2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Risk-constrained Motion Planning for Robot Locomotion: Formulation and Running Robot Demonstration

Jacob Hackett[1], Wei Gao[1], Monica Daley[2], Jonathan Clark[1] and Christian Hubicki[1]

***Abstract*— Robots encounter many risks that threaten the success of practical locomotion tasks. Legs break, electrical components overheat, and feet can unexpectedly slip. When all risks cannot be completely avoided, how does a robot decide its best action? We present a method for planning robot motions by reasoning about risk-of-failure probabilities instead of applying cost-penalty functions or inflexible path constraints. This work develops a risk-constrained formulation that can be straightforwardly included in existing motion planning optimizations. The risk constraints scale tractably with many risk sources, and in some cases, only add linear constraints to the optimization problem and are therefore compatible with model-predictive control techniques. We present a toy "Puck World" proof-of-concept example and a practical implementation on a planar monopod robot that runs at 3.2 m/s when permitted to take high-risk maneuvers. We believe this risk approach can be used to optimize robot behaviors under numerous conflicting task pressures and model risk-conscious behaviors in animals.**

I. INTRODUCTION

The physical world is rife with risks that our robots must mitigate. Motors can overheat, load-bearing links can break and critical contact points can slip away (Fig. 1A). In the best cases, these risks can be avoided simultaneously by strictly obeying predefined safety limits. However, when the task becomes sufficiently demanding or risks become unavoidable, how do we tractably plan robot motions to allow for some manageable risk while still gaining maximum benefit? This work presents a probabilistic model for applying risk constraints that can be straightforwardly incorporated into prevalent motion planning methods. As a robotics proof-of-concept, we demonstrate risk-constrained motion planning in a high-speed hopping robot [1] that actively balances the tradeoff between risky leg forces and running at high speeds.

Trajectory optimization is at the core of many robot control methods; legged locomotion in particular. Researchers commonly use direct collocation methods [2] to transcribe trajectory optimization problems into nonlinear programs (NLP's) with finite decision variables to optimize. Physical limits [3] and task constraints [4] are typically encoded as inflexible equality or inequality constraints for the optimizer to solve. Commonly, energy economy is chosen as an objective function to plan efficient walking [5]–[9] and running [10], [11] gaits. These motions are largely generated such that they are agnostic to stability, but frequently serve as an operating point for stabilizing feedback controllers.

[1]These authors are with the Department of Mechanical Engineering, Florida State University, FAMU-FSU College of Engineering, Tallahassee, FL 32310, USA. Corresponding author: hubicki@eng.famu.fsu.edu

[2]Monica Daley is with the Ecology & Evolutionary Biology Department, University of California Irvine, Irvine, CA 92697, USA

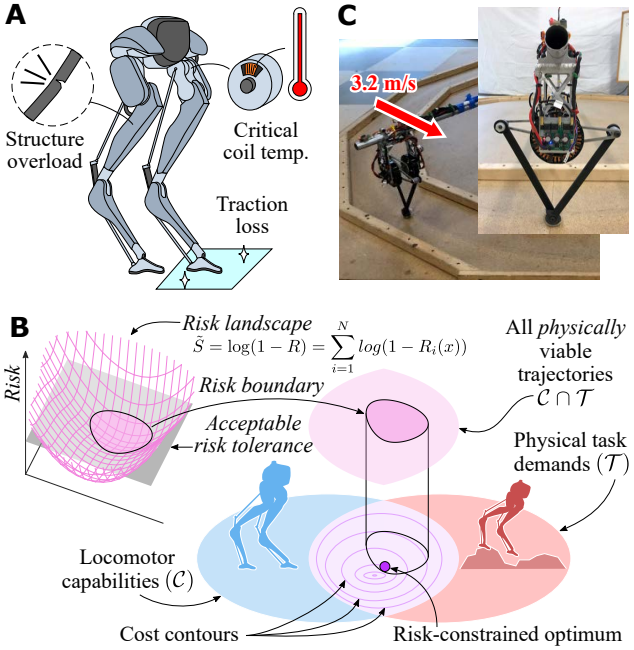

Fig. 1. **A.** Robots are subject to many potential modes of failure. This work presents an approach to planning robot motions while balancing multiple sources of probabilistic risk. **B.** Concept illustration of risk constraints, where a predetermined risk tolerance defines the space of admissible locomotion strategies. **C.** Implementation of risk-constrained motion planning with 3.2 m/s running on a testbed monopod hopper.

However, the optimized motions themselves can be stable or stabilizable in some form. By encoding a trajectory's sensitivity to perturbations into the motion planner, motions can be designed to be open-loop stable [12], stabilizable [13], robust to disturbances [14], reachable [15], and insensitive to sensory delays [16]. Further, stabilization methods use concepts of safety [17] and barriers [18] to ensure the planned motion can be tracked. Generally, these planning and control methods determine whether the robot will attenuate state perturbations, but not necessarily the likelihood it will fail at its task.

Probability, in contrast, can serve as a mathematically viable means of quantifying the likelihood of task success in practical environments with hazards. One example concept is metastability [19], a metric that takes the view that failure can be inevitable in systems with random influences. Instead of categorizing inevitable failure as "unstable," metastability quantifies the likely time until failure of a controlled system. However, this likely time-to-failure metric requires significant computation time to evaluate, making the metric currently computationally intractable for real-time or near-

real-time optimization. To achieve real-time optimization, "chance constrained" convex optimization methods have been developed for systems with polynomial dynamics subject to stochastic disturbances [20].

We present a risk-constrained motion planning method for designing optimal trajectories in the presence of probabilistic chances of failure. As illustrated in Fig. 1B, the approach finds the lowest cost solution for achieving the task while respecting constraints on acceptable risk tolerance. Further, the presented slack-variable formulation enables the optimization to accommodate many sources of risk while remaining computationally tractable. We further highlight situations where such risk constraints are compatible with fast convex optimizations for model-predictive control (MPC). Section II introduces and explains the formulation and Section III attempts to clearly illustrate how the method works on a simple toy problem called "PuckWorld." Section IV describes the proof-of-concept experiments with a running robot, shown in Fig. 1C, and Section V explores the implications of the approach for robotic applications.

II. APPROACH

We begin with a typical trajectory optimization formulation in continuous time. We introduce the state vector, $q(t) \in \mathbb{R}^{N_q}$, and control input vector, $u(t) \in \mathbb{R}^{N_u}$, both of which are dependent on time variable, t. N_q and N_u represent the numbers of states and control inputs respectively. We take a direct approach to trajectory optimization which discretizes the continuous-time problem into N discrete time points called nodes (sometimes called "knots"). Taking a direct collocation approach [21], we discretize both system states and inputs into a set of decision variables, q and u containing q_k and u_k respectively, where $k \in 1, 2, 3, \ldots, N$. The discretized time points are evenly spaced between $0 = t_1 < t_2 < \cdots < t_N = T$ where T is the total duration of the trajectory.

We then define the system dynamics and constraints. The system states evolve as per first-order system dynamics,

$$\dot{q}(t) = f(t, q(t), u(t)), \tag{1}$$

where the dynamics function $f(t, q, u)$ is vector-valued in $\mathbb{R}^{N_q}$. We now define the scalar-valued objective function to be minimized, $J(t, q, u)$, and the vector-valued path constraints function, $h(t, q, u)$. By approximating the integral of the dynamics via explicit Euler methods with a time step $\Delta T = T/(N-1)$, the resulting optimization problem takes the form

$$\begin{aligned} \min_{q,u} \quad & J(t, q, u) \\ \text{s.t.} \quad & q_{k+1} - q_k - f(t_k, q_k, u_k)\Delta T = 0 \\ & h(t, q, u) \leq 0, \ \ \forall k \in 1, 2, \ldots, N. \end{aligned} \tag{2}$$

This work takes this well-established trajectory optimization approach [2] and adds an additional set of constraints that ensure the solution does not exceed a predefined tolerance of acceptable risk of failure.

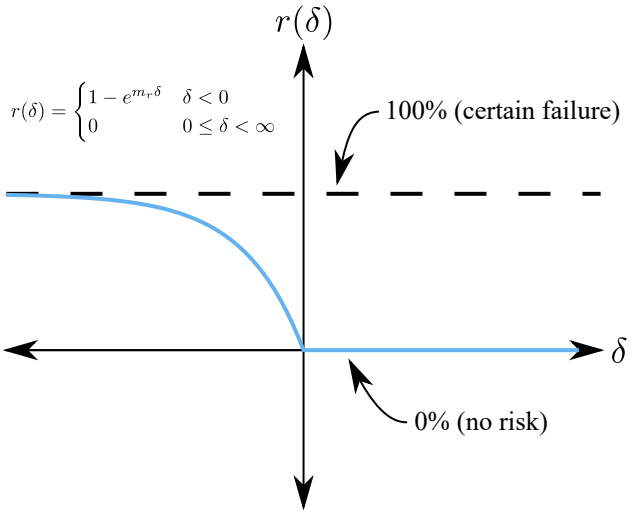

Fig. 2. An example failure probability function, $r(\delta)$, as a function of a risk source, δ. Positive δ has zero failure risk, but as δ becomes increasingly negative, failure probability approaches 100%. Any time the system spends in situations where $\delta \geq 0$ incur no risk, but time spent in situations where $\delta < 0$ will accrue risk of failure and count toward the acceptable risk tolerance for the task, R_a. Note: δ can be situationally dependent (i.e. varying with time, state, and input).

A. Definitions

The presented formulation models risk as an overall probability of task failure as aggregated from many instantaneous probabilities of failure. Repeatedly overheating a robot motor during a task is more likely to result in a burnout occurring than a single bout of overheating. Risk constrained optimization requires three *a priori* definitions: (1) **risk sources**, δ, (2) **failure probability functions**, $r(\delta)$, and (3) an **acceptable risk tolerance**, R_a.

We first introduce a potential cause of failure or **risk source**, $\delta(t, q(t), u(t))$, which is a scalar-valued function that can be dependent on the system time, state, and control input (which we shorten to δ for brevity). The risk source could represent temperature for a motor that could burn out, or euclidean distance from a source of danger, or the magnitude of force that could be injurious to the robot. A risk source, δ, is used by the failure probability function to compute the likelihood of failure at a given instant during the trajectory. Multiple sources of risk are permissible, in which case we enumerate each individual risk source as δ_j.

Secondly, we introduce **failure probability functions**, $r(\delta)$, which are dependent on our risk source, δ. The failure probability function defines the likelihood that the task will fail if the risk source, δ, remains at a given value for some small time duration, ΔT. Fig. 2 illustrates an example function $r(\delta)$, where a positive δ has zero probability of failure and is thus completely safe. Whereas, once δ becomes negative, probability of failure rapidly increases toward one, which is extremely dangerous. Unlike formal probability distribution functions, $\int_{-\infty}^{\infty} r(\delta) d\delta$ need not equal one. We will later demonstrate a convenient way to design failure probability functions in a piecewise fashion using logarithmic functions.

Thirdly, we introduce an **acceptable risk tolerance**, R_a. The trajectory will be deemed feasible if and only if the probability of failure of the entire trajectory, R, is less than or equal to R_a. Raising R_a above zero allows the trajectory to accept some risk if it provides improved performance (as defined by the objective function, J). R must account for all risk sources at every time point during the trajectory. For instance, if two risk sources δ_1 and δ_2 happen to each yield coin-flip failure probabilities, *i.e.* $r_1(\delta_1) = 0.5$ and $r_2(\delta_2) = 0.5$, and the system remains in this situation for three time durations of ΔT, then the task will fail if it loses any one of the six coin flips. In this example, $R = 1 - S$, where S is the probability of surviving the entire trajectory, and that survival requires winning both of two coin flips and repeating that success three times ($R = 1 - S = 1 - \prod_{k=1}^{3}\prod_{j=1}^{2}(1 - r(\delta)) = 1 - 0.5^6 = 98.4\%$ of the time).

B. Constraint Formulation

For the task to succeed, it must survive every random draw from all defined failure probability functions, $r_j(\delta_j)$, and at every time window during the trajectory, $[t_k \;\; t_{k+1}]$. As such, the total probability of the motion surviving the entire task, $S = 1 - R$, is the product of the survival probabilities $S_{jk} = 1 - r_j(\delta_j(t_k, q_k, u_k))$ for all N_r risk sources ($j \in \{1, 2, \ldots, N_r\}$) and time segments ($k \in \{1, 2, \ldots, N-1\}$), or

$$\begin{aligned} S &= \prod_{\forall j}\prod_{\forall k}\left[1 - r_j(\delta_j(t_k, q_k, u_k))\right] \\ &= \prod_{\forall j}\prod_{\forall k} S_{jk}. \end{aligned} \tag{3}$$

With all of the above definitions, one could trivially define the risk constraint as

$$S = \prod_{\forall j}\prod_{\forall k} S_{jk} \geq 1 - R_a, \tag{4}$$

meaning that the total survival probability, S, must be greater than $1 - R_a$, where R_a is the acceptable risk tolerance. However, this constraint as listed is highly nonlinear and thus can be difficult to solve. So we further develop the constraint formulation so it can be solved more quickly.

First, we take the logarithm of the constraint in Eq. 4 and convert the product to a sum via log identities,

$$\begin{aligned} \log\prod_{\forall j}\prod_{\forall k} S_{jk} &\geq \log\left(1 - R_a\right) \\ \sum_{\forall j}\sum_{\forall k}\log S_{jk} &\geq \log\left(1 - R_a\right). \end{aligned} \tag{5}$$

We next introduce new decision variables (or slack variables) to the optimization problem called $\tilde{S}_{jk}$. Specifically, we define $\tilde{S}_{jk} = \log S_{jk} \;\; \forall j, k$ and replace them in the risk constraint. This simplifies the risk constraint to

$$\sum_{\forall j,k}\tilde{S}_{jk} \geq \log\left(1 - R_a\right), \tag{6}$$

which is linear in decision variables $\tilde{S}_{jk}$ and right side, $\log\left(1 - R_a\right)$, is a constant. Theoretically, this renders our probabilistic constraint on total risk a *linear constraint*, and therefore far simpler for solvers to satisfy. Now, we must link our new $\tilde{S}_{jk}$ log-survival decision variables to our failure probability functions, $r_j(\delta_j)$. It is at this point we realize that it can be much more straightforward to define our $r_j(\delta_j)$ functions in this log-survival space.

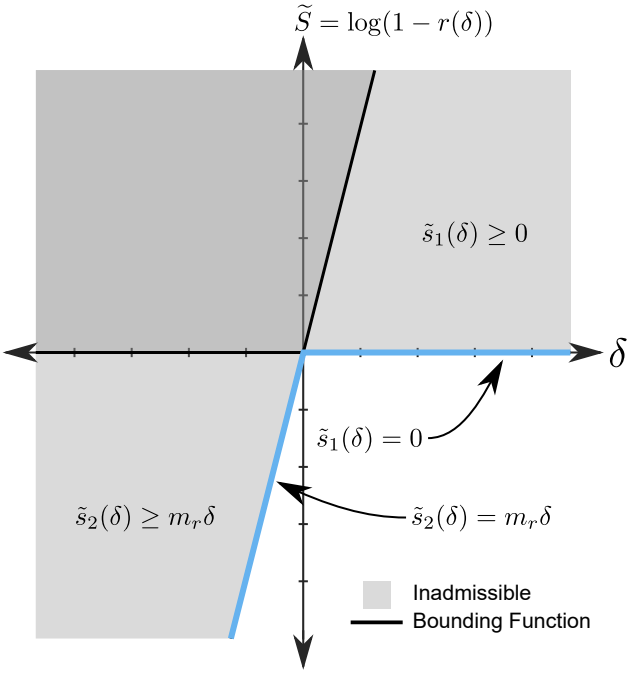

Fig. 3. The failure probability function, $r(\delta)$, from Fig. 2 constructed using linear upper bounds in log-survival space, $\tilde{S} = \log(1 - r(\delta))$. These upper bound functions, $\tilde{s}_i(\delta)$, are used to design the shape of failure probability functions to reflect possible dangers from a risk source, δ.

We begin this linking by defining bounding functions, $\tilde{s}_{ij}(\delta)$, that place upper bounds on the $\tilde{S}_{jk}$. The i subscript indexes into each of the bounding functions for a particular risk source δ_j. The bounding constraint takes the form

$$\tilde{S}_{jk} \leq \tilde{s}_{ij}(\delta) \;\; \forall i, j. \tag{7}$$

In effect, these bounding functions, $\tilde{s}_{ij}(\delta)$, are what we can use to define the failure probability function. As an example, refer to the piecewise failure probability function in Fig. 2. That function is actually piecewise linear after transforming into a log-survival, $\tilde{S}$, by taking $\log(1 - r(\delta))$. Consequently, we can partly define this piecewise curve by applying overlapping linear bounds on $\tilde{S}_{jk}$. Fig. 3 shows what the failure probability function from Fig. 2 looks like as bounded by two linear functions, $\tilde{s}_1(\delta) = 0$ and $\tilde{s}_2(\delta) = m_r\delta$.

The bounding functions themselves only serve to limit the upper bound on slack variables, $\tilde{S}_{jk}$. The need to prevent $\tilde{S}_{jk}$ from being too low is satisfied by the risk constraint in Eq. 6, which incentivizes large $\tilde{S}_{jk}$ values wherever possible. The resulting final risk-constrained optimization formulation

simplifies to the following program:

$$\begin{aligned} \min_{\tilde{S},q,u} \quad & J(t,q,u) \\ \text{s.t.} \quad & q_{k+1} - q_k - f(t_k,q_k,u_k)\Delta T = 0 \\ & h(t,q,u) \leq 0 \\ & \sum_{\forall j,k} \tilde{S}_{jk} \geq \log\left(1 - R_a\right) \\ & \tilde{S}_{jk} \leq \tilde{s}_{ij}(\delta_j(t_k,q_k,u_k)), \quad \forall i,j,k. \end{aligned} \tag{8}$$

We further note that when the functions $\tilde{s}_{ij}(\delta_j)$ are linear in δ_j and $\delta_j(t_k,q_k,u_k)$ is linear in t_k, q_k, and u_k, the risk-constrained formulation adds only linear constraints to the program. This makes the formulation highly scalable to many sources of risk.

III. TOY EXAMPLE

As a clear example of how risk constraints can shape a motion plan, we use a simple toy problem called "PuckWorld." A nod to the ubiquitous "GridWorld" problem in reinforcement learning tutorials [22] where an agent must learn to navigate obstructions in discretized 2D environment, PuckWorld simulates a mass navigating a 2D continuous environment in Cartesian space [23]. It is substantially similar to the 1D "double integrator" or "Block-Move" example [21], except the 2D motion and low ground friction gives the aesthetic feel of a hockey puck being maneuvered on ice.

A. Example Problem: PuckWorld

We begin by defining the dynamics of our PuckWorld problem. The puck has a point mass, $m = 1$ kg, and glides on a 2D surface under low linear friction, $c = 0.5$ Ns/m. The state vector $q(t) = [X(t)\ \dot{X}(t)\ Y(t)\ \dot{Y}(t)]^T$ where X and Y are the positions in Cartesian space and $u(t) = [F_X(t)\ F_Y(t)]^T$ are time-varying control input forces applied to the puck. We define the system dynamics, $f(t,q,u)$, as

$$\dot{q} = \begin{bmatrix} \dot{X}(t) \\ (F_X(t) - c\dot{X}(t))/m \\ \dot{Y}(t) \\ (F_Y(t) - c\dot{Y}(t))/m \end{bmatrix}. \tag{9}$$

The puck is tasked to start at position $(X,Y) = (-5\text{ m}, 0\text{ m})$ and end at $(X,Y) = (5\text{ m}, 0\text{ m})$, with velocities beginning and ending at rest. We constrain the time duration of the motion to $T = 10$ s and have the optimization minimize an efficiency-based objective function,

$$J(t,q,u) = \int_0^T F_X^2 + F_Y^2 dt. \tag{10}$$

In effect, the puck must travel to a distance 10 m away in 10 s with minimum effort. Without further complications, the optimal solution would look nearly identical to many publications which have previously solved a similar problem [21].

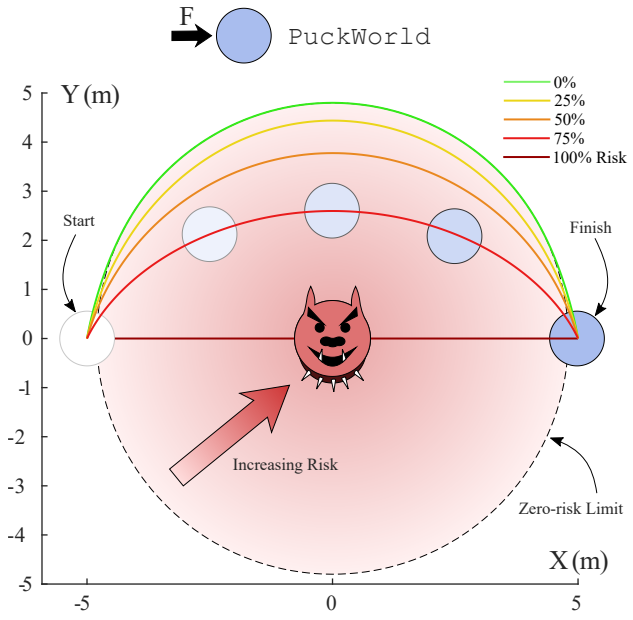

Fig. 4. Risk-constrained motion planning in a simple PuckWorld example, where a sliding puck is commanded to efficiently cross the field. However, when inside a circular danger zone, the puck faces repeated random chances of failing, a probability which increases as the puck approaches the center. When commanded to tolerate zero risk, the puck avoids the danger entirely. However, increasing the acceptable risk tolerance, R_a, results in the puck crossing closer to the dangerous circle center.

B. Example Risk: Proximity

To demonstrate the risk-constrained formulation, we introduce a risk source to complicate the solution. We place a proximity risk at the position between the start and the goal, $(X,Y) = (0\text{ m}, 0\text{ m})$, which threatens failure if you venture too close (inside a distance of $L_r = 4.8$ m). Imagine a guard dog tethered to a post that might decide to attack as you approach. The puck now faces a dilemma. It can be maximally safe by staying outside the danger zone completely or it can cut corners a bit to save some energy. This problem demonstrates by simply changing the acceptable risk tolerance, R_a, **the risk-constrained motion planner will generate an efficient trajectory while only being as risky as the user specifies**.

First, we define our sole **risk source**, $\delta(t,q,u)$, as being a function of the Euclidean distance between the puck and the danger source at $(X,Y) = (0\text{ m}, 0\text{ m})$, specifically,

$$\delta(t,q(t),u(t)) = \sqrt{X(t)^2 + Y(t)^2} - L_r. \tag{11}$$

By this definition, once the puck approaches the danger source closer than L_r, then δ becomes negative. Next we define our **failure probability function**, $r(\delta)$, for our risk source. This function is effectively defined by bounding functions in the log-survival space, $\tilde{s}_i(\delta)$. We will choose bounding functions of the same form as in Fig. 3 because, like in this proximity risk example, it ascribes a negative δ as incurring an increasing probability of failure. Thus, we

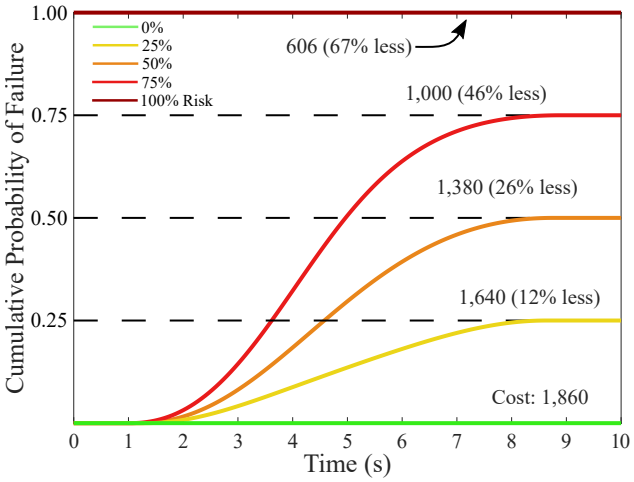

Fig. 5. Cumulative probability of failure for PuckWorld trajectories, showing how risk accrues with time spent in risky situations. However, this increased risk comes with a benefit, increased efficiency of the trajectory, which is listed above each probability line.

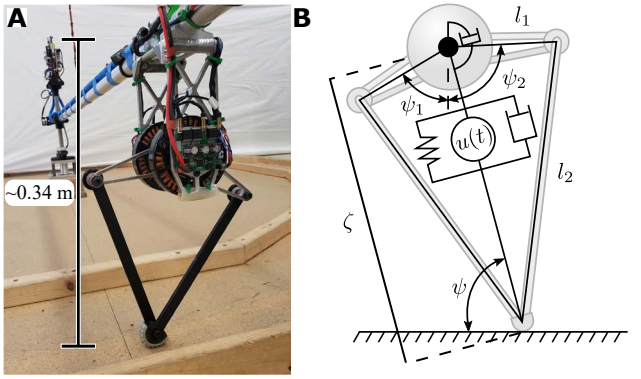

Fig. 6. **A.** An image of the hopper which is a modified leg from the Minitaur quadrupedal robot [26] **B.** An illustration of the math model of the hopper. Building atop a SLIP model, this model includes parallel actuation and damping with the left most element representing a spring, the middle element an actuator, and the right most element a linear damper. We also model a torsional damping force between the hip and attached boom.

define our bounding functions as

$$\begin{aligned} \tilde{s}_1(\delta) &= 0 \\ \tilde{s}_2(\delta) &= m_r \delta, \end{aligned} \tag{12}$$

where $m_r = 0.01$ models the level of danger incurred as the puck continues to cross the $\delta = 0$ threshold.

Finally, we set our **acceptable risk tolerance**, R_a for the motion planning problem. In this case, we generate solutions for R_a = 0, 25, 50, 75, and 100% to show how the strategy changes with increasing risk. Theoretically, an acceptable risk of 0% should yield the same solution as a hard inequality constraint on proximity, and 100% should behave as though the proximity danger doesn't exist.

Constraints were formulated in the manner shown in Eq. 8 using $N = 150$ nodes to discretize the problem into decision variables. The objectives, constraints, and their derivatives w.r.t. all decision variables for the NLP were written as symbolically generated MATLAB functions using the optimization parsing package, COALESCE [24]. The problem was solved using IPOPT [25] using a straight-line path and zero force as an initial guess, and primal and dual tolerances were set to 10^{-9} and 10^{-6} respectively. The NLP's were each solved with a computation time under one second.

C. PuckWorld Results

The Cartesian trajectories for the puck are plotted in Fig. 4. As predicted, when the acceptable risk tolerance was R_a= 0%, the puck dutifully avoided the danger zone completely, but stayed as close to the limit as possible as not to require more effort than necessary (taking a longer path requires higher speed and, thus, more force as tabulated by the objective). As R_a was increased, the puck began to risk venturing inside the danger circle and increasingly closer to the center, culminating in the R_a = 100% trajectory completely ignoring the danger's existence.

We further show how these solutions accrue risk of failure throughout the motion in Fig. 5. As soon as the puck crosses the zero-risk limit, the cumulative probabilities begin to aggregate, tallying up the likelihood it would fail a particular dice roll, until it finally exits the circle near the end of the trajectory. The plot shows the tradeoff associated with permitting more risk of failure, which is increased efficiency of the trajectory. Allowing for 25, 50, 75, and 100% chance of failure reduced the objective function value by, 12%, 26%, 46%, and 67% respectively.

IV. ROBOT EXPERIMENTS

As a practical test of risk-constrained motion planning, we optimized the behavior of a robot with significant sources of mechanical/electrical risk: a high-speed single-legged planar hopping robot, represented in Fig. 6.

A. Legged Simulation Model

The testbed hopper, as seen in Fig. 6, is based on a Spring-Loaded Inverted Pendulum (SLIP)-based model, named the Force-driven and Damped SLIP (FD-SLIP) model. The FD-SLIP dynamics are used to simulate the dynamics of a single-legged hopper. A brief explanation of the model is presented here, but for a complete description see [1] for details.

The model[1] was a modification of the classic SLIP model [27], with added dampers and actuators. Each step (*i.e.* gait cycle) consists of a stance phase, where the foot is pinned down to the ground, and a flight phase, where the body follows a ballistic trajectory. The dynamics of the model are described by the following equations of motion. The stance phase dynamics in polar coordinates are defined as,

$$\begin{aligned} \ddot{\zeta} &= \zeta\dot{\psi}^2 - g\sin\psi - \frac{k_o}{m}(\zeta - l_o) - \frac{b_l}{m}\dot{\zeta} + \frac{u(t)}{m}, \\ \ddot{\psi} &= -\frac{2\dot{\zeta}\dot{\psi}}{\zeta} - \frac{g\cos\psi}{\zeta} - \frac{b_t}{m\zeta^2}\dot{\psi}, \end{aligned} \tag{13}$$

where ζ is the leg length, ψ is the leg angle, $u(t)$ is the time-varying control input from the optimized trajectory (See

[1]Model parameters: $g = 9.8$ m/s, $m = 1.2$ kg, $l_o = 0.17$ m, $k_o = 1700$ N/m, $b_l = 13$ Ns/m, $b_t = 0.081$ Nms/rad

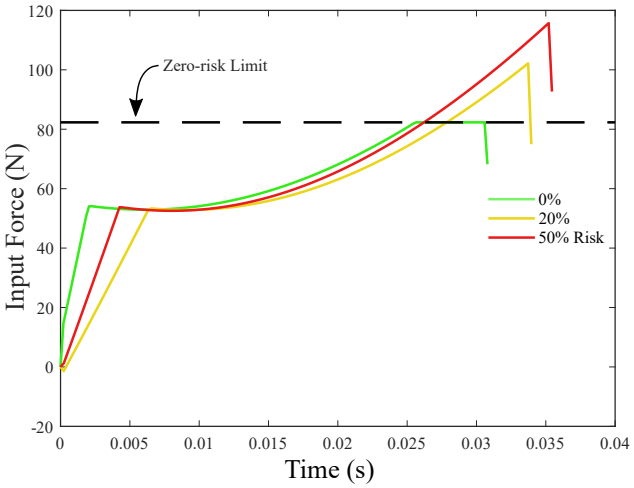

Fig. 7. Optimal force input profiles, $u(t)$, generated for 0%, 20%, and 50% risk of failure per step. While the 0% risk solution dutifully avoids breaching the zero-risk limit, the higher-risk solutions are reshaped to achieve higher-speed running while not exceeding 20% or 50% risk of failure respectively.

Fig. 7 for examples), g is gravitational acceleration, m is the mass at the hip, l_o is the nominal leg rest length, k_o is the nominal leg stiffness, b_l is the leg linear damping coefficient, and b_t is the hip torsional damping coefficient. The flight phase dynamics in Cartesian coordinates are defined as,

$$\begin{aligned} \ddot{x} &= 0, \\ \ddot{z} &= -g. \end{aligned} \tag{14}$$

where x is the horizontal position and z is the vertical position.

The equations of motion are then transferred to constraints in the optimization setup. The control input only takes effect in the stance phase, so only the stance phase needs to be optimized. As a result, the flight phase dynamics can be analytically solved and act as boundary constraints for the stance phase. Meanwhile, the stance phase dynamics are converted to trajectory constraints using trapezoidal integration. Additionally, a simple linear motor model is also incorporated to reflect a realistic power limit. More detailed definitions of physical hopper constraints can be found in [1].

B. Running Motion Planning

Described previously, we applied risk formulations as constraints in a trajectory optimization problem simulating a SLIP-like runner at high speeds using the same software package as described in Section III. The optimization problem passes through a target apex condition, height and speed $[z^{apex}, \dot{x}^{apex}]$, as well as a desired touchdown angle $[\psi^{TD}]$ for the running robot while minimizing the control input. The optimization problem is highly similar to that described in [1], except we add a source of risk to high input force, $u(t)$. This is a simple model of motor current limits, which if exceeded consistently can lead to electronic failures. We model the failure probability function using the form

$$\begin{aligned} \tilde{s}_1(\delta) &= m_r \delta, \\ \tilde{s}_2(\delta) &= 0, \end{aligned} \tag{15}$$

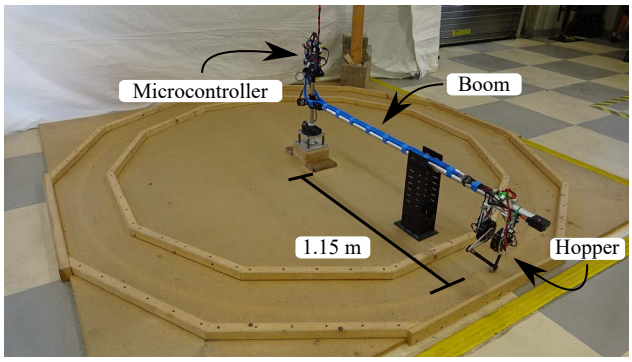

Fig. 8. The experimental setup for the hopping robot, where the robot is mounted on the end of a rotating boom, constraining the robot to planar locomotion. The boom allows for forward locomotion (via sweep angle), height change (via azimuth angle), but not pitching (via roll angle).

where in the first function, $m_r = 6.6 \times 10^{-4}$ is a scalar that sets the severity of the risk and δ the risk source. The second risk function acts as a constraint which restricts the work-space prohibiting the probability of success to be over 100%. The sole risk source, δ, is defined as,

$$\delta = u_r - u(t), \tag{16}$$

where u_r is an input force limit and $u(t)$ is the control input from the trajectory optimization. This limit begins to incur risk when seven body weights of the robot is exceeded. Eq. (16) follows the convention from Eq. (12) where negative δ is associated with accruing risk.

The objective of the optimization problem is to minimize the following function

$$J(u(t)) = \int_0^T u(t)^2 dt \tag{17}$$

and achieve a desired apex height, apex velocity and touchdown angle $[z^{apex}, \dot{x}^{apex}, \psi^{TD}]$ while constrained to a acceptable risk tolerance, R_a.

We chose this relatively simple risk source and task for the monopod hopper proof-of-concept for the primary reason to expediently evaluate and validate the risk-formulation. By keeping the risk source simple this permitted intuition to possible solutions. Therefore, by restricting the input force profile with the risk constraint, it is expected to also restrict the maximum velocity the robot can achieve. To test this hypothesis, a range of desired apex velocities were assessed. We swept desired speeds between 1.8 $\mathrm{m/s}$ and 4.0 $\mathrm{m/s}$ in increments of 0.1 for the risk tolerances of 0%, 20%, and 50%. z^{apex} and ψ^{TD} were kept at a constant value of 0.18 m and 77°, respectively. z^{apex} and ψ^{TD} were chosen based on the findings of previous experiments on the hardware [1]. It was found that an apex height of 0.18 m resulted in the most stable gaits. Through iterative testing on the physical platform it was found that a ψ^{TD} of 77° most often produced a stable gait. The stability criterion is defined simply as whether the robot falls after three laps, and thus we did not compute exponential stability in the form of Floquet multipliers.

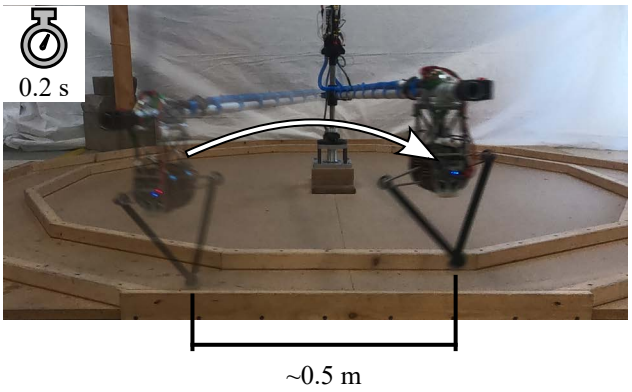

Fig. 9. Still frame images of the monopod hopper running at 2.5 $\mathrm{m/s}$. Three running gaits were optimized using risk-constrained motion planning and implemented on the planar hopper as shown.

We expected to see that an acceptable risk tolerance (R_a) of 0% would act as a hard constraint on the input force profile at the zero-risk limit of seven body weights. As we raise R_a to 20% and 50%, we expect to see the input profile breach the seven body weight limit to achieve higher speeds. As expected, this is the trend we observed in Fig. 7. Increasing the risk tolerance resulted in optimal solutions with the predicted average step velocity of 3.3 $\mathrm{m/s}$, 3.5 $\mathrm{m/s}$, and 3.7 $\mathrm{m/s}$ for risk tolerances of 0%, 20%, and 50%, respectively.

C. Hardware Setup

The same single-legged hopper, as introduced in [1] to instantiate the SLIP-like running behaviors in the sagittal plane, was used in this work for experimental demonstration. It uses a modified version of the single leg from the Minitaur quadrupedal platform [26], where the controllable transparent transmission between the hip and the toe can be treated as a combination of a virtual spring, damper and actuator, as illustrated in Figure 1C. The recent modification to the leg design, as mentioned in [1], has made the behaviors of the hopper well-described by the SLIP-based model.

The hopper has two Hall Effect based absolute encoders for the two hip motors to realize closed-loop control of the leg length and angle based on the leg kinematics. Additionally, the hopper has two Accu-Coder model 15s encoders located at the base to monitor the orientation of the boom and thus the global position of the hip attached at the end of the boom, using a boom length of 1.15 m. All the data are recorded on a Teensy 3.6 microcontroller at 500 Hz for post-processing. The setup of these components is depicted in Fig. 8.

To implement the optimized controller trajectory from the risk-constrained approach onto the physical platform, the generated array of the force trajectory is uploaded to the main controller board at the base of the hopper through USB-TTL serial communication. The controller code on the main board later calls this stored array at 1000 Hz to apply the correct controller input during the stance phase while the hopper is operating.

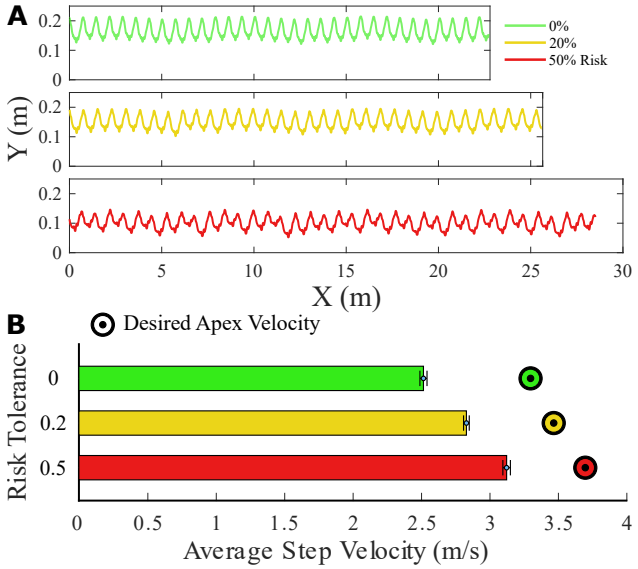

Fig. 10. **A.** Cartesian trajectories for the monopod hopper as it runs for approximately 10 seconds. Note that the higher risk trajectories move further in that plotted time window, and are therefore faster. **B.** Average step velocity from experimental results for max allowable risks of 0%, 20%, and 50% per step (error bars $\pm 1\sigma$). While the hopper was unable to achieve the velocity predicted by the simulation model, the resulting trend clearly shows that allowing a higher risk tolerance results in a faster gait as predicted.

D. Experimental Results

When executed on the hopper, the optimized gaits ran in a self-stable fashion for several laps around the circular track, as depicted in Fig. 9. In the same 10-second time window, the higher-risk trajectories traveled further than their risk-averse counterparts as seen in Fig. 10A. Fig. 10B confirms the fact that average hopper velocity did in fact increase with higher risk tolerance as predicted, effectively trading off safety for speed. We note that the actual measured speed did not achieve the target speed and attribute the difference to mechanical modeling discrepancies.

V. DISCUSSION

A. Application

This paper focused on rather straightforward applications of risk models to motion planning with simple risks on states and inputs. However, more complex constraints are possible. Motor curves, for instance, could allow the controller to briefly push outside the rated torque-speed relationship of the actuator. Slipping risk can be modeled as a risk source by quantifying the violation of the estimated friction cone.

B. Possible Extensions

There are numerous possible extensions of the constrained formulation that could broaden its use. Currently the time duration for these optimization problems is fixed, but many robot optimizations make use of variable time horizons. A variable-duration extension of this approach would be a clear improvement. Further, each individual random draw from the failure probability functions is assumed to be independent. Finding ways to couple potential failures early in the trajectory with effects later in the trajectory could lead

to greater flexibility in modeling risks. For instance, instead of a failure being terminal for the task, perhaps it hobbles the robot and forces it to plan new strategies.

C. Learning Risk

One key advantage to using risk to represent undesirable behaviors is that probabilities of failure are measurable quantities. While an abstract penalty term could be added to the objective function to discourage undesired features in motion plans, it is unclear how to automatically design these penalty terms from physical measurements of the system. Failure probability functions, however, could be fitted to data collected from a robot's experience. If a robot were to shut down when the motor heated up, the failure probability function could be updated (*i.e.* learned) to reflect that new source of risk and subsequently plan motions with newfound risk-awareness.

VI. CONCLUSIONS

We presented a risk-constrained motion planning method for finding optimal robot trajectories in the presence of failure probabilities. By introducing slack variables to the optimization (bounded by curves in log-survival space), we created a formulation with easy-to-solve (even linear) constraints, allowing for fast solving times (under one second in our presented applications). We validated the method on a toy problem called PuckWorld where changing the acceptable risk tolerance yielded a smooth tradeoff between safety and efficiency. When applied to a testbed monopod hopper, risk-constrained motion planning demonstrated how allowing for some failure risk from large leg forces enabled faster top speeds, up to 3.2 $\mathrm{m/s}$ in our experiments. Future work will extend the formulation to demonstrate applicability to the many sources of risk in our complicated world.

ACKNOWLEDGMENT

This work was supported by the Bruce & Deborah Morrison Undergraduate Research Award and the FAMU-FSU Mechanical Engineering department. The authors also thank Charles Young for technical work on the hopper platform, Olugbenga Moses Anubi for discussions on the concept, Hunter Kramer and Noah Lang for experiment assistance, and Tianze Wang, Adwait Mane, and Milovan Dakic for review of the manuscript.

REFERENCES

[1] W. Gao, C. Young, J. Nicholson, C. Hubicki, and J. Clark, "Fast, versatile, and open-loop stable running behaviors with proprioceptive-only sensing using model-based optimization," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[2] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[3] W. Xi, Y. Yesilevskiy, and C. D. Remy, "Selecting gaits for economical locomotion of legged robots," *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1140–1154, 2016.

[4] C. Hubicki, M. Jones, M. Daley, and J. Hurst, "Do limit cycles matter in the long run? stable orbits and sliding-mass dynamics emerge in task-optimal locomotion," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5113–5120.

[5] I. R. Manchester, U. Mettin, F. Iida, and R. Tedrake, "Stable dynamic walking over uneven terrain," *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 265–279, 2011.

[6] P. A. Bhounsule, J. Cortell, A. Grewal, B. Hendriksen, J. D. Karssen, C. Paul, and A. Ruina, "Low-bandwidth reflex-based control for lower power walking: 65 km on a single battery charge," *The International Journal of Robotics Research*, vol. 33, no. 10, pp. 1305–1321, 2014.

[7] T. L. Brown and J. P. Schmiedeler, "Gait transitions and disturbance response for planar bipeds with reaction wheel actuation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 3393–3398.

[8] A. Hereid, C. M. Hubicki, E. A. Cousineau, and A. D. Ames, "Dynamic humanoid locomotion: A scalable formulation for HZD gait optimization," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 370–387, 2018.

[9] X. Da and J. Grizzle, "Combining trajectory optimization, supervised machine learning, and model structure for mitigating the curse of dimensionality in the control of bipedal robots," *The International Journal of Robotics Research*, vol. 38, no. 9, pp. 1063–1097, 2019.

[10] K. Sreenath, H.-W. Park, I. Poulakakis, and J. W. Grizzle, "Embedding active force control within the compliant hybrid zero dynamics to achieve stable, fast running on MABEL," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 324–345, 2013.

[11] W.-L. Ma, S. Kolathaya, E. R. Ambrose, C. M. Hubicki, and A. D. Ames, "Bipedal robotic running with DURUS-2D: Bridging the gap between theory and experiment," in *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, 2017, pp. 265–274.

[12] K. Mombaur, "Using optimization to create self-stable human-like running," *Robotica*, vol. 27, no. 3, pp. 321–330, 2009.

[13] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1366–1373.

[14] H. Dai and R. Tedrake, "Optimizing robust limit cycles for legged locomotion on unknown terrain," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 1207–1213.

[15] S. Kousik, P. Holmes, and R. Vasudevan, "Safe, aggressive quadrotor flight via reachability-based trajectory design," in *ASME 2019 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection, 2019.

[16] J. Van Why, C. Hubicki, M. Jones, M. Daley, and J. Hurst, "Running into a trap: numerical design of task-optimal preflex behaviors for delayed disturbance responses," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2537–2542.

[17] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[18] X. Xiong and A. D. Ames, "Bipedal hopping: Reduced-order model embedding via optimization-based control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3821–3828.

[19] K. Byl and R. Tedrake, "Metastable walking machines," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 1040–1064, 2009.

[20] A. Jasour and C. Lagoa, "Convex chance constrained model predictive control," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 6204–6209.

[21] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[22] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[23] A. Karpathy. (2015) Puckworld: Deep Q Learning. [Online]. Available: https://cs.stanford.edu/people/karpathy/reinforcejs/puckworld.html

[24] M. S. Jones, "Optimal control of an underactuated bipedal robot," 2014.

[25] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[26] D. J. Blackman, J. V. Nicholson, C. Ordonez, B. D. Miller, and J. E. Clark, "Gait development on Minitaur, a direct drive quadrupedal robot," in *Unmanned Systems Technology XVIII*, vol. 9837. International Society for Optics and Photonics, 2016, p. 98370I.

[27] R. J. Full and D. E. Koditschek, "Templates and anchors: neuromechanical hypotheses of legged locomotion on land," *Journal of experimental biology*, vol. 202, no. 23, pp. 3325–3332, 1999.