UNIVERSITY OF CALIFORNIA SAN DIEGO

Virtual leg compliance improves swing-phase collision response when walking on uneven ground

A Thesis submitted in partial satisfaction of the requirements for the degree Master of Science

in

Engineering Sciences
(Applied Mechanics)

by

Henry Chang

2020

The Thesis of Henry Chang is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2020

EPIGRAPH

To achieve great things, two things are needed:
a plan & not quite enough time.

Leonard Bernstein

TABLE OF CONTENTS

LIST OF SUPPLEMENTAL FILES

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

I would like to acknowledge Professor Nicholas Gravish for his support as the chair of my committee. His guidance and insight help my research in both macro and micro level during past years. Also, his encouragement keeps me being enthusiastic in robotics research. All of his continued support has proved to be invaluable.

I also like to appreciate all the members in Gravish Lab, it is such an energetic and warm team. I got plenty useful advice and support from many teammates during the hard time. This team makes me feel like I am part of them, also, filled with joy and hard working.

Chapter 1, in full, has been submitted for publication of the material as it may appear in International Conference on Intelligent Robots and Systems (IROS), 2020. Henry Chang, Justin Chang, Glenna Clifton and Nicholas Gravish. The thesis author was the primary investigator of this paper. Also, special thanks to postdoctoral researcher Glenna Clifton and Professor Nicholas Gravish for clear data presentation with nice figures and helpful guidance.

Chapter 2, in part, thanks to teammate Weilun Hsieh for working on model building and part of the data analysis together. Thanks to Professor John Hwang and classmate Jaiyao Yan for lots of useful guidance and supports.

Chapter 3, in part, thanks to teammate Philip Hsieh for fighting with technical and communication issues together. We came up with lots of early stage idea of our robotic system and these inspiring accomplishments keeps me being interesting in legged research.

I also like to thank undergraduate researcher Justin Chang assisting me with more than half part of my research during past year. No matter in hardware or software part, especially in Chapter 3and 1, we solved many of communication issues together.

Lastly, thank to classmate Philip Hsieh for collaborating on the research in the early stage and roommate Yishen Yeh for spiritual supporting, who is also working on the master thesis, during the rough COVID-19 home quarantine life.

ABSTRACT OF THE THESIS

Virtual leg compliance improves swing-phase collision response when walking on uneven ground

by

Henry Chang

Master of Science in Engineering Sciences (Applied Mechanics)

University of California San Diego, 2020

Professor Nicholas Gravish, Chair

Uneven substrates in natural environments impose a lot of challenges on legged robot locomotion, no matter causing body instability during stance phase or obstacle collisions during swing phase. Through the swing phase, the leg lifts off the ground and arcs forward for searching a secure next foothold. The strategies for swing phase motion control are usually conservative, constantly sensing or observing the environment and re-planning the end-effector trajectory if collision is detected. Inspired from the fast and stable movement of small insects like cockroaches, our target is to design a passive control strategy that overcomes swing-collision without re-planning the trajectory. We implement a virtual compliance method to swing phase actuation using the direct-drive robotic leg. Through the systematic experiments with different obstacle-height and position, we compare the actuator positional control with virtual compliance control. We find out that positional control mode resulted in the leg swing collision that stuck the leg from moving through the obstacle. However, when the leg is actuated through virtual compliance mode, we observed a successful obstacle negotiation across a range of obstacle-height and position. Also, we seek to improve the accuracy and mobility of the leg virtual compliance control under quasi-static assumptions. Through a gradient-based optimizer, we present an optimal design that lowering the inertia effect of the leg assembly.

Introduction

The natural world imposes great challenges on animals and legged robots' mobility. Large structures within the environment prompt navigational planning [1] while small structures irregularities disrupt limb motion and foot placement [2]. During stance phase, the substrate unevenness induces body instability. During swing phase, when the limb arcs forward to search a subsequence foothold, nearby ground unevenness may cause toe stubs or miss footing that leading to catastrophic falls. A lot of research has been devoted to leg locomotion on natural substrates, however, most of the work has focused on stance phase and the methods to generate proper ground reaction force to improve robot stability and agility [2], [3]. To comprehensively navigate the natural substrate also requires sufficient study on limb swing phase.

In general principal, control of swing phase should be substantially simpler than stance phase. During stance phase, the limb produces ground reaction force to support and propel the body while adjusting for stability and agility to prevent slipping at the same time. On the other hand, swing phase simply resets the limb position to foothold for the next stance. This perspective can be validated by the observation that stance duration shifts with speed, but swing duration is mostly conserved [4]-[6]. In addition, some experimental findings show that limb swing is metabolically costly [7], suggesting that simple mechanisms for controlling swing may improve walking efficiency. Previous work on swing-limb control basically falls into one of three categories: 1) tactile, force, or inertial sensors on the legs and feet to detect obstacles and re-plan the trajectory [8]-[11], 2) visually plans the trajectory to avoid obstacle collisions [12]-[15], 3) customized swing trajectory to different substrates [2], [16]. Most of these methods implement positional control of the leg to track and actively adjusted the swing trajectory. In this chapter, a virtual compliance control method is implemented to leg swing-phase that allows the legs to passively conform and adapt to obstacle during their swing path.

Virtual compliance control makes a rigid linkage-based leg emulate a spring-damper-mass system. This control method has been successfully applied on various legged robots. However, in most of the cases, this has been used to control stance dynamics as running robot emulate spring-mass systems. Virtual compliance can be achieved by directly measuring the interaction forces through a force sensor on the foot, or through proprioceptive measurements such as measuring the joint torque within the actuators. Critically, proprioceptive based methods for compliance control rely on actuator and transmission systems that have low gear-ratios, low friction and high back-drivability.

The last ten years has seen dramatic growth in legged robotics hardware and design, specifically focused direct-drive robot legs for effective and compliant locomotion. In the regime of medium-small legged robots there is a broad class of small dog-sized, legged robots capable of dynamics movement (Fig. 1.1) [21]-[27]. These robots share lots of common features which include: 1) direct or quasi-direct drive actuation, 2) parallel linkages for reduced inertia, 3) high-torque density brushless DC motors. This class of robot presents an optimal testbed for studying swing-phase compliance.

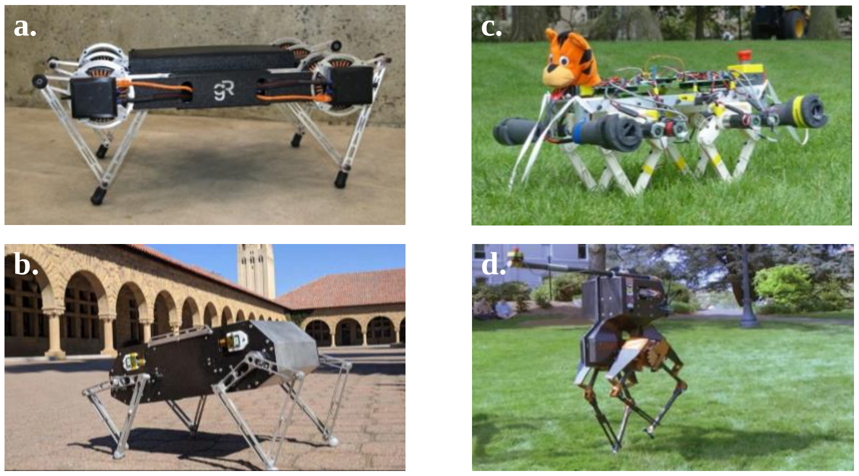


Figure 1.1: A class of small robot that use planar 5-bar linkages design, capable of compliance control through direct or quasi-direct drive motors. a) Ghost Robotics Minitaur [17]. b) The Stanford Doggo [18]. c) MIT Mini Cheetah [19]. d) The T Robot [20].

In Chapter 1, we present the design of a simple direct-drive robot leg capable of dynamic movement and control. We study the performance of this robot walking over an uneven substrate and subject to swing-phase collisions. More specifically, we are interested in two type of swing-phase control: 1) positional control and 2) compliance control. We hypothesize that high compliance during swing phase will enable the leg to collide with obstructions and accommodate these collisions through passive leg movement.

In Chapter 2, basing on the observations and conclusions from the virtual leg compliance experiments, we aimed to improve the mobility of limb by reducing its weight. To reach this goal, we will need shorter limbs. However, there is a trade-off between linkage length and working space. A short limb performs high agility but prone to be stuck in uneven substrates and hindered the ability to jump higher. We thus define a reasonable end-effector trajectory as the limb working space which is feasible for most of the leading-edge direct-drive legged robot [17]-[19]. An optimization package, called OpenMDAO [40], is implemented to produce the optimal limb design. It is a multidisciplinary open-source framework and can be applied in Python.

In Chapter 3, we will demonstrate some communication upgrades to our robotic leg system. The communication speed of a robot is very critical that defines its capability of completing tasks. The more efficient connection it has, the more agile it can response to real world perturbations. According to past experiences and the authentic works *Modern Robotics, Mechanics, Planning and Control* [41], a computer can usually connect with a motor controller up to $10^2 \sim 10^3$ Hz and a motor controller can connect with a motor up to a much higher 10k Hz. The frequency may drop drastically when a robot is programed to perform complex dynamic locomotion. We thus came up with a few ideas to break through the bottleneck in this section.

In Appendix, a sliding wedge model is introduced because we try to proof the virtual leg compliance experiments (mentioned in Chapter 1) have a reasonable trend. The behaviors of wedge effect may help readers to understand the obstacle negotiation when shin collision is happened. In the last part, we also provide the detail trajectory records of virtual leg compliance experiments with each obstacle position and height combination.

Chapter 1

Virtual Leg Compliance Improves Obstacle Negotiation

1.1 Virtual Compliance

We seek to develop a virtual compliance control for our robotic leg (Fig. 1.2). Many methods exist for controlling the compliance of robotic appendages, including admittance and impedance-based control. In this work we construct a compliance controller using quasi-static assumptions to simplify the controller design and reduce the computational complexity of the leg controller.

The leg geometry is a planar five-bar linkage (Fig. 1.2a) with dimension describe in. Table I. Notably, unlike other five-bar planar leg design such as in the Ghost Minitaur [17] or Stanford Doggo [18] the motors are not concentric and instead are separate by a distance w from each other. This geometry is chosen to reduce the mechanical design complexity while maintaining a relatively large workspace and vertical effective mechanical advantage [17].

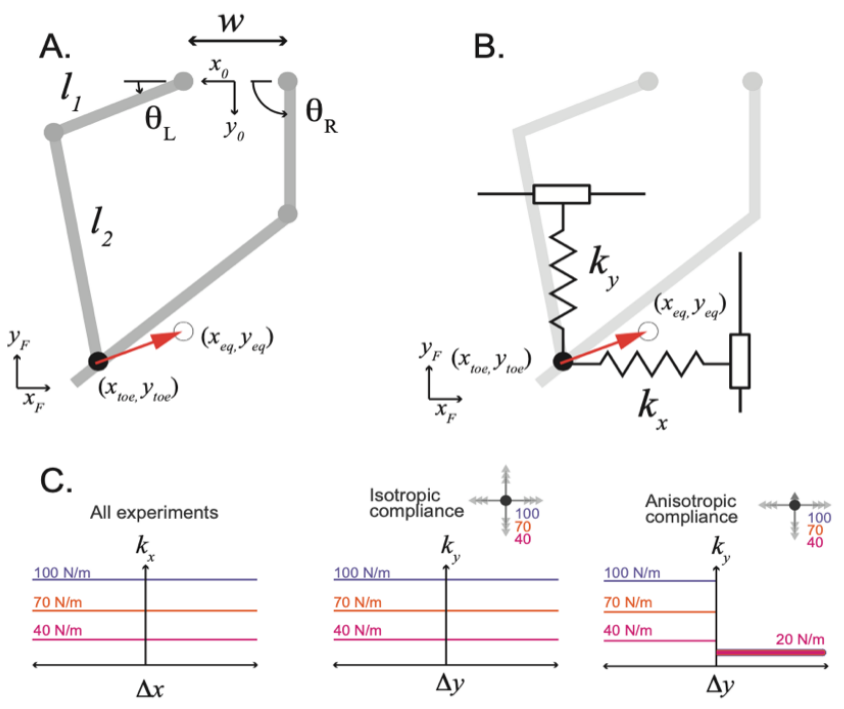


Figure 1.2: a) Leg geometry of the 5-bar leg assembly. b) We implement a virtual leg compliance model with horizontal k_x and vertical k_y stiffness. c) Plots of stiffness components with respect to displacement for two set of experiments, isotropic and anisotropic compliance. In all experiments, horizontal stiffness is independent of x displacement. For isotropic case, $k_x = k_y$ in all directions, and for anisotropic case, the positive vertical stiffness is always low (20 N/m) while in all other directions the stiffness is constant.

From the knee coordinates we compute the knee-span, L, and the knee angle α as follows

$$L = \sqrt{(x_{knee,R} - x_{knee,L})^2 + (y_{knee,R} - y_{knee,L})^2}$$

$$\alpha = \operatorname{atan2}(y_{knee,R} - y_{knee,L}, x_{knee,R} - x_{knee,L})$$

Lastly, we define the two-dimensional homogeneous transformation matrix which transform points between task-space and workspace coordinate systems as

$$T(\beta, t_x, t_y) = \begin{bmatrix} \cos(\beta) & -\sin(\beta) & t_x \\ \sin(\beta) & \cos(\beta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

While β is the rotation angle between transformations, and $[t_x, t_y]^T$is the column vector specifying the translation distance of the transformation. The forward kinematics of the five-bar linkage are given by the following transformation, where x, y specifies the location of the toe joint with respect to the motor angles and leg geometry

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = T\left(\theta_R, -\frac{w}{2}, 0\right) T(\alpha - \theta_R, l_1, 0) \begin{bmatrix} L/_2 \\ \sqrt{(l_2)^2 - (L/_2)^2} \\ 1 \end{bmatrix} \quad (2)$$

When the motor spacing, w, is zero the forward kinematics collapse to the simpler formula present in [17]. However, our leg with non-zero w, equation 2 is fairly complicated and we derived it using symbolic toolkit, SymPy [28]. We note that since this leg is a parallel linkage, there are two solution corresponding to the foot up or foot down configuration. Equation 2 explicitly selects the foot down configuration of the forward kinematics. From the forward kinematics we derive the standard leg Jacobian $J(q)$.

We then develop a simple compliance controller for our leg using standard force-torque relationship, $\tau = J^T F_{foot}$. For the foot to emulate a spring-mass system we first define the equilibrium position of the foot, $[x_e, y_e]^T$. We than calculate the horizontal error, $\Delta x = (x_{foot} - x_e)$, and vertical position error $\Delta y = (y_{foot} - y_e)$. The positive x direction is in direction of the forward swing trajectory (x_F, Fig. 1.2a) and the positive y direction points up away from the ground, (y_F, Fig. 1.2a). We define the force then to be

$$\begin{bmatrix} F_x \\ F_y \end{bmatrix} = \begin{bmatrix} k_x(\Delta x, \Delta y)\Delta x \\ k_y(\Delta x, \Delta y)\Delta y \end{bmatrix} \quad (3)$$

where the horizontal and vertical stiffness can be a constant or dependent upon toe location (Fig. 1.2). In this chapter we study two modes of compliance control for swing trajectory tracking, isotropic and anisotropic compliance. In isotropic mode, k_x and k_y are equal constant across all ranges of displacement (Fig. 1.2c) and we vary k_x, k_y = $[40, 70, 100]\ N/m$.In anisotropic mode, we reduce the positive vertical stiffness to $20\ N/m$ to allow the leg to more freely deviate from the swing trajectory in the positive vertical direction. In all other direction the stiffness is constant and held at one of three values k_x, k_y = $[40, 70, 100]\ N/m$. Lastly, in all experiments we compensate for the gravitational effect on the linkage through a feedforward added to the control signal. The gravity compensation control is generated from experimental measurements in which the leg is held at various locations inside the 0.2 x $0.1\ m^2$ workspace with 121 measurement points. At each point, the leg will be held for 10 seconds and we measure the average motor current that required to hold the leg statically. Then, the leg smoothly moves on to the next point and do the same measurement until all 121 points are measured. Finally, a lookup table is generated using interpolation.

1.2 Leg Design and Control

1.2.1 Hardware and Motor Control

Table 1: Summary of leg geometry (left) and leg actuation (right)

Name	Detail
l_1	9 cm
l_2	16 cm
w	7 cm

Name	Detail
Motors	Quanum 5250
Encoders	CUI AMT 102
Motor controller	ODrive

We implement positional and virtual compliance control strategies on the planar five-bar leg assembly. The leg linkages are fabricated from 1.22 cm thick aluminum with geometries inspired by [18]. Leg linkages are joined at rotational joints with deep groove bearings. Leg dimensions are given in Table I.

We use two brushless DC motors (BLDC) at the hip joints to actuate the legs. The motors are low-profile outrunner motors (Quanum 5250) which have seen extensive use in recent small dog-sized legged robots [21]-[26]. Motor communication and control are performed by ODrive motor controller (ODrive Robotics, Richmond, CA) which perform the communication for each motor at 10kHz. Additionally, the ODrive comes with closed-loop positional control and closed-loop current control that can be commanded through an external microcontroller or computer. Motor position is measured by the encoder mounted on the motor base (AMT-CUI 102). Each encoder provides quadrature signals with a step-resolution of 8192 counters per revolution.

Also, the testbed is consisted of two 6-feet linear bearing shafts which are capable of finishing 12 test strides of the bipedal leg unit. The bipedal leg unit connects to the shafts through linear bearings and sliding freely with the ODrive motor controllers. However, in this chapter, we

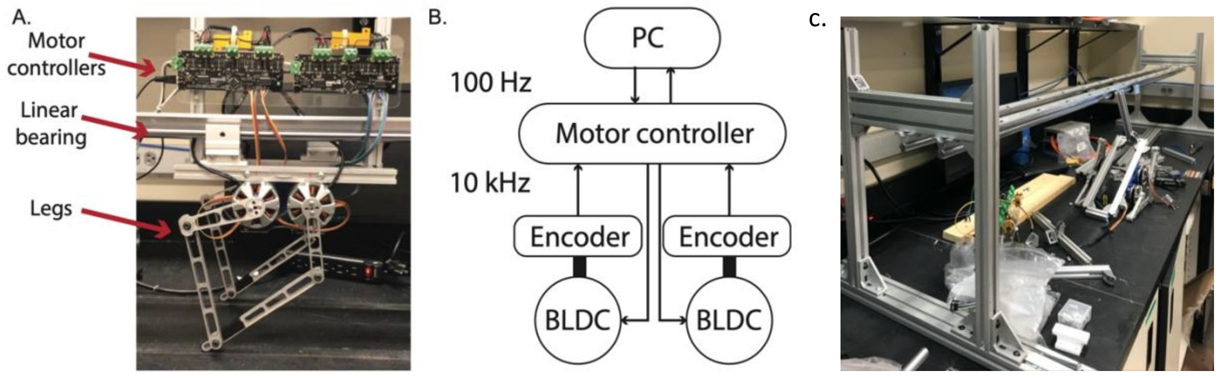


Figure 1.3: Experiment setup and control. a) The bipedal leg system is mounted on the linear rail to enable planarized walking motion through linear bearings. The leg controllers are mounted above the legs. b) Schematic of control system. A computer sends position or current commands to the motor controllers at 100Hz, and the motor controller runs at 10kHz. c) The testbed setup. The structure consists of two 6-feet linear rails and supported by aluminum T-slots.

decide to fix the bearing and focus on the single step that interact with the design obstacles. Figure 1.3 shows the leg and motor hardware, the controller communication and the linear shafts testbed.

Trajectory generation and control is performed on a computer and position and current commands are send to the ODrive at 100 Hz over serial. While 100 Hz is relatively slow for closed-loop limb control, our experiment focus on low-speed swing movement (0.5, 1 and 2s swing duration) where this rate is sufficient. We execute all motor control in Python through NumPy [29] and SciPy libraries [30]. We implement three control methods for the swing phase: 1) position control and compliance control with 2) isotropic stiffness, 3) anisotropic stiffness. In position control, angular position commands directly sent from the computer to each motor defined the swing trajectory. The motor position control was performed on the ODrive using a high-gain proportional integral. In isotropic and anisotropic cases, desired motor torques are translated to currents using $\tau = K_T I$ and set as set-point to each motor.

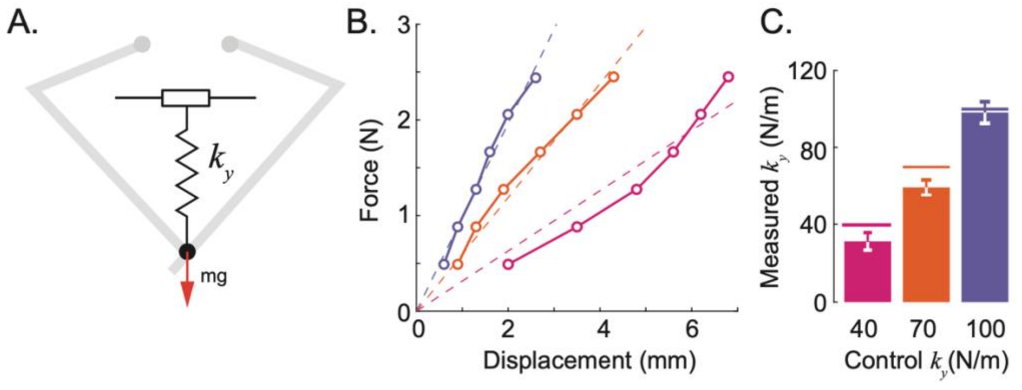


Figure 1.4: Comparison between specified and actual limb stiffness. a) We attached known masses to the toe joint in the y direction and measured the displacement. b) Force versus displacement for the three specified stiffness conditions, $k_y = [40, 70, 100]\ N/m$. Dash lines are the fit functions of $F = k_y \Delta y$. c) Comparison of measured and specified stiffness k_y. Error bars represent the 95% confidence intervals of the linear regression. Solid white or colorful lines represent the specified stiffness.

1.2.2 Stiffness Measurement

The simplifying quasi-static assumptions used in designing the compliance controls for our robotic leg produce some inaccuracies. To estimate the difference between specified and actual leg stiffness, we calculate the vertical stiffness of stationary limbs for each experimental stiffness. Downward forces are generated by hanging mass at the toe joint and the resulting deflections are measured. The slope of plotting forces versus deflections defines the stiffness, $F = k_y \Delta y$.

As expected, deflection of the toe increased approximately linearly with the applied forces (Fig. 1.4b). For the specified conditions $k_y = [40, 70, 100]$ N/m, we calculate an actual stiffness of $k_y = [31.5 \pm 9.0, 59.5 \pm 7.8, 98.2 \pm 11.2]$ N/m respectively. The 40 N/m deviated the most from a linear relationship and had the least stiffness accuracy. The higher stiffness values are within the measure 95% confidence interval from the linear regressions. The disagreement between the specified and actual stiffness for 40 N/m is likely the result of bearing friction in the

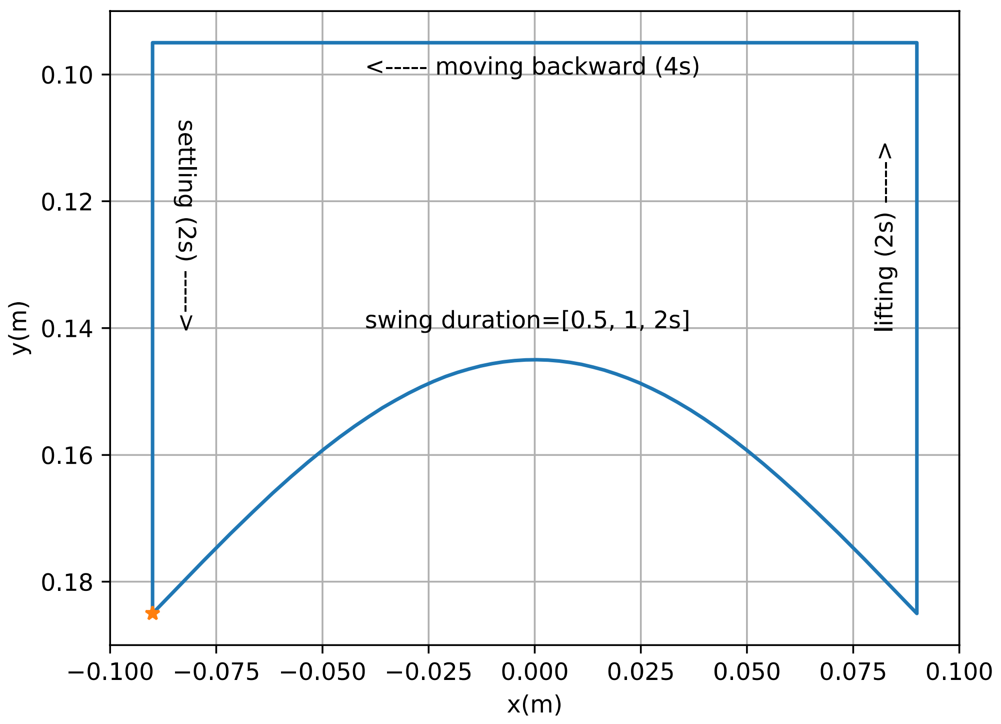


Figure 1.5: Specified end-effector trajectory. Starting from the orange point, swing phase takes three different duration. Then, the lifting (2s), moving backward (4s) and settling (2s) processes bring the toe back to stating position.

joints and the motor cogging torque. Motor cogging torques results from magnetic interactions within the motor producing a small non-zero torque that must be overcome for the motor to move. At low specified torque values, the resistance due to bearing friction and cogging torque may appreciably contribute to the motor dynamics. However, at higher specified torques the relative influence is reduced or negligible.

1.2.3 Swing Trajectory Generation and Tracking

Before performing experiment with obstacles, we first verified that the robot leg accurately tracked the specified swing-phase trajectory. To test the swing-phase performance of our leg under varying motor control modes, we generated a sinusoidal swing-trajectory for the foot to follow (Fig. 1.5). The height of the swing-phase is 4 cm with a stride length of 18 cm. A going home

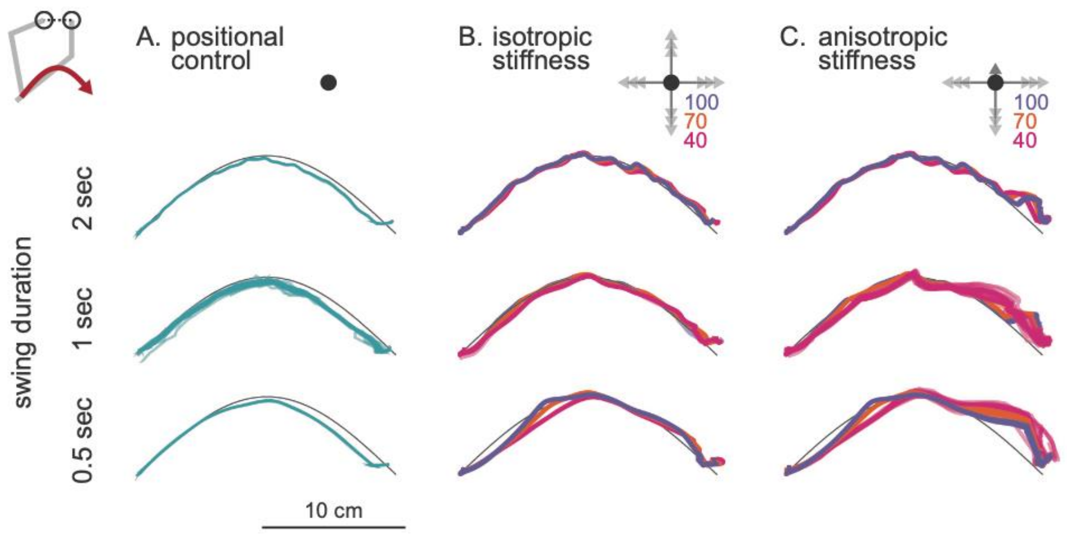


Figure 1.6: Swing-phase trajectory tracking without a step under three different motor control methods and for three different swing durations. Each plot shows 25 trials overlaid. a) Position control mode. b) virtual compliance mode with isotropic stiffness. c) virtual compliance mode with anisotropic stiffness.

process comes after the swing phase and takes the toe back to starting position (the orange star shown in Fig. 1.5) which is defined as one testing cycle. After 25 cycles, we then compare the tracking performance for each control mode of swing phase under three different durations, 0.5, 1 and 2 second. For convenience, the going-home process would be always using position control mode given that this is not the part we focus on.

From figure 1.6, the position control demonstrates the most accurate tracking, especially for short swing durations. For the virtual compliance mode, isotropic and anisotropic conditions tracked the first half of the swing phase within a few millimeters but deviated strongly toward the end of swing phase. The deviation from the specified trajectory was getting worse for low stiffness (40 N/m) and at faster speed (0.5 second duration). The anisotropic mode showed the worst tracking, with the toe location consistently above the specified trajectory. This overshoot in the vertical direction likely results from relatively low stiffness (20 N/m) in the positive vertical

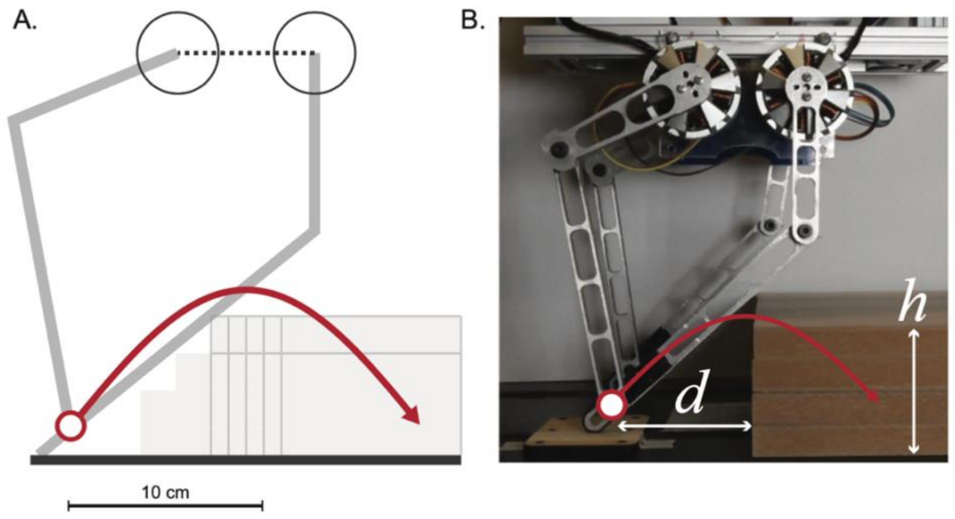


Figure 1.7: a) Overview of step experiment illustrating the swing trajectory and the corresponding obstacle locations. b) A picture of the experiments with $d = 50\%$, and $h = 5.6\ cm$

direction. Thus, any deviations above the commanded trajectory are compensated with low torque compared to the deviations below the commanded trajectory.

1.3 Swing Phase Collision Resistance

1.3.1 Collision Experiments

We performed systematic experiments to determine how the control of the leg affect obstacle negotiation during the swing phase. The obstacles used in the experiments consist of stacks of smooth MDF particle board, each 1.9 cm tall. Particle board stacks were aligned so that they presented the leg with single vertical step. Accounting for the of the toe, the step heights included $h = [1.8,\ 3.7,\ 5.6]\ cm$ above the toe, set at a distance d from the starting location. The step distance was measured as a percentage of the forward swing length, with mid-swing occurring at $d = 50\%$. We performed the step experiments over the range $d = [20\%\ 30\%\ 35\%\ 40\%\ 45\%\ 50\%\ 55\%\ 55\%\ 60\%]$, but we did not test the combinations that were precluded due to the

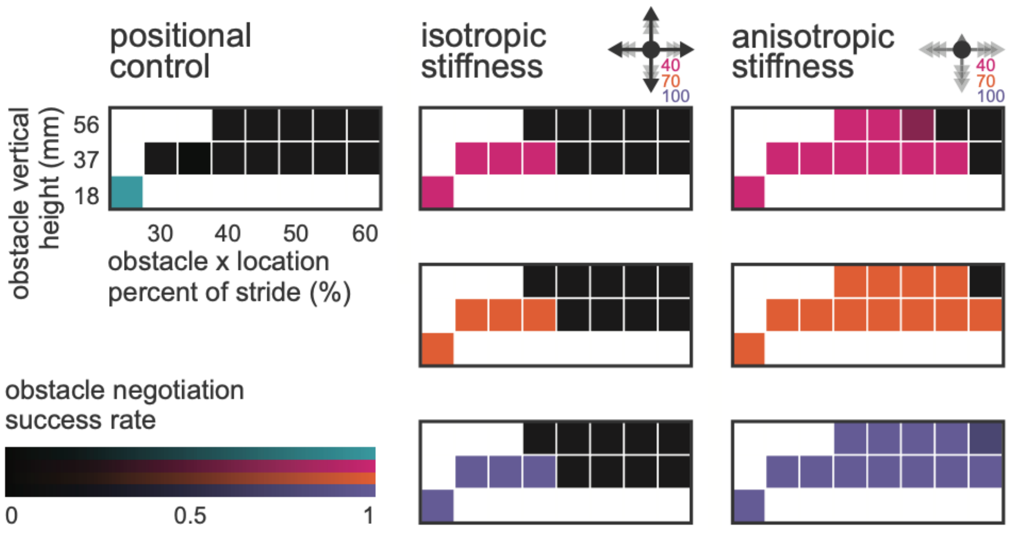


Figure 1.8: Collision negotiation success for step obstacles. Each box represents the outcome of 25 experiments at that distance, height, and control method combination. Black indicates 0% success and color indicates the level of success. White boxes are not tested.

limb geometry or that did not lead to a limb collision. The swing trajectory and duration was constant across all experiments and is consistent with the trajectory shown in figure 1.7. For all experiments we performed 25 replicates to determine the robustness of the swing phase obstacle interaction.

1.3.2 Comparison of Control Methods

We define a successful swing phase as one where the foot completes swing with the foot on top of the step (Fig. 1.10) Unsuccessful swing phase movements results from the toe or shin hitting the obstacle and then not progressing to the top of the step. We only included experimental conditions in which the swing trajectory result in a limb collision with the step.

1) Position control: The first method we tested was direct angular position control of the motors. Motor position control of the leg during swing movements lead to step failures in all but the shortest step height, $h = 1.8\ cm$ (Fig. 1.8). With the taller steps, the leg constantly hit the edge of the step and became stuck (Fig. 1.10). Averaging across all of the step height and step distance experiments (325 experiments) we found that the position control lead to a successful swing motion in less than 10% of the experiments (Fig. 1.9).

Our result indicated that open-loop motor position control is highly problematic for swing phase motions. As the limb collides with the step, the foot is displaced above the specified swing trajectory, which results in the motors attempting to push the leg downwards. Consequently, the foot become jammed against the obstacle and is unable to reach the top of the step, as illustrated in figure 9a.

Swing phase collision resistance using position control can be robust but requires trajectory re-planning when a collision is detected. This re-targeting is aided by the accurate tracking associated with motor position control and has been investigated in previous studies [10], [31], [32]. Alternatively, limbs controlled using motor position control could improve obstacle negotiation by retracting the foot to highest possible swing phase trajectory within the workspace. However, comparing to lower swing heights, this high swing tactic increases energy usage and requires either longer swing duration or faster swing speed.

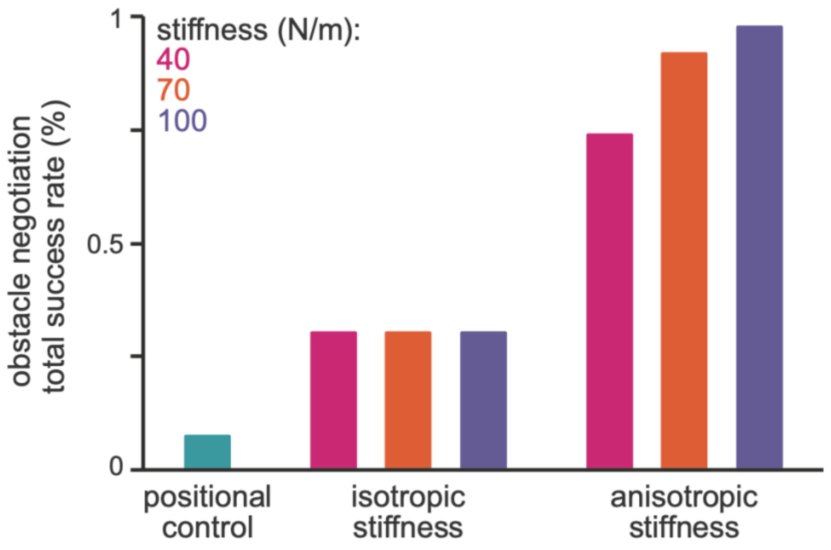


Figure 1.9: Aggregate success averaged across all step height and distance combinations. Position control had lowest success rate, isotropic stiffness was the second lowest, and the anisotropic stiffness worked the best. Within the anisotropic stiffness, higher stiffness values lead the highest success rate.

2) isotropic stiffness control: Under isotropic stiffness control the virtual stiffness was the same in horizontal and vertical directions. Despite the leg now exhibiting spring-like behaviors, its ability to successfully continue through swing after colliding with a step obstacle was still below 50% across all trials (Fig. 1.9). More specifically, the leg demonstrated 100% success for steps that less than programed swing height (4 cm) and placed within the first 40% of the stride (Fig. 1.8). However, when the step is further away from the beginning of the swing trajectory, and when the step was getting higher, the isotropic stiffness control failed every time.

While the poor performance of the isotropic stiffness control may seem counterintuitive it can be understood by considering the toe position is either higher or lower than the specified swing trajectory. When the foot collides with an obstacle within the first 50% of the stride, the toe is displaced in the $-x$ direction with a continued $+y$ actuating enabling the foot to move onto the step. However, with the step heights above the specified swing trajectory or at step location after

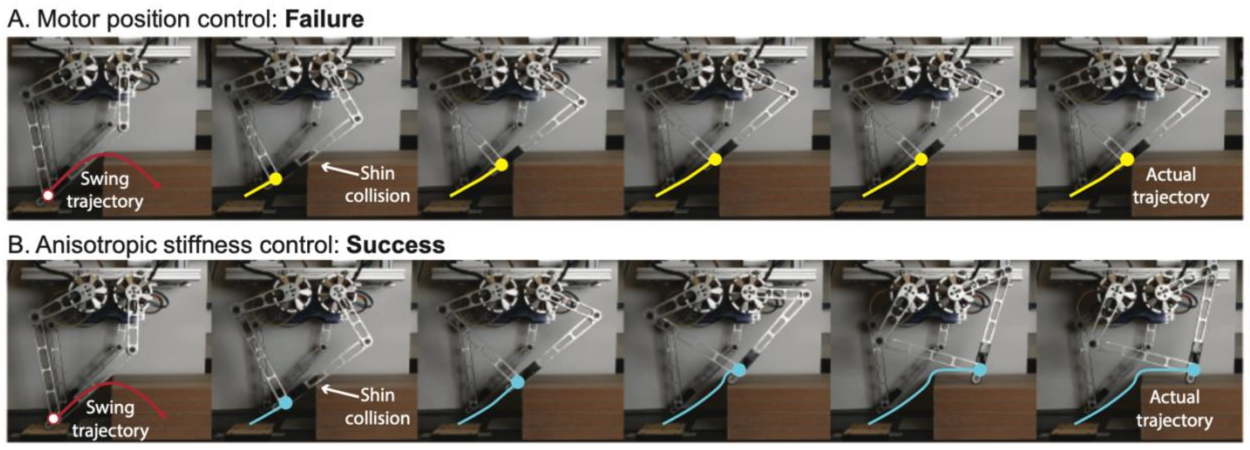


Figure 1.10: Comparison of a swing phase failure and success (step variables $d = 50\%, h = 7.6\ cm$). a) In motor position control mode, the leg collision at the shin causes the toe trajectory to deviate from the commanded swing trajectory. The foot is not able to recover and reach the top of the step. b) In anisotropic stiffness control mode, the foot is capable of continuing along the forward swing trajectory while deviating in the vertical direction to reach the top of the step.

the first 50% of the stride, the $-x$ displacement of the foot is accompanied by a continued $-y$ actuation which pushes the leg downward and preventing it from successfully moving onto the step. Also, the magnitude of isotropic stiffness did not affect the leg performance. Using higher leg virtual stiffness does not boost nor deteriorate the step negotiation success rate.

3) anisotropic stiffness control: Anisotropic stiffness resembled isotropic stiffness except with a positive vertical stiffness reduced to $20\ N/m$. Across all three stiffness conditions, the limb successfully advanced forward to the top of the step in over 75% of the tests. Step negotiation success rate increased with stiffness, with a lowest success rate of 75% for stiffness at $40\ N/m$ and highest success rate of 98% for stiffness at $100\ N/m$ (Fig. 1.9).

The anisotropic stiffness collision resistance process is illustrated in Figure 10b. In the anisotropic stiffness scheme the leg relatively freely able to deflect vertically upwards, and thus the obstacle collision at the shin acts like a wedge to push the leg up and onto the step. A high horizontal stiffness ensures that the leg will advance forward and thus through the wedge action of the shin will also be lifted vertically. The wedge-like behavior of the leg also explains why the higher anisotropic stiffness performed better on average: the wedge action requires substantial horizontal force to lift the foot. When the anisotropic stiffness is too low the motors do not generate enough lateral force to push the leg up onto the step. For more mechanics and mathematical details, a proof using simple wedge model sliding upward a slope is attached in the Appendix.

While the anisotropic stiffness worked best in the collision resistance experiments, it also performed the worst in the trajectory tracking. Thus, presents a trade-off of this method: allowing the limb to freely deflect when colliding with an obstacle can also result in deflections due to link inertia or friction. To resolve this issue, a design optimization project aiming to lower the inertia effect of our leg assembly by implementing gradient-based optimizer will be introduced in Chapter 2. Also, on the other hand, a more complete control could be constructed in which the full dynamics equations are considered in the control system and the link inertial and Coriolis forces can be canceled in the control signal. Despite the simplicity of our controller we still identify that anisotropic stiffness control provides a robust method for limbs to passively deflect and advance forward onto the step perturbation during swing phase.

1.4 Conclusions

Control of limb compliance during has improved robotic walking on uneven and complex substrates [2], [20]. We have found that swing phase collisions between a leg and an obstacle can also be overcome quite easily when the limb is actuated through a rudimentary stiffness control method. Our formulation of stiffness control makes several simplification assumptions, including that leg motion is slow and quasi-steady (ignoring inertia and Coriolis terms in the dynamics equation). The stiffness control method we have implemented is in many ways similar to proportional position control in operational space [33], [34]. Operational space control methods have been widely applied to legged locomotion control, primarily for control of stance phase ground reaction force [35], [36].

A critical motivation for stiffness-based swing control is that running movements can be fast and re-planning leg trajectories after a collision may be infeasible. Thus, stiffness control of the leg allows for robust trajectory tracking with and without a swing phase collision. Critically, the ability to successfully reach the top of a step does not require any trajectory re-planning in stiffness control mode. Thus, while stiffness control requires feedback and is close-loop, trajectory commands are open loop.

The perturbation resistance we observed through stiffness control are support by observation in animal walking. The legs of cockroaches contain elastic materials [37] which enable them to bend and flex passively in response to external perturbations [38]. When subjected to a perturbation in the swing phase cockroach limbs recover within 40 ms [39]. A critical distinction between the passive response of cockroach limbs and our experiments however is that there are no elastic materials in our robotic leg. Our leg compliance is feedback-controlled through the dual motor system, however the swing trajectory is never re-planned. As such, limb compliance control strategies could reduce computational requirements to successfully traverse uneven terrain without needing to detect collisions and adjust swing trajectories.

Chapter 1, in full, has been submitted for publication of the material as it may appear in International Conference on Intelligent Robots and Systems (IROS), 2020. Henry Chang, Justin Chang, Glenna Clifton and Nicholas Gravish. The thesis author was the primary investigator of

this paper. Also, special thanks to postdoctoral researcher Glenna Clifton and Professor Nicholas Gravish for clear data presentation with nice figures and helpful guidance.

Chapter 2

Design Optimization Using Gradient-Based Optimizer

2.1 Robotic Leg Model

2.1.1 Hardware

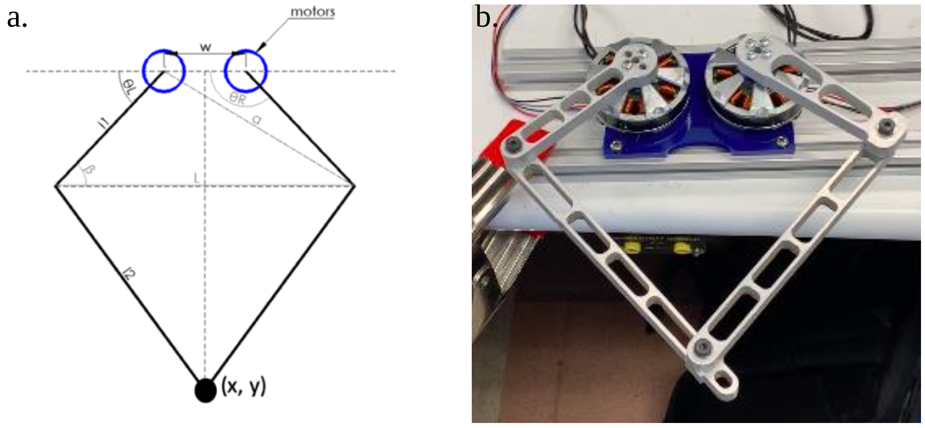


Figure 2.1: a) Schematic draw and b) picture of parallel five-bar linkage assembly

To get better control of virtual leg compliance basing on quasi-static assumptions (ignoring inertia and Coriolis terms in the dynamics equation), a lightweight design may perform a relevant result that we need. In this chapter, we will demonstrate a limb design optimization using gradient-based optimizer OpenMDAO and compare the final results. The original limb design we are interested in is identical to Chapter 1. It is a five-bar linkage assembly (Fig. 2.1) which is controlled by one ODrive motor controller (ODrive Robotics, Richmond, CA) and two Quanum brushless DC motors (BLDC). Distance between two motor is $w\ (7cm)$. There are two primary links $l_1\ (9cm)$ connect to the motors which are also hip joints and two secondary links $l_2\ (16\ cm)$ are

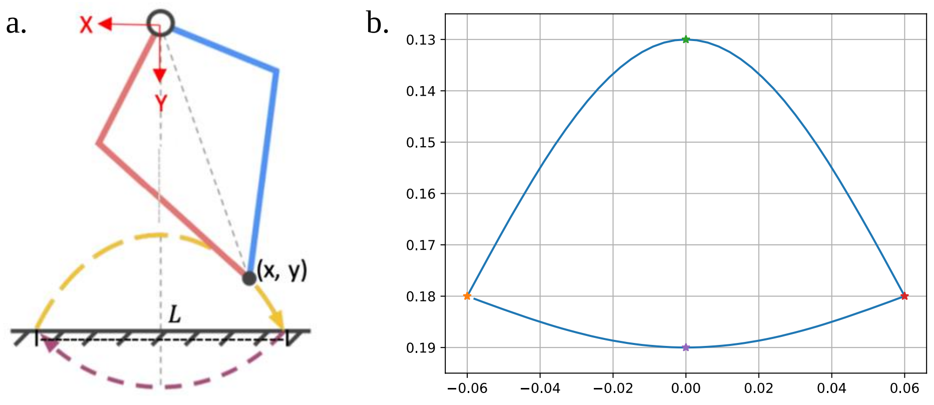


Figure 2.2: Piecewise sinusoidal end-effector trajectory. a) Swing phase is the yellow curve while stance phase is the purple curve. b) the example scales in meter.

connected at the toe. We define the toe position (x, y) with origin located at the middle of two motors.

2.1.2 End-Effector Workspace

To generate a proper trajectory which defines the maximum and necessary workspace, we chose several state-of-the-art quadruped studies as our reference for gait development. Inspired by open-looped trajectory introduced in Ghost Robotics Mintaur [17] and The Stanford Doggo [18], we then obtained an approach using piecewise sinusoidal functions as shown in Figure 2.2. Both swing and stance phase are sinusoidal function with different amplitudes. Also, the distance between touch down and lift off point defines the stride length L. Overall, the design flexibility is nice that the gait can be easily modified by varying the amplitudes, frequencies and stride length.

For consistency, we decided to use the same way as Chapter 1 to generate the toe trajectory (Fig. 2.3). That is defining certain amount of discrete control points to form smooth curves. These control points were used as the positional or torque commands for motor controller, ODrive. Although we do not send commands to motor controller in the optimization process, it is better to

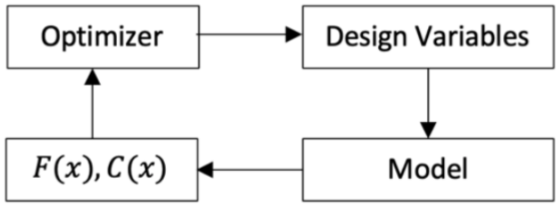
```
TOETRAJECTORY (DownAMP, UpperAMP, H, L):
 1  $p \in [0, 1]$
 2  $ST = 0.6$
 3  $SW = 1 - ST$
 4  initialize $x,\ y$ # for storing control points
 5  **Forall** $x,\ y$:
 6      **If** $p < ST$:
 7          $x = -(L/2) + (p/ST) * L$
 8          $y = DownAMP * \sin(\pi * p/ST) + H$
 9      **Else**:
10          $x = (L/2) - ((p - ST)/SW) * L$
11          $y = -UpperAMP * \sin(\pi * (p - ST)/SW) + H$
12  return $x, y$
```

Figure 2.3: The pseudocode for generating toe trajectory. Notations: stance phase amplitude = DownAMP, swing phase amplitude = UpperAMP, robot stance height = H, stride length = L and current percentage = p

let two projects share high compatibility for any future works. $ST\%$ and $SW\%$ represent the percentage of control points of stance and swing phase in a gait cycle. They satisfy the equation: $ST\% + SW\% = 1$.

2.2 Structure of Optimization Process

Optimization process is an algorithm to figure out a local minimum of objective function inside the model (Fig. 2.4). The model is the main part of the process and it consists of design variables. With gradient-based method, the optimizer keeps updating the design variables in the model and searching for the minimum of objective function in the direction basing on derivatives and this minimum implies the best value that fulfills certain engineering purpose. Also, the objective function can be comprehensive that contains multidiscipline. For example, designing an air foil would consider fluid mechanics, structural theory and manufacturing cost simultaneously.



$$\min_{l_1, l_2} \quad F(l_1, l_2) = l_1 + l_2 \ (unit\text{: } cm)$$

$$s.t. \quad C(\cdot) = \left\{ \begin{array}{c} l_1 > 0,\ l_2 > 0,\ w = 7 \\ 0^{\circ} < \theta_L < 90^{\circ} \\ 90^{\circ} < \theta_R < 180^{\circ} \\ SW = 5,\ ST = 1,\ L = 12 \end{array} \right\}$$

Figure 2.4: On the left side, we demonstrate the flow chart of the optimization process. Also, the general form of optimization is shown on the right side. We try to minimize objective function $F(l_1, l_2)$ with respect to design variables l_1, l_2, and subject to constrain functions $C(\cdot)$.

1) Design Variables: the length of primary link ($l_1 = 0.09\ m$) and secondary link ($l_2 = 0.16\ m$), the angle of left (θ_L) and right motors (θ_R). A reasonable range for $\theta_L$is 0°~ 90° and $\theta_R\ is$ 90°~ 180°.

2) Model: The forward kinematics (Fk) of five-bar parallel limb is calculated through planar homogeneous transformation matrix of hip and knee joints. The detailed derivation can be found in Chapter 1 equation 1~3. The simplified expression would be: $(x, y) = Fk(l_1, l_2, \theta_L, \theta_R)$.

3) Objective Function $\boldsymbol{F(x)}$: The goal is to minimize the inertia of the limb through decreasing its length and mass. Thus, we set the total length of primary and secondary link to be the objective function, $F(x) = l_1 + l_2$, and supposed to get the optimal values of the limbs when $F(x)$ is reaching to the minimum.

4) Constraint Function $\boldsymbol{C(x)}$: The maximum workspace defined by two piecewise sinusoidal curves is the constraint in the optimization process. This guarantees the optimizer provides the result which is capable of reaching the workspace fully. If there is any other discipline need to be concerned, we can implement multiple constraints here. In addition, leg geometry and motor workspace are also part of the constraint function.

Table 2: Preparation for Fk subcomponent

Inputs	Outputs	Symbolic Compute Partials
l_1, l_2, θ_L and θ_R	x, y	$\frac{\partial x}{\partial l_1}, \frac{\partial x}{\partial l_2}, \frac{\partial y}{\partial l_1}, \frac{\partial y}{\partial l_2}$ and $\frac{\partial x}{\partial \theta_L}, \frac{\partial x}{\partial \theta_R}, \frac{\partial y}{\partial \theta_L}, \frac{\partial y}{\partial \theta_R}$

5) Optimizer: The gradient-based Sequential Least Square Quadratic Programing (SLSQP), is the quasi–Newton method with a BFGS update of the B–matrix sourced from Scipy Library. The algorithm finds optimal solution instantly under small amount of input variables.

To run the OpenMDAO framework, we need several subcomponents to build up the main script. Fist, all the design variables l_1, l_2, θ_L and θ_R are consider as a subcomponent called $IndepVarComp$ since they are independent. l_1, l_2 are scalars ($l_1 \in \mathbb{R}, l_2 \in \mathbb{R}$) of linkage length and θ_L, θ_R are vectors ($\theta_L \in \mathbb{R}^n, \theta_R \in \mathbb{R}^n$, where n is the number of control points). By definition, they are both inputs and outputs of this component. Second, the limb forward kinematics (Fk) is defined as a subcomponent called $ExplicitComp$ which all the outputs variables are explicit. Besides specifying the inputs and outputs, the symbolic partial derivatives of the outputs with respect to all the inputs have to be precalculated too (Table II). The partials with respect to motor angles are also known as the Jacobian matrix of limb forward kinematics. The 2×2 Jacobian matrix represents the transformation between working space (toe space) and joint space (motors). Lastly, we defined the objective function $C(x)$ as another subcomponent which has inputs l_1, l_2, outputs $l_1 + l_2$ and relatively simple partial derivatives.

In the main script, three above mentioned subcomponents are included and all identical variables are connected. Then, we specify the upper and lower bound of design variables and declare the constraint function. Since the model is under planar assumptions and most of the subcomponents are relatively simple, optimization process usually will not take more than 15

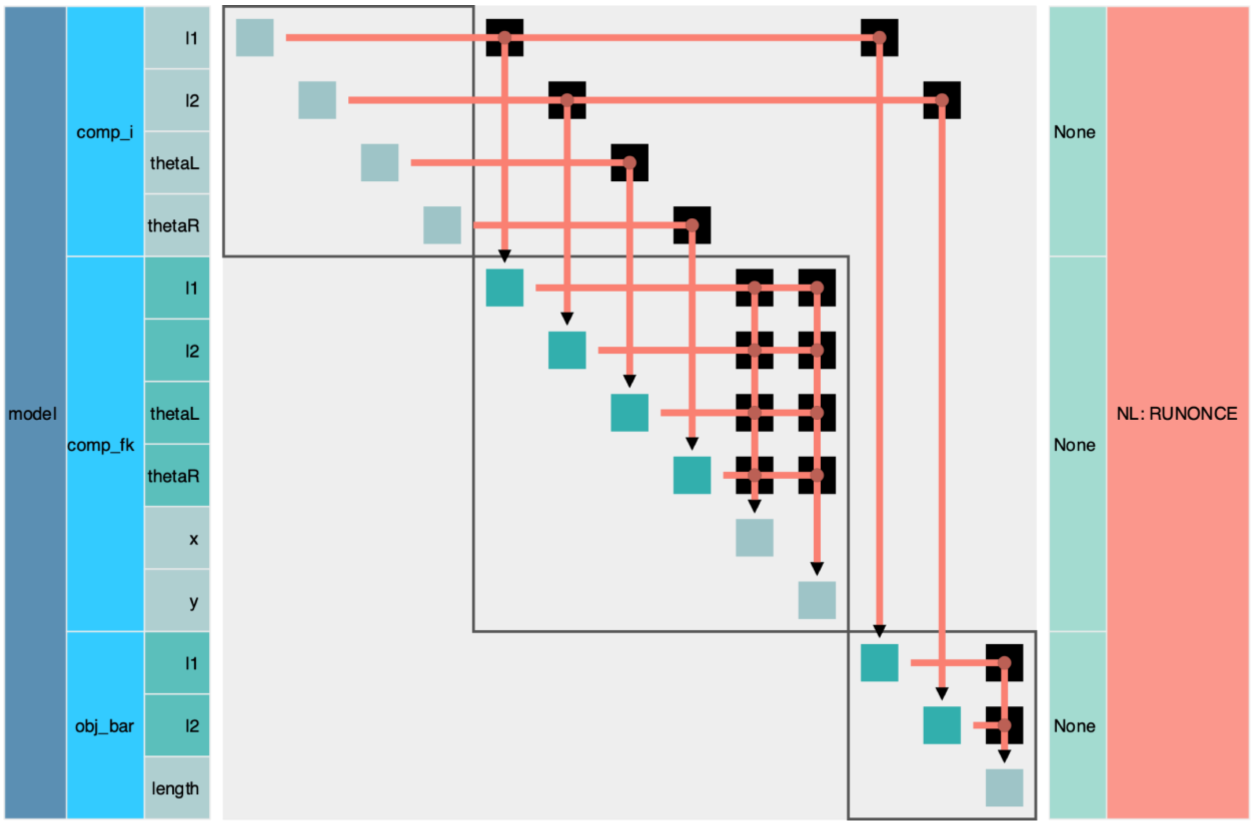


Figure 2.5: Model tree of the optimization process. Three gray boxes indicate the independent variables, forward kinematics and objective function subcomponents, respectively. Inside the subcomponents, the cyan blue dots are the inputs and light cyan blue dots are the outputs. The magenta arrows represent the information flow. Once the variables are passed to the objective function, system will start a new iteration to update design variables until a local minimum is found.

second to converge. Figure 2.5 clearly shows the subcomponents and information flows in the model tree.

2.3 Optimization Results

Before moving to the results, the reasonable left motor working angle θ_L is $0° \sim 90°$ and right motor working angle θ_R is $90° \sim 180°$. Any values beyond this range will likely to cause a singularity in our five-bar mechanism and this should be strictly prevented. From the result shown in figure 2.6, we consider two different kinds of optimal values. The first case, second column, which has the full range of motor working angle has the best optimization result for both links.

	Original values	Optimal values (without offset)	Optimal values (with offset)
Primary link, l_1 (m)	0.09	0.0578	0.0608
Secondary link, l_2 (m)	0.16	0.1548	0.1562
Total length (m)	0.25	0.2126	0.2171
l_1 improvement		35.78%	32.45%
l_2 improvement		3.25%	2.38%
Total improvement		14.96%	13.16%

Figure 2.6: The comparison between initial values and optimal values. Two sets of optimal values are calculated with different motor working angles. The second column with fully stretched knee joints, while the third column with safety offset (5.4°) of knee joints. The improvements are the percentage of length decrease compare to the original values.

However, this case brings some potential problems when the end-effector is at the furthest point of the trajectory. When the limb reaches to the furthest point with the shortest design length, the knee joints are nearly fully stretched and angle between l_1, l_2 is almost $180°$. The dynamically moving joints will easily run into singular point because of perturbations from uneven terrain or obstacle collision. Thus, in the second case, we implement an experimental offset to keep the knee joints from running into singularity when the toe reaches to the furthest point. It is a $5.4°$ ($\approx 0.03\pi$) safety offset that decreases the range of motor working angle θ_L to $0° \sim 84.6°$ and θ_R to $95.4° \sim 180°$.

Now back to figure 2.6, for the optimal values without offset, the primary link l_1 decreases the length by 35.78% and secondary link l_2 decreases the length by 3.25%. The total length is thus 14.96% shorter than original values. On the other hand, if the offset is considered, the primary link l_1 decreases the length by 32.45% and secondary link l_2 decreases the length by 2.38%. The total length is thus 13.16% shorter than original values.

2.4 Conclusions

The optimal result provides a reasonable lighter and shorter design of the robot leg. This may result in better agility and maneuverability of robot since the inertia effect is decreased. However, the exact improvement of cost of transportation requires further dynamics analysis which is not included in the chapter.

On the other hand, there are some interesting aspects worth to do detailed study. The new primary link l_1 is much shorter than original design relative to the new secondary link l_2 and here come a few questions: Is it going to hinder the ability to jump highly because of the shorten thighs (primary link)? Or is it going to have different obstacle negotiation patterns since the shin collision angle is different? This chapter may not have answers for most of the questions, but this might be an initiation for implementing multidiscipline objective function and more aspects of performance tests. For example, adding objective functions for measuring motors torque consumption may get to an optimal design with energy efficiency and locomotion agility. Since the model is ready and comes in handy, several thrilling studies are looking forward to being done in the future.

Chapter 2, in part, thanks to teammate Weilun Hsieh for working on model building and part of the data analysis together. Thanks to Professor John Hwang and classmate Jaiyao Yan for lots of useful guidance and supports.

Chapter 3

Communication Protocol Upgrades and Miscellaneous Work

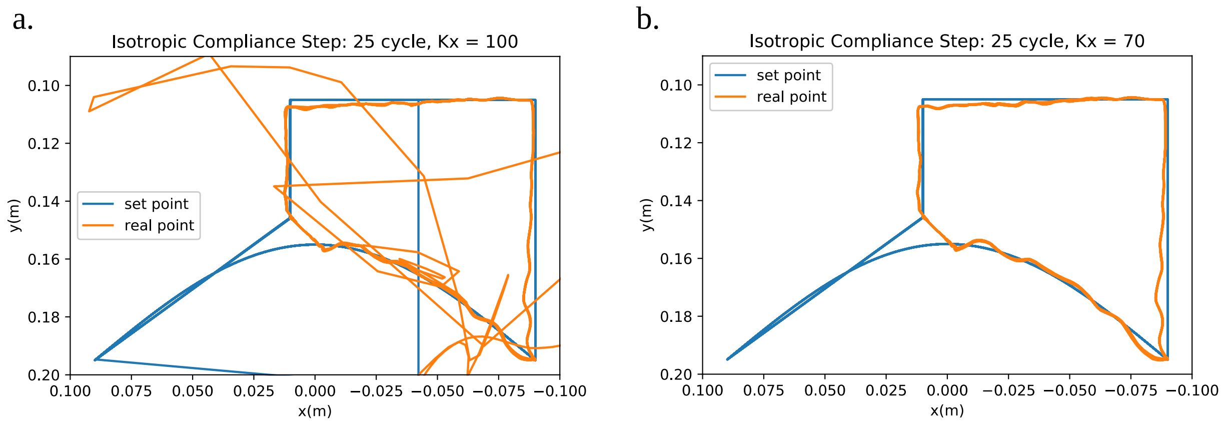
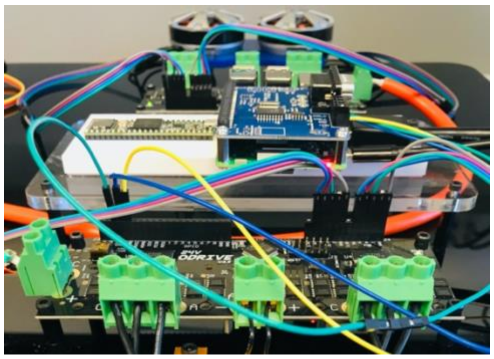


Figure 3.1: Comparison between a lost-control case and a well-function case. Orange line represents actual toe positions actuated by motors while blue line is the specified trajectory. a) A demonstration of motor losing control due to sudden communication delay. Controller fails to follow the specified trajectory and causes the leg moving randomly. b) In well-function cases, actual toe position follows specified trajectory unless an obstacle collision happened.

Communication system in a robot is just like the neural network in a human body. A more efficient system can help a robot perform more accurate control. Especially in a dynamic motion, information need to be updated quickly during each command iteration when reacting to real world perturbations. Relatively insufficient speed or delay in communication may cause a catastrophic fall or miss reacting. Although our previous study of passive limb obstacle negotiation was carried out under quasi-static assumption and the limb was able to finish most of the tasks smoothly, it did show a few signs of delay and miss control occasionally (Fig. 3.1). The default communication between the computer to the motor controller ODrive is through USB or UART (Universal Asynchronous Receiver-Transmitter). It is a user-friendly interface that comes with lots of ready-to-use Python packages. The tested speed is 80Hz~150Hz which is however at the lower bound of reasonable speed range (100Hz~1000Hz) in general cases [41]. Therefore, we came up

a. 

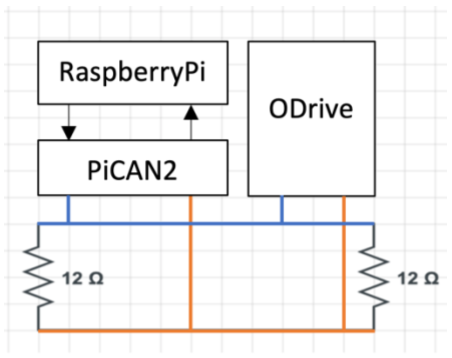
b. 

Figure 3.2: CAN bus wiring setup a) The black PCB (bottom) is ODrive and it connects with the dual-layer PCB (top, RaspberryPi and PiCAN2 transceiver) through two wires. b) The blue line represents low voltage. The orange line represents high voltage. We will need an aftermarket PiCAN2 CAN-bus-transceiver to decode data transition for RaspberryPi while ODrive is embedded with transceiver ready to be used.

with a few ideas to either upgrade the communication protocol or parallel compute the information. Following sections will cover the potential of the new systems that we have tried and the challenges or technical issues that we encountered.

3.1 CAN Bus with Raspberry-Pi 3b+

CAN bus (Controller Area Network) is a two-wire communication that has been widely used for electrical wiring in vehicles. The network is able to speed up to 1Mb/s which is faster than operating USB or UART on ODrive. The wiring is basically consisting of a high voltage

and a low voltage wire. The data is transmitted in binary form which will be decoded by a CAN bus transceiver and received by each node. This system is not only high speed but also easy wiring. However, our lab PCs or laptops are not built with the pins that connect to CAN bus. We decided to use Raspberry Pi 3b+ as our main controller and it has several advantages: 1) comes with a lot of built-in pins that compatible with most of the common protocols, 2) it is a tiny and portable computer that can be placed on the robot itself, and results in simple wiring and higher robot mobility.

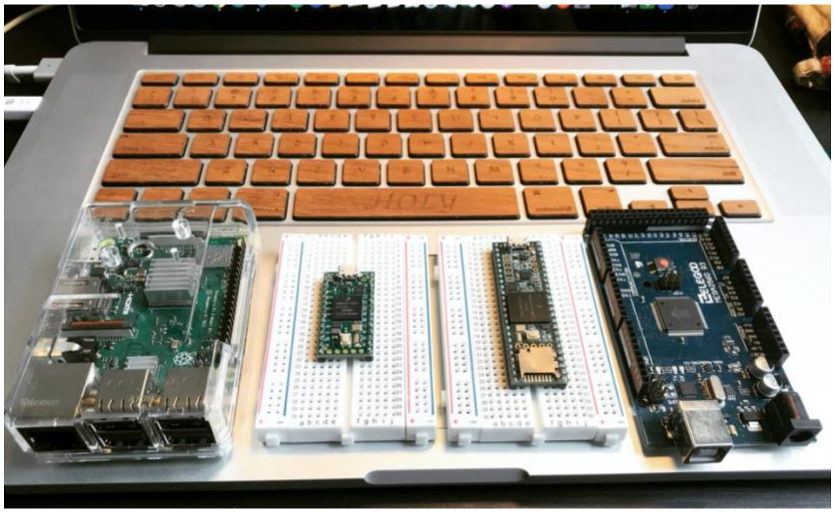
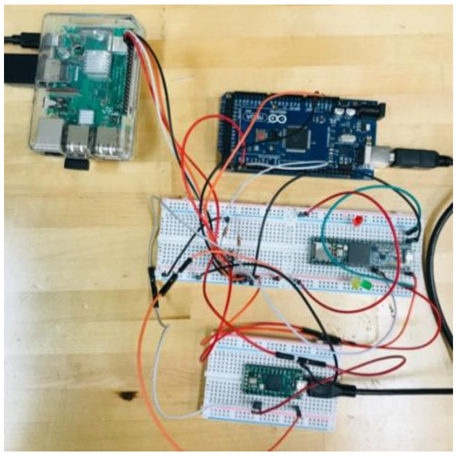


Figure 3.3: Popular microcontrollers for medium-small size robots. Besides a laptop, a small size and high-speed controller might also be a perfect choice to perform agile and accurate locomotion in most robotics research. Starting from the left: RaspberryPi 3b, Teensy 4.0, Teensy 3.5 and Arduino Mega 2560.

After quite amount of time studying in our new device setup (Fig 3.2), the hardware was basically completed and ready to run. However, we been through some hard time figuring out the firmware and software issues. First, the aftermarket CAN bus transceiver, PiCAN2, was relatively lack of a step-by-step instruction. It was not compatible with the newest Linux Kernel version (v. 4.19.79-v7+) in RaspberryPi for some reasons. We tried downgrading the OS to get an older kernel version and finally cleared the problem. Then, a driver was required to be installed to enable CAN interface. To sum up, those steps were not extremely hard to do but it took lots of trial and error to make a correct step, because the devices were not built perfectly matched. Second, the most critical point, the ODrive tools for CAN bus were still under development. There were no sufficient tools in Python or C++ at that time can be found. At last, we thought it was not worth to spend lots of time on an incomplete system just for faster communication and finally paused this project. Still, there were some nice experiences that helped us to choose a feasible device in the following trials.

3.2 I2C with Multiple Microcontroller

a. 

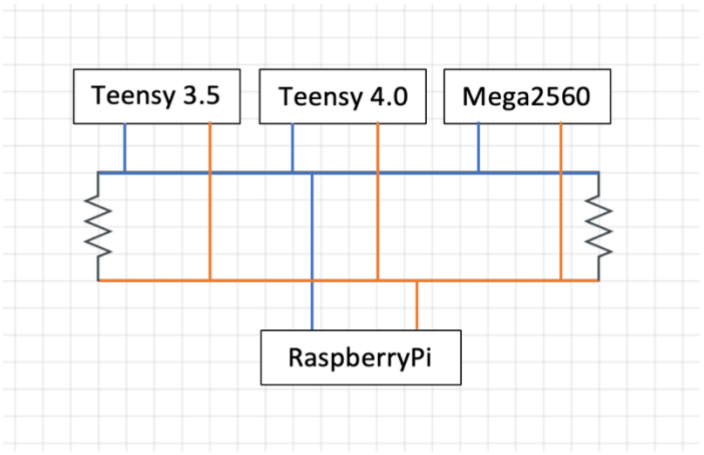
b. 

Figure 3.4: I2C bus is a two-wire communication, with serial data (SDA), serial clock (SCL) and two pull-up resistors. Master sends data to slaves by specifying the address and the speed can up to 1~4Mb/s. a) I2C bus wiring setup. b) The blue line represents SDA which carries data and the orange line represents SCL which synchronize the data transfer between the devices using clock signal.

With the experience in the last section, we generalized some points that need to pay attention to when build up new communication protocol and new controllers. A comprehensive software tools for ODrive would be first priority. Changing the communication usually means changing the tools or programing language to command ODrive. Currently, the developer provides mature and stable tools in Python and Arduino-based C++ for ODrive, so we decided to stick to Arduino or Arduino-based microcontroller in this section (Fig. 3.3).

Arduino Mega2560, Teensy 3.5 and 4.0 are three targets we have done complete speed tests through a same C++ program. Within certain time, Mega2560 can transfer and receive the messages with single ODrive up to 123Hz, with is relatively decent. On the other hand, Teensy 3.5 and 4.0 are able to communicate with single ODrive with outstanding 250Hz. In addition, the speed can be even 10 times faster than Mega2560 depends on how we flash the ODrive firmware.

(By default, 115200 b/s is the highest baud rate that Mega2560 can hold while ODrive and Teensy can hold much higher.)

Next step, we try to keep every microcontroller runs with highest speed by just connecting one ODrive to each of them. If we are building up a quadruped using four ODrives, we will need four controllers to manipulate each limb. Then, we picked RaspberryPi as the main controller that coordinates with four local controllers through the information bus I2C (Inter-Integrated Circuit). Arduino has <Wire.h> package that specifically built for data transmission through I2C bus while there is also Python library for I2C bus to run on RaspberryPi. Therefore, we setup a simple I2C testbed with master and slave controllers to demonstrate control signal transmission (Fig. 3.4).

Chapter 3, in part, thanks to teammate Philip Hsieh for fighting with technical and communication issues together. We came up with lots of early stage idea of our robotic system and these inspiring accomplishments keeps me being interesting in legged research.

Appendix

I. Wedge Model

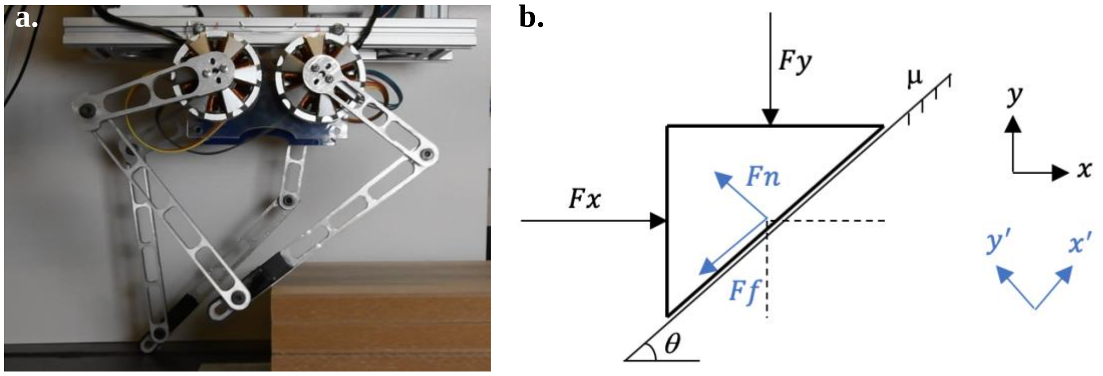


Figure 4.1: Wedge model a) Robotic leg performs the wedge-like behavior sliding up to the step as a shin collision happened. b) Free body diagram of a wedge block sliding on a tilted surface with friction coefficient μ and angle θ.

In Chapter 1, we find out the difference between isotropic and anisotropic stiffness control through the obstacle negotiation experiments. Anisotropic stiffness control cases are more capable of making successful step through each obstacle position and height combination. To make the analysis easier to understand, we relate the experiment to a sliding wedge model and provide some simple mechanics calculations here (Fig. 4.1).

Imagine the shin that collides with the step is a triangular block sliding on the θ degree tilted surface. Fx, Fy are the horizontal and vertical forces generated by the virtual spring system from the motors. The friction coefficient between aluminum bar and fiberboard is μ. By calculating the force equilibrium in rotation coordinate (blue coordinate), we can easily get the relation between θ and virtual spring force Fx, Fy (We ignore the gravity effect here).

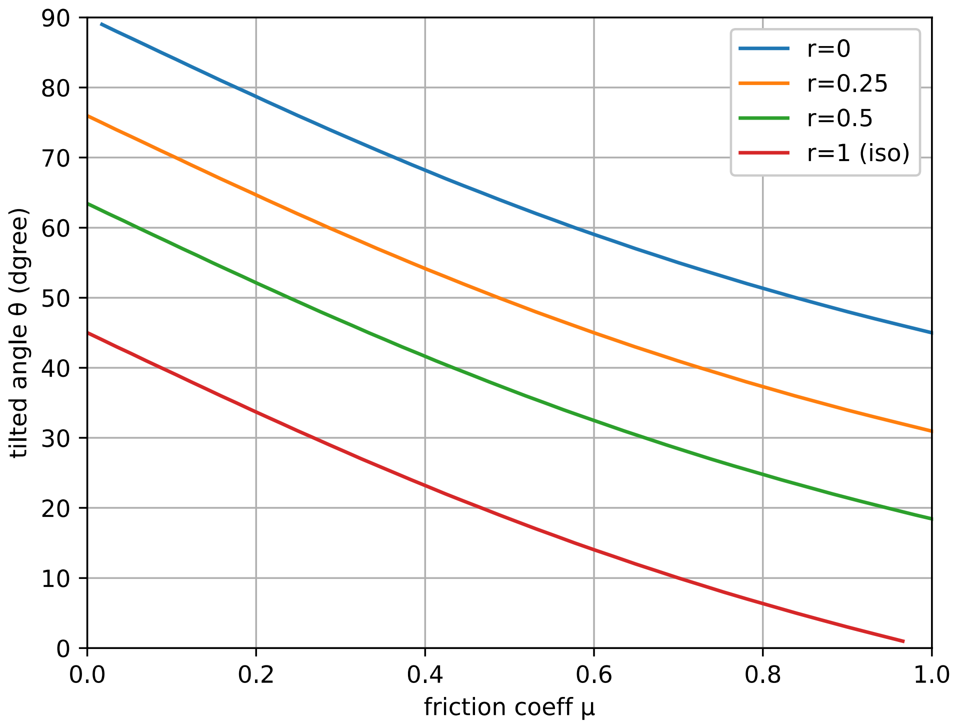


Figure 4.2: The maximum angle that lateral force Fx able to move the block upward with respect to friction coefficient µ. The red curve (r=1) represents the isotropic stiffness while rest of the curves represent anisotropic stiffness of virtual springs.

$$tan\theta = \frac{Fx - \mu Fy}{\mu Fx + Fy}$$

We simplify the above equation by assuming the horizontal and vertical displacements are isotropic, which means $\Delta x = \Delta y = 1$, and we also know the Hooke's Law for linear springs, $Fx = kx\ \Delta x$, $Fy = ky\ \Delta y$. Thus, the equation can be written as:

$$tan\theta = \frac{kx - \mu ky}{\mu kx + ky} = \frac{1 - \mu r}{\mu + r}$$

r is the stiffness ratio of horizontal and vertical springs, $r = ky/kx$. We then plot the maximum angle that the horizontal spring force is capable of moving the block upward with respect to friction coefficient (Fig. 4.2). Besides, four different stiffness ratios are implemented:

$r = [0, 0.25, 0.75, 1]$. If $r = 1$, this represents the isotropic stiffness control case. Any case with $r < 1$, represents the anisotropic stiffness control case.

From figure 4.2, there are two conclusions regarding the plot. First, the lower the friction coefficient is, the larger the angle that the horizontal spring can push the block upward. However, this is not the part we are mainly interested in. Given that the friction coefficient is always the same in our experiments. Second, under the same friction coefficient, the lower the stiffness ratio is, the larger the angle that the horizontal spring can push the block upward. The result here validates our experiments in Chapter 1. The isotropic stiffness case has the worst ability to overcome the step. For anisotropic stiffness cases, the lower the vertical stiffness, the more possible that the limb can move up to the step. Although the block is sliding upward at once does not necessarily means the leg overcome the obstacle in the end. It still indicates a larger probability for the leg to make a successful obstacle negotiation.

II. Detailed Experiment Results

The following are the detailed results of figure 1.8 & 1.9 with 7 different motor control modes in each obstacle position and height combination: 1) position control, 2) isotropic stiffness control, $k = [40, 70, 100]$ and 3) anisotropic stiffness control, $k = [40, 70, 100]$. On the other hand, 3 different obstacle heights, $h = [18,\ 37,\ 56](mm)$ (2, 3 or 4 stacks of fiberwood) and 8 different obstacle positions, $d = [25\%, 30\%, 35\%, 40\%, 45\%, 50\%, 55\%, 60\%]$ (located at d% of stride length) build up 13 combinations excluding unfeasible cases.

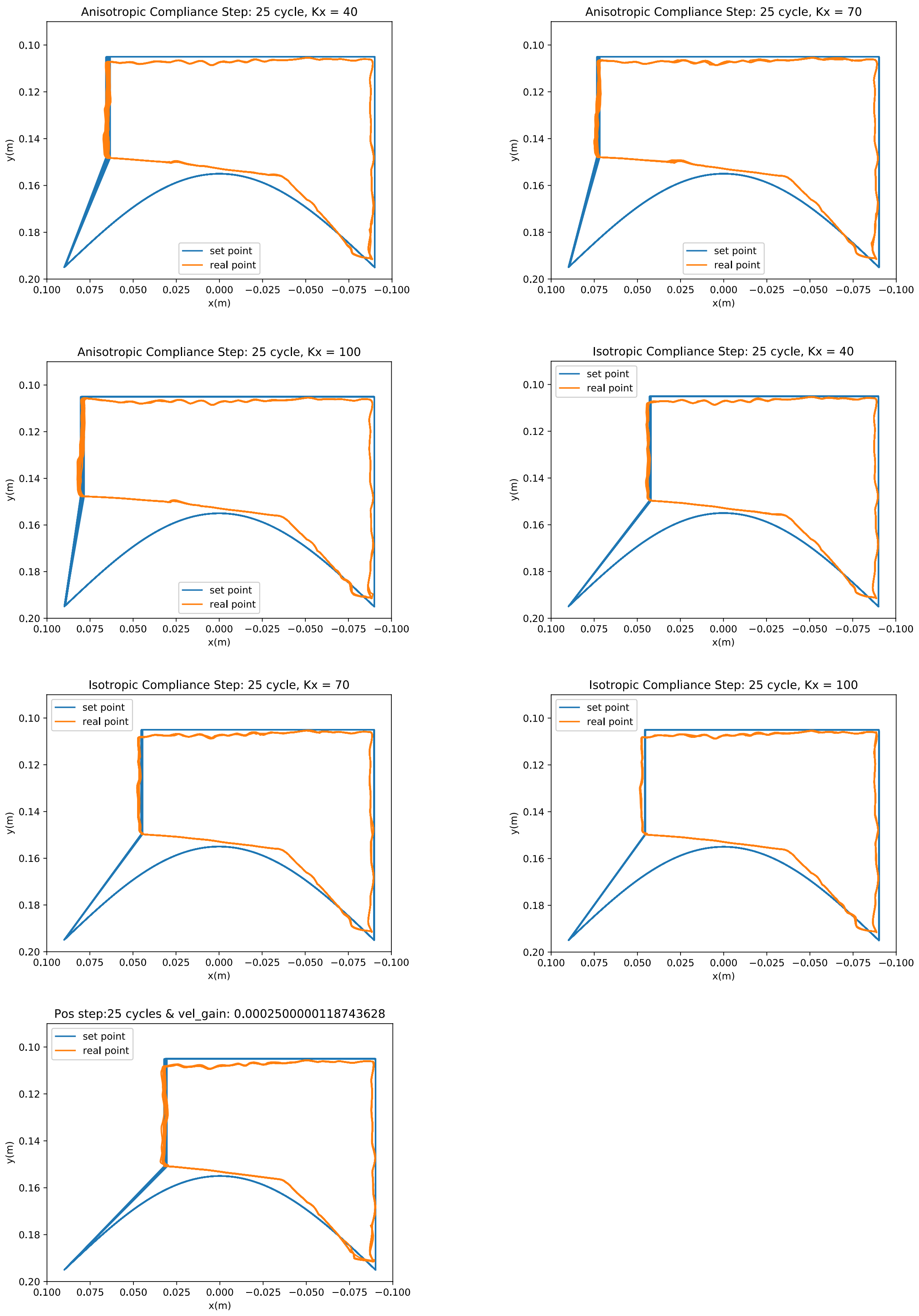


Figure 4.3: Obstacle height: 18mm (2 stacks), position: 25% of stride length

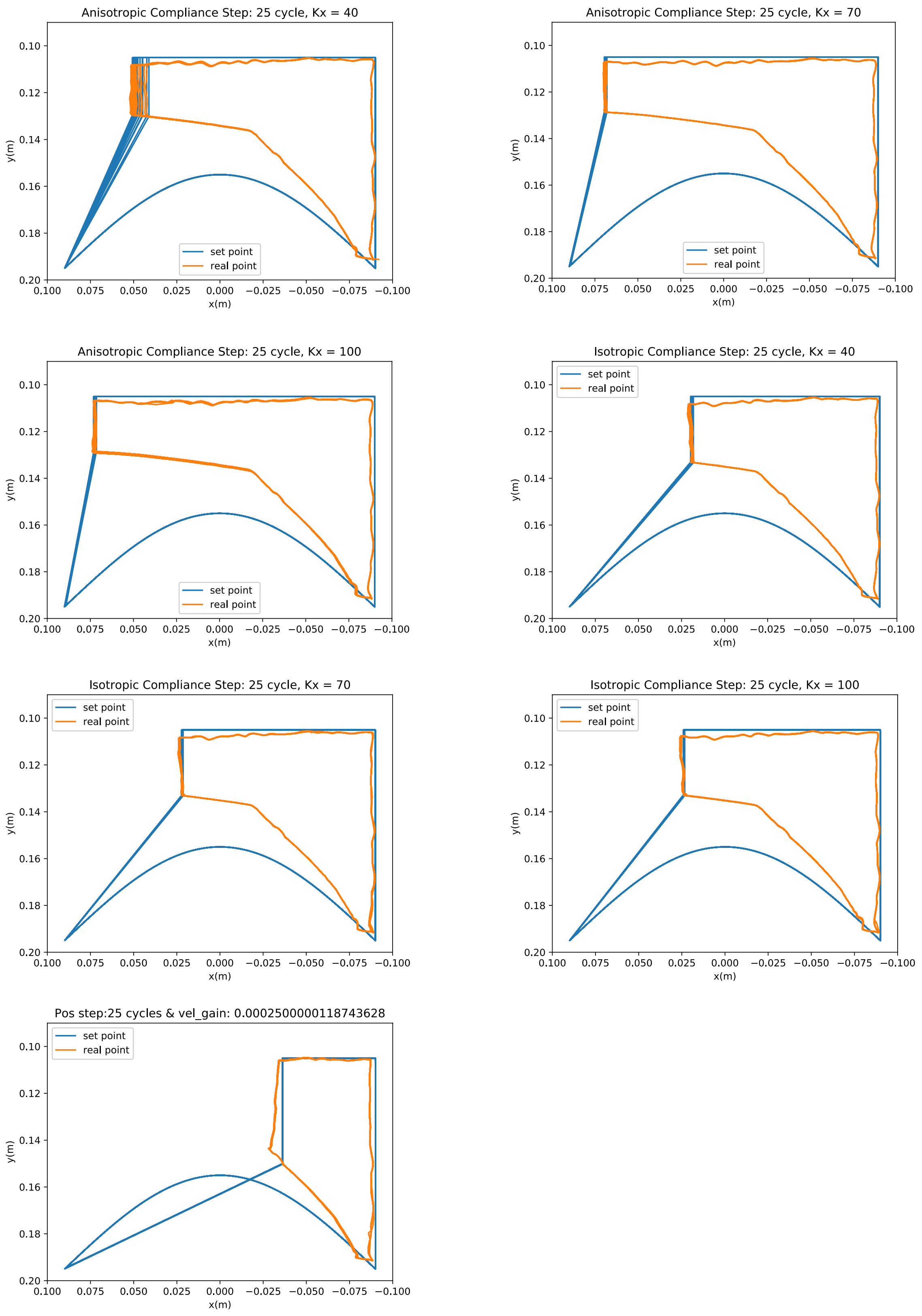


Figure 4.4: Obstacle height: 37mm (3 stacks), position: 30% of stride length

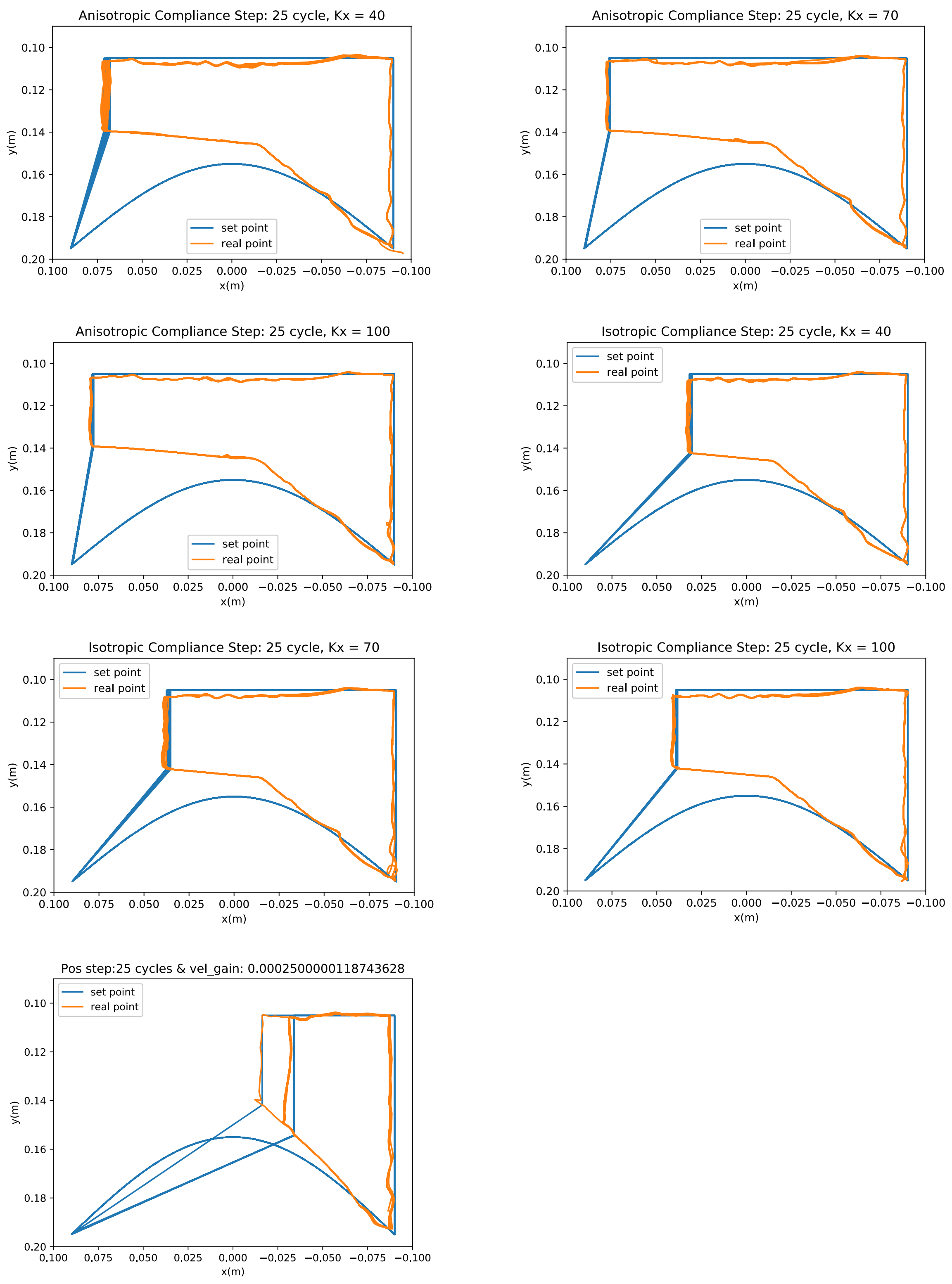


Figure 4.5: Obstacle height: 37mm (3 stacks), position: 35% of stride length

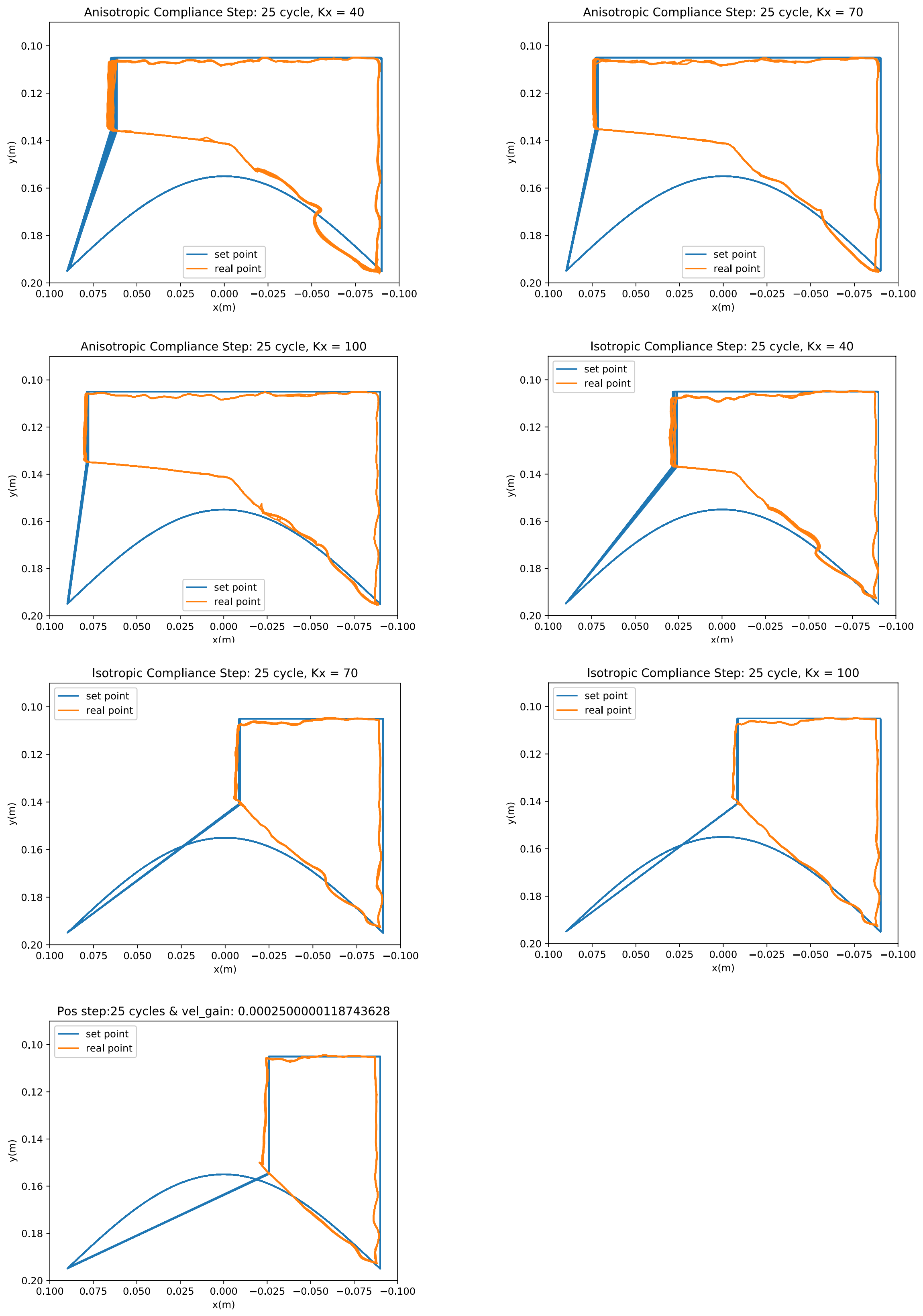


Figure 4.6: Obstacle height: 37mm (3 stacks), position: 40% of stride length

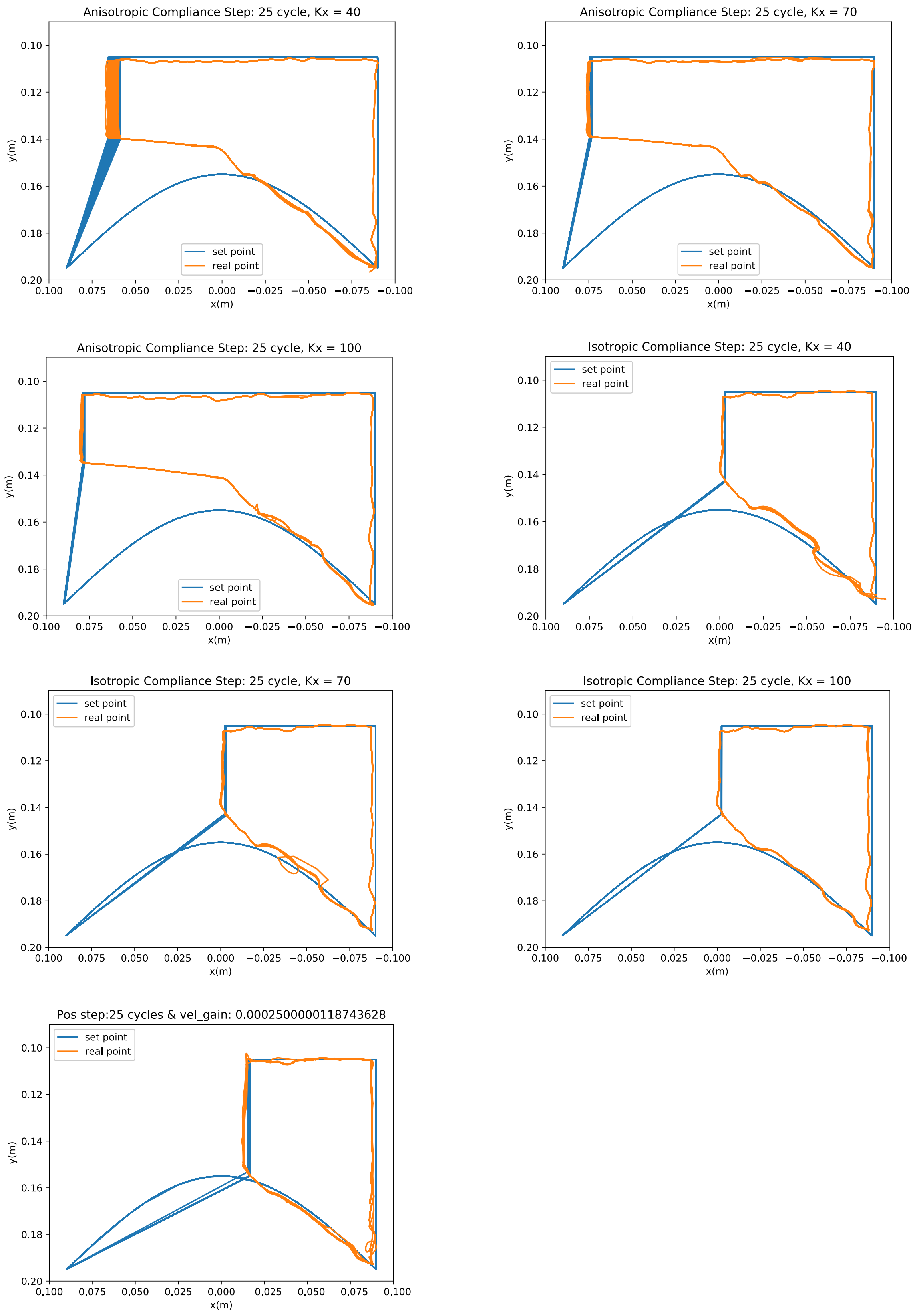


Figure 4.7: Obstacle height: 37mm (3 stacks), position: 45% of stride length

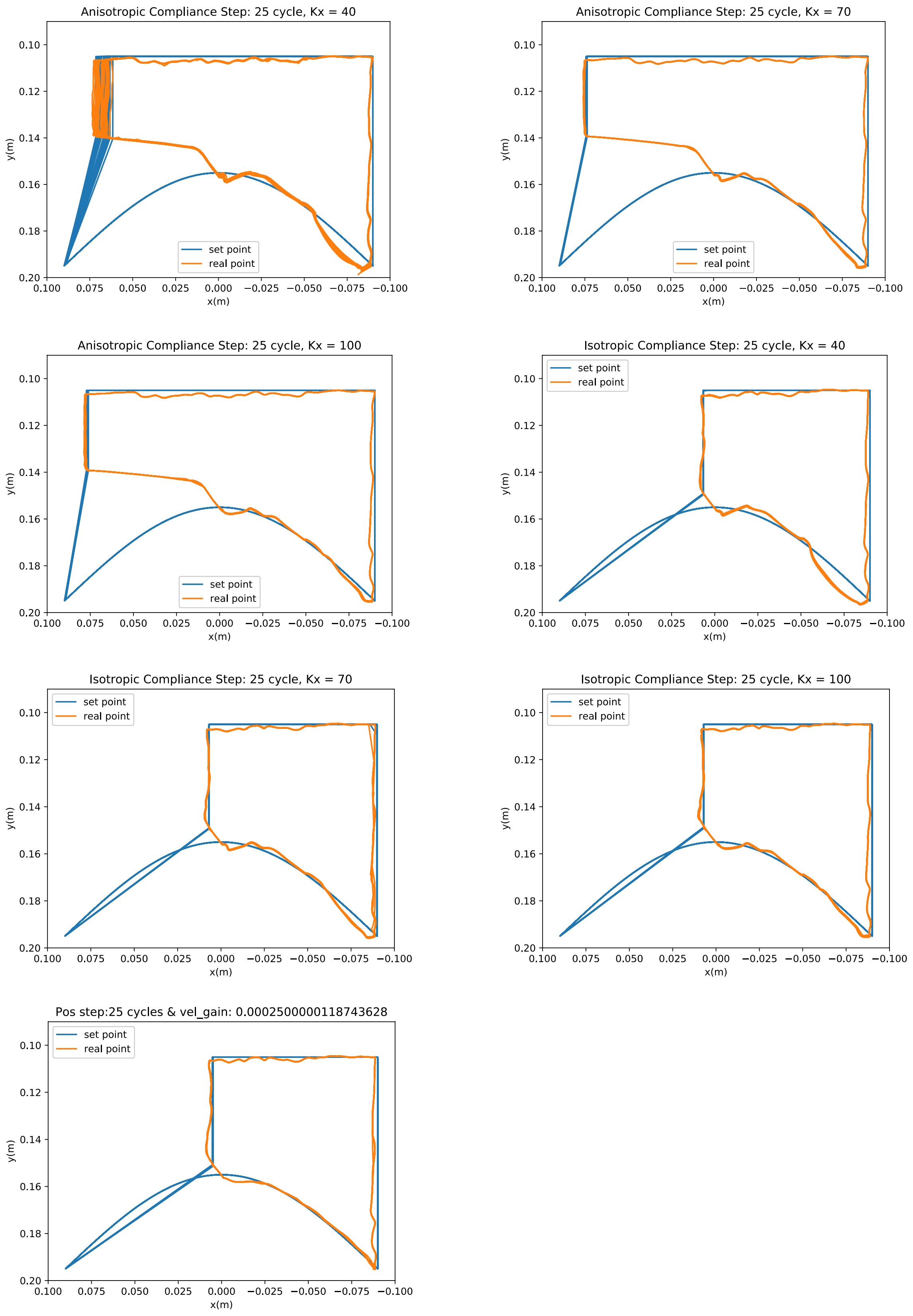


Figure 4.8: Obstacle height: 37mm (3 stacks), position: 50% of stride length

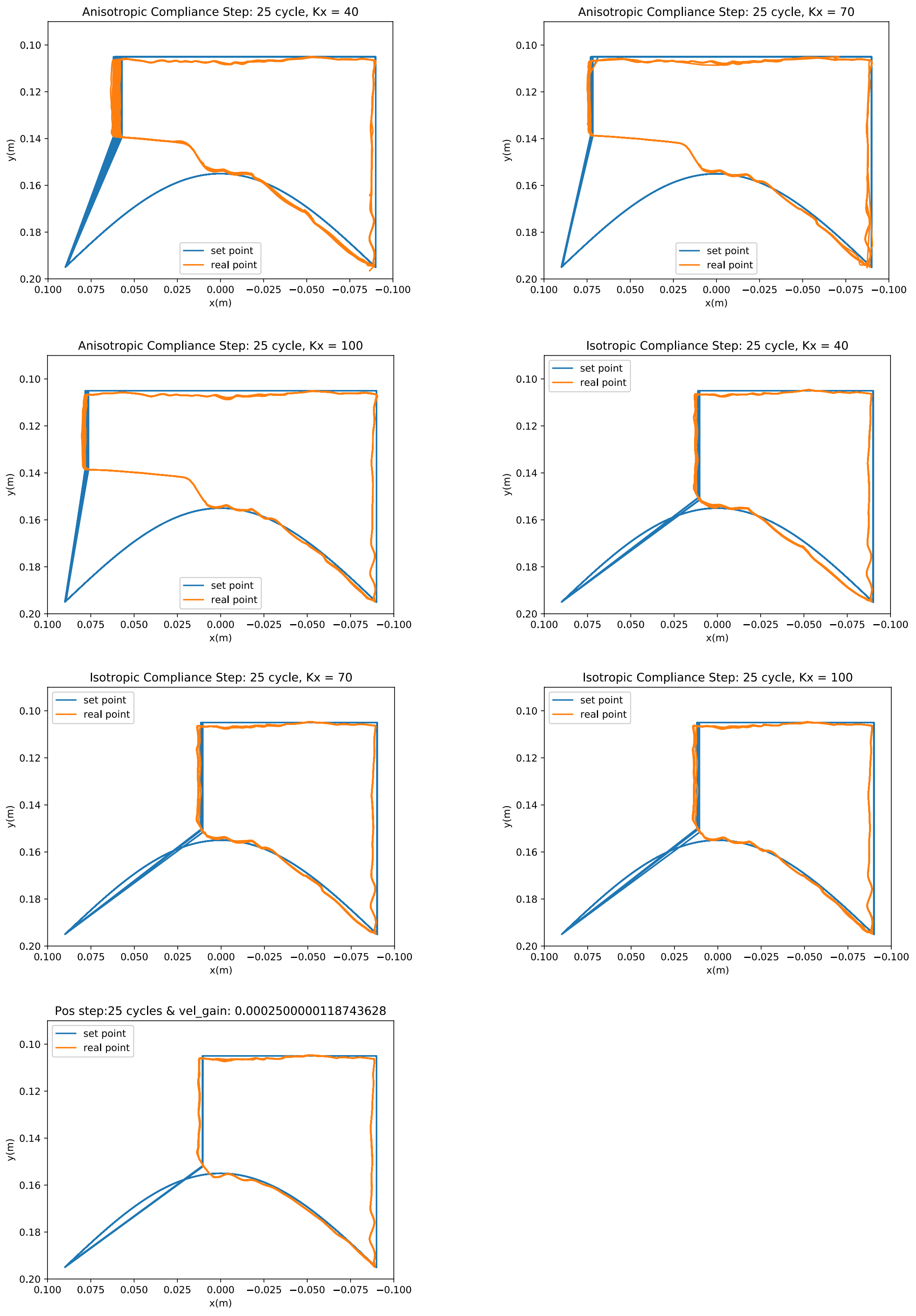


Figure 4.9: Obstacle height: 37mm (3 stacks), position: 55% of stride length

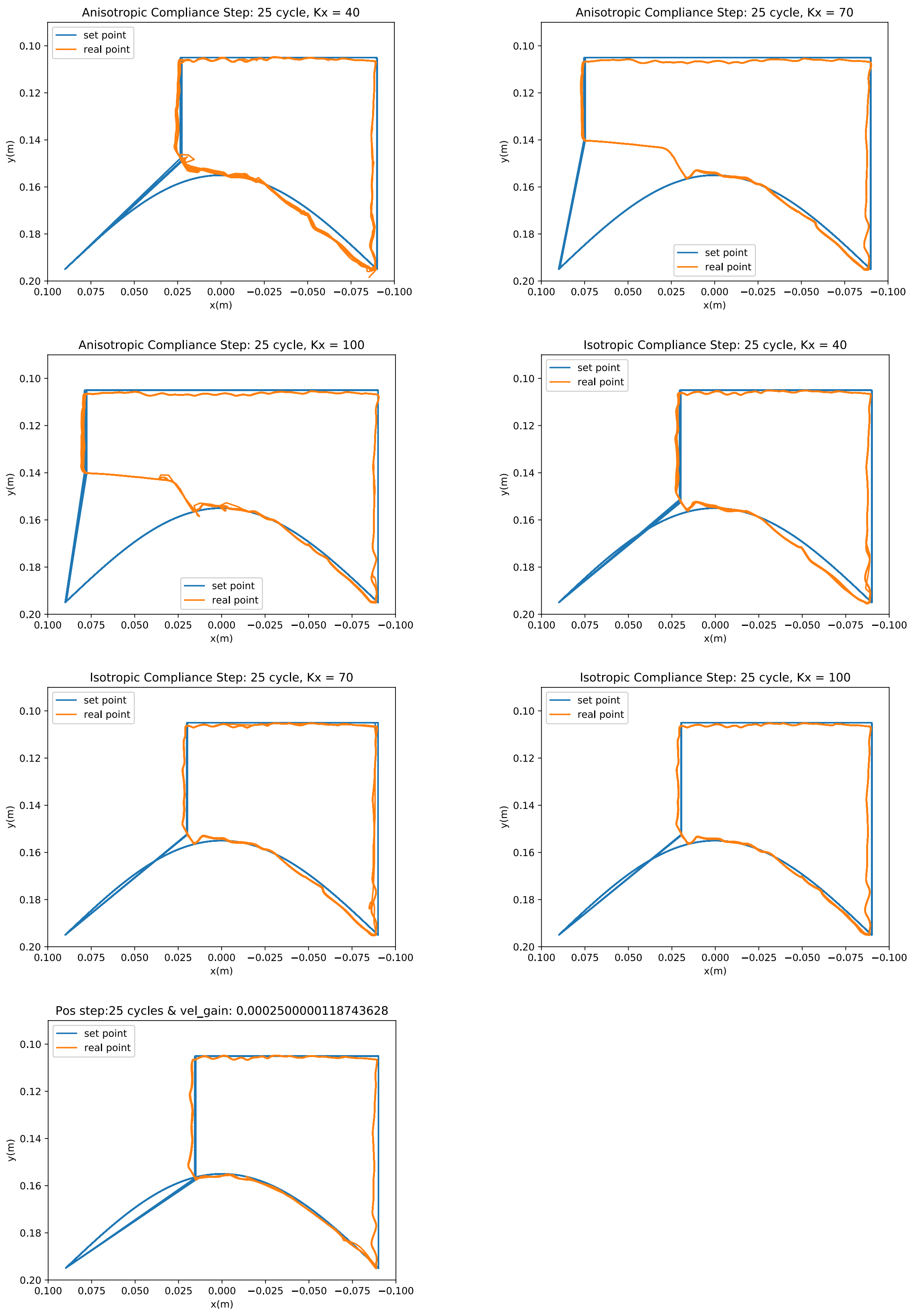


Figure 4.10: Obstacle height: 37mm (3 stacks), position: 60% of stride length

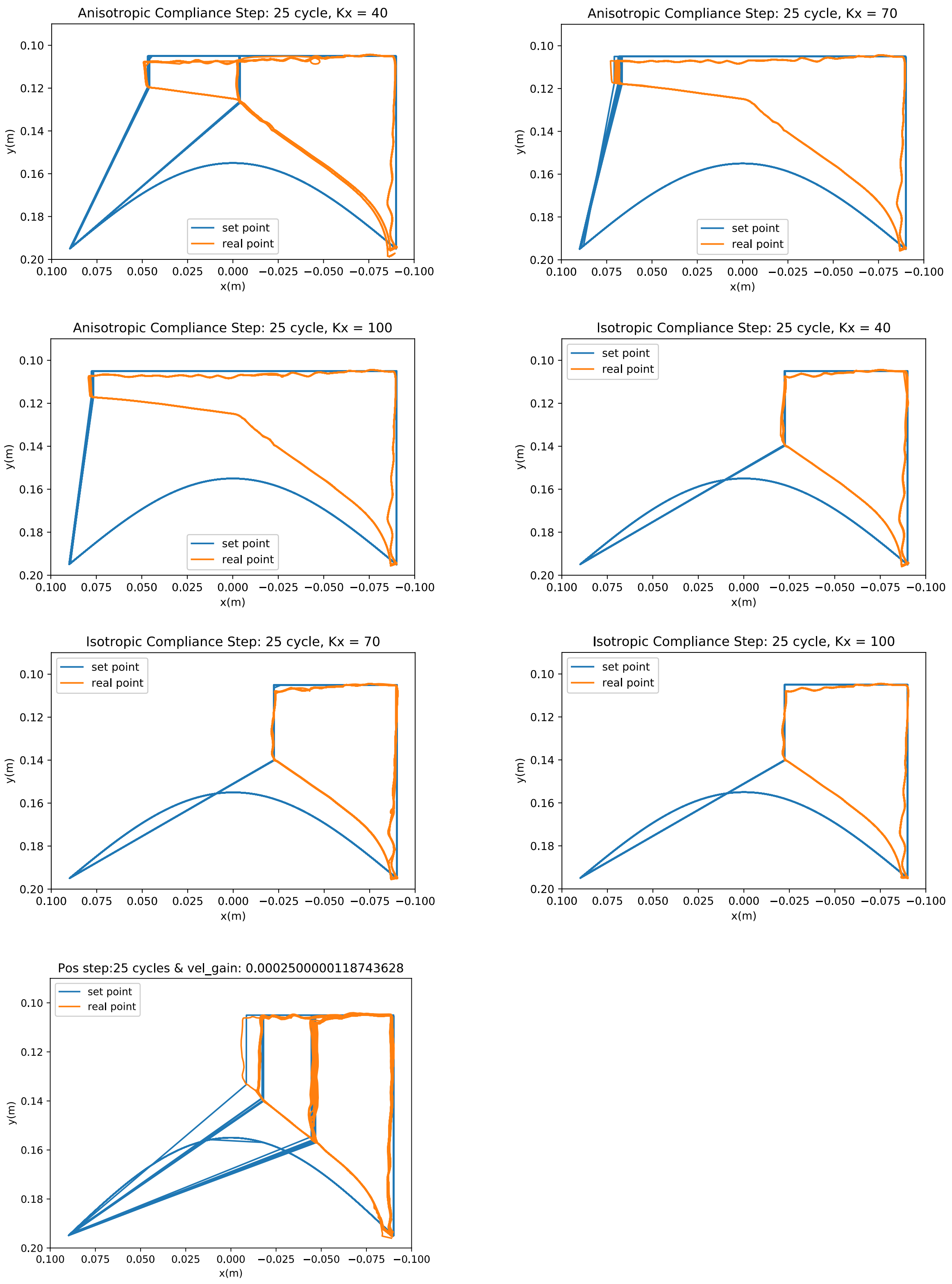


Figure 4.11: Obstacle height: 56mm (4 stacks), position: 40% of stride length

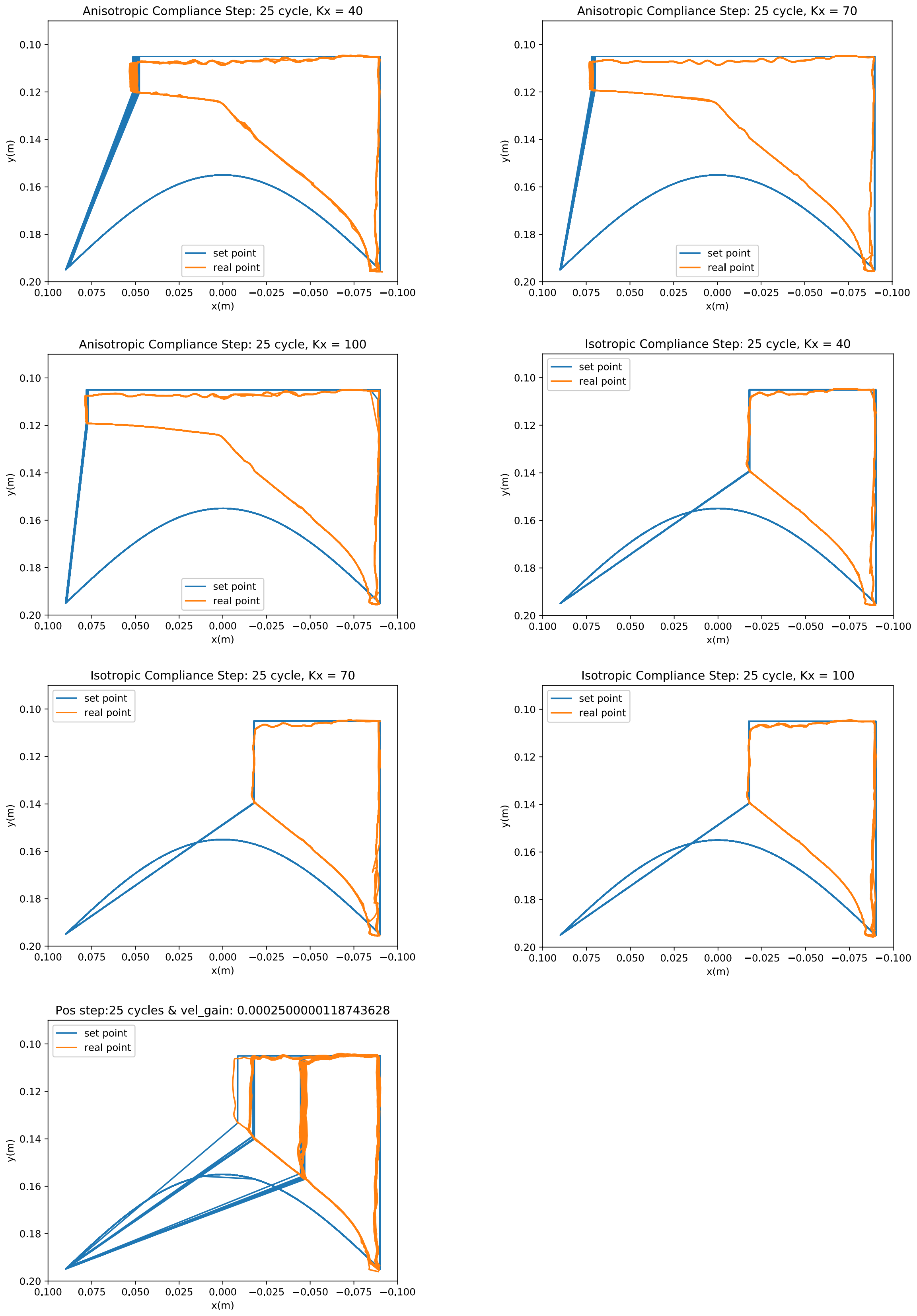


Figure 4.12: Obstacle height: 56mm (4 stacks), position: 45% of stride length

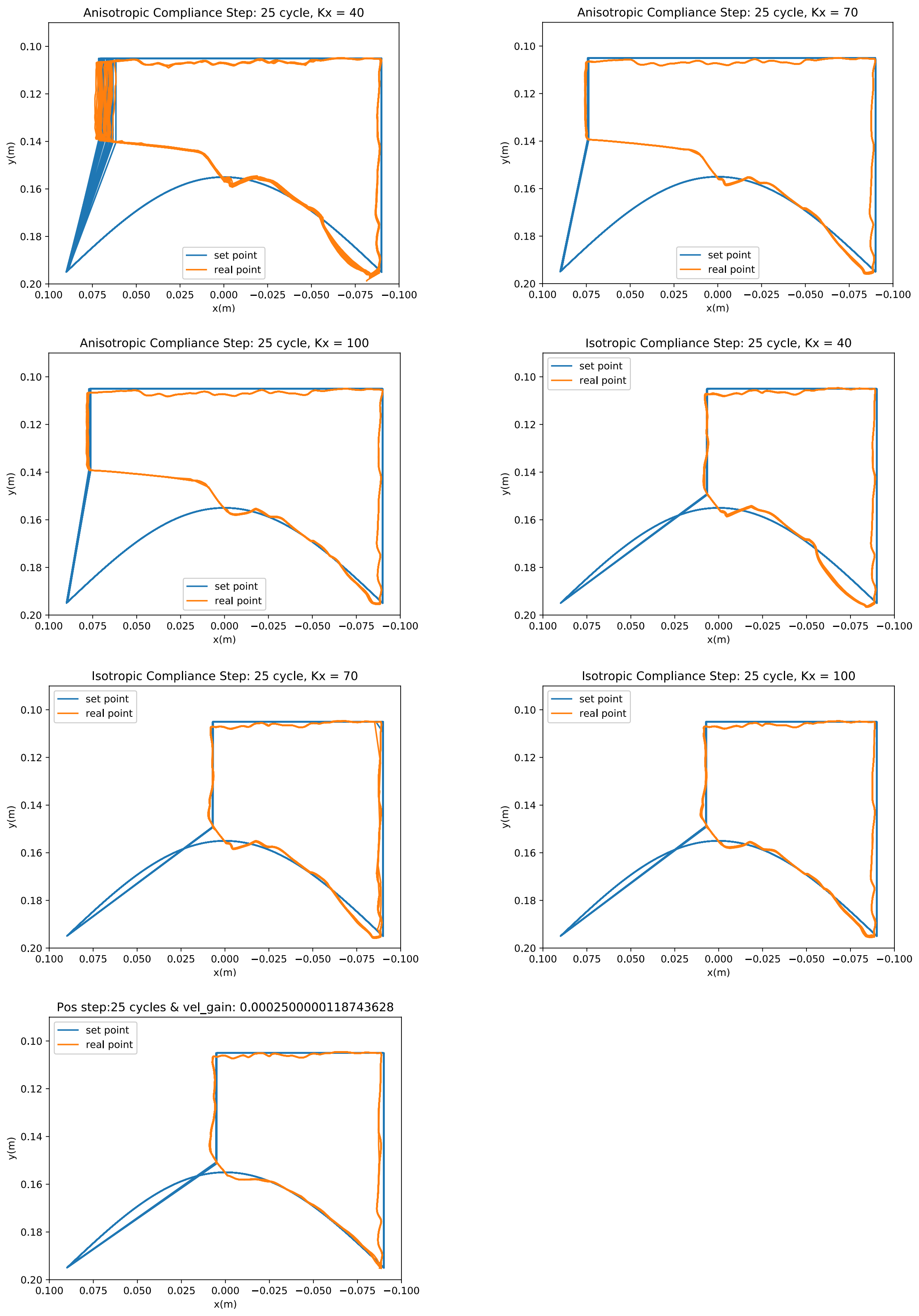


Figure 4.13: Obstacle height: 56mm (4 stacks), position: 50% of stride length

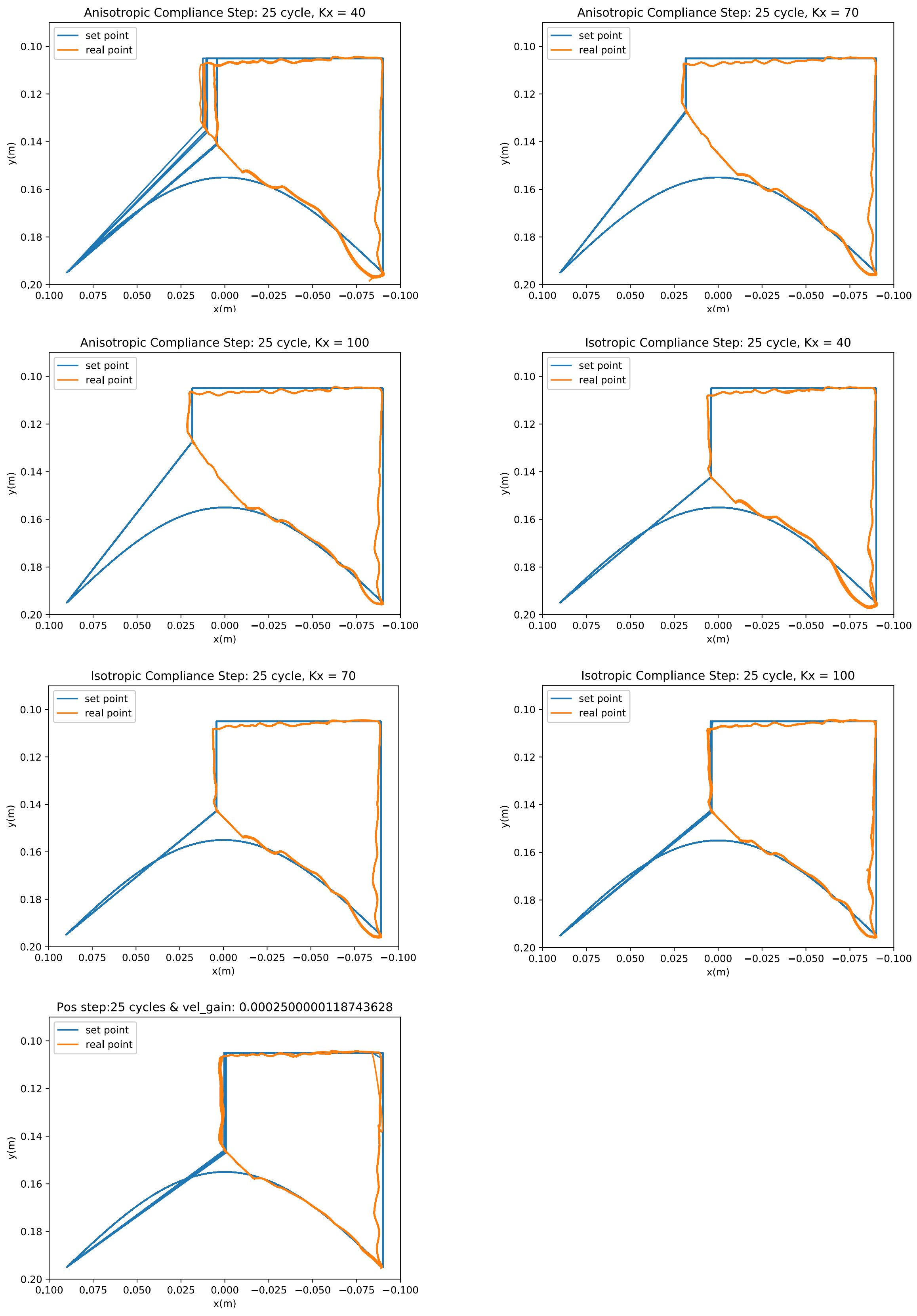


Figure 4.14: Obstacle height: 56mm (4 stacks), position: 55% of stride length

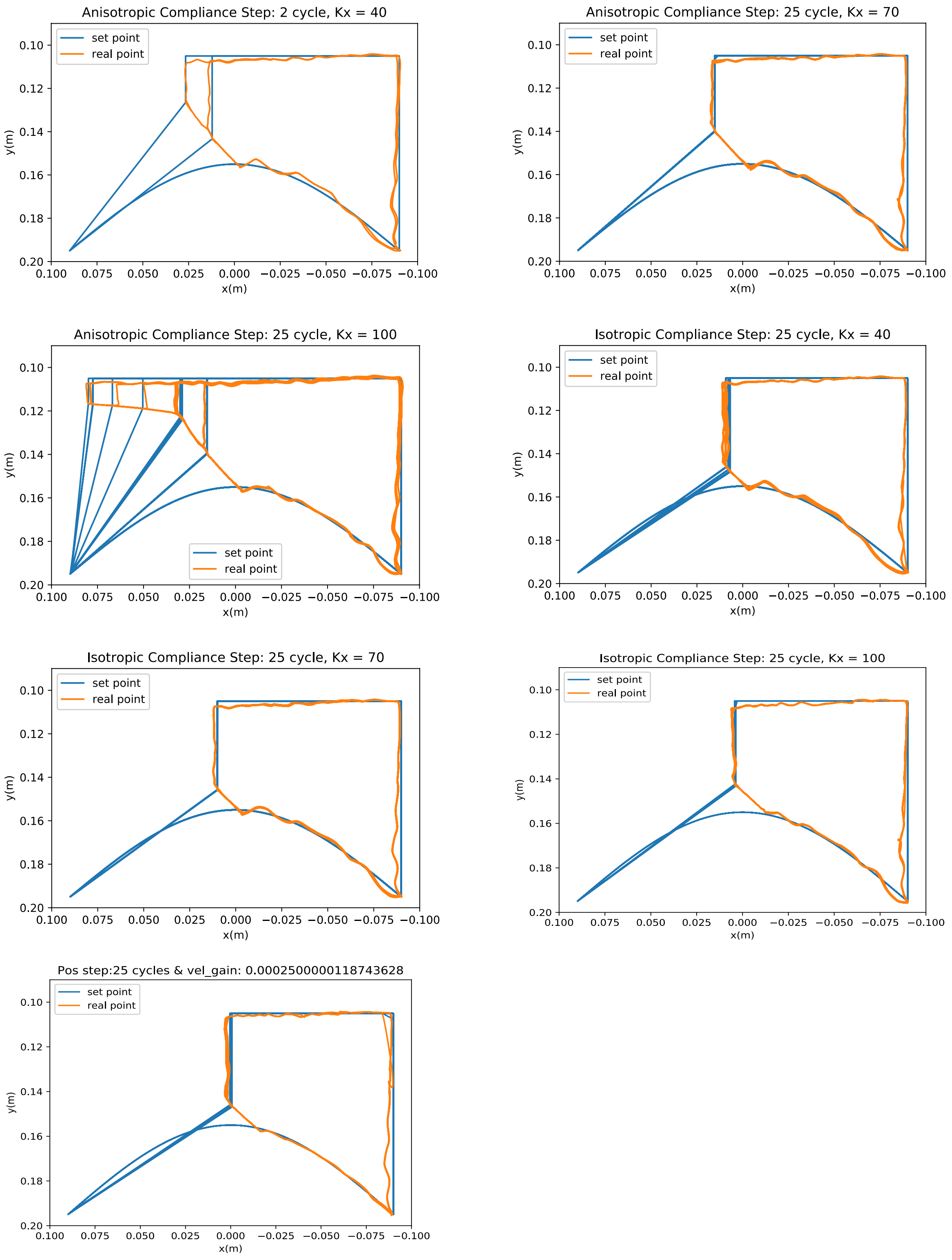


Figure 4.15: Obstacle height: 56mm (4 stacks), position: 60% of stride length

References

[1] A. Chilian and H. Hirschmüller, Stereo camera-based navigation of mobile robots on rough terrain, 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, 2009, pp. 4571-4576.

[2] J. K. Hodgins and M. N. Raibert, Adjusting step length for rough terrain locomotion, in IEEE Transactions on Robotics and Automation, vol. 7, no. 3, pp. 289-298, June 1991.

[3] Hauser, K., Bretl, T., Latombe, J.-C., Harada, K., & Wilcox, B. (2008). Motion Planning for Legged Robots on Varied Terrain. The International Journal of Robotics Research, 27(11–12), 1325–1349.

[4] N. C. Heglund, C. R. Taylor, and T. A. McMahon, Scaling stride frequency and gait to animal size: mice to horses, Science, vol. 186, no. 4169, pp. 1112–1113, 1974.

[5] Pijnappels M, Bobbert MF, van Dieën JH. Changes in walking pattern caused by the possibility of a tripping reaction. Gait Posture. 2001;14(1):11-18.

[6] Kuo, A. D. (January 11, 2001). A Simple Model of Bipedal Walking Predicts the Preferred Speed–Step Length Relationship. ASME. *J Biomech Eng*. June 2001; 123(3): 264–269.

[7] Marsh, Richard L., David J Ellerby, Jennifer A. Carr, Havalee T. Henry and Cindy I. Buchanan. Partitioning the energetics of walking and running: swinging the limbs is expensive. *Science* 303 5654 (2004): 80-3.

[8] P. Čížek, J. Kubík and J. Faigl, Online Foot-Strike Detection Using Inertial Measurements for Multi-Legged Walking Robots, 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, 2018, pp. 7622-7627.

[9] Pearson, K.G., and R. Franklin. Characteristics of Leg Movements and Patterns of Coordination in Locusts Walking on Rough Terrain. The International Journal of Robotics Research 3, no. 2 (June 1984): 101–12.

[10] T. Ishikawa, Y.Kojio, K. Kojima, S. Nozawa, Y. Kakiuchi, K.Okada and M. Inaba Bipedal walking control against swing foot collision using swing foot trajectory regeneration and impact mitigation, 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, 2017, pp. 4531-4537.

[11] Zucker, Matt, Nathan Ratliff, Martin Stolle, Joel Chestnutt, J Andrew Bagnell, Christopher G Atkeson, and James Kuffner. Optimization and Learning for Rough Terrain Legged Locomotion. The International Journal of Robotics Research 30, no. 2 (February 2011): 175–91.

[12] D. Kanoulas and M. Vona, Bio-inspired rough terrain contact patch perception, 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, 2014, pp. 1719-1724.

[13] A. Albert, M. Suppa and W. Gerth, Detection of stair dimensions for the path planning of a bipedal robot, 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Proceedings (Cat. No.01TH8556), Como, Italy, 2001, pp. 1291-1296 vol.2.

[14] A. Chilian and H. Hirschmüller, Stereo camera-based navigation of mobile robots on rough terrain, 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, 2009, pp. 4571-4576.

[15] K. Okada, T. Ogura, A. Haneda and M. Inaba, Autonomous 3D walking system for a humanoid robot based on visual step recognition and 3D foot step planner, Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, 2005, pp. 623-628.

[16] Ö. Arslan and U. Saranli, Reactive Planning and Control of Planar Spring–Mass Running on Rough Terrain, in IEEE Transactions on Robotics, vol. 28, no. 3, pp. 567-579, June 2012.

[17] G. Kenneally, A. De and D. E. Koditschek, Design Principles for a Family of Direct-Drive Legged Robots, in IEEE Robotics and Automation Letters, vol. 1, no. 2, pp. 900-907, July 2016.

[18] N. Kau, A. Schultz, N. Ferrante, and P. Slade, Stanford doggo: An Open-Source, Quasi-Direct-Drive quadruped, in 2019 International Conference on Robotics and Automation (ICRA), May 2019, pp. 6309– 6315.

[19] W. Bosworth, S. Kim and N. Hogan, The MIT super mini cheetah: A small, low-cost quadrupedal robot for dynamic locomotion, 2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), West Lafayette, IN, 2015, pp. 1-8.

[20] C. Hubicki, A. Abate, P. Clary, S. Rezazadeh, M. Jones, A. Peekema, J. V. Why, R. Domres, A. Wu, W. Martin, H. Geyer, and J. Hurst, Walking and Running with Passive Compliance: Lessons from Engineering: A Live Demonstration of the ATRIAS Biped, in IEEE Robotics & Automation Magazine, vol. 25, no. 3, pp. 23-39, Sept. 2018,

[21] P. Arm, R. Zenkl, P. Barton, L. Beglinger, A. Dietsche, L. Ferrazzini, E. Hampp, J. Hinder, C. Huber, D. Schaufelberger, F. Schmitt, B. Sun, B. Stolz, H. Kolvenbach and M. Hutter, SpaceBok: A Dynamic Legged Robot for Space Exploration, 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 2019, pp. 6288-6294.

[22] M. P. Austin, J. M. Brown, C. A. Young, and J. E. Clark, Leg design to enable dynamic running and climbing on BOBCAT, in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2018, pp. 3799–3806.

[23] Brown, Jason M., Charlie P. Carbiener, John V. Nicholson, Nicholas Hemenway, Jason L. Pusey and Jonathan E. Clark. Fore-Aft Leg Specialization Controller for a Dynamic Quadruped. 2018 IEEE International Conference on Robotics and Automation (ICRA) (2018): 1-9.

[24] D. J. Blackman, J. V. Nicholson, J. L. Pusey, M. P. Austin, C. Young, J. M. Brown, and J. E. Clark, Leg design for running and jumping dynamics, 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO), Macau, 2017, pp. 2617-2623.

[25] D. J. Blackman, J. V. Nicholson, C. Ordonez, B. D. Miller, and J. E. Clark, Gait development on minitaur, a direct drive quadrupedal robot, in Unmanned Systems Technology XVIII, vol. 9837. Inter- national Society for Optics and Photonics, May 2016, p. 98370I.

[26] M. Austin, J. Brown, K. Geidel, W. Wang, and J. Clark, Gait design and optimization for efficient running of a direct-drive quadrupedal robot, in Unmanned Systems Technology XIX, vol. 10195. Interna- tional Society for Optics and Photonics, May 2017, p. 1019504.

[27] Hubicki, Christian, Jesse Grimes, Mikhail Jones, Daniel Renjewski, Alexander Spröwitz, Andy Abate, and Jonathan Hurst. ATRIAS: Design and Validation of a Tether-Free 3D-Capable Spring-Mass Bipedal Robot. The International Journal of Robotics Research 35, no. 12 (October 2016): 1497–1521.

[28] Meurer A, Smith CP, Paprocki M, Čertík O, Kirpichev SB, Rocklin M, Kumar A, Ivanov S, Moore JK, Singh S, Rathnayake T, Vig S, Granger BE, Muller RP, Bonazzi F, Gupta H, Vats S, Johansson F, Pedregosa F, Curry MJ, Terrel AR, Roučka Š, Saboo A, Fernando I, Kulal S, Cimrman R, Scopatz A. 2017. SymPy: symbolic computing in Python. PeerJ Computer Science 3:e103 https://doi.org/10.7717/peerj-cs.103

[29] S. van der Walt, S. C. Colbert and G. Varoquaux, The NumPy Array: A Structure for Efficient Numerical Computation, in Computing in Science & Engineering, vol. 13, no. 2, pp. 22-30, March-April 2011.

[30] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, I. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cim- rman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors, SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, Nature Methods, 2020.

[31] P. Kryczka, P. Kormushev, N. G. Tsagarakis and D. G. Caldwell, Online regeneration of bipedal walking gait pattern optimizing footstep placement and timing, 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, 2015, pp. 3352-3357.

[32] H. Kaminaga, J. Englsberger and C. Ott, Kinematic optimization and online adaptation of swing foot trajectory for biped locomotion, 2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012), Osaka, 2012, pp. 593-599.

[33] Nakanishi, Jun, Rick Cory, Michael Mistry, Jan Peters, and Stefan Schaal. Operational Space Control: A Theoretical and Empirical Comparison. The International Journal of Robotics Research 27, no. 6 (June 2008): 737–57.

[34] O. Khatib, A unified approach for motion and force control of robot manipulators: The operational space formulation, in IEEE Journal on Robotics and Automation, vol. 3, no. 1, pp. 43-53, February 1987.

[35] J. Nakanishi, M. Mistry and S. Schaal, Inverse Dynamics Control with Floating Base and Constraints, Proceedings 2007 IEEE International Conference on Robotics and Automation, Roma, 2007, pp. 1942-1947.

[36] Nicholas Roy; Paul Newman; Siddhartha Srinivasa, Hybrid Operational Space Control for Compliant Legged Systems, in Robotics: Science and Systems VIII, MITP, 2013, pp.129-136.

[37] Dudek, Daniel M, and Robert J Full. Passive mechanical properties of legs from running insects. The Journal of experimental biology vol. 209, Pt 8 (2006): 1502-15. doi:10.1242/jeb.02146

[38] Daniel M. Dudek, Robert J. Full An isolated insect leg's passive recovery from dorso-ventral perturbations Journal of Experimental Biology 2007 210: 3209-3217.

[39] D. M. Dudek, S. Dastoor, and R. J. Full, Rapid recovery from an impulse perturbation to a leg in running insects, in INTEGRATIVE Manuscript 2207 submitted to 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems. Received March 1, 2020. AND COMPARATIVE BIOLOGY, vol. 45. RD, CARY, NC ..., 2005, pp. 989–989.

[40] J. S. Gray, J. T. Hwang, J. R. R. A. Martins, K. T. Moore, and B. A. Naylor, OpenMDAO: An Open-Source Framework for Multidisciplinary Design, Analysis, and Optimization, Structural and Multidisciplinary Optimization, 2019.

[41] Kevin M. Lynch and Frank C. Park. 2017. Modern Robotics: Mechanics, Planning, and Control (1st. ed.). Cambridge University Press, USA.