC can use the model to do global path planning by VI method. We create a long-range locomotion task for our Qingzhui hexapod robot in simulation, where the scene is $20m \times 20m$. We deploy our CapPlanner on our robot to find its target. After around 60 iterations in 23.47s , the result shows convergence in Fig 6(a).

Finally, the TLC split the guidance trajectory to many local targets suitable for the ability of BLC-robot system. And at the start of each BLC mission, the next BLC mission is given by TLC according to current robot position and



(a) Environment mark and meshing for the real robot experiment



(b) Utility map of VI result, extracted guidance trajectory and VI process

Fig. 7. CapPlanner for real robot experiment.

Fig. 8. Video screenshots of Qingzhui long-range locomotion under CapPlanner's control

expected end position of current BLC mission, which will give BLC enough time to calculate trajectories and also avoid the accumulation of position errors of the robot. We finished the global target search task in simulation, the results are shown in Fig 6(b).

To demonstrate the advantages of our TLC with the learned ability, we conduct a comparison experiment. Terrains with stairs, obstacles and both are created in simulation, and the complexity are distinguished by simple, medium and hard, which have corresponding stair width, height and obstacle density. Then, we give Qingzhui 20 locomotion tasks in these terrains under 2 controllers. They share the same BLC of CapPlanner, but different TLC. One use A*, and the other use our TLC of CapPlanner. We analyze the success rate and the distance ratio of successful tasks. In Fig 9, it can be concluded that TLC that learned the ability of BLC-robot system can achieve much higher success rate in medium and hard terrains. And the displacement ratio result also shows that our robot with CapPlanner can pass obstacles when they are easy to across so that saves displacement, and get rid of obstacles beyond its traverse ability, gaining much higher success rate.

C. Real Robot Experiment

We deploy our CapPlanner on Qingzhui hexapod robot. First, we give it a long-range locomotion task which includes crossing terrain of different heights, walking up and down stairs, passing narrow path and avoiding obstacles. We

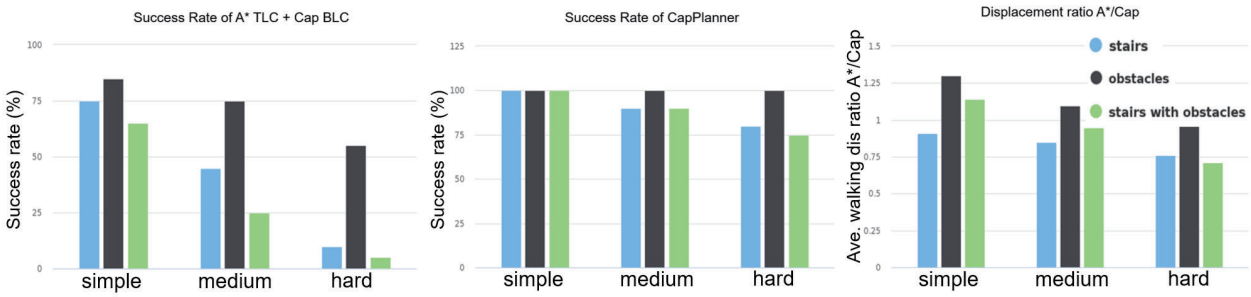


Fig. 9. Comparison result of different TLCs

use a HESAI Pandar 64P Lidar to map the environment and position the robot with LeGO-LOAM [27]. After the terrain information is mapped, we define the start position and destination for our robot and mesh the terrain(Fig. 7(a)). The TLC gives guidance trajectory (Fig.7(b)) according to the ability model it learned and controls the BLC use suitable topology in separated local tasks. Finally, CapPlanner lead Qingzhui successfully reach its destination (Fig.8). And Qingzhui will change topology (first three figs in Fig.8) to better adapt to tough terrain.

IV. Conclusion

In this paper, we propose CapPlanner, a learning based hierarchical control strategy for hexapod robots, which is adaptable to topology and locomotion capability of hexapod robots. CapPlanner has 2 layers, where BLC uses non-linear feasibility optimization to find feet and body trajectory given terrain information and local target as well as hold the robot balance. TLC uses learning based method to get the transition probability of robot in different topology and terrains which represents the ability of BLC-robot system. Then, it uses the learned model and value iteration method to find global trajectory and send suitable command to BLC to finish global locomotion task. CapPlanner reduces the problem that the hexapod's trajectory optimization is hard to complement in long-way task. And our method is suitable for any bottom-level controller due to the hierarchical design. The success of our method is verified by experimental results both in simulation and real robot.

References

[1] Chencheng Dong, Chen Xuechao, and Yu Zhangguo. A novel hierarchical control strategy for biped robot walking on uneven terrain. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2019.

[2] Jianwen Luo, Su Yao, and Ruan Lecheng. Robust bipedal locomotion based on a hierarchical control structure. *Robotica*, 37(10):1750–1767, 2019.

[3] Farbod Farshidian, Edo Jelavić, Alexander W Winkler, and Jonas Buchli. Robust whole-body motion control of legged robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4589–4596. IEEE, 2017.

[4] Hae-Won Park, Patrick M Wensing, and Sangbae Kim. High-speed bounding with the mit cheetah 2: Control design and experiments. *The International Journal of Robotics Research*, 36(2):167–192, 2017.

[5] Jared Di Carlo, Patrick M Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1–9. IEEE, 2018.

[6] Gerardo Bledt, Matthew J Powell, Benjamin Katz, Jared Di Carlo, Patrick M Wensing, and Sangbae Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252. IEEE, 2018.

[7] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 international conference on robotics and automation (ICRA)*, pages 6295–6301. IEEE, 2019.

[8] Donghyun Kim, Jared Di Carlo, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *arXiv preprint arXiv:1909.06586*, 2019.

[9] Alexander W Winkler, C Dario Bellicoso, Marco Hutter, and Jonas Buchli. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, 2018.

[10] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019.

[11] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47), 2020.

[12] Boon-Chong Seet, Genping Liu, Bu-Sung Lee, Chuan-Heng Foh, Kai-Juan Wong, and Keok-Kee Lee. A-star: A mobile ad hoc routing strategy for metropolis vehicular communications. In *International conference on research in networking*, pages 989–999. Springer, 2004.

[13] Alex Nash, Kenny Daniel, Sven Koenig, and Ariel Felner. Thetaˆ*: Any-angle path planning on grids. In *AAAI*, volume 7, pages 1177–1183, 2007.

[14] Daniel Harabor and Alban Grastien. Online graph pruning for pathfinding on grid maps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 1114–1119, 2011.

[15] František Duchoň, Andrej Babinec, Martin Kajan, Peter Beňo, Martin Florek, Tomáš Fico, and Ladislav Jurišica. Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 96:59–69, 2014.

[16] Roland Geraerts and Mark H Overmars. A comparative study of probabilistic roadmap planners. In *Algorithmic foundations of robotics V*, pages 43–57. Springer, 2004.

[17] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.

[18] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[19] Iram Noreen, Amna Khan, and Zulfiqar Habib. Optimal path planning using rrt* based approaches: a survey and future directions. *International Journal of Advanced Computer Science and Applications*, 7(11), 2016.

[20] Jonathan D Gammell, Timothy D Barfoot, and Siddhartha S Srinivasa. Batch informed trees (bit*): Informed asymptotically optimal anytime search. *The International Journal of Robotics Research*, 39(5):543–567, 2020.

[21] Yue Zhao, Feng Gao, and Yunpeng Yin. Obstacle avoidance and terrain identification for a hexapod robot. 2020.

[22] Kevin M Lynch and Frank C Park. *Modern robotics*. Cambridge University Press, 2017.

[23] Michael Bloesch Hannes Sommer Peter Fankhauser Marco Hutter Roland Siegwart Christian Gehring, C. Dario Bellicoso. Kindr library – kinematics and dynamics for robotics, 2016.

[24] Alexander W Winkler. Towr–an open-source trajecetory optimizer for legged robots in c++, 2018.

[25] Krishnendu Chatterjee and Thomas A Henzinger. Value iteration. In *25 years of model checking*, pages 107–138. Springer, 2008.

[26] Dimitri P Bertsekas. *Dynamic programming and optimal control 3rd edition*. Belmont, MA: Athena Scientific, 2011.

[27] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.