Improved Performance of CPG Parameter Inference for Path-following Control of Legged Robots

Nathan D. Kent[1], David Neiman[2], Matthew Travers[2], and Thomas M. Howard[1]

***Abstract*— The difficulty associated with the coordinated locomotion of legged robots grows quickly as the number of joints increases. Although prior approaches have addressed this problem through sampling-based planners, learning-based techniques have recently been explored as a means to handle such complexity. Among these recent approaches are systems that utilize probabilistic graphical models in order to infer parameters for central pattern generators (CPGs) which enable the path-following locomotion of highly-articulated legged robots through unstructured terrain. This paper presents a novel formulation of a CPG parameter inference-based path-following controller. The new inference process and accompanying CPG formulation enforce oscillator convergence to the limit-cycle specified by the inferred parameters in addition to biasing towards parameters that quickly reach stable-state. This formulation is shown to improve the performance of CPG parameter inference-based path-following control for legged robots across a number of simulated and physical experiments.**

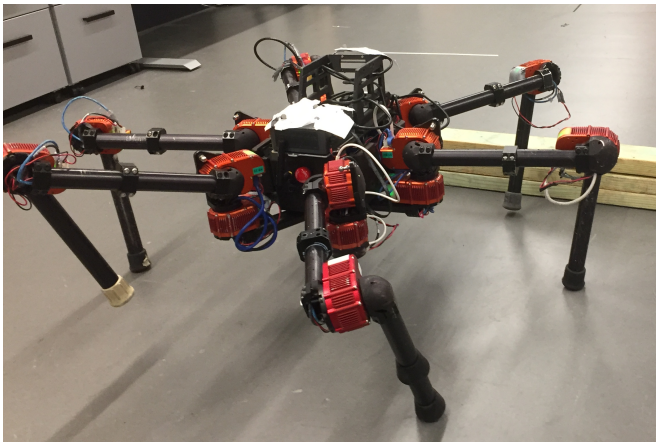


Fig. 1: The Hebi Robotics Daisy hexapod robot used for both the simulated and physical experiments. When standing, this robot is approximately $1.1\,\mathrm{m}$ by $1.1\,\mathrm{m}$.

I. Introduction

Highly-articulated legged robots, such as the robot in Fig. 1, are particularly well suited for traversing cluttered and unstructured terrain. However, the high-dimensional configuration spaces of these robots significantly increase the complexity of the joint and limb coordination required for locomotion. One studied approach for overcoming this challenge is to utilize sampling-based planning methods to generate motion plans in high-dimensional spaces in order to guide feedback controllers [1, 2]. Unfortunately, the use of these sampling techniques requires a compromise between exhaustively searching the state-action space to find the optimal motion plan versus the ability to quickly react to changes in the state of the robot, the environment, or the required locomotive behavior. By using a dense sampling strategy, the motion planner is able to reason over a wider variety of behaviors but limits the rate at which the robot is able to respond to changes. A more sparse sampling strategy reduces the amount of time required to generate a motion plan but is limited in the portion of the state-action space in which it can search for a solution, likely resulting in the selection of a sub-optimal action.

Recently, machine learning techniques have become a common solution to handling the complexity of locomotion in legged robots [3–5]. One class of these methods has been developed which encodes information about the environment, robot kinematics, and motion commands within a probabilistic graphical model (PGM) [6, 7]. Once encoded, information about the desired motion commands can be fed to the inference process via a path-following controller along with information concerning the terrain from an on-board depth camera. This information is then used to efficiently search the space of parameterized central pattern generators (CPGs) which represent coordinated walking behaviors and enable the robot to locomote along the specified path. Inspired by both earlier gait-controlled methods of controlling legged robots [8] and work in PGMs for grounded language communication [9–11], these methods exploit conditional independence assumptions and information gleaned from the robot's kinematic model for efficient search in a high-dimensional space.

However, a limitation of these earlier works is that the current positions of all robot limbs are not considered when selecting a new set of CPG parameters, which leaves the potential to infer parameters that may delay or prevent the convergence of the oscillators. If such a behavior is used for path-following control that replans at a fixed rate, such as in the work of Kent et al. [7], the selected parameters may not ever allow the CPG to converge to the desired behavior before new parameters have been inferred, resulting in undesirable expressed behaviors. Thus, the work pre-

*This work was supported by the National Science Foundation under grants IIS-1724000 and IIS-1723972.

[1] Nathan D. Kent and Thomas M. Howard are with the Robotics and Artificial Intelligence Laboratory, University of Rochester, Joseph C Wilson Blvd. Rochester, NY 14627, USA. nkent2@cs.rochester.edu, thoward@ece.rochester.edu

[2] David Neiman and Matthew Travers are with the Biorobotics Lab, Carnegie Mellon University, 5000 Forbes ave, Pittsburgh, PA 15213, USA. <dneiman, mtravers>@andrew.cmu.edu

sented herein is an improvement to CPG parameter-based path-following control for highly-articulated robots [7] by enforcing that inferred parameters produce consistent and predictable behaviors regardless of the relative positions of the robot limbs.

This is achieved with two modifications. First, the CPG formulation is modified to ensure that the expressed relative positions of the legs at stable-state matches the desired relative positions regardless of legs' initial states while also enforcing that the expressed step shape is identical to the desired step shape. Second, the inference process is modified to search over only the set of CPG parameters that can quickly reach stable-state from the current positions of the limbs. The proposed closed-loop controller is then compared against the controller found in Kent et al. [7] by comparing the performance of a simulated robot when traveling through a series of complex, unstructured environments. Additionally, a set of experiments are also performed on a physical robot.

Simulation results show a performance improvement compared to the previous framework both on flat and non-flat (a flat environment ridden with rigid obstacles) terrain, where the robot is following predefined paths. The results of the physical experiments show that framework is applicable to physical robots as well as simulated, which was not previously shown. Thus, the work presented here represents an improvement in the locomotion performance exhibited by robots following the framework specified in Kent et al. [7] both in terms of path accuracy and the ability to traverse cluttered terrain.

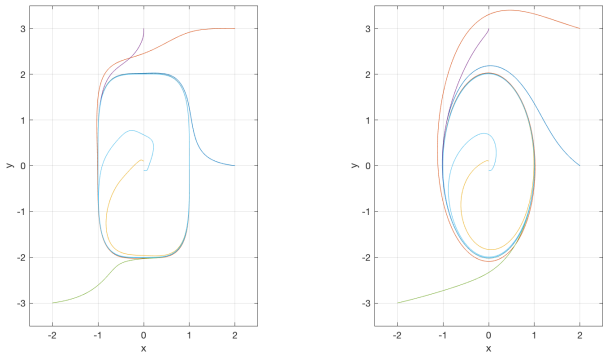
The remainder of this paper is organized as follows. Section II contains background information on CPGs with a technical overview of the CPG used within this work. In Section III, overviews of controller and the inference process are provided, detailing the proposed improvements to the path-following controller. Section IV specifies the experimental setup used to compare this work against the previous work as well as the experiments on the physical system, with Section V providing comparative experimental results. Finally, Section VI details the current limitations and future work.

II. Central Pattern Generator Modeling

CPGs are biological neural circuits that produce rhythmic output without requiring rhythmic input, though they can be modulated by various inputs in order to modify the output. These circuits are pervasive in animals and are the networks that drive many of the cyclical activities done by these animals, such as breathing, walking, and chewing. Inspired by these biological examples, various models of CPGs have been developed for robot locomotion [12–14], most often in the form of coupled oscillators with various parameters modulating the resulting limit cycles. However, the ability to transition between sets of CPG parameters while maintaining effective movement is still an active area of research which largely focuses on transitioning between a small set of well-known gaits [15]. This work thus presents a new CPG formulation inspired by the CPG models presented by Sartoretti et al. [16] and Kent et al. [7], with modifications to allow for more predictable stable-state behavior when transitioning between CPG parameters which vary both gait and limit cycle. As with the previous works, this work assumes that the integral curve associated with the i^{th} oscillator's dynamics are representative of the position of the robot's i^{th} end-effector and is constrained to be super-elliptical, such that the trajectory is defined by

$$H_i(x,y) = \left|\frac{x}{a_i}\right|^{d_i} + \left|\frac{y}{b_i}\right|^{d_i} = 1 \tag{1}$$

where a_i and b_i are the lengths of the semi-major/minor axes of the limit super-ellipse and d_i specifies the shape of the limit cycle (as seen in Fig. 2). These curves are then transformed to a coordinate frame that is fixed relative to the robot's base coordinate fame.



(a) Rectangle step shape. (b) Ellipse step shape.

Fig. 2: Comparison between CPG oscillators with various starting positions for rectangular and elliptical shapes. The elliptical step has been found to be more efficient on smooth terrains while the rectangular step is less prone to becoming stuck.

These oscillators are quantified in terms of the relative phase differences such that every point in $\mathbb{R}^2$ is assigned a phase corresponding to its position on the super-ellipse concentric and belonging to the same family as $H_i(x,y) = 1$. The absolute phase of the oscillator θ_i is defined as the four-quadrant tangent inverse of the point $f_i(x_i, y_i)$, where f is a function that maps the current position of the oscillator to the unit circle:

$$f_i(x,y) = \begin{bmatrix} \text{sign}(x)\left|\frac{x}{ra_i}\right|^{\frac{d_i}{2}} \\ \text{sign}(y)\left|\frac{y}{rb_i}\right|^{\frac{d_i}{2}} \end{bmatrix} \tag{2}$$

where

$$r = \sqrt[d_i]{\left|\frac{x}{a_i}\right|^{d_i} + \left|\frac{y}{b_i}\right|^{d_i}} \tag{3}$$

In the prior work, desired coupling was specified as phase differences between adjacent limbs and was enforced by driving the oscillators directly to the position with the desired offset. However, directly applying this penalty term to the equations governing the oscillation can result in expressed

behaviors that are unexpected when considering the parameters. Specifically, the penalty term can result in expressed super ellipses with shapes that do not correspond to the stated values of a_i and b_i. An example of this can be seen in Fig. 3 and is compared to the formulation found below in Eq. (4). Additionally, coupling all of the oscillators together potentially results parameters that have multiple possible stable states depending on the initial positions of the oscillators. To account for these issues, the CPG formulation here achieves robust coupling between limbs via modulating the tangential velocity of the oscillator along the super ellipse. Defining k_i to be the desired phase difference between oscillators i and 1 (with $k_1 = 0$), the differential equations governing the dynamics are modeled as

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} = M_i(k_i + \theta_1 - \theta_i)\omega_i \hat{t}_i + \gamma_i(1 - H_i(x_i, y_i))\hat{n}_i \quad (4)$$

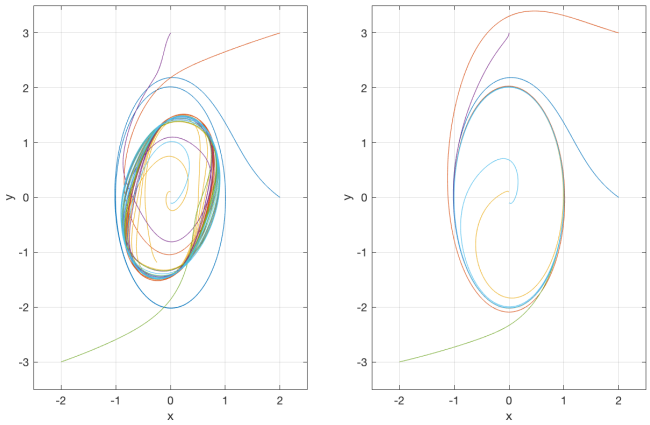
where $\hat{n}_i$ is the unit vector in the normal direction relative to the super ellipse and $\hat{t}$ is the unit vector tangential to the super ellipse. Additionally, ω_i is proportional to oscillation frequency, γ_i is proportional to the speed at which an oscillator converges to the limit cycle, and M_i is a function such that the following hold:

$$\begin{aligned} M_i(\theta) &> 1 \quad \text{if } \theta > 0 \\ M_i(\theta) &< 1 \quad \text{if } \theta < 0 \end{aligned} \quad (5)$$

This formulation has two major advantages when compared to the formulation present in Kent et al. [7] which result in more consistent stable-state behavior of the CPG regardless of the initial state of the oscillators. The primary change is that oscillator coupling is created by modulating the angular velocity of the oscillators rather than via an additive penalty vector, meaning that the expressed shape of the super ellipse is exactly defined by a_i and b_i. The second advantage is that reasoning about the relative offsets for the oscillators only requires reasoning about the behavior of M_i rather than the initial state of the system.

III. CPG Parameter Inference

The inference process utilized in this work is an extension of the frameworks described in Chavali et al. [6] and Kent et al. [7] to infer the optimal set of ten parameters $\mathcal{P}^*$ in the space of possible CPG parameters $\mathcal{P}$. Specifically, we utilize the block diagram shown in Fig. 4 for path-following control of legged robots. This controller utilizes a learned model of the CPG parameters that depend on estimates of the robot's environment and desired behavior. Terrain information is estimated by compressing information from an on-board RGB-D camera into a parametric representation of terrain roughness and path-following information is provided using an estimate of the robot's position (x, y, ψ) and attempts to follow a path that is either defined a priori or continuously updated by a path planning module. The CPG parameter inference estimates the most likely CPG parameters $\mathcal{P}^*$ from these estimations using the PGM described below, which are then used to integrate Eq. (4) and provide desired positions for the robot's end-effectors.



(a) Converging trend of the formulation from Kent et al. [7].

(b) Converging trend of Eq. (4).

Fig. 3: Examples of converging behavior of the CPG formulations. Oscillators are in an "alternating tripod" configuration, where oscillators 3 and 5 are in-phase with oscillator 1 and oscillators 2, 4, and 6 are 180° out-of-phase with oscillator 1. For all oscillators, $a = 1$ and $b = 2$. Note that Fig. 3a does not readily stay on the specified super ellipse.

For the hexapod robot utilized in this work, we assume a space of parameters which include the values the shape of the step (d), the angular offsets between the first limb and the successive limbs ($k_{2\ldots6}$), the width of the left and right steps ($a_{1,2}$), the height of all steps (b), and the height of the robot's chassis from the ground (h). This framework exploits information from the robot's kinematic model for efficient search over $\mathcal{P}$ and treats the inference process as a search over a factor graph, which is completed at a fixed rate in order to enable path-following control. Specifically, the inference process is formulated as the search for the CPG parameters that are most likely to be optimal given the behavior and environment:

$$\mathcal{P}^* = \underset{d,k_i,a_1,a_2,b,h}{\arg\max}\; p(h)p(a_2)p(a_1)p(b)p(d)\prod_{i=2}^{6} p(k_i) \quad (6)$$

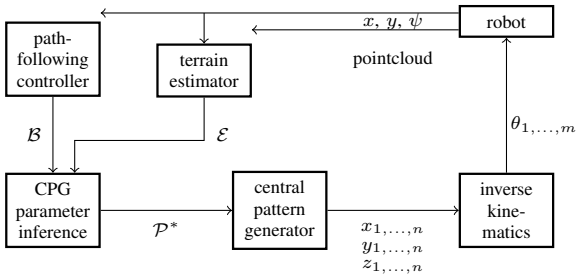


Fig. 4: The block diagram for path-following control using CPG parameter inference. This is based on the work described in Kent et al. [7].

However, the approach used here deviates from the framework described in Kent et al. [7] in two major ways. First,

CPG parameters that will not reach a stable state within a fixed time are removed from the search process and those that do converge within that time constraint are scored by a linear combination of probability and convergence time. The result of this additional step is that the expressed behavior of the CPG will match the behavior indicated by the inferred parameters rather than the behavior expressed during the converging phase. This is expressed as a function $s(\mathcal{P})$, which filters the CPG parameters from Eq. (6) that do not converge within the time constraint and weigh those that do by a combination of the time required to reach stable-state and their probabilities. Second, the curvature parameter of the super ellipse is also inferred, allowing for the system to select between an ellipse step shape and a rectangle step shape, with the differences in these shapes seen in Fig. 2. The work of Sartoretti et al. [16] found that a elliptical step was more efficient on smooth surfaces while a rectangular step was less prone to becoming stuck in cluttered terrains, and the modification to the inference process here allows the system to decide between the two as appropriate.

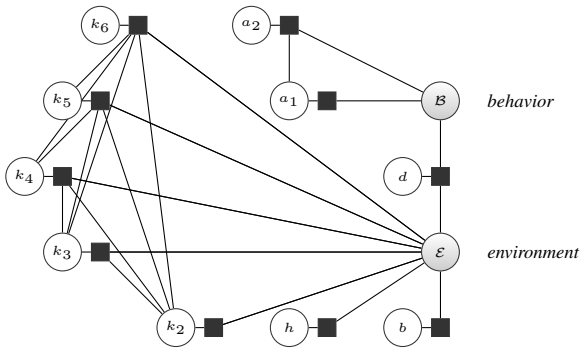


Fig. 5: An illustration of the factor graph used to infer a distribution of parameters for this CPG model. The model exploits conditional independence assumptions to efficiently infer the most likely set of CPG parameter values $k_{2\ldots6}, a_{1,2}, d, b, h$ from models of the environment ($\mathcal{E}$) and behavior ($\mathcal{B}$). Note that the robot's model $\mathcal{M}$ is implicit within the inference process as only a single robot model is considered in the experiments presented in Section IV. Additionally, the value of k_1 is specified as $0\,\mathrm{rad}$.

These expressions are simplified by assuming that the probability of behavior $\mathcal{B}$ is conditionally independent from the probabilities of the environment model $\mathcal{E}$. Just as in Kent et al. [7] the environment models and behaviors are integrated at latent variables, permitting a distribution of these values to be used during inference when estimation of these values is uncertain. In order to simplify the inference procedure, it is assumed that the probability of the behavior $\mathcal{B}$ is conditionally independent from the probability environment model $\mathcal{E}$. Additionally, other assumptions of conditional independence between parameters are made, with the factor graph for this system being illustrated in Fig. 5 and the factors in this graph being approximated using neural networks.

In order to make the inference process tractable, a beam search is used to efficiently generate a distribution of effective CPG parameter sets. These sets are then passed on to the path-following controller which prioritizes the CPG parameter sets based on a linear combination of the likelihood of the sets and the amount of time required for those sets to converge. The highest scoring set of parameters is then executed until the next iteration of the inference process.

IV. Experimental Design

In order to evaluate the performance of the model, a set of simulated experiments identical to those found in Kent et al. [7] were performed within the Gazebo simulator [17]. Additionally, a set of physical experiments were completed in order to demonstrate that the model can be realized on a complex physical system. For both sets of experiments, a mapping of environment and behavior to ideal CPG parameters was generated using a genetic algorithm process. The environment was modeled as the average obstacle height for obstacles within $0.5\,\mathrm{m}$ of the robot and the heights used to create these mappings were one of $0.0\,\mathrm{m}$ (flat ground), $0.01\,\mathrm{m}$, $0.025\,\mathrm{m}$, $0.035\,\mathrm{m}$, $0.05\,\mathrm{m}$, and $0.1\,\mathrm{m}$ with the obstacles being distributed across a simulated terrain using a Poisson Random Process (see Fig. 6). The behaviors included in the mapping consisted of locomotion with desired curvatures of $0\,\mathrm{m}^{-1}$, $0.2\,\mathrm{m}^{-1}$, $0.286\,\mathrm{m}^{-1}$, $0.4\,\mathrm{m}^{-1}$, and $1.0\,\mathrm{m}^{-1}$.

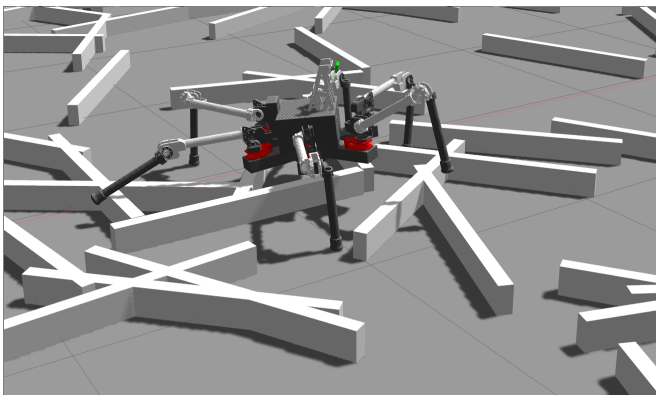


Fig. 6: Example of $0.1\,\mathrm{m}$ terrain and simulated robot used in both the genetic algorithm-based training and the simulated evaluation.

A base score for each individual within the genetic algorithm process was calculated as the line integral of the individual's path through the vector field caused by the normalization of $F(x, y)$ from Eq. (7), where c is the desired curvature value. Figure 7 contains an example vector field that is generated by Eq. (7) for the purpose of finding CPG parameters for turning with a radius of $2.5\,\mathrm{m}$. Unlike previous works, measures to prevent damage to the physical robot were taken in the form of reducing the base score based on whether the individual's CPG parameters resulted in either self-contact or contact between the robot's chassis and the terrain. When self-contact occurred between the robot's

legs, the score was reduced to 75% of the original score. Furthermore, the base score was reduced by the percentage of time the chassis was in contact with the terrain. E.g., an individual that spend 50% of its time in contact with the terrain received a score that was 50% of the base score. The highest scoring individuals at the end of the genetic algorithm process were added to the mapping.

$$\begin{aligned} \Phi(x,y) &= c^2x^2 + (cy-1)^2 \\ W_x(x,y) &= (1-\Phi(x,y))\frac{\partial \Phi}{\partial x}(x,y) \\ W_y(x,y) &= (1-\Phi(x,y))\frac{\partial \Phi}{\partial y}(x,y) \\ F(x,y) &= \left\langle \frac{\partial \Phi}{\partial y} + \frac{W_x(x,y)}{2c},\ -\frac{\partial \Phi}{\partial x} + \frac{W_y(x,y)}{2c} \right\rangle \end{aligned} \tag{7}$$

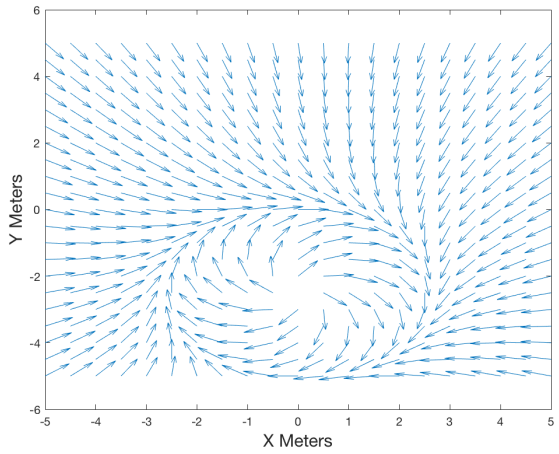


Fig. 7: Example vector field with $c = 0.4m^{-1}$ used for evaluating individuals while creating the mapping via genetic algorithm.

Using this method, approximately 200 data points were collected. As with Kent et al. [7], the mapping of environments and behaviors to probability distributions of CPG parameters was extracted from the data and used to train neural networks approximating the factors within the PGM. In contrast to the work in Kent et al. [7], these neural networks consisted of a single hidden layer of 10 units for factors involving $k_{2\ldots6}$ and no hidden units otherwise. These smaller networks were utilized as they were found to be sufficient to learn the required distributions.

Path following controllers for both the simulated and physical robots were evaluated on flat and non-flat terrain across predefined paths. Terrain information was fed into the system as a 2.5D heightmap generated using the on-board depth camera. The desired curvature for path-following was supplied through an adaptation of the Pure Pursuit algorithm [18] with path tracking performance being quantified as a function of the average distance between the desired path and the path that the robot executes (i.e., crosstrack error) calculated at a rate of 1 Hz. The CPG parameter inference process was run at a rate of 1 Hz and the selected set of parameters $\mathcal{P}$ were the candidate parameters that maximized $p(\mathcal{P}) + 0.1(2 - \mathcal{T}_\mathcal{P})$ where $\mathcal{T}_\mathcal{P}$ was the time required for the CPG to reach stable-state, as determined by repeated evaluation of Eq. (4).

For the simulated experiments, the non-flat terrain was generated using a Poisson Random Process, with the height of the obstacles increasing linearly from flat ground to $0.1\,\mathrm{m}$. Three different paths were used to evaluate the system with all evaluations repeated 20 times, resulting in a total 120 simulated experiments (3 paths $\times$ 2 terrains $\times$ 20 evaluations). In order to provide a baseline for analyzing the performance of these controllers, a non-inference based controller was designed, tuned, and evaluated on the same sets of paths and terrains for a total of 60 evaluations. This hand-crafted controller was utilized a fixed alternating tripod gait, an elliptical step shape, and held both the step height and body height (b and h, respectively) at fixed values of $0.2\,\mathrm{m}$, which was double the height of the largest obstacles in the experiment. Left and right step widths ($a_{1,2}$) were selected based on the desired curvature reported from the pure-pursuit algorithm. Specifically, $a_1 = 0.15\mathcal{B} + 0.05$ and $a_2 = 0.15(1-\mathcal{B}) + 0.05$ with both a_1 and a_2 being clamped to the range of $[0.05, 0.15]$ and $\mathcal{B}$ being the desired curvature.

The physical experiments utilized the robot seen in Fig. 1 with an attached forward-facing Intel Realsense camera. Pose information for this system was provided using a motion capture system that was able to observe an area of approximately $6\,\mathrm{m}$ by $15\,\mathrm{m}$. Two different paths through this space were used for evaluation and both flat and non-flat terrain was considered. For the non-flat terrain, wooden boards were placed in the path of the robot resulting in obstacles that were $0.1143\,\mathrm{m}$ tall and $0.0889\,\mathrm{m}$ wide. Notably, these obstacles were taller than those that had been used in the genetic algorithm-based training process. A total of 40 physical experiments were performed (2 paths $\times$ 2 terrains $\times$ 10 evaluations).

V. Experimental Results

A. Simulated Experiments

Following the procedure outlined in Section IV, the average crosstrack error was recorded for three predefined paths through flat and non-flat terrain, using the new CPG formulation and inference process. Tables Ia and Ib contain a performance comparison between the controller presented within this paper, the previous results found in Kent et al. [7], and the baseline controller described in the previous section. The expressed paths of all three controllers plotted against the desired paths are found within Tables III and IV, for flat and non-flat terrain respectively. As expected, the new CPG formulation and the inference process which takes into account the time required for the CPG to converge greatly improved the ability of the simulated robot to both remain on the desired path and to overcome obstacles within the terrain. In all cases, the proposed system was more successful at traversing the terrain and reaching the end of the path and, with the exception of a single combination of path and

	Path 1	Path 2	Path 3
Baseline	0.24 m (100)	0.22 m (100)	**0.38 m (100)**
Kent et al. [7]	0.35 m (100)	0.45 m (100)	0.49 m (100)
Proposed	**0.14 m (100)**	**0.12 m (100)**	0.61 m (100)

(a) Flat Terrain

	Path 1	Path 2	Path 3
Baseline	0.32 m (10)	0.33 m (0)	0.42 m (0)
Kent et al. [7]	0.81 m (50)	0.66 m (10)	0.58 m (0)
Proposed	**0.20 m (85)**	**0.22 m (60)**	**0.27 m (50)**

(b) Non-flat Terrain

TABLE I: Comparison of the average crosstrack error for both the controller described in this work, the controller described in Kent et al. [7], and a baseline controller with hand-tuned parameters. In parenthesis are the percentage of experimental runs that resulted in the robot successfully reaching the end of the pre-defined path.

terrain, the average crosstrack error was significantly reduced when compared to the previous work.

In the case of the Path 3 with no obstacles, we theorize that the CPG parameter filtering process described in Sections III and IV resulted in a system that was less able to respond to abrupt changes in the desired path when compared to the previous work. For Paths 1 and 2, the changes in the desired path were less abrupt and, in the case of Path 3 with obstacles, the improvement in the robot's ability to traverse obstacles was sufficient to see an improvement in the expressed path.

B. Physical Experiments

The average crosstrack error for the physical experiments can be found in Table II. Additionally, plots of the desired and expressed paths of the robot can be found in Fig. 8. As seen from the results, the effectiveness of the proposed system on a physical platform mirrors that of the larger scale simulated experiments. Based on the results of the baseline controller in simulation, the experiments on the physical platform were concentrated on the performance of the proposed controller.

	Path 1	Path 2
Flat	0.18 m (90)	0.27 m (90)
Non-flat	0.07 m (90)	0.11 m (100)

TABLE II: The average crosstrack error during physical experiments. In parentheses is the percentage of experimental runs in which the robot successfully reached the end of the pre-defined path. Issues with the actuators on the robot are believed to be the cause of the unexpected decrease in crosstrack error when obstacles were introduced. In parenthesis are the percentage of experimental runs which resulted in the robot successfully reaching the end of the pre-defined path.

In order to apply this approach on the physical robot within the chosen experimental space, certain modifications to the experimental process were made. Primarily, the calculation of the obstacle height was modified to accommodate the presence of furniture and walls in the room. Specifically, all detected obstacles higher than $0.3\,\mathrm{m}$ were assumed to be walls and not obstacles. We also found that there was significant noise in the generation of the 2.5D height map due to interference between the motion capture system and the depth camera. Additionally, significant encoder drift in the robot's actuator modules was an issue during these experiments, requiring that the robot be reset between each run. These encoder issues may have been exacerbated by self-collision, so as a preventative measure the step length values (a_1 and a_2) were capped at $0.15\,\mathrm{m}$, with an equal reduction on the uncapped value in order to preserve the desired angular velocity.

While the robot was successful at path-following in the majority of cases, there were a few instances (noticeable in the paths without obstacles) where we believe the encoder issues resulted in the robot not correctly following the commanded motions. The experiments with suspected encoder issues are the cause of the unexpected improvement in crosstrack error when obstacles are introduced.

VI. Conclusion

General solutions for safe and robust control of legged robots remains a difficult and unsolved problem in robotics. While learning-based approaches eliminates the need for hand-engineering of gaits and controlled behaviors, such models can produce novel, unforeseen motions as the CPG parameters change from joint configurations not observed in the training data, causing suboptimal, unpredictable, and potentially unsafe locomotion. This paper presents an advance in our ability to use PGM-based models for inferring CPG parameters for legged robots by modifying the formulation of the CPG to produce more predictable behaviors and by filtering results that exhibit slow convergence rates. Simulation results show an improvement over the system presented in Kent et al. [7] across a number of different path and terrain shapes while the novel physical experiments demonstrated that this process is effective on physical robots as well.

We recognize that there are several limitations to this approach. Primarily among them is the limitation caused by the usage of a genetic algorithm-based process for generating a mapping of behavior and environment to ideal CPG parameters. While this process works for environments like those in these experiments, it may prove difficult to scale it to more complex terrains and behaviors. Future work will use techniques described in recent works in reinforcement learning in order to directly train the neural networks within the PGM. Removing this limitation would allow the system to utilize a more dense environmental model in order to infer more efficient sets of parameters that we hypothesize would allow the robot to better traverse obstacles.

Acknowledgement

The authors would like to thank Julian Whitman for advice given during this work and Judson Kyle for assistance in running experiments on the physical platform.

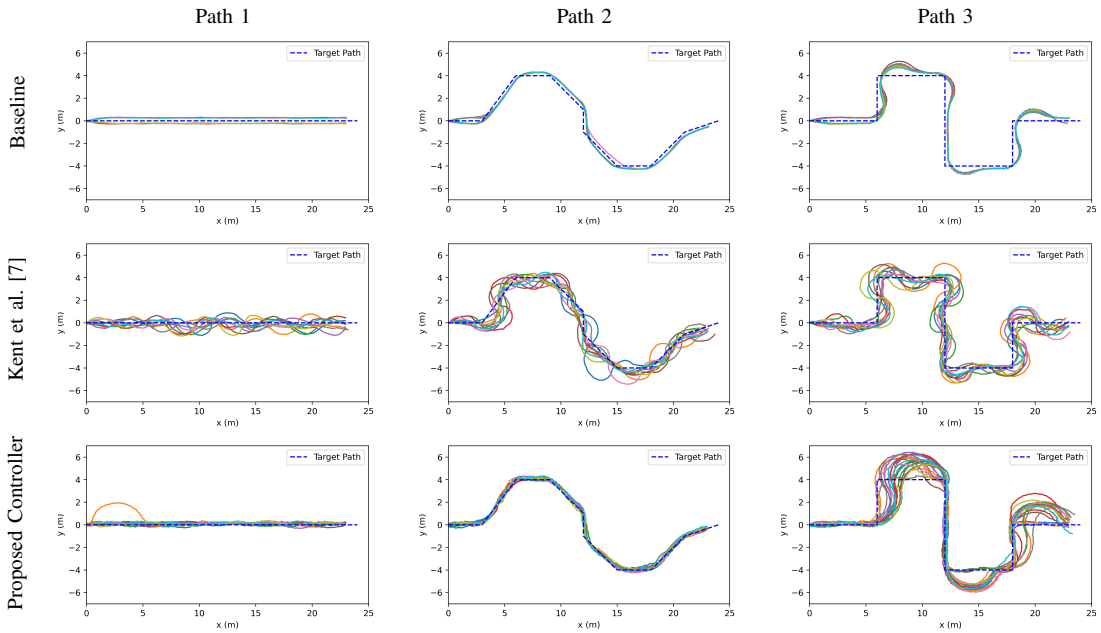


TABLE III: Performance of the simulated experiments on flat terrain. The outlying path seen in Path 1 for the proposed controller may be a result of the initial converging behavior of the CPG.

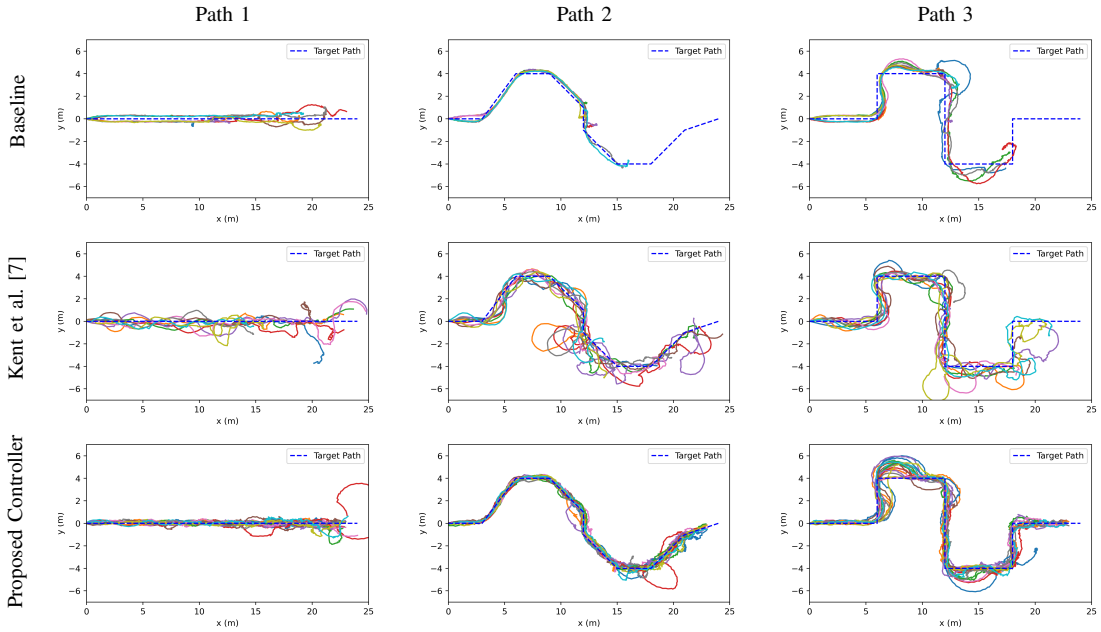
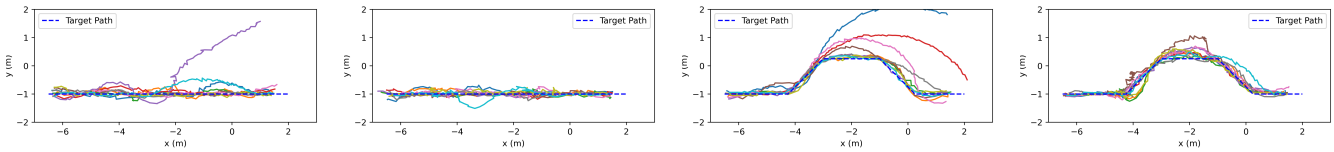


TABLE IV: Performance of the simulated experiments on non-flat terrain.



(a) Path 1 (flat) (b) Path 1 (non-flat) (c) Path 2 (flat) (d) Path 2 (non-flat)

Fig. 8: Physical path-following performance of inferred CPG parameters across two path shapes on both flat and non-flat terrain. The unusual paths in Figs. 8a and 8c are believed to be caused by issues with the actuators on the physical platform.

REFERENCES

[1] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Motion planning for humanoid robots," in *Robotics Research. The Eleventh International Symposium*, P. Dario and R. Chatila, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 365–374.

[2] K. Hauser, T. Bretl, J.-C. Latombe, and B. Wilcox, "Motion planning for a six-legged lunar robot," in *Algorithmic Foundation of Robotics VII: Selected Contributions of the Seventh International Workshop on the Algorithmic Foundations of Robotics*, S. Akella, N. M. Amato, W. H. Huang, and B. Mishra, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 301–316.

[3] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "RMA: Rapid Motor Adaptation for Legged Robots," in *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[4] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, 2020.

[5] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

[6] R. A. Chavali, N. Kent, M. E. Napoli, T. M. Howard, and M. Travers, "Inferring distributions of parameterized controllers for efficient sampling-based locomotion of underactuated robots," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 5767–5773.

[7] N. D. Kent, R. M. Bhirangi, M. Travers, and T. M. Howard, "Inferring task-space central pattern generator parameters for closed-loop control of underactuated robots," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8833–8839.

[8] U. Saranli, M. Buehler, and D. E. Koditschek, "Rhex: A simple and highly mobile hexapod robot," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 616–631, 2001. [Online]. Available: https://doi.org/10.1177/02783640122067570

[9] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation," in *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[10] T. M. Howard, S. Tellex, and N. Roy, "A natural language planner interface for mobile manipulators," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6652–6659.

[11] R. Paul, J. Arkin, D. Aksaray, N. Roy, and T. M. Howard, "Efficient grounding of abstract spatial concepts for natural language interaction with robot platforms," *International Journal of Robotics Research*, Jun. 2018.

[12] A. Crespi and A. J. Ijspeert, "Online optimization of swimming and crawling in an amphibious snake robot," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 75–87, 2008.

[13] J. Conradt and P. Varshavskaya, "Distributed central pattern generator control for a serpentine robot," in *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, 2003, pp. 338–341.

[14] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural networks*, vol. 21, no. 4, pp. 642–653, 2008.

[15] H. Yu, H. Gao, L. Ding, M. Li, Z. Deng, and G. Liu, "Gait generation with smooth transition using cpg-based locomotion control for hexapod walking robot," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 9, pp. 5488–5500, 2016.

[16] G. Sartoretti, S. Shaw, K. Lam, N. Fan, M. Travers, and H. Choset, "Central pattern generator with inertial feedback for stable locomotion and climbing in unstructured terrain," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–5.

[17] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[18] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, Tech. Rep., 1992.