A Methodology to Quantify Risk of Failure for Dynamic Robots

by

Albert D. Wang

S.M., Massachusetts Institute of Technology (2012)
S.B., Massachusetts Institute of Technology (2010)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Sept 2019

© Massachusetts Institute of Technology 2019. All rights reserved.

Signature redacted

Author ..................

Department of Mechanical Engineering
August 19, 2019

Signature redacted

Certified by ..................

Sangbae Kim
Associate Professor
Thesis Supervisor

Signature redacted

Accepted by ..................

Nicolas Hadjiconstantinou
Chairman, Department Committee on Graduate Theses
A Methodology to Quantify Risk of Failure for Dynamic Robots

by

Albert D. Wang

Submitted to the Department of Mechanical Engineering on August 19, 2019, in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

Humans possess an innate sense of danger that guides the execution of extraordinary dynamic maneuvers. They can also use this sense to generate creative recovery strategies to eventually come to a safe stop. This capability is not yet available to robots, fundamentally because there is no clear metric that represents the quantified risk of failing. Possessing such a metric would allow robots to explore their dynamic capability up to their physical limitations. This thesis attempts to address this problem by introducing a methodology to quantify the risk of failure for dynamic robots. It employs a sampling-based network constructed using the principles of viability theory, which focuses on the avoidance of failure instead of the regulation to specific movements. Simplifications that specifically target complex hybrid systems are explored to extend the usage of viability theory for practical application to legged robots. The results of this methodology are the Viable State Network, a network showing the non-failing and failing states, and the Risk Map, the quantified risk of failure. These concepts are demonstrated for a planar hopping robot model.

Thesis Supervisor: Sangbae Kim
Title: Associate Professor
Acknowledgments

I would like to first thank my advisor Sangbae Kim. It has been a pleasure to study with you. What an incredible journey! When I first inquired about an undergraduate thesis project, I never could have imagined what you would have challenged me to accomplish in the last decade. Thank you for your support and encouragement to pursue the hardest problems. As I leave your direct guidance, I hope to carry on your vigor for my future career.

My thesis committee of Neville Hogan, Alberto Rodriguez, and Scott Kuindersma have been an incredible help during this project. Thank you for your insights, you help in uncovering my blind spots, and life advice.

During my time in the lab, I have had the fortune of working alongside colleagues who are some of the most intellectually stimulating people I have met. To Will Bosworth, Matt Haberland, Jongwoo Lee, Carmen Graves, Yichao Pan, Sangin Park, John Mayo, Wyatt Ubellacker, Gerardo Bledt, Erich Meinig, Ben Katz, Andrew Saloutos, Sid Trehan, Chiheb Boussema, Daniel Carballo and Donghyun Kim, thank you. Sangok Seok and Haewon Park, thank you for your mentorship to shape me into the person I am now. To Patrick Wensing, I consider you to be my second advisor. Thank for your advice, both in technical subjects and in life. Two special thankyous go to Michael Chuah and João Ramos, who have become brothers that I never had. Michael, it has been wonderful to have you as a friend and labmate every step of the way through graduate school. I will always remember fondly our years of working hard and playing hard. João, you are the most industrious person I know and a true inspiration. I never would have been able to complete a project like HERMES without you. Traveling the world together with our robot has been one of the most memorable experiences during graduate school.

My research has been enhanced by the work of several undergraduate students who I have mentored over the years. Justin Cheung, Stacy Mo, Nancy Ouyang, Sam Ingersoll, Evan Brown, and George Roudebush, thank you for all the hard work. It has been wonderful to see you grow and flourish. I hope you have learned from me...
as I have from you.

The HERMES project was a highlight of the first half of the PhD. John Mayo, Wyatt Ubellacker, and João Ramos, it was pleasure to work alongside you. The project and our ambitious demos were made possible due to the help of Dave Otten, John Freidah, Alissa Mallinson, Daniel Herrick, Joseph MacLeod, Tasker Smith, Steve Haberek and the Pappalardo Lab staff.

For this thesis, I have enjoyed the support and advice from colleagues and friends. I would like to thank Patrick Wensing, Michael Yu, Jeremy Lai, Chiheb Boussema, Yiping Liu, Ben Katz, Sid Trehan, Nima Fazeli, Donghyun Kim, Meghan Huber, Jeffery Xing and Fan Liu. Your drive and excitement have helped me to keep pushing through the hard times. I would also like to acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing resources that have contributed to the results reported within this thesis. Both Julie Mullen and Lauren Milechin have been a great help in adapting my code and educating me on the intricacies of high performance computing.

Many staff have made my time at MIT a richer experience. Alexandra Cabral, thank you for all that you do for the lab. Harrison Chin, David Carrasco, and Chip McCord, I always enjoy your excitement and life advice when we meet in the halls. Leslie Reagan, you are the rock of the department. You seem to have a sixth sense for supporting students. Over the years, I have found support, motivation and compassion from you. Thank you.

I would like to thank all of my friends and colleagues at MIT and around the world who have been an integral part of my life outside of the lab. I hope to see you all soon.

Finally I would like to thank my parents and family for providing me with the foundations and support during my education to achieve my dream of attending MIT. I hope I have made you proud.
# Contents

## 1 Introduction

1.1 Motivations ........................................... 23

1.2 HERMES tele-operated humanoid ....................... 26

  1.2.1 HERMES as a motivation for quantifying risk of failure ........ 26

  1.2.2 Introduction of the HERMES system .................... 26

  1.2.3 Principles of full-body teleoperation .................. 30

  1.2.4 Design principles of the HERMES humanoid robot ....... 34

  1.2.5 Experiment ....................................... 37

  1.2.6 Conclusion ...................................... 42

1.3 Related work ......................................... 44

1.4 Non-regulation approaches inspired by Viability Theory ........ 46

1.5 Thesis overview and contributions ........................ 48

## 2 A Process to quantify risk in dynamic systems 

2.1 Data-driven, simulation based network ................... 51

  2.1.1 Node placement at hybrid transitions boundaries ........ 56

  2.1.2 Binning ......................................... 61

  2.1.3 Exploration and processing ................................ 63

2.2 Quantification of risk .................................. 66

  2.2.1 Absorbing Markov Chains ............................... 66

  2.2.2 Additional noise considerations ........................ 69

2.3 Manipulation of the network post creation ................. 70

2.4 Discussion ........................................... 71
3 Study of the planar hopping robot model

3.1 Dynamic model ............................................. 75
3.2 Constructing the Viable State Network ..................... 78
3.3 Exploration results and construction of the VSN .......... 80
3.4 Risk Labeling .................................................. 86
3.5 Observations on the VSN and Risk Map ................... 92
3.6 Variations on controlled behavior ............................ 103
3.7 Conclusions .................................................. 108

4 Extensions ............................................................ 109

4.1 Usage in control .............................................. 109
4.2 Whole capability analysis ..................................... 111
4.3 Adapting to Pogo-bot hardware ........................... 114
4.3.1 Modifications to the design parameters ................ 120

5 Conclusion .......................................................... 123

5.1 Contributions ................................................ 123
5.2 Limitations .................................................... 125
5.3 Future work .................................................. 125
List of Figures

1-1  The HERMES humanoid system showing the operator and humanoid robot. The operator is standing inside the balance feedback mechanism and is wearing the upper body motion capture suit. ........................................ 28
1-2  Human hip displacement due to step input of proprioceptive stimulus force on hips and visual stimulus .............................................................. 31
1-3  Block diagram of HERMES system with human-in-the-loop feedback. The bold arrows show the balance feedback loop in which the operator uses non-visual force feedback on the waist in order to stabilize the robot. The open arrows show the feedback path of visual data and higher level cognitive planning. ......................................................... 32
1-4  Flow diagram showing the expected sequence of events for human in the loop balance feedback .............................................................. 33
1-5  CAD drawing of the HERMES humanoid highlighting the power plane on the upper right limb .............................................................. 35
1-6  Cutout view of the HERMES robot composite leg design .................. 35
1-7  HERMES humanoid robot foot showing the location of the 3 load cells. 36
1-8  Arm of the HERMES humanoid robot showing the location of proximal motors driving Bowden cable that actuate the fingers. ....................... 36
1-9  Hand of the HERMES humanoid robot showing a grasp of a power drill and a fist. Note that the fingers can independently actuate the drill power switch. .............................................................. 37
1-10 The trajectory of the robot CoP inside of the support polygon from 0 to 4 seconds after the impact disturbance. ........................................ 38
1-11 Time trajectory of multiple signals during a impact disturbance on the robot. Signals are normalized and bias is removed to aid in visualizing the propagation of information. (a) Unexpected impact disturbance on robot (b) Unexpected impact disturbance on robot zoomed in to show the propagation of signals through the system (c) Expected impact disturbance on robot

1-12 Sequence showing the HERMES system breaking through a wall. The safety ropes on humanoid are loose. (a) Starting position (b) Operator commands the robot to grab the frame (c) Operator commands a punch (d) Robot pushes off the wall with balance feedback enabled (e) Operator restabilizes the robot with balance feedback

1-13 Abstract representation of reachable and viable states. Black dots are example points in state space and green arrows represent control actions between pairs of states. The terminating states are user defined. Viable states are states in which there exists a control action that keeps the robot in the set of viable states. Conversely, if there is a state that leads to an inevitable failure at a terminating condition, then that state is non-viable. Within the viable space lies the statically stable states as well as the quasi-static controllers that operate in a conservative region. Clearly, from observation of biological systems, there exist highly dynamic and interesting behaviors that may be available to robots. These lie in the yet unexplored viable regions and are not typically captured by linear controllers that regulate the system around fixed points or static stability.

1-14 Abstract representation of reachable states, viable states, home set, and home set reachable states. For this thesis, the capability is defined to be the intersection of the reachable, viable and backwards reachable states which is shown by the shading.
2-1 A falling block which experiences a plastic collision upon impact. The velocity after impact is zero but it is not possible to recover the velocity prior to impact. ............................................................. 53

2-2 The pogo-bot is a prismatic inverted pendulum with a reaction wheel and massless leg. The leg can apply unidirectional linear force and the reaction wheel can apply torques on the body. ......................... 56

2-3 Hybrid transition node types for the pogo-bot. ....................... 57

2-4 Event types for the pogo-bot. The hybrid contact transition boundary is crossed at takeoff and landing. The pre-takeoff and post-takeoff states refer to the state of the robot prior and post transition. Similarly during landing, there are pre-impact and post-impact states around the impact transition. Solid arrows indicate ground contact dynamics. Dashed arrows indicate flight phase dynamics. ......................... 58

2-5 Hybrid transition node types for a biped ............................ 59

2-6 IMU ringing at hybrid transitions. In this experiment a hardware prototype of the pogo-bot was dropped in the aerial phase and the robot completed a jump. Impact with the ground occurred at 250 ms and takeoff occurred at 1100 ms. The MEMS IMU produces erroneous readings shortly after the impact event and therefore cannot be relied upon for state estimation. If the control of the ground contact phase were dependent on the state estimation at the time of impact, the ringing would likely cause poor decision-making by the robot. ........ 60

2-7 Example gridding of the space of pre-takeoffs into bins ........... 62

2-8 Drunkard’s walk example of the Absorbing Markov Chain. The home state and the pub are absorbing states. Once entered it is not possible to leave. This classic textbook example illustrates the likelihood of ending in any of the absorbing states when starting in one of the transient states. ................................................................. 66

2-9 The absorbing Markov chain formulation for the robot. This is analogous to the textbook example of the Drunkard’s Walk in Figure 2-8 . 67
2-10 The control block diagram of the robot. The feedback path inside of the shaded region labeled "Network Path Search" shows the dynamics that are implied by using the graph search of the Viable State Network. The feedback path labeled "True feedback path" includes the dynamics that the robot experiences with process noise or estimation noise.

3-1 The pogo-bot is a prismatic inverted pendulum with a reaction wheel and massless leg. The leg can apply unidirectional linear force and the reaction wheel can apply torques on the body.

3-2 (a) Control actions chosen for pogo-bot network. Each combination of force and torque is applied when the robot is in contact with the ground. (b) The robot applies a constant force and constant torque immediately following the impact event, through the ground phase, until the takeoff event.

3-3 Viable and non-viable states in the space of pogo-bot takeoffs. States shown in green are viable and have a path that leads to the home states. States shown in red are non-viable and will eventually fail.

3-4 Fewest number of steps from home state. Each subfigure shows states at pre-takeoff categorized by the shortest path (in steps) to the home states. The number above the plot indicates the fewest number of additional takeoffs before the robot can reach the home states. The coloring is for clarity of visualization. States are colored by distance from $\theta = \dot{\theta} = 0$. The color of a region of state space is consistent across all subfigures.
3-5 Fewest number of steps to guaranteed failure. The failure states are those indicated by subfigure (0), in which all sampled actions from these pre-takeoff states resulted in the robot reaching the failure condition, without reaching any home state or another successful takeoff. Each subfigure shows states at pre-takeoff categorized by the shortest path (in steps) to a failure state in subfigure (0). The number above the plot indicates the fewest number of additional takeoffs before the robot can reach a state in which all actions directly lead to the failure condition. The coloring is for clarity of visualization. States are colored by distance from $\theta = \dot{\theta} = 0$. The color of a region of state space is consistent across all subfigures.

3-6 Pre-takeoff states that will reach home states after the next landing. The coloring is for clarity of visualization. States are colored by distance from $\theta = \dot{\theta} = 0$. The color of a region of state space is consistent across all subfigures.

3-7 Four types of transitions for the generic pogo-bot.

3-8 In some trajectories, the robot will successfully land and pass through the set of home states. At this point, the robot has two choices. It can choose to stop or continue to apply the same action. If the robot chooses to stop, it will apply a controller that takes it to the starting states. If the robot continues the same control action, it may successfully takeoff again or fail before reaching takeoff. Each control action is split into two graph edges representing the two outcomes. The random control policy gives these edges equal weights. (a) Case in which the robot passes the set of home states and can then takeoff. (b) Case in which the robot passes the set of home states and can then fail if it does not choose to stop.

3-9 Risk map with the safety values of states in the takeoff space of the pogo-bot. The color value corresponds to log10(safety value) at each pre-takeoff state.
3-10 Risk map with the safety values of states in the takeoff space of the pogobot. The color value corresponds to log10(safety value) at each pre-takeoff state. The surrounding surface shown in light red corresponds to non-viable states.

3-11 Fewest number of steps to the home states. Each subfigure shows states at pre-takeoff categorized by the length of the shortest path (in steps) to the home states. The number above the plot indicates the fewest number of additional takeoffs before the robot can reach the home states. The color value corresponds to log10(safety value) at each pre-takeoff state.

3-12 Sections of the risk map plotted with the non-viable states. The color value corresponds to log10(safety value) at each pre-takeoff state. The surrounding points in light red are the non-viable states. This figure illustrates the difficulty of reasoning about boundaries and the safety values. (a) On the left side boundary, there is a gradient in safety values, going from high to low as the distance to boundary decreases. (b) On the left side boundary, there is no such gradient in safety values near the boundary. Viable states at the boundary have relatively high safety values $-2 < \log_{10}(safety \ value) < 0$.

3-13 Relationship between Cartesian distance margin to non-viability and safety value in the risk map. Cartesian distance is measured in the number of state space discretizations. (a) Scatter plot with the safety value on a linear scale (b) Scatter plot with the safety value on a log10 scale.
3-14 Relationship between Cartesian distance margin to non-viability and the safety value in the risk map. The coloring represents the relative proportion of viable states appearing in each histogram bin. Bins with no data-points are left blank. Cartesian distance is measured in the number of state space discretizations. (a) Histogram with the safety value on a linear scale (b) Histogram with the safety value on a log10 scale. ................................................................. 95

3-15 Illustration of capability reduction by loss of viability and loss of reachability. (a) A set of states and possible transitions in the VSN. (b) State B is removed for conservative control. (c) As a consequence of removing state B, state A becomes non-viable and state C becomes unreachable. ................................................................. 97

3-16 Effect of removing viable states within distance 1 to a non-viable state. The red states are those that are removed. The black states are additional states that become non-viable. The green states are the additional viable states that are no longer reachable. These plots correspond to data point A1 in Figure 3-20 ................................................................. 98

3-17 Effect of removing viable states within distance 4 to a non-viable state. The red states are those that are removed. The black states are additional states that become non-viable. The green states are the additional viable states that are no longer reachable. These plots correspond to data point A4 in Figure 3-20 ................................................................. 99

3-18 Effect of removing viable states with safety values less than 10^{-3.59}. The red states are those that are removed. The black states are additional states that become non-viable. The green states are the additional viable states that are no longer reachable. These plots correspond to data point B1 in Figure 3-20 ................................................................. 100
3-19 Effect of removing viable states with safety values less than $10^{-2.63}$. The red states are those that are removed. The black states are additional states that become non-viable. The green states are the additional viable states that are no longer reachable. These plots correspond to data point B4 in Figure 3-20.

3-20 Tradeoff in capability by removing viable states. A1-A8 denote instances in which all viable states within distance 1-8 of a non-viable (NV) state are removed. Distance is measured in the number of state space discretizations. B1-B8 denote instances in which all states below a threshold safety are removed. B1: safety threshold $10^{-3.59}$, B2: safety threshold $10^{-2.94}$, B3: safety threshold $10^{-2.70}$, B4: safety threshold $10^{-2.63}$, B5: safety threshold $10^{-2.55}$, B6: safety threshold $10^{-2.47}$, B7: safety threshold $10^{-2.41}$, B8: safety threshold $10^{-2.35}$. The dotted line represents the baseline loss in capability purely due to removing states.

3-21 Viable states and corresponding safety values for three different step length parameters. The color value corresponds to $\log_{10}(\text{safety value})$ at each pre-takeoff state.

3-22 Cumulative distribution of safety values for the transient pre-takeoff states shown for three different step length parameters.

3-23 Risk map with the safety value of states in the takeoff space of the pogobot. This risk map combines 3 risk maps that are generated for variations in the leg angle at landing. The color value corresponds to $\log_{10}(\text{safety value})$ at each pre-takeoff state.

4-1 Behavior of a risk-constrained controller. (a) Safety value minimum threshold $10^{-1}$. (b) Safety value minimum threshold $10^{-10}$. The robot starts at takeoff 0 in the standing position. In (a) the $\log_{10}(\text{safety})$ of 0 at steps 2.5, 4.5, 6.5 and 8.5 indicate that the robot chose to stop at a safe home state prior to the next takeoff.
4-2 Behavior of a risk gradient controller. The robot starts at takeoff state with safety value $10^{-12}$ and chooses the action that results in the safest next takeoff available.

4-3 Risk Maps and capability space comparisons for physical parameter variations. The first shows the Risk Maps for each model variation. The coloring represents the log10(Safety Value) of each state and is consistent across all of the Risk Maps. The second and third rows show operations on the set of viable states, for each model variation. In the second and third rows, the coloring is for clarity of visualization. States are colored by distance from $\theta = \dot{\theta} = 0$. The color of a region of state space is consistent across all of these subfigures.

4-4 Hardware pogo-bot

4-5 Actions sampled for the hardware pogo-bot model. Each point represents a single combination of force and torque profile that the robot applies during the ground contact phase.

4-6 Viable and non-viable states in the space of pogo-bot takeoffs for the hardware model. States shown in green are viable and have a path that leads to the home states. States shown in red are non-viable and will eventually fail. The set of states shown in black do not have a path leading to the home state but can avoid failure by remaining indefinitely within the set. For the purposes of this research, these states are considered non-viable.

4-7 Computed Risk Map for the hardware pogo-bot. The color value corresponds to log10(safety value) at each pre-takeoff state.

4-8 Computed Risk Map for the hardware pogo-bot. The color value corresponds to Safety value at each pre-takeoff state.

4-9 Computed Risk Map for the hardware pogo-bot with 20% body inertia compared to the original hardware model. The color value corresponds to log10(safety value) at each pre-takeoff state.
4-10 Computed Risk Map for the hardware pogo-bot 10% more body mass compared to the original hardware model. The color value corresponds to $\log_{10}(\text{safety value})$ at each pre-takeoff state.
**List of Tables**

1.1 Table of time constants for components in system. 33
1.2 Onset of signal change for unexpected impact at time t 41

3.1 Pogo-bot model parameters 76
3.2 Pogo-bot control parameters 78
3.3 Nominal Pogo-bot exploration parameters 79
3.4 Pogo-bot exploration and propagation summary 82
3.5 Variations in step length 103
3.6 Variations in step length 104

4.1 Hardware Pogo-bot model parameters 114
4.2 Hardware Pogo-bot exploration and propagation summary 117
Chapter 1

Introduction

1.1 Motivations

When observing the wonderfully complex and dynamic behaviors seen in legged animals, it is clear that there exist a multitude of limitations that must be overcome in order for engineered machines to match the sophisticated behaviors that are seen in nature. One major identified limitation in robots is the understanding of risk of failure, something that humans and animals seem to possess with ease. This thesis provides a guide to a methodology that can be used as a starting point towards robots being able understand risk, which may contribute to robots exceeding the capabilities of animals in the future.

The motivation for this thesis can be viewed from several perspectives and categorized into three major thrusts. First, we would like to extend the capability of engineered robots to match and exceed that of the animals that serve as inspiration. This requires the understanding of which motions truly lead to failure. Second, the sequencing of motions and fluidity in blending multiple goals during complex tasks suggest that animals and humans possess intuition and triggers. One of these may be the continuous evaluation of the risk of failure. We would like to enable robots to do the same. Third, we would like to construct a risk metric that serves as a generic language between systems, a quantity that is system independent and provides fair comparison and understanding between systems that may not share the
same topology or architecture.

The first motivation is to enable robots to carry out a wider range of behaviors. The current state of the art weakly couples the desired abilities of the robot with the mechanical design. However, by definition, the behaviors that can be achieved on a robot are a subset of its full abilities. In many cases, control designers run the robot in situations that are overly conservative compared to its true ability. Like a child learning how to walk, there are many instances in which the full capability is bounded by hardware limitations and it is up to the controller to access abilities up to those limits. If robots were to be aware of their own limitations, then they would be able to explore their maximal physical potential while remaining as safe as required. Additionally, if the robot is in a state that is known to result in failure, it can attempt actions that reduce physical damage \cite{16}. Another interpretation of this motivation is to propose a method that can find the capability of the robot independent of specific controllers.

Understanding the boundaries of robot capability can also benefit the design of robots. Currently, hardware is designed too conservatively in order to accommodate the worst case scenarios that the designer can imagine. In such cases, there are potentially many creative behaviors that the designer did not imagine and can be used if identified. With the understanding of robot capability, hardware designs can instead be evaluated on whether desired behaviors are contained within the capability space.

The second motivation is that robots lack a sense of danger. When performing dynamic tasks near the boundary of safety, humans possess an innate sense of danger that guides the execution of extraordinary dynamic maneuvers while retaining the ability to eventually stop safely. This capability is due, in part, to a self-awareness of feasible motions, feasible forces and impulses, risk and safety \cite{5} \cite{54} \cite{12}. This research is a method to step beyond the traditional conservatism seen in robot movements by computing the boundaries of safety for the dynamic system and providing a sense of how close the robot is to the boundary.

Particularly when regulating to a state or a trajectory, deviation from the desired
states is considered to be the proxy for danger. However, for most tasks in legged
locomotion or manipulation, there exist large sets of controls that can achieve effec-
tively the same task. The typical optimization formulation of control and planning
can return feasible solutions but do not provide feedback on the level of danger for
states in the returned trajectory. Sometimes, the solutions are near extreme states
and such methods would benefit from a metric that encodes the risk of failure. A
quantified risk value can provide a complementary set of information to help decide
whether certain decisions are critical. For instance, when picking up a cup of wa-
ter, there are many possible motions that can accomplish the task, and humans may
be indifferent to many of the possibilities. Quantified risk can determine whether
following the planned trajectory tightly truly matters. In general, controllers could
be compared not only by performance in attaining motion goals but also by their
propensity to guide the robot towards and away from risk.

The third motivation is to provide a generic quantity that has a consistent inter-
pretation across systems. An example of this motivation is the teleoperated system
HERMES [67], [50] which is described in Section 1.2. In this system, we intended
for a human operator to give proper motion trajectories for the humanoid to follow.
These motion commands would keep the robot balanced, provided that the robot
could give proper feedback about the state of balance, which was used as a proxy
of risk of falling. In order to properly communicate the risk, a generic quantity is
required. Additionally, a generic quantity used in teleoperation can allow each side of
the system to reason about their own risk and override commands that are potentially
unsafe.

Motivated by the aforementioned goals, we aim to create a methodology to quan-
tify the risk of failure. Such a quantity would be a valuable parameter in designing
physical systems, informing the design of controllers and providing a system inde-
pendent quantity that retains consistent meaning across applications. Application of
these quantities will expand the range of available robot capability and enable be-
haviors that are effortlessly intuitive to humans but have thus far evaded engineered
machines.
1.2 HERMES tele-operated humanoid

1.2.1 HERMES as a motivation for quantifying risk of failure

In an effort to expand robot capabilities towards humans’, we investigated full body teleoperation with the HERMES system. HERMES was composed of humanoid robot controlled by a human operator. The state of balance was sent as haptic feedback to the human operator to communicate the risk of falling. In this system we achieved two major goals: (1) The human was able to prevent the robot from falling after an applied disturbance. (2) The human operator was able to effortlessly coordinate high degree of freedom limbs, which allowed the robot to exploit the human’s innate intelligence and creativity.

Significant effort was directed towards maintaining high bandwidth to create a seamless and immersive experience. We hypothesized that by doing so, the robot’s abilities could approach that of its operator. However, an open question was the method to communicate danger. HERMES used the margin between the center of pressure and the boundary of the support polygon as the proxy of the true risk of falling. This method was sufficient for a simple task such as two foot standing. However as the task complexity was increased, outside of an easily characterizable range of behavior, the same method became insufficient. The robot was constrained to a limited set of behaviors and we were not able to utilize the robot up to the anticipated limit of the hardware. To address this limitation, Chapter 2 provides a methodology to better approximate the true risk of failure using a probabilistic measure. The rest of Section 1.2 provides detailed account of the HERMES system.

1.2.2 Introduction of the HERMES system

Disaster response has long been considered the ideal application for robotics but current robots have failed to address the needs of the recent incident at the Fukushima nuclear power plant [40]. Track-robots have entered and partially traversed the rubble in order to survey the area, a formidable challenge in itself. However, there have
not been any robots that possess the ability to navigate the highly unstructured environment as well as do physical work inside of the facility such as lifting, pushing or breaking through impact, a combination of tasks that a typical human would be able to do.

We identify two major needs for robots that humans can achieve with ease: the ability to ‘creatively’ handle highly unstructured environments, and the ability to stably manipulate objects with high force using its own dynamics to maximize the capability per volume - a bigger and heavier robot will be stronger but might not be able to go through the space designed for a human. Current artificial intelligence is far from ‘creatively’ tackling situations even in a very simple situation, particularly when dynamic movements or dynamic interaction with environments are required\(^2\).

We believe that direct human-in-the-loop approach with appropriate sensory feedback will enable virtual presence of an operator and dramatically improve the performance of robots in disaster situations. If the robot is sufficiently anthropomorphic and the operator receives the proper sensory feedback, the operator can leverage the innate control ability of its own body to control the robot including the balancing.

We introduce the teleoperated HERMES humanoid robot system with the Balance Feedback Interface shown in Figure 1-1 that addresses these key deficiencies.

For humanoid robots, maintaining balance is crucial to accomplishing any manipulation task and one that has not been well addressed by whole-body teleoperation. Furthermore, complex momentum control of the human body is essential for many tasks such as swinging a hammer, and opening a heavy spring-loaded door. Previous model-based posture control approaches successfully stabilize the body but are unable to interpret and execute the operator’s intention of complex movements in various situations [59],[44]. Nor are they used to provide feedback back to the operator [73],[74],[29]. In addition, these approaches are highly dependent on prior knowledge which is prohibitive for high force manipulation in unknown disaster environments.

Visual feedback, commonly used in teleoperation gives no indication of changing

\(^2\)Manufacturing robots are optimized for position control and are usually not capable of force control tasks such as dish washing, cleaning a car without damaging it
Figure 1-1: The HERMES humanoid system showing the operator and humanoid robot. The operator is standing inside the balance feedback mechanism and is wearing the upper body motion capture suit.
balance conditions until movement occurs. Often when tipping is observed on camera after a significant vision processing delay, there is little that can be done. Humans however, are able to achieve balance naturally through a multitude of sensors in the body (vestibular, proprioceptive and visual) [34]. Furthermore, humans can creatively use strategies that proactively suspend balance control when it conflicts with a task, such as dynamically shifting body mass to open a heavy spring loaded door.

In this study we explore the use of the operator’s entire body as the control input to the humanoid robot and develop a force feedback to display the robot’s balance state in an intuitive way. Previous efforts using joysticks [39] or indirect mapping of operator hand movements to robot body movements [62] cannot use intuitive reflex responses without significant prior training. Upper body teleoperation systems such as the RE$^2$ HDMS [1] and DLR JUSTIN [31] provide a haptic, immersive experience for manipulation and we extend those principles to full body balance.

By immersing the human operator in the experience of a humanoid robot body during teleoperation, the operator can rely on the intuitiveness of its own control frameworks for coordination. Existing work has addressed visual feedback from the robot, but an open area of research is how to display nonvisual, non-audio balance information back to the user. For instance, with properly identified and tuned feedback, the robot falling could transmit the feeling of falling to the operator and any reflex-based corrective action could be mapped back to the robot to correct its balance.

In addition to the intuitiveness of whole body human-in-the-loop teleoperation, there are distinct advantages in capturing human movement. First, the learning capability of the human will be able to compensate for the differences in dynamics and kinematics between the robot and operator. This is a property shared across all teleoperated systems. Second, the system can observe strategies of the human for balance in response to a stimulus and use these data to synthesize autonomous strategies for the robot. We expect that over time, not only will the operator learn to use the robot, but the robot will be able to correct for aberrations in the human control as well as exceed human performance, given that robot can adequately anticipate the
command. Third, the human interface side can observe the correspondence between applied force to the human and the human’s response to better formulate feedback strategies to the operator. Finally, as long as the robot is sufficiently similar to the human operator, this strategy can reduce the reliance on purely model-based approaches. Unexpected perturbations as well as loading conditions due to handling heavy objects will be felt similarly as disturbances to the balance of the robot.

This section is an exploration into the system design for full body human-in-the-loop balance control with balance feedback. Section 1.2.3 addresses the system design principles. Section 1.2.4 show the major design components of the humanoid robot and section 1.2.5 presents results from preliminary strategies for balancing with the human in the loop.

1.2.3 Principles of full-body teleoperation

Anthropomorphic design of the humanoid robot While teleoperation using joysticks has been readily explored, to provide an immersive experience for the operator, the humanoid robot should move similarly to the human operator. Therefore, HERMES is designed with dimensions scaled to approximately 90% of an average human female. At that size, the robot is able to interact with human designed environments and operate tools. The arms however are longer to match the leg length so that the robot will be capable of quadruped locomotion in the future.

Balance feedback and motion capture of the operator In order to create the immersive telepresence experience, the operator stands in a Balance Feedback Interface and wears a motion capture suit that is used to command the robot posture. Design details are provided in [49] and [50]. The center of pressure (CoP) is used as the measure of balance for the robot and it will stay within the boundaries of the support polygon formed by the convex hull of the feet when the robot is dynamically stable [66]. The Balance Feedback Interface applies planar forces to the operator’s hips that correspond to the movement of the robot’s CoP in order to maintain the CoP inside of the support polygon. The mapping between the CoP position inside the
Figure 1-2: Human hip displacement due to step input of proprioceptive stimulus force on hips and visual stimulus

robot’s support and the force vector on the operator is constructed with a potential function described by equations 1.1 and 1.2.

\[ V = f(x_{CoP}, \text{SupportPolygon}) \]  
\[ F_{\text{operator}} = -\frac{dV}{dx_{CoP}} \]  (1.1)

When the CoP of the robot approaches the boundary of the support polygon, an increasing force is applied to the operator in the direction of movement away from the boundary. We expect the human operator to comply to the force and move in the direction that regains balance of the robot. That is, if the robot is tipping too far forward, the Balance Feedback Interface will apply a backward force on the operator’s hips. The resulting hip motion is mapped to the robot so that a backward movement of the operator will rotate the robot’s torso backward. Since the motion of the operator’s upper body is also mapped to the robot, the operator can choose to use other movements to correct the balance. This is the first exploratory test using a simple feedback strategy and future strategies will incorporate more knowledge of human reflex actions and robot dynamics. An example of future study includes using Capture Point and stepping to maintain balance [44].
Figure 1-3: Block diagram of HERMES system with human-in-the-loop feedback. The bold arrows show the balance feedback loop in which the operator uses non-visual force feedback on the waist in order to stabilize the robot. The open arrows show the feedback path of visual data and higher level cognitive planning.

System design  Seamless virtual telepresence for the operator requires management of dynamics and delays of the entire feedback loop with the humanoid robot. Human proprioceptive reflexes have response times of 50 - 100 ms [6] and visual processing response times on the order of 200-250 ms [25] depending on the subject. We predict that for the strategy of balance feedback information displayed as force on the operator's hips, the reaction time after training should be between that of proprioceptive reflexes and visual processing.

To justify the hip force feedback strategy rather than visual, a hip position step reference is presented as proprioceptive input at the hips and through visual input. Four trials of each type of sensory feedback are presented in Figure 1-2. These show that proprioceptive response time is ~100 ms faster than the visual response. Therefore, force feedback on the hips should allow for more dynamic teleoperation performance compared to visual feedback.

A block diagram showing the overall control architecture of the system is shown in Figure 1-3 and expected sequence of events is shown in Figure 1-4. The characteristic time constants of major components in the system are listed in the Table 1.1. System components related to computation and data transmission are set to 1 ms to minimize additional delay on top of physical dynamics.
Figure 1-4: Flow diagram showing the expected sequence of events for human in the loop balance feedback

In typical operation, the system works as follows. Operator posture is captured by the Balance Feedback Interface and motion capture suit. These are transmitted over ethernet to the robot which transfers operator task space motions to robot coordinates. The robot controller carries out the commanded motion and the load cells in the feet monitor the center of pressure. The center of pressure data is transmitted over ethernet back to the Balance Feedback Interface which then determines the proper feedback force command on the operator. Force is applied to the operator's waist and the resulting motion of the operator is captured and commanded to the robot.

Table 1.1: Table of time constants for components in system.

<table>
<thead>
<tr>
<th>System component</th>
<th>Characteristic time constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human controller and body dynamics</td>
<td></td>
</tr>
<tr>
<td>Human proprioceptive reflex action [6]</td>
<td>$\sim$ 50 - 100 ms</td>
</tr>
<tr>
<td>Human visual processing [25]</td>
<td>$\sim$ 200 - 250 ms</td>
</tr>
<tr>
<td>Anticipatory postural adjustments [33]</td>
<td>$&lt;0$ ms</td>
</tr>
<tr>
<td>Experiment</td>
<td></td>
</tr>
<tr>
<td>Human hip proprioceptive response</td>
<td>100 ms</td>
</tr>
<tr>
<td>Human hip response to visual stimulus</td>
<td>200 - 250 ms</td>
</tr>
<tr>
<td>System computational delay</td>
<td></td>
</tr>
<tr>
<td>Motion capture of human</td>
<td>1 ms</td>
</tr>
<tr>
<td>Ethernet communication delays (2x)</td>
<td>2x 1 ms</td>
</tr>
<tr>
<td>Task space motion transfer</td>
<td>1 ms</td>
</tr>
<tr>
<td>CoP to BFI feedback force mapping</td>
<td>1 ms</td>
</tr>
<tr>
<td>Video camera (30 frames/sec)</td>
<td>33 ms</td>
</tr>
</tbody>
</table>
1.2.4 Design principles of the HERMES humanoid robot

The HERMES humanoid robot is 45 kg and sized to approximately 90% of an average human female. The hip height is 730 mm and the shoulder axis height is 1105 mm. Each limb has 6 degrees of freedom: 3 at the shoulder/hip, 1 at the elbow/knee, and 2 at the wrist/ankle. The following are major design features that consider the maximization of force control bandwidth and overall system bandwidth while providing features that allow the robot to accomplish human-like manipulation tasks.

Major motor axes operate on power planes The HERMES robot is designed to do most of the high force work on ‘power planes’. We define these as the planes in which the movement of the end effector occurs due to the parallel axis shoulder and elbow motors (in the lower body, the hip and knee). To maximize energy efficiency, high force movements such as hammering or throwing generally involve swinging a mass on a single plane. In order to maximize capability without adding significant weight, major work is expected to be done on these planes. If the humanoid is unable to reach a position with these planes, the operator can reorient the robot and replan the movement. Like humans, these planes are typically close in alignment with the sagittal plane of the robot and can deviate slightly. Since highly dynamic motions are executed in the power planes, the majority of the actuator mass in the robot is allocated to the power plane motors. The actuators used on the power planes are custom large gap-radius motors that are shared with the MIT Cheetah robot [56]. The remaining yaw and roll axes of the shoulder/hip are driven by a parallel actuator mechanism by two Dynamixel MX-106 compact servo actuators with custom electronics. These lower power, lightweight actuators are used to reorient to ‘power planes’. Figure 1-5 illustrates the power plane concept.

Limb design In order to minimize inertia of limbs for high bandwidth, the manufacturing was focused on high stiffness composites. The main structure is provided by commercially available braided carbon fiber tube. Complex interface geometry for mounting the limbs to the motors and the mechanical attachments made from ABS
Figure 1-5: CAD drawing of the HERMES humanoid highlighting the power plane on the upper right limb

Figure 1-6: Cutout view of the HERMES robot composite leg design.

on a Stratasys uPrint 3D printer. The hollow ABS parts are then glued over the carbon fiber tube. A cross sectional view is shown in figure 1-6. To aid the design of parts in FEA, several experiments were conducted on solid cast ABS and the 3D printed equivalents. 3D printed parts were found to have 90% of the weight and 85% of the stiffness compared to solid ABS. Final limbs on the HERMES robot have higher stiffness/weight ratio compared to a similar aluminum part.

Foot design The design of the foot is shown in figure 1-7. Each foot contains 3 contact points, each instrumented with a load cell that together provide an estimate of the center of pressure of the robot inside of the convex hull corresponding to the support polygon. Three load cells provide a minimal estimate of the CoP even when
Figure 1-7: HERMES humanoid robot foot showing the location of the 3 load cells.

Figure 1-8: Arm of the HERMES humanoid robot showing the location of proximal motors driving Bowden cable that actuate the fingers.

only a single foot is in contact with the ground.

**Hand design** For a disaster situation, hands that can grasp basic shapes such as an axe or door handle, yet have the strength to move or demolish rubble are necessary. Additionally, the hands must follow the overall design philosophy of the robot in making the limbs as light as possible to reduce inertia during dynamic operations. This was accomplished by utilizing three cable-driven underactuated fingers which allows the actuators to be mounted proximal to the body as seen in Figure 1-8.

To accomplish grabbing task, the fingers close until the object is fully bounded due to underactuation as Figure 1-9 demonstrates [20]. To support the robot when moving or be used as a tool themselves, the hand can also make a rigid fist.

The lower arm allows for 360 degrees of roll through an integrated motor towards the elbow, while the wrist provides 180 degrees of yaw movement.
Figure 1-9: Hand of the HERMES humanoid robot showing a grasp of a power drill and a fist. Note that the fingers can independently actuate the drill power switch.

1.2.5 Experiment

**Single Impact disturbance recovery**  In order to characterize the performance of the balance feedback system, the time trajectory of motions and forces in the system were monitored while applying a disturbance force to the robot. An instrumented mallet with a load cell was used to apply an impact disturbance at the top back edge of the body frame in the forward direction. The change in robot balance due to the disturbance is measured by the load cells in the feet by computing the change in center of pressure. When the robot CoP approaches the edge of the support polygon of the feet, a force is applied to the operator in the direction that the CoP should move to keep the robot balanced according to a potential function described in Section 1.2.3. The operator then commands corrective motions which are mapped to the robot posture. In these experiments, the operator uses a hip strategy to control the robot. Forward planar displacement of the operator’s hips commanded forward tilt of the robot torso at the robot hips with a gain of 5 rad/m. This gain was chosen empirically so that the operator possessed sufficient control authority to command large robot torso tilt angles that could cause the robot to lose balance, all while being able to maintain its own balance. Vertical displacement of the operator hips was mapped to the robot hip height.

Experiments were conducted with two different operator knowledge conditions. In the first experiment, the operator had no knowledge of when the impact was going to occur and had to rely on the feedback from the Balance Feedback Interface for
Figure 1-10: The trajectory of the robot CoP inside of the support polygon from 0 to 4 seconds after the impact disturbance.

The robot state of balance. In the second experiment, the hammer impact on the robot was preceded by an audible countdown so that the operator could anticipate the change in robot CoP.

**Impact Disturbance Recovery Results**  Results of impact disturbance experiments in Figure 1-11 show that the operator is able to stabilize the robot CoP and hence balance the robot. The trajectory of robot CoP remains inside of the support polygon as shown in Figure 1-10.

Plots 1-11a and 1-11b are the normalized time trajectories of system signals for the unexpected force disturbance on the robot. The signs of the signals are matched for ease of comparison. 1-11c show the results in which the human operator could anticipate the impact disturbance with a countdown.

The human operator strategy can be seen in the time trajectory of operator movement over 4 seconds after the impact disturbance in Figure 1-11a. This strategy emerged over several hours of testing during the development of the system. During the initial rise in feedback force, the operator's hips comply to the feedback force, which commands the robot to tilt the torso at the hips. However, after the force falls from its peak, the operator maintains a smooth transition back to equilibrium and allows the robot CoP oscillations to damp out. Notice that although the robot CoP oscillates around the new equilibrium after 1.5 seconds, the force feedback command to the operator is minimal because the robot CoP is moving in a safe zone.
Figure 1-11: Time trajectory of multiple signals during a impact disturbance on the robot. Signals are normalized and bias is removed to aid in visualizing the propagation of information. (a) Unexpected impact disturbance on robot (b) Unexpected impact disturbance on robot zoomed in to show the propagation of signals through the system (c) Expected impact disturbance on robot
Figure 1-11b shows the onset of change in signals of the system feedback loop and gives insight into the maximum performance of the balance feedback system. The results are compiled in Table 1.2. The 50 ms delay between the initial impact and the movement of the robot CoP is a fixed property of the humanoid hardware dynamics. We expect the maximum delay between actuator movements and change in CoP to be similar. The movement of the robot CoP generates a force feedback command on the operator but the due to the bandwidth of the actuators on the Balance Feedback Interface, there is a delay of 55 ms before the force is applied to the operator. The operator hip movement in the forward axis matches the applied force. Since the shortest typical human proprioceptive reflex response time in the literature is about 50 ms, this suggests that the initial operator movement is due to passive compliance. The operator is holding a position with a given stiffness. Although the experiment was conducted using the forward movements of the robot and operator, the operator’s sideways movement can provide insight on the human response. The operator shows sideways movement 50 ms after the force is applied which matches reflex times in literature. We hypothesize that since it is unlikely for a human to move perfectly in a single axis, that the sideways movement captures the delay of the operator’s active response to the feedback force. Therefore the system is lower bounded by 100 ms of fixed delay due to the robot and human operator dynamics.

Two delays between the change in robot CoP to the actual feedback force on the operator’s waist are flexible. The 20 ms delay between movement of the robot CoP and rise in the force feedback reference command on the operator is due to the neutral zone of the robot’s CoP in which nearly no force is commanded. Only when the robot CoP nears the edge of the support polygon, does the force feedback rapidly increase in magnitude. A controller that incorporates the CoM velocity would be able to predict the trajectory of the CoP and possibly command a feedback force on the operator earlier. The 55 ms delay between the force feedback command on the operator and the actual force is due to the bandwidth of the Balance Feedback Interface. With higher bandwidth actuators and control, this delay can be drastically reduced.

Figure 1-11c for the experiment in which the operator anticipated the impact
Table 1.2: Onset of signal change for unexpected impact at time t

<table>
<thead>
<tr>
<th>Event</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impact force</td>
<td>T + 0 ms</td>
</tr>
<tr>
<td>Robot CoP</td>
<td>T + 50 ms</td>
</tr>
<tr>
<td>Force feedback command on operator</td>
<td>T + 70 ms</td>
</tr>
<tr>
<td>Force applied to operator</td>
<td>T + 125 ms</td>
</tr>
<tr>
<td>Operator hip movement</td>
<td>T + 125 ms</td>
</tr>
<tr>
<td>Operator active response</td>
<td>T + 175 ms</td>
</tr>
</tbody>
</table>

disturbance show largely the same delays as the unexpected impact with one striking difference. The sideways movement of the human operator which commands the robot posture precedes the actual feedback force on the operator by 50 ms probably due to the audible countdown. The forward movement of the operator matches the applied force. This suggests that the human operator was able to prepare its body for the force feedback. With some combination of feedforward through either force feedback or other sensory input, it may be possible to reduce the loop delay and increase performance in dynamic tasks.

**High Force Dynamic Wall-breaking Task** A demo that highlights the integration of human cognition and motor skills to teleoperate the HERMES robot is breaking through drywall. This is a task that cannot be achieved with high force for humanoid robots that attempt to maintain static balance throughout the motion. In order to apply a large force, the robot must find a way to anchor itself or use the body momentum to apply impact force. The sequence in Figure 1-12 shows a strategy that is possible with whole body teleoperation and balance feedback. First, the operator uses the onboard camera to identify an object that can be used to anchor the body and holds onto it with one hand. Second, the operator commands the robot to pull on the anchored arm to create forward momentum towards the wall. Simultaneously, the operator commands a punch using the closed fist of the opposite arm. Finally, after the robot has successfully broken through the wall, the operator uses the arms to push off the wall frame to regain balance. The operator is aware of the balance of the robot through force feedback from the Balance Feedback Interface and can creatively use many strategies involving a mix of upper and lower body posture to regain
balance. The accompanying video shows wall breaking as well as other capabilities.

### 1.2.6 Conclusion

This is an introductory study of full body teleoperation with human-in-the-loop balance feedback. System design considerations are introduced and initial experiments show that human operator is able to stabilize the robot CoP after an impact disturbance. In addition, the advantages of using innate human strategies for dynamic movements are demonstrated in a wall-breaking task.

The experiment in which an impact disturbance was applied to the humanoid robot, and loop dynamics were recorded show that the humanoid and the human operator account for 50 ms each of delay, giving a performance bound of the system of 100 ms. The 175 ms of total system loop delay was found to be sufficient for controlling robot balance but in ideal situation, this should be minimized. We plan to improve the feedback mechanism to reduce delay due to the CoP to operator force mapping and force production in the Balance Feedback Interface. In addition, we intend to explore feedforward methods to use human anticipatory control.

The ultimate solution for humanoid teleoperation likely involves human creativity as well as high speed autonomous control. To enable more complex behaviors, additional work will explore alternate methods of that provide feedback on the robot’s likelihood of losing balance. Legs will be added to motion capture suit to allow operator control of stepping. Finally, common human control strategies will be observed by the human motion capture device to better predict human intention and autonomously execute learned strategies.
Figure 1-12: Sequence showing the HERMES system breaking through a wall. The safety ropes on humanoid are loose. (a) Starting position (b) Operator commands the robot to grab the frame (c) Operator commands a punch (d) Robot pushes off the wall with balance feedback enabled (e) Operator restabilizes the robot with balance feedback
1.3 Related work

A variety of risk related concepts have been explored by researchers. Some concepts can be described with intuitive rules while others explicitly operate on the robot state variables.

For legged systems, tumble stability and energy stability [19] aim to place the robot’s center of mass over a point within the convex support formed by the outer extremities of the contact with the ground, such that the robot can resist tumbling or tipping due to an external disturbance. An extension to these ideas came from ZMP-based metrics [66] which are instantaneously computable from a simple template. The numeric values that these provide are often in the form of a distance margin from a boundary. Metrics such as these are well suited as indicators. Violation of these metrics imply that the robot has exceeded a condition necessary for stability when the robot is using a limited strategy. For instance, controllers developed from these concepts [60] [23] can instantaneously exceed the bounds implied by the metric, while also producing working motions. However, the construction of these metrics constrains the robot to quasi-static motions. The goals of this research are different. We wish to compute a value that can serve as more than a momentary guideline in order to enable more aggressive behaviors.

Another class of physical notions incorporating the dynamic behavior of the robot is the idea of the capture point [44],[30],[51]. This extension allows the controller to determine whether a Linear Inverted Pendulum model of the robot could stop in a standing position. The construction of a risk value is derived from whether the foot center of pressure can be applied at the capture point. In an alternative use, the control of the destabilizing portion dynamics can be used to guide the robot [13], [14], [21].

The above concepts consider safety to be in terms of being able to regulate to a known ‘good’ state or not exceeding a conservative operating range in which regulation controllers are available. These ideas correspond with notions of stability and it is important to delineate them from the ideas of failure. Although there exist many
related concepts in stability that cover the definitions of non-failure, the usage of
stability in this thesis refers to the regulation to a state or trajectory, in the presence
of control. Being unstable in this context means that the robot state is diverging
from a known ‘good’ state and can reach a state that guarantees that the system will
fail. However, instability in itself does not imply that the robot will fail in a general
sense. It only fails to achieve a desired behavior that is considered to be ‘good’.

For complex tasks, the notion of risk becomes difficult to define. Abstract con-
cepts such as feasibility consider the possible forces that the robot can exert on the
environment to affect changes in motion. Attempts to attain a measure of ‘goodness’
have examined the distance between an instantaneous wrench to the boundary of fea-
sibility [75], or examined the volume of feasible impulses that the robot can apply [4].
Reachability concepts identify whether states will evolve to failure [68], [3] whereas
viability considers the full set of states that have guarantees of non-failure [72], [11],
[24], [2],[10], [35]. The above concepts provide a general approach with mathematical
rules but are typically difficult to explore in high dimensional systems such as legged
robots, which are also complicated by impact dynamics.

Approaches to quantify the risk of failure in complex systems have included prob-
abilistic descriptions such as metastability [9], [53]. These examine the long-term
robustness to external factors and primarily allow holistic comparison of different
controllers under similar operating conditions. However, this approach lacks the
specificity of quantifying risk at individual states that a real-time controller could
respond to while in motion.

Examples of work to define safe regions for complex robot systems have included
controller composition [65], [7], [8], [43] or large scale search of optimization-based
controllers [38]. This work builds upon the literature by attempting to simultaneously
find safe regions of state space in addition to providing a consistent quantification of
risk (as a risk of incurring a catastrophic failure) over the entire range of found
behaviors, without binding the robot to a stable controller.

The methods developed in this thesis can be viewed as an evolution of several
established works. The first is tabular control, which was successfully applied to a
hopping robot by Raibert [45]. Tabular control uses pre-computed tables to store the results of complex calculations needed for control. We similarly build large tables of state evolutions to model the complex dynamics of a legged system. The second is Metastability, in which Byl quantified the long-term likelihood of failure for a legged robot [9]. The usage of the Markov Chain was also adopted for modeling in this thesis to determine the forward projected risk of a state. The third is the work by Mummolo [38], in which a region of controller-independent non-failing states were identified for a legged robot. We adopt similar principles to map the detailed capability space of dynamic locomotion.

1.4 Non-regulation approaches inspired by Viability Theory

In order to truly understand the risk of failure, any methodology intending to do so must find failure. We are inspired by Viability theory [2], which provides a construct to reason about failing and non-failing states. Viable states are defined as those that are not guaranteed to reach a condition of failure. There exists at least one sequence of controls that can avoid failure. Figure 1-13 illustrates the definitions. An alternative interpretation is that viability implies ultimate aimlessness without failure. In contrast, the regulation paradigm only allows states that can be regulated to a desired state or desired trajectory. While there exist relaxations in strictness, with convergence and more loosely, boundedness, these concepts encode a goal. Therefore viability can encompass a much larger set of states and behaviors compared to those produced from regulation approaches.

The interpretation of failure in these contrasting approaches differs. In regulation ideas, a state that cannot be regulated to a desired goal is considered to be dangerous or failed. However, in the viability definition, the same state will not be considered failed unless all forward state evolutions result in guaranteed failure. Therefore any risk metric concerning failure must be approached from Viability Theory.
Figure 1-13: Abstract representation of reachable and viable states. Black dots are example points in state space and green arrows represent control actions between pairs of states. The terminating states are user defined. Viable states are states in which there exists a control action that keeps the robot in the set of viable states. Conversely, if there is a state that leads to an inevitable failure at a terminating condition, then that state is non-viable. Within the viable space lies the statically stable states as well as the quasi-static controllers that operate in a conservative region. Clearly, from observation of biological systems, there exist highly dynamic and interesting behaviors that may be available to robots. These lie in the yet unexplored viable regions and are not typically captured by linear controllers that regulate the system around fixed points or static stability.

While Viability Theory can inform whether a state is non-failing, it cannot provide a measure of riskiness. Neither is the use of margins in space to the viability boundary sufficient as a risk quantity. Figure 1-14 shows a generalized diagram relating the sets of interest. We first define a home set of states where it is possible to find strict theoretical guarantees of stability. For many robots, this could be states in which the robot is static or quasi-static. The set of reachable states are all the states that the robot can reach under any sequence of feasible controls. For the purposes of this argument, states encountered due to the simultaneous influence of control actions and unreliable environmental noise are not included. The set of viable states are those that encompass all non-failures. The set of backwards reachable states include all states can reach the set of home states, with at least one sequence of feasible controls. These may also be regarded as the maximal Region of Attraction to the home states or the states that the home states can easily capture. We define the intersection of these sets to be the true capability of the robot. More precisely, the true capability of the robot includes states that are reachable from the home states, are non-failing, and can return to the home states. Alternatively the capability set contains all states
Figure 1-14: Abstract representation of reachable states, viable states, home set, and home set reachable states. For this thesis, the capability is defined to be the intersection of the reachable, viable and backwards reachable states which is shown by the shading.

within all cycles of any length containing the home set.

While each of the boundaries in figure 1-14 provides indications of the operating regions of the robot, boundaries and the margin from the boundary often provide inexact measures of security. For instance, suppose the robot is operating at the shared boundary of capability and reachability inside of the boundary of backwards reachability. While any command to push the robot further may be ineffective, the robot is essentially not in danger of failing to reach the home state. However, if the robot is again operating on the boundary of backwards reachability, inside the reachable set, the robot is likely to be in more danger. Therefore proper quantification of the actual risk of irrecoverable failure is necessary to evaluate states within the capability of the robot.

1.5 Thesis overview and contributions

Chapter 2 details the development of the methodology to quantify risk. The design decisions of this methodology are guided by the challenges of hybrid locomotion in legged systems. The planar pogo-stick robot is an exemplary model of these challenges and will serve as a simple template model throughout the thesis. It acts as an abstraction of the core dynamics of legged locomotion without considering additional complexities that arise from the large number of degrees of freedom present in hu-
manoid and quadruped systems. This simple model begins to address the challenges faced while developing the HERMES humanoid system in Section 1.2.

The methodology in Chapter 2 introduces two constructs, the Viable State Network (VSN) and Risk Maps. The VSN is a network of states and actions labeled with failure and non-failure, and shows the space inside of the robot’s capability. Risk Maps computed on the Viable State Network provide each state with a value of safety or risk of failure. Chapter 3 demonstrates the usage of the risk quantification method applied to a generic pogo-stick robot model and present the results. Chapter 4 provides usage examples of the risk quantity as well as extensions of the methodologies to tackle other related challenges.

The intended contributions of this thesis are as follows:

1. A methodology that is intrinsically designed to approach robot control with the motivation of non-failure instead of regulation.

2. General methodology to create network maps by sampling hybrid transitions

3. Computation of the capability of the generic pogo-stick robot. The resulting map shows the unintuitive shape and bounds of robot capability that have previously been unexplored. These results demonstrate the unexpected complexity of behaviors that exist for even simple robot models.

4. Computation of a risk (or safety) value for states within the capability of the pogo-stick using on a random control policy and Absorbing Markov Chain.
Chapter 2

A Process to quantify risk in dynamic systems

2.1 Data-driven, simulation based network

We seek to provide the robot with a risk quantity at each decision-making point, projecting all future possibilities and examining those for actual failure. Using the viability concept, we identify all of the reachable failures and use these to inform the risk quantity. Importantly, because this methodology considers the long-term evolution from the current state, it allows us not only to posit the existence of non-failing behaviors but also to explicitly inform the robot of how to achieve them. This is distinctly different from metrics that are based on exceeding a recoverable margin. These margins are typically based on a predefined behavior or control strategy, where the instantaneous risk may be overstated because the robot may be capable of executing commands that are not considered in the construction of the bound.\(^1\)

The necessary step to find the risk quantity is identifying the capability of the robot. That is, identifying all states that the robot can reach that do not result in

\(^1\)An analogous example is when observing someone leaning forward and losing static balance. From the point of view of an external observer, this may look like a very risky state to be in. However, if taking a step is well within the capability and intent of the subject, this state is in fact unlikely to lead to failure. The methodology in this thesis would consider the possibility for the subject to take a step and thus avoid falling.
guaranteed failure. While there exist methods to reason about the dynamic behavior of continuous, smooth and differentiable systems, hybrid systems such as legged robots remain a challenge. To targeting the challenges that are present on hybrid dynamic systems, which are affected by contact and impact dynamics, we adopt a data-driven approach.

As computation power grows, the low cost of simulation has given rise to sampling based approaches that numerically explore large dimensional state spaces. Approaches such as RRT, RRT*, PRM, RRG [26], Kinodynamic RRT [69], RG-RRT [57] allow an algorithm to quickly find feasible (and sometimes optimal) paths through large state spaces. These methods rely on building a network of nodes representing states in state space as well as edges representing the control actions that connect two nodes. With sufficiently dense sampling and computation time, the network would be able to asymptotically cover the entire reachable state space. Compared to formulation of the dynamics that give theoretical guarantees for conservative portions of the state space, sampling based approaches are able to draw insight from virtually any model that can be simulated.

Given a data driven approach, three possibilities exist for exploring the space: forward simulation, backward simulation, or a combination of the two. In the forward simulation approach, one would start with a stable state and propagate forward until the robot reaches failure. In the backwards approach, one would start from stable states and propagate backwards to find all of the states that can lead to it. One can imagine a combination of the two, which has been applied to trajectory planning, such as bidirectional RRTs [32]. For the methodology presented in this thesis, we choose to use forward simulation only, in order to allow for flexibility in contact dynamics, especially impact. Impact can cause any method that relies on backward simulation to easily become unbounded. This problem is illustrated in Figure 2-1 which shows how the single post-impact state of a falling block with a plastic collision can have an infinite number of possibilities for the state prior to impact. There exist methods that can invert contact states [63], permitting both forward and backward simulation. However popular approaches using Linear Complementary Problem dynamics are
suited for forward simulation only. Thus, growing a forward simulation tree allows
the user to employ any contact modeling as they see fit and make adjustments to
produce the best results for a particular robot.

![Diagram of a falling block](image)

Figure 2-1: A falling block which experiences a plastic collision upon impact. The
velocity after impact is zero but it is not possible to recover the velocity prior to
impact.

While exploration of the full viable space for a dynamic system is intractable, user
specified design choices can greatly simplify the problem. Here, an understanding of
the robot and physical intuition can avoid unnecessary computations. All physical
systems have limits, which include actuator force, kinematics, structural strength,
controller bandwidth, mechanical bandwidth, and actuator command resolution. Any
description of the viability kernel that exceeds these real limits will not improve the
performance of the physical system. Therefore, we can use knowledge of specific
physical limits to inform the exploration strategy for building the network while
allowing new unexpected behaviors to emerge.\(^2\)

\(^2\)It is important to note that over-reliance on human intuition may constrain the robot to only
familiar and expected behaviors.

The network approach to finding risk combines the approach of sampling based
methods on complex dynamic models with the principles of Viability Theory. In this
formulation, graph nodes represent states of the robot and graph edges are control
actions that lead from one node to another. The requirement of the network is to
develop a map of states and control actions that can be used to control a robot. Here,
we take an intuitive approach to defining safety and risk. At any given state on the
network, if it is possible to reach a known non-failing state in finite time using the
control actions available in the network, then it is at least minimally safe. The map

53
should encode controls to get to any goal that has finite safety, despite the relative risk of any intermediate states. Probabilistic methods will be used to quantify risk allowing a comparative measure between different states. The definitions for translating the robot problem into the network are provided below. Another interpretation of the methodology is that we are trying to grow a discrete graph of states and control action sequences that occupy the backward invariant set [58] containing non-failing states. Starting from a known set of working states which we define as the home states, we attempt to expand it using sampling.

Definition 2.1.1. **Graph node** A graph node represents one possible state of the robot.

Definition 2.1.2. **Graph edge / Action** A graph edge represents a single choice of control action that connects two graph nodes (robot states). Control actions can be open loop control tapes, or specific control strategies. Each action is a specific and repeatable decision that when applied to the same starting graph node (state) should lead to the same ending graph node (state).

Definition 2.1.3. **Failure** User defined undesirable states

Definition 2.1.4. **Starting states** User defined state where the robot is assumed to be stable.

Definition 2.1.5. **Home states** User defined state where the robot is assumed to be easily regulated, under active control, to any of the starting states. These states can be thought of as being within Region of Attraction (ROA) to the starting states. The maximal boundary of the ROA is typically found for each specific controller and desired state. However, the purpose of the home states is to define a conservative safe region that can be used as a final target of state evolutions to imply safety. These are states that are well understood and easy to regulate with a variety of simple control methods if the robot decides to apply a stopping controller. It is impractical to distinguish the relative safety levels of states within this region and we consider all of these states to be equally maximally safe. This allows us to focus on behaviors
that take the robot outside the region of user defined home states and user defined starting states, where it is not clear that a working controller exists.

**Definition 2.1.6.** *Behavior class* User defined choices that constrain the movements, states and control strategies which are not intrinsic to the hardware definition. Since it is intractable to consider every possibility, we use this umbrella set of choices to balance the generality of the results with computational effort. The user must weigh the specificity of the constrained behavior against the possibility of discovering unexpected movements. Different definitions of the behavior class will produce different results. Suppose the user may want to target a family of movements such as walking, running, jumping, etc. Since the abstract notions of a family of movements may be difficult to define mathematically, this can be achieved by defining certain generic forms of control strategy, constraining the allowable state space, defining *actions* that bias towards known behavior or defining additional *failure* conditions. These strategies are all considered to be specifications of the behavior class.

The main technical challenge of this research is to mitigate the heavy computation requirements of exhaustive computation by introducing a number of practical simplifications that preserve the essence of the problem. Combined with a sampling based method, the result should be a low-resolution map that can achieve the simultaneous goals of quantifying risk in the known robot state space and finding the boundary of performance. The following subsections relate to the distinct approaches to formulate the various parts of the problem.

In order to provide context for the design decisions of the risk quantification methodology, a planar single-legged hopper (pogo-bot) will be used as the dynamic model. This model is the simplest legged system that captures the core dynamics of legged locomotion. An example that similarly uses an inverted pendulum model to condense complex humanoid dynamics is described by Wensing [70]. Figure 2-2 shows the general model of the robot. Analysis of the fully developed model is shown in Chapter 3.
2.1.1 Node placement at hybrid transitions boundaries

While robots are continuous systems, some level of data compression can be achieved by observing cycles of hybrid transitions, similar to the usage of the apex height Poincaré return maps in the study of continuous hopping [55]. In order to achieve continuous actions without failure, the robot state will pass through the same hybrid transition type repeatedly. The usage however, differs from typical limit cycle return maps, as we are primarily concerned with the achievable states rather than local stability around fixed points in the return map.

The use of hybrid transitions confer some advantages when storing state evolutions in a discrete graph. This approach is in contrast to sampling the reachability space of a kino-dynamic system with continuous dynamics [42]. The non-obvious temporal sampling resolution can easily inflate storage requirements unless a compression scheme is adopted. The node placement on hybrid transition cycles partly address this problem because it is possible to place as few as a single sample per locomotion cycle. Adding this refinement to the definitions for the general network graph representation in 2.1 allows us to address problems in hybrid locomotion.

Figure 2-3 shows the hybrid transitions of a single-legged hopper (pogo-bot) whose...
detailed model is described in Chapter 3. In this case, an event is taken to mean a set of consistent conditions that the robot experiences, with significance to locomotion, and has a measurable state. Figure 2-4 shows the progression of events that occur when the robot is operating. The ‘Pre-takeoff’ and ‘Post-takeoff’ states are the instantaneous states on the two sides of the hybrid transition from single contact dynamics on the ground to flight dynamics. The ‘Pre-impact’ and ‘Post-impact’ states are the instantaneous states on the two sides of the hybrid transition from flight dynamics to single contact dynamics. Similarly, the hybrid transitions are shown for a biped in Figure 2-5.

In principle, any state that lies in the sequence of periodic movement, and corresponds with a consistent and detectable event can be used to build the map. Additionally, all maps should have equivalent utility, regardless of the user’s choice of state types. This is sufficient if the user wishes to use the results for offline analysis of the robot model and stay within the realm of computation. However, if the user chooses to incorporate the results of the simulation into an online, realtime control process, the following guidelines are recommended. These provide a methodical approach to choose the proper hybrid transition event to use as nodes in the network. These questions and design considerations can avoid performance pitfalls that are a result of the known unpredictabilities of hardware.
Figure 2-4: Event types for the pogo-bot. The hybrid contact transition boundary is crossed at takeoff and landing. The pre-takeoff and post-takeoff states refer to the state of the robot prior and post transition. Similarly during landing, there are pre-impact and post-impact states around the impact transition. Solid arrows indicate ground contact dynamics. Dashed arrows indicate flight phase dynamics.

Practical considerations mean that the choice of using ‘pre-takeoff’ states as the nodes in the network has some distinct advantages. First, the pogo-bot can only takeoff if the leg reaches the maximum extension length. Since the example pogo-bot applies force on the ground until takeoff, the contact force reaches zero when the leg is at maximum length. Thus, on the ground the four variables $\theta, \dot{\theta}, L, \dot{L}$ can be reduced to the set of $\theta, \dot{\theta}$, and $\dot{L}$. The reduced set of three variables enables ease of visualization.

More important for the use of the results in real hardware is the understanding of state estimation around hybrid transitions. Hybrid transitions are easy to detect in simulations having perfect knowledge of the full robot state, but can present challenges in hardware. In hardware, detection may not be immediately obvious and state estimation following hybrid transitions can be unreliable. Among these, impact at landing presents difficulty in estimation [71] [48] and control may need to be properly designed to mitigate the effects [28] [22]. Impacts cause ringing in MEMS IMUs as shown in Figure 2-6 as well as structural vibrations. This can mean that end-effector
positions do not match the estimates from the motor collocated encoders. These effects subside over time and if estimation is performed throughout the ground phase, the last instance of time on the ground should have the most reliable state estimate.

Another popular measurement is at the apex of flight. While the detection of an apex event can be readily measured by non contact distance sensors, forward velocity $\dot{x}$ at the apex cannot be easily measured in the air. For robots without vision, the forward velocity is typically estimated at takeoff. By choosing the pre-takeoff event to use for creating the network, open loop control actions are first executed in the air, where there is time to refine the takeoff state estimate and make decisions accordingly.
Figure 2-6: IMU ringing at hybrid transitions. In this experiment a hardware prototype of the pogo-bot was dropped in the aerial phase and the robot completed a jump. Impact with the ground occurred at 250 ms and takeoff occurred at 1100 ms. The MEMS IMU produces erroneous readings shortly after the impact event and therefore cannot be relied upon for state estimation. If the control of the ground contact phase were dependent on the state estimation at the time of impact, the ringing would likely cause poor decision-making by the robot.
2.1.2 Binning

We aim to reduce the possibility of wasting computation on exhaustive simulation of very close states that are likely to produce indiscernible results. Each state variable is discretized into a number of bins. Every state within a single bin is considered equivalent. The states corresponding to nodes chosen in Section 2.1.1 are located at the bin centers. During simulation, nodes that are to be simulated, start from the state at the bin center. The resulting state is recorded using the corresponding bin and the node located at the center of the bin. Since the network assumes that all actions lead from one bin center to another, we expect sufficient local linearity that is consistent within the size of the bin. An illustration of the binned space of the pogo-bot takeoffs is shown in Figure 2-7.

**Guideline:** We pick a resolution that is consistent with the sensor specifications and within achievable certainty. After selecting the type of hybrid transition event to use as the node in the network, employing physical intuition of the robot and hardware system architecture can inform the binning limits. To inform the lower limit on bin size, the planned sensors and sensing methodology should be examined. For instance the pogo-bot in Section 4.3 uses a cable drive mechanism. The cable pulley that drives the leg extension is measured by a typical encoder with 13-bit resolution. With a pulley diameter of 90 mm, this is equivalent to a 0.05 mm resolution on the length of the leg. A typical IMU [64], has angular resolution 0.05° and repeatability 0.2°. These can inform the lower limit of bin size. Additional flex in the mechanisms would raise this limit further.
Figure 2-7: Example gridding of the space of pre-takeoffs into bins
2.1.3 Exploration and processing

The exploration process begins with definition of the dynamics model, as well as the special states in Definitions 2.1.4, 2.1.5, 2.1.6, and 2.1.3. We pick a starting state in which the robot is stable. For the pogo-bot, this is the robot standing straight up at a fixed leg length. We then define a home set of states that we are reasonably certain can converge to one of the starting states. For the pogo-bot, this is a small region around the starting state for which a typical controller such as LQR would be able to drive the robot to the stable starting state. Failure in the pogo-bot is defined as anytime the robot’s center of mass falls below a height threshold. Whereas this could also be defined with the robot touching the ground, any height in which the user is reasonably confident about the irrecoverability can allow early termination of simulations to save computation. Lastly, the user chooses a set of actions that the robot is allowed to carry out. These should be chosen to represent hardware capabilities of the robot.

The exploration is outlined in Algorithm 1. Starting from the starting node (representing the starting states), simulations are run for every action in the predefined Actions list. Any resulting states that have not been explored are added to the Unexplored queue for simulation. Any simulation step that observes the robot reaching the *home states* is recorded. Similarly, any state in which every action in the action list leads to a failure condition and does not pass the *home states* are recorded in the Failed list.

Algorithm 1 will terminate once every reachable node within the allowable states has been explored. The process returns the set of explored nodes, the nodes that lead to inevitable failure in the next step, the nodes that can reach the *home states* in the next step, and the connections between nodes by action.

We wish to further categorize nodes into those that are non-viable and those that are home set reachable beyond the one-step horizon. The node status propagation is shown in Algorithm 2. In this process, all nodes can be labeled as ‘viable’, ‘non-viable’ or ‘unknown’. The initial viable-nodes are those that can reach the *home states* in
**Algorithm 1** Explore space

1: Initialize queue Unexplored ← starting nodes ▷ The nodes that have yet to be explored. Actions have not yet been applied to derive child nodes.

2: Initialize set Explored ← ∅ ▷ The nodes that have been explored (child nodes have been derived through applying Actions).

3: Initialize set Failed ← ∅ ▷ The nodes that have been classified as Failures.

4: Initialize set Visited ← Queue ▷ All nodes that have been reached. This is the union of Unexplored and Explored.

5: Initialize list TransitionGraph ← ∅ ▷ Graph that stores all transition edges (starting node, action, result node) from the result of our exploration.

6: Initialize list PassedHome ← ∅ ▷ List of (node, action) pairs where the action applied to the node leads to a home state.

7: for all nodes in Unexplored do

8:    for all actions in Actions list do

9:        [next node, next status, passed home set] ← SimulateStep(node, action)

10:       if next status is not fail then

11:           Add next node to Visited

12:           Add entry (node, action, next node) to TransitionGraph

13:           if next node is not in Explored then

14:               Add next node to Unexplored

15:           if passed home set is true then

16:               Add entry (node, action) to PassedHome

17:       if all simulations from node failed and none passed the home set then

18:           Add node to Failed

19:       Add node to Explored
the next step and are taken from the 'Passed home list'. The initial non-viable nodes are those contained within the 'Failed list'.

During exploration, the state may reach a failure state at a known failure condition or home states which are known guaranteed safe states. Guaranteed safe states are viable by definition. Failure states are a subset of non-viability. The principle of the node status propagation is as follows. If a state is guaranteed safe, then any parent state that directly leads to it has at least one possible control that reaches safety so the parent is viable. For a state in which every control action leads to a failure or non-viability, then that state is non-viable. Algorithm 2, propagates the status from labeled children to their parent nodes. Any node whose status is changed at each step is added to the queue for processing. If after running Algorithm 2, there may exist unlabeled nodes after the queue is empty. These are nodes (states) that do not have clear paths to guaranteed safety or failure. These may include infinite limit cycles that are technically viable according to Viability Theory, but are not particularly useful in the formulation of safety or risk for a real system. In a real robot system, if the robot is trapped in a non-failing cycle, then it should also fail as well.³ Thus these remaining nodes are labeled as 'non-viable'.

³A real life example of this phenomenon is the childhood activity of running down a long slope. For those less experienced at controlling their bodies, there may be an instance in which the legs seemingly cycle automatically but it is very difficult to speed up or slow down. Any attempt to do so leads to falling. You are in essence trapped at a certain running speed, at least until the slope flattens.
The result of the exploration and status propagation is the Viable State Network (VSN). For all actions that are available to the robot, the VSN contains all reachable states that match the user specified node definition and marks the viability or non-viability for each state (represented by a node in the network). Additionally, links between states are preserved. The portion of the VSN labeled as viable can be considered as the maximal capability of the robot. This set contains all of the states that robot can reach and remain inside indefinitely under self-actuation, without considering external influences such as noise.

2.2 Quantification of risk

2.2.1 Absorbing Markov Chains

While a network based purely on the viability concept can inform the existence of a safe set of controls, it misses the relative safety or danger when comparing states. In any system, some states must be safer than others. The intuition is to capture the likelihood of reaching the safe home states at the end of an action sequence prior to reaching irrecoverable failure. For this, we use the concept of an Absorbing Markov Chain.

![Absorbing Markov Chain Diagram]

**Figure 2-8:** Drunkard's walk example of the Absorbing Markov Chain. The home state and the pub are absorbing states. Once entered it is not possible to leave. This classic textbook example illustrates the likelihood of ending in any of the absorbing states when starting in one of the transient states.

The absorbing Markov chain is a special type of Markov chain that includes states that once reached, cannot be left [17]. All other states are known as transient states.
In the context of the risk network, once the robot state reaches a non-viable state it will remain in the non-viable set. Thus any non-viable node in the VSN is also absorbing. For this particular computation, all home states are also considered to be absorbing. Once the robot reaches a state in the home set, it can choose to remain there indefinitely without failure. An example choice for such states could be the statically stable states along with its region of attraction.

This method requires the user to define the transition probabilities between nodes. In the case of the robot picking randomly from any possible action presented to it, at every node, the transition probabilities of every edge leaving a parent would be equal. Then the risk quantity would encapsulate the likelihood of reaching a guaranteed safe state by randomly traversing the Viable State Network. The safety value can be similarly computed for robots possessing a controller with some noise on the chosen action. At each node, the transition probabilities could be set non-uniformly with the desired next state having the highest probability.

Suppose there are $n$ transient states, $m$ safe states and $p$ non-viable states. The safe states and non-viable states are absorbing. The absorbing Markov chain is represented by the full Markov transition matrix $P \in \mathbb{R}^{(n+m+p) \times (n+m+p)}$, where $P_{ij}$ is the probability of moving from state $i$ to state $j$ in a single step and $\sum_j P_{ij} = 1$. If the states are arranged such the first $n$ states are transient, the next $m$ states are safe states and the last $p$ states are the non-viable states, then $P$ can written as Equation (2.1) where, $Q \in \mathbb{R}^{n \times n}$ is the transition matrix between the transient states and $R \in \mathbb{R}^{n \times (m+p)}$ holds the probabilities of each transient state transitioning to an absorbing state. The $(m + p)$ absorbing states can only transition to themselves and
are hence represented by the identity matrix \( I^{(m+p)\times(m+p)} \). \( R \) can be further split into the transitions going to the safe states and transitions to non-viable states shown in Equation (2.2).

\[
P = \begin{bmatrix} Q & R \\ 0 & I^{(m+p)\times(m+p)} \end{bmatrix}
\]

(2.1)

\[
R = \begin{bmatrix} R_{safe} & R_{NV} \end{bmatrix}
\]

(2.2)

\[
N = \sum_{k=0}^{\infty} Q^k = (I - Q)^{-1}
\]

(2.3)

\[
B = NR
\]

(2.4)

\[
\bar{s} = NR_{safe}\tilde{1}^{n\times1}
\]

(2.5)

The fundamental matrix \( N \) in Equation (2.3) represents the aggregate transition probabilities for the transient states for all sequences of any length. The probability of each transient state to eventually reach each absorbing state is described by \( B \) in Eq. (2.4). Equation (2.5) shows the computation for the safety value.\(^4\) \( \bar{s} \) is a column vector that holds the probability of each transient state reaching any safe home state and \( \tilde{1} \) is a column vector of ones. We define each entry in \( \bar{s} \) as the safety value for each of the \( n \) transient states. The equations shown are adapted from the textbook example [17].

Taking the results of the Viable State Network, the risk quantification is setup as follows. All of the viable states in the VSN are transient in the Absorbing Markov Chain. The home states are the safe absorbing states. The non-viable states in the VSN are the non-viable absorbing states in the Markov Chain. The TransitionGraph from Algorithm 1 is used to define all of the state transitions that can have non-zero probability. The PassedHome list provides the transitions from transient states to the

\(^4\)Equivalently, the representation of Equation (2.5) as the linear system \((I - Q)\bar{s} = R_{safe}\tilde{1}^{n\times1}\) allows the flexibility to use various solvers. Iterative methods such as the Generalized minimal residual method (GMRES) or Biconjugate gradient method can efficiently compute \( \bar{s} \) when the size of \( Q \) grows large [15].
safe absorbing states. The result of the risk quantification computation is the Risk Map $\tilde{s}$ that encodes a safety value for each state in the viable set of the VSN.

**Remark.** The risk quantification methodology presented here provides safety values, which give the probability of reaching a *home state* prior to failure. Similarly, an equivalent risk value could be defined using $\text{risk} = NR_{NV} \mathbf{1}^{n \times 1}$. In this methodology the two can be used interchangeably with $\tilde{s} + \text{risk} = \mathbf{1}$. This is because the absorbing Markov chain only includes two absorbing classes and every viable state will be absorbed. The process of labeling infinite cycles as failures, described in Section 2.1.3, guarantees that every state considered can be absorbed. This is a slight departure from the strict definition of viability.

### 2.2.2 Additional noise considerations

The offline process of computing the capability and risk quantity for the robot model ignores some of the "dirty" realities that are present in any hardware system. These are the unmodeled dynamics of the system. Figure 2-10 illustrates the difference.

![Control block diagram of the robot](image)

Figure 2-10: The control block diagram of the robot. The feedback path inside of the shaded region labeled "Network Path Search" shows the dynamics that are implied by using the graph search of the Viable State Network. The feedback path labeled "True feedback path" includes the dynamics that the robot experiences with process noise or estimation noise.

In order to incorporate the effects of noise as seen in real processes, an additional Markov process is introduced. For instance, a noise process could distribute probability from each node to neighboring nodes in state space. Depending on the conservatism of the user, the influence of noise can be constructed in two ways. If the noise is allowed to act on all states, then we construct a noise matrix as fol-
\( W_{ij} = f(\text{dist}(i, j)) \) with \( \sum_j W_{ij} = 1 \) where \( f(\cdot) \) is a user defined function that indicates the behavior of the noise.

\[ P_{\text{new}} = PW \tag{2.6} \]

In this formulation, previously non-viable states can become viable if the noise can transition a non-viable state to a viable state. This can in effect, return a larger set of states that are within the robot’s capabilities. However, it is not possible to make decisions on noise, so this is not a practical representation.

If the noise is only allowed to act on viable states, then

\[ W = \begin{bmatrix} W_{\text{viable}} \\ 0 & I \end{bmatrix} \tag{2.7} \]

where \( \sum_j W_{ij} \leq 1 \).

### 2.3 Manipulation of the network post creation

A feature of representing the results of robot simulations in a network is that it can be easily manipulated after creation. Each simulation appears in the network as a single edge entry, which is a triple consisting of the starting node, an action, and ending node. These represent the starting robot state, the applied action and the final robot state. There are several scenarios in which additions or deletions may be necessary.

- **Reducing allowed actions:** For instance, if the user wishes to constrain the type of allowed actions, all edges associated with non-conforming actions can be removed. The propagation process in outlined in Algorithm 2 and the risk quantification process in Section 2.2 can be repeated to obtain the updated risk map.

- **Changing states to non-viable:** The user may decide to label a state in the network that is within the capability of the robot as failing. This can occur
for many reasons. For instance, the state may be undesirable but may not have been specified in the failure conditions. Another possibility is the state always fails in experiments.

- **Adding new actions** The user may choose to add additional actions to the robot. This could be due to expanded hardware capability or a desire to increase the resolution of the map. First all nodes in the visited list should be simulated with the new action. If any new nodes are found, these should be added to the Unexplored list of nodes and Algorithm 1 run without initialization.

## 2.4 Discussion

The methodology presented in this chapter is a first step towards quantifying risk in a hybrid dynamic system. It provides a value of safety at every state, directly related to the likelihood of failure. However, there are several limitations.

First, as a data-driven approach, the methodology suffers from scalability issues. The size of the state space, the complexity of the dynamic model and the number of actions have a great effect on the amount of computation required. There are possible variants on the method that could reduce the required computation. A method that could be explored in the future would be to designate control policies instead of enumerating over a large list of actions, replacing the inner loop of Algorithm 1 on Line 8.

Second, due to the construction of the VSN as a discrete network, it does not reason about local behavior. Every entry is a specific starting state, a specific action and a specific resulting state. Future refinement may be able to incorporate regions of states connected by families of actions.

Third, the results of the risk quantification can be biased by the choice of actions. Since the discrete network cannot sample every possible action that the hardware can produce, there may exist some unused actions that allow a state to be marked viable. In the VSN, the non-viable states are guaranteed to fail only for the choices of actions presented to the robot. Thus they represent an overestimation of the truly
non-viable states. In the usage of the VSN, if a state and an applied action is not simulated, then it is treated as if the result would fail.

Fourth, the discretization of the state space assumes that all states in each bin are equivalent. That is, for each bin to bin transition in the network, there exists a control that can connect any state within the starting bin to any state within the resulting bin. This assumption may not hold true for dynamics with high sensitivity to initial conditions, such as chaotic systems. If the user is aware of this sensitivity, then the bin size could be decreased to temper its effect albeit with significantly more computational effort.

The Risk Map computation presented here provides a safety value at each state that is risk neutral. It gives the expected value of the probability of reaching a safe home state or failure over all possible sequences of steps. This value may not be suited for all applications. Within the provided framework, the user can define more relevant control policies by manipulating the transition probabilities in the Absorbing Markov Chain. Or, using the VSN, which encodes the viable states and possible transitions between states, other risk measures can be applied [36] that are better suited to the user’s preferences.

The process of labeling each state with a safety or risk value as in the Risk Map has parallels to Reinforcement Learning [61], albeit with different goals. This popular technique has enabled learning agents to find suitable controllers to complex tasks in examples such as video games [37] or legged locomotion [18]. In Reinforcement Learning, the computed action-value and state-value arrays could serve the same purpose as the safety value at each state in the Risk Map. The process to generate the equivalent values in Reinforcement Learning would prioritize exploration completely over the exploitation process. Whereas Reinforcement Learning is learning what to do in each situation, the process presented in this thesis has different goals. The methodology in this thesis provides a systematic process to identify non-failing state evolutions and label them with a value corresponding to a likelihood of failure. The purpose of this research is to provide a process by which the boundaries between a failure and non-failure can be identified, without regard to optimality in any sense.
There is not necessarily a goal state to reach or reward to be obtained. Rather, we seek to satisfy the primal goal of first not failing without considering any controllers.
Chapter 3

Study of the planar hopping robot model

3.1 Dynamic model

The Pogo-bot is a prismatic inverted pendulum with a reaction wheel. The leg is massless and can apply unidirectional forces in the line that connects the foot contact point with the center of mass. The reaction wheel can apply torque on the body. A schematic is shown in Figure 3-1 and physical model parameters are given in Table 3.1.

Figure 3-1: The pogo-bot is a prismatic inverted pendulum with a reaction wheel and massless leg. The leg can apply unidirectional linear force and the reaction wheel can apply torques on the body.
Table 3.1: Pogo-bot model parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body mass</td>
<td>$M_B$</td>
</tr>
<tr>
<td>Body inertia</td>
<td>$I_B$</td>
</tr>
<tr>
<td>Reaction wheel mass</td>
<td>$M_R$</td>
</tr>
<tr>
<td>Reaction wheel inertia</td>
<td>$I_R$</td>
</tr>
<tr>
<td>Leg force on the ground</td>
<td>$F_{leg}$</td>
</tr>
<tr>
<td>Reaction torque input</td>
<td>$\tau$</td>
</tr>
<tr>
<td>Allowed leg length</td>
<td>$L$</td>
</tr>
<tr>
<td>Gravity</td>
<td>$g$</td>
</tr>
</tbody>
</table>

During the ground phase, foot is assumed to be stuck to the ground. A constant leg thrust force $F_{leg}$ and a torque from the reaction wheel motor $\tau$ are applied to the body.

\[
M_{total} = M_B + M_R \\
\dot{\theta} = \frac{\tau + M_{total}L(g \sin(\theta) - 2\dot{L}\dot{\theta})}{M_{total}L^2 + I_B} \\
\ddot{L} = \frac{M_{total}\dot{L}\theta + F_{Leg} - M_{total}g \cos \theta}{M_{total}} \\
\phi = -\frac{\tau}{I_R}
\]  

The transition to flight occurs when the robot leg length has reached $L = L_{takeoff}$, which is set to the be the maximum leg travel length. This is the pre-takeoff state that is used as the basis for constructing the Viable state network. The pre-takeoff state maps to the post-takeoff state with

\[
\dot{L}(post \ takeoff) = 0
\]

During the flight phase, the center of mass follows ballistic dynamics.

\[
\dot{x} = 0 \\
\dot{z} = -g
\]

The controls applied during the flight to cycle the leg to the desired landing
configuration are given by Equations (3.8)-(3.12). As these are not intrinsic to the physical hardware of the robot, we consider these choices to be part of the definition for the behavior class. The landing foot location plays a critical role in many one legged hoppers and are often chosen to stabilize a certain desired behavior. These are typically constructed in some feed-forward fashion that relies on the forward speed of the robot, wherein the foot is placed further forward with increasing forward speed. In the Raibert formulation [47] [46], the landing position of the foot is chosen so that the leg angle at landing is about equal and opposite the angle at liftoff, to ensure steady state forward velocity. In relation to this nominal step position, a shorter step causes forward acceleration and a longer step causes deceleration. In the Capture Point formulation [44], the foot landing position is placed to attempt to stop the robot in an upright position after landing. In relation to the capture point, a shorter step has a tendency to allow the robot to continue moving forward and a longer step can reverse the direction of travel. For the nominal pogo-bot, we are primarily interested in exploring additional behavior beyond a defined controller. The capture point is used as a reference for ‘unbiased’ placement position for the foot upon landing for transient start-stop behavior. These inform the choice of $K_{\text{step}}$ in (3.8).

\[
\text{step length} = K_{\text{step}} \dot{x} \quad (3.8)
\]

\[
|\text{step length}| < L_{\text{land}} \quad (3.9)
\]

\[
\theta_{\text{ref}} = \arcsin\left(\frac{\text{step length}}{L_{\text{retract}}}\right) \quad (3.10)
\]

\[
\tau = K_{F,\theta}(\theta_{\text{ref}} - \theta) - K_{D,\theta}\dot{\theta} \quad (3.11)
\]

\[
\dot{L} = K_{P,L}(L_{\text{retract}} - L) \quad (3.12)
\]

The end flight phase occurs when the foot first touches the ground, regardless of whether the robot is in the desired landing configuration. The foot is assumed to stick to the ground upon contact, resulting in the following impact dynamics.
\[
\begin{align*}
\dot{\theta}^+ &= \frac{M_{\text{total}}(-\dot{z}L\sin(\theta) - \dot{x}L\cos(\theta)) + I_B\dot{\theta}}{M_{\text{total}}L^2 + I_B} \\
\dot{L}^+ &= -\sin(\theta)\dot{x}^- + \cos(\theta)\dot{z}^-
\end{align*}
\] (3.13) (3.14)

Table 3.2: Pogo-bot control parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leg length at takeoff</td>
<td>(L_{\text{takeoff}}) 0.8 m</td>
</tr>
<tr>
<td>Nominal leg length after landing</td>
<td>(L_{\text{land}}) 0.65 m</td>
</tr>
<tr>
<td>Retract leg length</td>
<td>(L_{\text{retract}}) 0.7 m</td>
</tr>
<tr>
<td>Foot landing adjustment*</td>
<td>(K_{\text{step}}) 0.26 s</td>
</tr>
<tr>
<td>Aerial phase leg angle gain</td>
<td>(K_{P,\theta}) 100 Nm/rad</td>
</tr>
<tr>
<td>Aerial phase leg angle gain</td>
<td>(K_{D,\theta}) 20 Nm/(rad/s)</td>
</tr>
<tr>
<td>Aerial phase leg length gain</td>
<td>(K_{P,L}) 100 s(^{-1})</td>
</tr>
</tbody>
</table>

\* The capture point is a reference for choosing \(K_{\text{step}}\). It is defined as \(K_{\text{step}} = \sqrt{\frac{h}{g}}\) where \(h\) is constant height of the center of mass. Since the pogo-bot does not make any attempt to maintain body height during the stance phase, \(L_{\text{land}}\) is used instead as the body height which provides a baseline \(K_{\text{step}} = \sqrt{\frac{L_{\text{land}}}{g}} = 0.26\). Behaviors are defined by choosing a \(K_{\text{step}}\) in reference to this value.

3.2 Constructing the Viable State Network

In order to generate the Viable State Network, specific to the generic pogo-bot, we define the regions of failure and for the home state using the process in Chapter 2.

**Failure Region** The failure region is defined using a threshold height of the center of mass given by \(z < z_{\text{thresh}}\). Failure can occur at any phase of the robot’s operation so long as the condition is met.

**Starting States** For the nominal pogobot, we choose the starting state to be the robot standing upright on the ground with \(\theta = \dot{\theta} = \dot{L} = 0\) and \(L = L_{\text{start}}\).
**Home States** The home states are set to be a conservative region of states when the robot is operating in the ground phase, in which a reasonable controller can bring the robot to an upright standing stop. Equation (3.15) constrains the space of $\theta$ and $\dot{\theta}$ depending on the height length of the leg. The strictness of the states is tuned by parameter $\alpha_1$, where smaller values of $\alpha_1$ require the robot to be closer to upright and have slower speed at some point during the ground phase. Equation (3.16) constrains the velocity in the direction of leg shortening to below what a fraction ($\alpha_2$) of the maximum force can stop the body from crashing into the ground. The values chosen for the nominal pogo-bot are shown in Table 3.3

$$\theta^2 + \dot{\theta}^2 - \alpha_1/L < 0$$  
(3.15)

$$-\dot{L}^2 + \frac{2L(\alpha_2 F_{\text{max}} - g M_{\text{total}})}{M_{\text{total}}} < 0$$  
(3.16)

Table 3.3: Nominal Pogo-bot exploration parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$z_{\text{thresh}}$</td>
<td>0.1 m</td>
</tr>
<tr>
<td>$L_{\text{start}}$</td>
<td>0.5 m</td>
</tr>
<tr>
<td>$\alpha_1$</td>
<td>0.0025</td>
</tr>
<tr>
<td>$\alpha_2$</td>
<td>0.7</td>
</tr>
</tbody>
</table>

**Binning parameters:** The simulation for the nominal pogo bot uses 501 discretized bins for each variable in the pre-takeoff space. $-1.2 < \theta < 1.2 \text{ rad}$, $-6 < \dot{\theta} < 6 \text{ rad/s}$, $0 < \dot{L} < 5 \text{ m/s}$ with bin sizes $\Delta \theta = 0.005 \text{ rad}$, $\Delta \dot{\theta} = 0.02 \text{ rad/s}$, $\Delta \dot{L} = 10 \text{ mm/s}$. If a simulation returns a pre-takeoff state outside of these bounds, it is considered to have reached a failure condition. These give the robot $501^3 = 125,751,501$ distinct pre-takeoff states that the robot can explore.

**Graph edges / Actions:** In the nominal pogo-bot, the choice of action specifies an open loop control during the ground phase. For the chosen behavior class, the robot executes a fixed controller during the aerial phase.
When the robot is in the ground phase, the robot executes a flat force and torque profile for the duration of stance. A total of 231 combinations of force and torque are sampled. The values are shown in Figure 3-2. The space of sampled actions does not uniformly cover the space of possible forces and torques. Since the robot requires a vertical force of $M_{\text{total}g}$ to stand vertically, any forces below this threshold are unlikely to produce interesting behavior. The average vertical impulse must counteract gravity for any sustained locomotion, thus the vertical force during intermittent ground contact must exceed $M_{\text{total}g}$ [41]. To focus computation on actions that are likely to produce more hopping, the sampled leg forces are biased to those between $M_{\text{total}g}$ and $2M_{\text{total}g}$.

![Figure 3-2: (a) Control actions chosen for pogobot network. Each combination of force and torque is applied when the robot is in contact with the ground. (b) The robot applies a constant force and constant torque immediately following the impact event, through the ground phase, until the takeoff event.](image)

**Implementation:** Dynamic simulations were implemented in MATLAB using ODE45 with event detection. Computation was done using the pMATLAB toolbox [27] on Lincoln Lab Supercloud [52].

### 3.3 Exploration results and construction of the VSN

The results of the exploration and propagation steps outlined in Section 2.1.3 are shown in Figure 3-3. All takeoff states that can reach the home states are shown...
in green and the remaining reachable states that inevitably fail are shown in red. Notable statistics of the process are shown in Table 3.4.

Figure 3-3: Viable and non-viable states in the space of pogobot takeoffs. States shown in green are viable and have a path that leads to the home states. States shown in red are non-viable and will eventually fail.

Figure 3-4 shows the complex paths in state space that the robot in a viable state can take to reach a home state. The labeling of the viable states by the fewest number of steps to a home state can be interpreted as a possible risk quantity. For instance being fewer steps from a home state could be considered to be safer, whereas a state with large number of steps to the home state is more risky. Depending on user preferences, this may be a sufficient quantification. Section 3.4 provides a more detailed quantification based on probabilistic methods.

A detailed examination of the pre-takeoff states that can reach the home states upon landing are shown in Figure 3-6) which are the same set as in Figure 3-4(0). This set contains a central group of pre-takeoff states centered around $\theta = \dot{\theta} = 0$. 

81
Table 3.4: Pogo-bot exploration and propagation summary

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viable states</td>
<td>6,443,011</td>
</tr>
<tr>
<td>Non-viable states</td>
<td>2,614,470</td>
</tr>
<tr>
<td>Total number of state discretizations</td>
<td>125,751,501</td>
</tr>
<tr>
<td>Coverage of reachable states</td>
<td>7.2 %</td>
</tr>
<tr>
<td>Coverage of viable states</td>
<td>5.1 %</td>
</tr>
<tr>
<td>Number of state action pairs simulated</td>
<td>$2.09 \times 10^9$</td>
</tr>
<tr>
<td>Number of state action pairs resulting in direct failure</td>
<td>$1.10 \times 10^9$</td>
</tr>
</tbody>
</table>

These states are consistent with local stability notions wherein the configuration and state of the robot is close in Cartesian space to the states in the set of home states. There exist additional states outside of the central cone that will also reach the home states upon landing. In these, $\text{sign}(\dot{\theta}) = -\text{sign}(\theta)$, which means that the robot becomes more vertical in the air and can reach upright standing upon landing. A small group of states that are less intuitive around $\theta = 0.6$, $\dot{\theta} = 2$ show that upon liftoff, the robot becomes less vertical in the air. Through a combination of the aerial phase control and a ground phase action, the robot is able to reach the home states after landing.

Figure 3-5 demonstrates one of the features of approaching the problem from the perspective of non-failure. Figure 3-5(0) shows the pre-takeoff states that will fail before reaching any home state or another takeoff for any of the applied control actions in the sampled set. In the graph, the associated nodes will terminate to the failure node. Each sub diagram shows the states that are certain to fail and the fewest number of hops required to reach a state in which any controls will cause the robot to fail. In the case of the nominal pogo-bot, there exist pre-takeoff states that will eventually lead to failure but may require 5 hops to reach. A one-step planner could easily allow the robot to complete several successful hops without detecting danger, having already passed the point of no return. While there would likely be actions that lead to immediate failure at each step, there would be at least one successful next takeoff that the planner could choose. However, the robot would reach the failure condition after a sequence of steps. This example illustrates the value of identifying the non-viable states. Typical methods that look ahead, such as Model Predictive
Control with limited horizons, would require extensive computation to avoid such states. Using a precomputed VSN, a controller could choose to avoid hard-to-detect non-viable states.

Although the VSN is built on a Poincaré section of the pogo-bot's trajectory, it is important to avoid interpreting it in the context of local stability. Figure 3-3 shows that a significant portion of the viable states satisfy $\text{sign}(\dot{\theta}) = \text{sign}(\theta)$. This condition implies that the robot has a tendency to become less upright in the air and local stability notions would view these states with skepticism. However, since the robot is under control in the aerial phase and attempts to land according to (3.8), these states do not necessarily lead to failure.

Figure 3-4: Fewest number of steps from home state. Each subfigure shows states at pre-takeoff categorized by the shortest path (in steps) to the home states. The number above the plot indicates the fewest number of additional takeoffs before the robot can reach the home states. The coloring is for clarity of visualization. States are colored by distance from $\theta = \dot{\theta} = 0$. The color of a region of state space is consistent across all subfigures.
Figure 3-5: Fewest number of steps to guaranteed failure. The failure states are those indicated by subfigure (0), in which all sampled actions from these pre-takeoff states resulted in the robot reaching the failure condition, without reaching any home state or another successful takeoff. Each subfigure shows states at pre-takeoff categorized by the shortest path (in steps) to a failure state in subfigure (0). The number above the plot indicates the fewest number of additional takeoffs before the robot can reach a state in which all actions directly lead to the failure condition. The coloring is for clarity of visualization. States are colored by distance from $\theta = \dot{\theta} = 0$. The color of a region of state space is consistent across all subfigures.
Figure 3-6: Pre-takeoff states that will reach home states after the next landing. The coloring is for clarity of visualization. States are colored by distance from $\theta = \dot{\theta} = 0$. The color of a region of state space is consistent across all subfigures.
3.4 Risk Labeling

In order to find the safety value of each state in viable pogo-bot takeoff states, the risk quantification method in Section 2.2 is applied to the VSN computed in Section 3.3. To encompass all behaviors and goals the robot might want to achieve, we apply a random control policy. For every explored state, all actions in the set of allowed actions were simulated. The random control policy gives each action equal probability. Figure 3-7 shows the four hypothetical transition types that can occur in the network. In this scenario, action 1 leads to another takeoff state. Action 4 leads to failure. However, action 2 and action 3 both pass through the user defined home states before reaching a successful takeoff or failure, respectively. Critically, the home states are defined when the robot is in contact with the ground and if the robot satisfies those conditions during its operation, it can stop safely. We introduce a choice to stop safely when the robot is inside of the home states. If the robot does not choose to stop in the home states, it continues the same action until it reaches another takeoff or failure. In the generic pogo-bot with the random control policy, the robot chooses to stop safely in half of the cases. The probability of an action is then split between the two outcomes as shown in Figure 3-8.

The results of the risk quantification are shown in the computed risk maps in Figure 3-9 and Figure 3-10. In these, all of the viable states in the VSN are shown with the corresponding computed safety value. In general, the safety value shows smoothness with respect to Cartesian state. That is, local groups of states have similar safety values. Due to this fact, there appears to be structure, which would allow the risk map to be compressed or interpolated, a subject of future work.

In Figure 3-11, we observe the minimum number of the steps needed to reach the home states and computed safety values. As expected in the Markov Chain formulation of the step to step transitions, there is a weak correspondence between the two. In general, the safety value drops as the number of steps to reach the home states increases.
Figure 3-7: Four types of transitions for the generic pogo-bot.
Figure 3-8: In some trajectories, the robot will successfully land and pass through the set of home states. At this point, the robot has two choices. It can choose to stop or continue to apply the same action. If the robot chooses to stop, it will apply a controller that takes it to the starting states. If the robot continues the same control action, it may successfully takeoff again or fail before reaching takeoff. Each control action is split into two graph edges representing the two outcomes. The random control policy gives these edges equal weights. (a) Case in which the robot passes the set of home states and can then takeoff. (b) Case in which the robot passes the set of home states and can then fail if it does not choose to stop.
Figure 3-9: Risk map with the safety values of states in the takeoff space of the pogobot. The color value corresponds to $\log_{10}(\text{safety value})$ at each pre-takeoff state.
Figure 3-10: Risk map with the safety values of states in the takeoff space of the pogobot. The color value corresponds to $\log_{10}(\text{safety value})$ at each pre-takeoff state. The surrounding surface shown in light red corresponds to non-viable states.
Figure 3-11: Fewest number of steps to the home states. Each subfigure shows states at pre-takeoff categorized by the length of the shortest path (in steps) to the home states. The number above the plot indicates the fewest number of additional takeoffs before the robot can reach the home states. The color value corresponds to log10(safety value) at each pre-takeoff state.
3.5 Observations on the VSN and Risk Map

The results obtained in Section 3.4 show the difficulty of using bounds and Cartesian
distance measures in the state space as proxies of risk. Figure 3-10 shows that the
boundary of the computed viable space in the state space visualizations does not have
consistent correspondence with the safety value. We identify two types of boundaries
in the Risk Map. In the first type, parts of the viable space do not border (in a
Cartesian sense) the space of non-viable states. These form the reachable boundary
of the robot. If the robot had access to additional actions, possibly from increased
capability of the actuators, then this boundary might be extended. In the second
type of boundary, the viable states directly border the space of non-viable states.

The complexity of the border between viable and non-viable states is shown for
two examples in Figure 3-12. In Figure 3-12b, the safety value does not diminish for
states nearing the boundary with the non-viable states. In Figure 3-12a, the safety
values of viable states noticeably decrease for states nearer to the boundary with the
non-viable states. This case embodies an assumption when working in the regulation
paradigm. The regulation paradigm views states as safer when they are ‘closer’
(in a Cartesian sense) to stable or desired states. Similarly, it follows that states
are more dangerous when they are ‘closer’ to a known failure condition. Indeed, the
central cone of states in Figure 3-6 have the highest safety values among viable states.
However, this paradigm does not generalize throughout the Risk Map.

To further explore the relationship of distance in Cartesian space compared to
the safety value, a generic $L^2$ norm is constructed. This metric is based on the dis-
cretization of the state space during the pre-takeoff event for the pogo-bot. Distance
between two states located at bin centers $X_i$ and $X_j$ in the discretized state space is
defined by

$$D = (X_i - X_j)^T \begin{bmatrix} 1/(\Delta \theta)^2 & 0 & 0 \\ 0 & 1/(\Delta \dot{\theta})^2 & 0 \\ 0 & 0 & 1/(\Delta \dot{L})^2 \end{bmatrix} (X_i - X_j)$$

(3.17)
Figure 3-12: Sections of the risk map plotted with the non-viable states. The color value corresponds to $\log_{10}(\text{safety value})$ at each pre-takeoff state. The surrounding points in light red are the non-viable states. This figure illustrates the difficulty of reasoning about boundaries and the safety values. (a) On the left side boundary, there is a gradient in safety values, going from high to low as the distance to boundary decreases. (b) On the left side boundary, there is no such gradient in safety values near the boundary. Viable states at the boundary have relatively high safety values $-2 < \log_{10}(\text{safety value}) < 0$.

\[
X_i = [\theta_i, \dot{\theta}_i, L_i]^T.
\]

$\Delta \theta$, $\Delta \dot{\theta}$, and $\Delta L$ are sizes of the discretization for each variable as defined in Section 3.2.

Figure 3-13 shows a plot of all viable states compared by computed safety value and distance to a non-viable state. Figure 3-14 presents the same results in histogram. With the safety values on a linear scale, there appears to be no correspondence between the distance to a non-viable state and its safety value. However, with the safety values on a logarithmic scale, it is possible to draw two conclusions. First, the lower bound of the safety value increases with increasing distance from the non-viable states. Second, the upper bound on the safety value does not appear to depend on the distance from the non-viable states. Thus it is safer to be far away from failing states, but not necessarily more dangerous to be close to failing states.
Figure 3-13: Relationship between Cartesian distance margin to non-viability and safety value in the risk map. Cartesian distance is measured in the number of state space discretizations. (a) Scatter plot with the safety value on a linear scale (b) Scatter plot with the safety value on a log10 scale.
Figure 3-14: Relationship between Cartesian distance margin to non-viability and the safety value in the risk map. The coloring represents the relative proportion of viable states appearing in each histogram bin. Bins with no data-points are left blank. Cartesian distance is measured in the number of state space discretizations. (a) Histogram with the safety value on a linear scale (b) Histogram with the safety value on a log10 scale.
Examine the implications of conservative control    The above results demonstrate the value of reasoning about the safety or risk of a state by forward projecting all possibilities until failure. Using this method, we have identified complexities beyond the conventional reasoning that there exists a simple correspondence between a state’s similarity to danger and its risk. One limitation however, is that this analysis only applies to the behaviors caused by the robot’s own actuation. In order to add robustness against external influences or unmodeled dynamics, it may be prudent to incorporate ideas of margins at the boundary that minimize the likelihood of unexpectedly being in the non-viable states.

Noise and uncertainty could drive any state to some nearby states, so there is a possibility that viable states may be driven to non-viable states. To protect against that possibility, it is possible to apply conservative control policies to gain robustness to noise. In order to model this situation, we use the framework in Section 2.2.2. In the most conservative assumption, a catastrophic noise acts on viable states that are near non-viable states. This noise causes each affected viable state to transition to a non-viable state with probability 1. Using the Cartesian distance to the nearest non-viable state as shown in Figure 3-13, we define a set of affected states that are below a threshold distance value. Viable states within the threshold distance are off-limits and are now considered en route to guaranteed failure. These assumptions have similar effects as using cost functions to avoid non-viable states in optimization-based planning.

The result of marking viable states within the capability set as off-limits has the possibility of reducing the capability beyond the naive removal of those states. States whose forward evolutions must pass through a removed state to reach a Home State, become non-viable. States can also become unreachable if they depend on the forward evolution from the removed states. This process is illustrated in Figure 3-15. Figure 3-16 and Figure 3-17 show the loss of viable states due to the removal of states within a threshold distance of 1 and 4 respectively from the nearest non-viable state. For comparison, the same number of states were removed using the Safety Value as the criterion. Figure 3-18 and Figure 3-19 show the loss of viable states due to the
removal of states below the Safety Value threshold of $10^{-3.59}$ and $10^{-2.63}$ respectively.

Figure 3-15: Illustration of capability reduction by loss of viability and loss of reachability. (a) A set of states and possible transitions in the VSN. (b) State B is removed for conservative control. (c) As a consequence of removing state B, state A becomes non-viable and state C becomes unreachable.

In order to compare the loss of states due to the removal of ‘dangerous’ states in different situations, we examine the normalized loss of capability. The normalized number of removed states is defined as the proportion of viable states that are removed due to conservative control assumptions. The normalized loss of capability is the proportion of viable states that are removed, become non-viable or become unreachable. Figure 3-20 shows the normalized loss of capability by the normalized number of removed states for two removal methods. Especially when removing states by distance to a non-viable state, there is considerable loss in total capability, shown in blue. This effect is much less pronounced when removing states by Safety Value, shown in red. Although the Safety Value appears superior in minimizing loss of capability, it is important to note that the two removal methods consider robustness on fundamentally different criteria. It is clear, however, that conservative control assumptions can incur additional loss of capability. Users of this methodology can choose which assumptions are most appropriate for the situation.
Figure 3-16: Effect of removing viable states within distance 1 to a non-viable state. The red states are those that are removed. The black states are additional states that become non-viable. The green states are the additional viable states that are no longer reachable. These plots correspond to data point A1 in Figure 3-20.

**Geometric significance of the Viable states in state space** Although the VSN and Risk Maps appear to occupy geometric volumes in state space, it is important to note that figures shown in this chapter are Cartesian projections of a Poincaré section. Step to step transitions do not necessarily move the robot state incrementally through the volume of state space. Thus it is difficult to define a clear metric of risk in the Cartesian state space. Therefore this thesis focuses on a probabilistic construction of the risk quantity.
Figure 3-17: Effect of removing viable states within distance 4 to a non-viable state. The red states are those that are removed. The black states are additional states that become non-viable. The green states are the additional viable states that are no longer reachable. These plots correspond to data point A4 in Figure 3-20.
Figure 3-18: Effect of removing viable states with safety values less than $10^{-3.59}$. The red states are those that are removed. The black states are additional states that become non-viable. The green states are the additional viable states that are no longer reachable. These plots correspond to data point B1 in Figure 3-20.
Figure 3-19: Effect of removing viable states with safety values less than $10^{-2.63}$. The red states are those that are removed. The black states are additional states that become non-viable. The green states are the additional viable states that are no longer reachable. These plots correspond to data point B4 in Figure 3-20.
Figure 3-20: Tradeoff in capability by removing viable states. A1-A8 denote instances in which all viable states within distance 1-8 of a non-viable (NV) state are removed. Distance is measured in the number of state space discretizations. B1-B8 denote instances in which all states below a threshold safety are removed. B1: safety threshold $10^{-3.59}$, B2: safety threshold $10^{-2.94}$, B3: safety threshold $10^{-2.70}$, B4: safety threshold $10^{-2.63}$, B5: safety threshold $10^{-2.55}$, B6: safety threshold $10^{-2.47}$, B7: safety threshold $10^{-2.41}$, B8: safety threshold $10^{-2.35}$. The dotted line represents the baseline loss in capability purely due to removing states.
3.6 Variations on controlled behavior

As seen in the legged locomotion literature, the leg landing position has a critical effect on the local limit cycle stability of single leg hoppers. In contrast, the study of the generic pogo-bot focuses on actions in the ground phase actuation profiles. The behavior class (Definition 2.1.6) encompasses the design choices within the use of the risk quantification methodology which constrain the controlled behavior of the robot. The actual control is defined in primitives and the use of different primitives may allow the robot to exhibit different capability (viable states) and safety values.

In this section, we examine the results of changing the step length definition by varying $K_{step}$ in Equation (3.8). We use the values of $K_{step}$ shown in Table 3.5 and compute a complete VSN and Risk Map for each instance, otherwise keeping the same model and parameters as shown in Sections 3.1 and 3.2.

<table>
<thead>
<tr>
<th>$K_{step}$</th>
<th>Short step</th>
<th>Nominal step</th>
<th>Long step</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.17</td>
<td>0.26</td>
<td>0.43</td>
</tr>
<tr>
<td>Number of Viable states</td>
<td>6,552,943</td>
<td>6,443,011</td>
<td>3,973,399</td>
</tr>
<tr>
<td>Number of Non-Viable states</td>
<td>1,125,160</td>
<td>2,614,470</td>
<td>6,829,316</td>
</tr>
</tbody>
</table>

Figure 3-21 shows the computed Risk maps for the three step length parameters. While the capability spaces appears to show a similar structure in the pre-takeoff state space, there are clear differences. For instance, the long step $K_{step} = 0.43$ finds a region of viable states around $\theta = 0.6, \dot{\theta} = 3$ that are not found with the other two. Additionally, there is a notable difference in safety values across the three Risk Maps. Figure 3-22 shows the cumulative probability distributions of the safety values for each Risk Map. These give the proportion of viable states that have safety values within the range of 0 to $x$. Irrespective of the ground phase control actions, the behavior class with shorter steps produces overall safer Risk Maps. That is, with a fixed control strategy in the air and a random control policy for ground phase actuation, a more vertical landing angle tends to produce safer viable states. These viable states are statistically more likely to reach a home state.
Since all variations of the behavior class are achievable with the same robot hardware, the results of exploration can be combined incrementally to expand the set of known viable states. In the above examples, there are 3 self contained VSNs, each representing a different prescribed behavior in the air. The most conservative approach is to combine the sets of viable and nonviable states for the three behaviors. $V_{new} = \bigcup_i V_i$ and $NV_{new} = \bigcup_i V_i \setminus V_{new}$ where $V_i$ and $NV_i$ are the sets of viable and non-viable states generated for each behavior. For a less conservative estimate, assemble all the graph edge entries from all 3 variations and rerun Algorithm 2. Figure 3-23 shows the combined Risk Map after merging graph entries, recomputing the VSN with Algorithm 2 and the safety values using the process in Section 2.2.

<table>
<thead>
<tr>
<th>Merge method</th>
<th>Viable states</th>
<th>Non-viable states</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set</td>
<td>10,276,260</td>
<td>4,885,956</td>
</tr>
<tr>
<td>Graph edge</td>
<td>10,488,376</td>
<td>4,673,840</td>
</tr>
</tbody>
</table>
Figure 3-21: Viable states and corresponding safety values for three different step length parameters. The color value corresponds to log10(safety value) at each pre-takeoff state.
Figure 3-22: Cumulative distribution of safety values for the transient pre-takeoff states shown for three different step length parameters
Figure 3-23: Risk map with the safety value of states in the takeoff space of the pogobot. This risk map combines 3 risk maps that are generated for variations in the leg angle at landing. The color value corresponds to log10(safety value) at each pre-takeoff state.
3.7 Conclusions

In this chapter, the methodology presented in Chapter 2 was applied to the study of a generic pogo-stick hopping robot. The results show that the simple expression of the legged locomotion using the planar single legged hopper can produce complex behavior, far beyond what typical regulation based controllers imply.

We have successfully generated the Viable State Network (VSN), which indicate the failing and non-failing states of the robot. This will enable the understanding of the maximal capability of the robot model. We have also successfully computed Risk Maps, which encode a safety value for each state in the VSN. This risk quantity forward projects all possible state evolutions that lead to the safe *Home States*, and assigns an appropriate value of safety or risk.

The tools developed in Chapter 2 provide a framework that is well suited to answer additional questions that typically arise in the design of robot controllers. Using these tools, we showed the correspondence between a Cartesian risk measure and the Safety Value generated from an Absorbing Markov Chain. We established that for the example model, Cartesian distance in state space relates to the lower bound of the Risk Map Safety Value but not the upper bound. Farther distance from failing states results in high safety values but states near the boundary formed by viable and non-viable states can show any value of safety.

Further exploring the typical techniques for conservative control, we showed the trade-off in capability that results from restricting the robot to a reduced set of robust states in response to risk measures.

We additionally explored variations in control parameters by choosing leg landing angles for the robot. The computed Risk Maps demonstrated how control policies can influence the general safety of all states.
Chapter 4

Extensions

4.1 Usage in control

With the capability space and risk quantity computed in Chapter 3, it is possible to incorporate these values in the design of controllers. For instance, the user may choose to bound the amount of acceptable risk that the robot is allowed to experience. As with anything in life, the consequence of risk of failure (or falling in this case) must be weighed against the desire to achieve the desired motion goals. Consider the following example in which the pogo-bot is commanded to follow a forward velocity trajectory at each takeoff shown in Equation 4.1. With the robot operating on the state-action directed network computed in Chapter 3, the robot chooses the next state that most closely matches the desired velocity and is within the risk threshold.

$$\min_{\text{next action}} |\dot{x}_{\text{next}} - \dot{x}_{\text{des}}|$$

subject to \( \text{safety value(next state)} > \text{threshold} \)  \hspace{1cm} (4.1)

Figure 4-1 shows the resulting behavior under different risk tolerance thresholds and the computed risk values presented in Chapter 3. There is a clear tradeoff in ability versus risk tolerance. The tight threshold of \(10^{-1}\) constrains the states to a set in which the most unsafe takeoff state has a 1 in \(10^1\) chance of reaching the home state with a random control policy. In this case the robot barely moves and is
clearly constrained by the risk threshold. Figure 4-1b shows the same control scheme operating with a relaxed risk threshold of $10^{-10}$. In this case, the robot was able to follow the velocity target trajectory with much greater success. The takeoff states that were chosen had a worst case safety of $10^{-7}$, well within the desired threshold. While it is simple to make a comparison of timid versus confident behavior, the actual threshold values would need to be tuned by the user.

A related controller is one that optimally decreases the risk or increases the safety at each step. Such a controller would be useful when the robot senses that the probability of failure is too high and needs to temporarily prioritize safety over any other goal or control target. This risk gradient control is given by Equation 4.2 and an example result is shown in Figure 4-2. In this example, the state at the first takeoff has a safety value of $10^{-12}$. Over subsequent steps, the robot is able to consistently increase the safety value at each takeoff. The state at each takeoff does not monotonically converge to the safe standing state which is $\theta = 0, \dot{\theta} = 0$. This behavior is different compared to when the robot uses a controller that regulates to a safe standing state using state space metrics.
Figure 4-2: Behavior of a risk gradient controller. The robot starts at takeoff state with safety value $10^{-12}$ and chooses the action that results in the safest next takeoff available.

$$\text{maximize} \quad \text{safety value}(\text{next state})$$  \hspace{1cm} (4.2)

Additionally, graph search can be employed as a method of planning sequences of steps in real-time while adapting to changes in the external environment. The VSN contains all viable states and the control actions that link each pair of states, with the assumption of ideal conditions. There may be instances during real-time control of the robot in which certain states or actions may not be available or desired. In response to these situations, the corresponding nodes and edges of the graph can be removed and replaced. The remaining subgraphs of the VSN would indicate the forward looking capabilities of the robot as bound by the updated constraints. With changing conditions at every step, the robot could build these subgraphs and use them to plan the subsequent step.

### 4.2 Whole capability analysis

The Viable State Network and Risk Maps produced using the methods introduced in Chapter 2 give the user a wide snapshot of a robot's capabilities that is generally applicable to all scenarios that the robot faces. As an analytical tool, the results from a single model can inform control strategies and new behaviors. Additionally, the
computed Viable State Network represents the full capability of the robot.

Instead of viewing hardware as fixed when computing the robot’s capabilities, it may be possible to encode the user’s desires directly in the VSN and Risk maps. One can imagine a myriad of applications in which this could be used. For instance, a designer might want a robot with the greatest number of viable states in the VSN. Alternatively, a need might arise for a robot with a specific region of viable states in the VSN with consideration of other factors present in robot design, such as minimizing hardware complexity. In another case, the user may choose to maximize the safety values for a region of desired capabilities. With quantities describing the capability of the robot as well as the safety value at each state, it is possible to explore the contributions of underlying design parameters to those quantified values.

In this proposal, if the desires of the user can be described in terms of the VSN or risk map, then it will be possible to observe how model variations can get closer to these goals. This differs from the analysis in Section 3.6, which examined control parameter choices. Changes to the controller are within the same robot hardware model, so computations with varied control parameters provided add to the viable states of the robot. In contrast, physical parameter changes mean that each instance of fixed hardware has its own unique viable and non-viable space.

Figure 4-3 shows how the Risk Map and the set of viable states change in response to hardware parameter variations. Starting from the nominal model, four variations of body mass and body inertia are simulated while keeping all Behavior Class choices constant. Each column shows the Risk Map, the added viable states compared to the nominal model and the missing viable states compared to the nominal model. The capability measure can be inferred from the number of discretized state space bins that are in each Risk Map. A larger value means that the robot can utilize more states. From this sweep of parameter choices, users can observe how the VSN responds and thus make intelligent choices to guide the design of the robot to the desired capability. The plot labeled ‘common to all’ shows the states that are included in the viable set for all parameter variations. This can be interpreted as the set of states robust to small design changes or model error.
Figure 4-3: Risk Maps and capability space comparisons for physical parameter variations. The first shows the Risk Maps for each model variation. The coloring represents the log10(Safety Value) of each state and is consistent across all of the Risk Maps. The second and third rows show operations on the set of viable states, for each model variation. In the second and third rows, the coloring is for clarity of visualization. States are colored by distance from \( \theta = \dot{\theta} = 0 \). The color of a region of state space is consistent across all of these subfigures.
4.3 Adapting to Pogo-bot hardware

Figure 4-4 shows the proposed hardware design of the pogo-bot. The leg is a carbon fiber tube, driven by a cable-drive mechanism. Physical parameters taken from the CAD model are provided in Table 4.1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body mass</td>
<td>$M_B$ 2 kg</td>
</tr>
<tr>
<td>Body inertia</td>
<td>$I_B$ 0.05 kg-m$^2$</td>
</tr>
<tr>
<td>Reaction wheel mass</td>
<td>$M_R$ 0.21 kg</td>
</tr>
<tr>
<td>Reaction wheel inertia</td>
<td>$I_R$ 0.016 kg-m$^2$</td>
</tr>
<tr>
<td>Allowed leg length</td>
<td>$L$ [0.3,0.8] m</td>
</tr>
<tr>
<td>Leg force on the ground</td>
<td>$F_{leg}$ [0, 60] N</td>
</tr>
<tr>
<td>Reaction torque input</td>
<td>$\tau$ [-1,1] Nm</td>
</tr>
</tbody>
</table>

In an effort to find the full capability of the hardware model, the VSN and Risk Maps are computed. The model follows the nominal model in Chapter 3 except in the physical parameters, binning parameters and the action samples. The remaining Behavior Class definition remains the same.

Binning parameters: The simulation for the nominal pogo bot uses 251 discretized bins for each variable in the pre-takeoff space. $-1.2 < \theta < 1.2$ rad, $-6 < \dot{\theta} < 6$ rad/s, $0 < \dot{L} < 5$ m/s with bin sizes $\Delta \theta = 0.01$ rad, $\Delta \dot{\theta} = 0.04$ rad/s, $\Delta \dot{L} = 20$ mm/s. If a simulation returns a pre-takeoff state outside of these bounds, it is considered to have reached a failure condition. These give the robot $251^3 = 15,813,251$ distinct pre-takeoff states that the robot can explore. Compared to the simulations in Chapter 3, the state space resolution is reduced by about 8 times to conserve computational effort.
Figure 4-4: Hardware pogo-bot
Figure 4-5: Actions sampled for the hardware pogo-bot model. Each point represents a single combination of force and torque profile that the robot applies during the ground contact phase.

**Actions:** The actions specify an open loop control during the ground phase. The specific combinations of force and torque are given in Figure 4-5. The leg force is sampled at 45 levels from 10 N to 60 N. The torque applied on the body is sampled at 41 levels from -1 N to 1 N. In total there are 1845 action combinations. Prior simulations with 231 action combinations as in Chapter 3 were inadequate for providing a sufficient number of viable states for visualization and control.

The computed VSN is shown in Figure 4-6. The Risk Maps are shown in Figure 4-7 with safety values on a logarithmic color scale, and in Figure 4-8 with safety values on a linear color scale. A summary of the computation results is provided in Table 4.2. The hardware model has a much smaller viable region compared to the example model in Chapter 3, Sections 3.3 and 3.4. While the coverage of the state space (reachable states) is similar between the two models, only 0.06% of the possible state space is viable. In this model, the viable states tend to form a narrow region around $\theta = 0$ and $\dot{\theta} = 0$. These are relatively vertical takeoffs with low angular velocity. Interestingly, this model includes non-safe states that do not necessarily lead to guaranteed failure, which are discussed in Section 2.1.3. They are shown in black in the Figure 4-6 and considered to be non-viable.
Table 4.2: Hardware Pogo-bot exploration and propagation summary

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viable states</td>
<td>9,574</td>
</tr>
<tr>
<td>Non-viable states</td>
<td>1,223,600</td>
</tr>
<tr>
<td>Total number of state discretizations</td>
<td>15,813,251</td>
</tr>
<tr>
<td>Coverage of reachable states</td>
<td>7.8%</td>
</tr>
<tr>
<td>Coverage of viable states</td>
<td>0.06%</td>
</tr>
<tr>
<td>Number of state action pairs simulated</td>
<td>$2.28 \times 10^9$</td>
</tr>
<tr>
<td>Number of state action pairs resulting in direct failure</td>
<td>$1.55 \times 10^9$</td>
</tr>
</tbody>
</table>

Figure 4-6: Viable and non-viable states in the space of pogo-bot takeoffs for the hardware model. States shown in green are viable and have a path that leads to the home states. States shown in red are non-viable and will eventually fail. The set of states shown in black do not have a path leading to the home state but can avoid failure by remaining indefinitely within the set. For the purposes of this research, these states are considered non-viable.
Figure 4-7: Computed Risk Map for the hardware pogo-bot. The color value corresponds to \( \log_{10}(\text{safety value}) \) at each pre-takeoff state.
Figure 4-8: Computed Risk Map for the hardware pogo-bot. The color value corresponds to Safety value at each pre-takeoff state
4.3.1 Modifications to the design parameters

Using a similar qualitative analysis as in Section 4.2, we observe the effect of slight design parameter changes. For instance, adding a protective roll cage around the body would result in higher mass and body inertia. Figure 4-9 shows the Risk Map of the model with 20% higher body inertia. Figure 4-10 shows the Risk Map of the hardware model with 10% higher body mass. A cursory observation of Risk Map for the model with higher body inertia shows two new viable regions with $|\dot{\theta}| \geq 0.8$. In contrast, the model with increased mass has largely the same Risk Map as the original hardware model.

![Figure 4-9: Computed Risk Map for the hardware pogo-bot with 20% body inertia compared to the original hardware model. The color value corresponds to log10(safety value) at each pre-takeoff state](image-url)
Figure 4-10: Computed Risk Map for the hardware pogo-bot 10% more body mass compared to the original hardware model. The color value corresponds to log10(safety value) at each pre-takeoff state.
Chapter 5

Conclusion

5.1 Contributions

This thesis provides a step towards the vision of highly capable and dynamic robots, which can execute maneuvers that seem innate to humans and animals. In this research, we specifically tackle the problem of identifying the maximal capability of the robot as defined by hardware limitations, and provide a measure of risk for states within that capability. The inspiration for this work was partly due to challenges experienced when constructing the HERMES humanoid system which is described in Section 1.2. Specifically in the HERMES tele-operated system, the risk perceived by the operator differed from a conservative estimate of what the robot could accomplish. There was a clear need for a method to identify the capability and quantify the risk for individual systems, to enable more complex tasks.

This thesis presents a methodology to quantify the risk of failure in dynamic robots. Applying the principles of Viability Theory for dynamic hybrid systems, we adopted a simulation data-driven approach to create a graph representing the outcomes of state evolutions. Using this graph, we identify viable and non-viable spaces for the robot, the result of which is introduced as the Viable State Network. To quantify the risk of failure among viable states, we use the Absorbing Markov Chain to model the likelihood of reaching a safe home state or failure. This framework is introduced the Risk Map.
The methodology was applied to the study of a planar single-legged hopper model, studied in Chapter 3. The computed VSNs and Risk Maps show that simple dynamic models can produce complex behavior, that was previously unidentified. Additionally this framework allowed examination of the Safety value in the Cartesian state space. We tested the assumptions of state similarity in Cartesian state space as a proxy of risk. It was observed that while increased Euclidean distance from failing states produced higher safety values, the converse did not hold true. Decreased distance to a failing state did not imply a lower safety value. We explored the implications of increasing robustness by removing dangerous states. We find that conservative control can greatly reduce the realizable capability of the robot, much greater than if only the dangerous states were unavailable.

This methodology was applied to several extensions as described in Chapter 4. First, an example usage of the Risk Map in control was illustrated. Second, a potential design application was proposed that demonstrates the framework as a guide to refine physical designs to satisfy specific demands in capability or safety values. This application is the reverse process to the main methodology of this thesis, in which the dynamic model and its constraints were projected into capability and safety. Finally, the VSN and Risk Maps were computed for a realistic hardware robot, which will enable future hardware demonstrations.

Collectively, the work presented in this thesis is a collection of tools that can be widely applicable and provides a new perspective on the understanding and analysis of dynamic systems. These tools have enabled a quantified understanding of the capability as well as the risk of carrying out highly dynamic movements. Furthermore, this thesis has led to a standardized risk quantity that has a consistent interpretation across different systems. This risk quantity has the potential to guide the design of better robot controllers. Overall, the methodology developed in this thesis builds a foundation towards more capable robots.
5.2 Limitations

This thesis formulates the basic concepts of estimating the full capability of a robot and quantifying the risk of failure within that capability. In its current form, the developed methods rely on exhaustive computation and thus suffer from the curse of dimensionality. Additionally, the reliance of the method on discretization necessarily results in omissions, and current estimates of the viable space are a subset of the true space. Future optimization will determine how to best adapt discretization methods to retain details while considering computational effort. Separately, much of the computed VSN and Risk maps are dependent on user choices. While this thesis provides working examples, application to other systems may require additional domain knowledge.

5.3 Future work

Following the results of this thesis, several new research directions can be pursued. First, the methodology has provided the VSN and Risk Maps for a single-legged hopper. These results can be further studied for adaptation to physical hardware. In addition, the results may be able to inform the control of more complex legged systems such as humanoids. Since the single-legged hopper is a template model for locomotion, future work would focus on creating maps between the simple robot and a complex one. While current methodologies focus on individual behaviors for robots, the methodologies presented in this thesis specifically target complex tasks. The methodology can be similarly applied to other robots in order to enable more complex behaviors. Second, the safety values generated through this framework can be used for the development of new risk-aware controllers. Third, the current workflow to estimate capability and quantify risk has potential for further optimization with more sophisticated modeling. Last, the data produced in the pogo-bot example can initiate a more formal examination of risk concepts in hybrid systems that can be more generally applicable to legged robots.
Outlook  It is striking to see the differences between today’s legged machines and humans. Modern robots can only execute tasks that are comparable in complexity to walking or running. However, athletes inspire the imagination: humans can easily perform complex footwork, far beyond the reach of the most sophisticated robots, all while being keenly aware of the risk of failure. As humanity expects robots to perform increasingly complex tasks, the development of robots must move beyond the ad hoc creation of controllers for specific behaviors. This thesis presents one possible framework that enables robots to accomplish more, through the quantification of capability and risk. We expect further studies in this broad topic which will allow future robots to approach the capabilities of humans, and possibility exceed them.
Bibliography


