Efficient Anytime CLF Reactive Planning System for a Bipedal Robot on Undulating Terrain

Jiunn-Kai Huang and Jessy W. Grizzle

***Abstract*—We propose and experimentally demonstrate a reactive planning system for bipedal robots on unexplored, challenging terrains. The system consists of a low-frequency planning thread (5 Hz) to find an asymptotically optimal path and a high-frequency reactive thread (300 Hz) to accommodate robot deviation. The planning thread includes: a multi-layer local map to compute traversability for the robot on the terrain; an anytime omnidirectional Control Lyapunov Function (CLF) for use with a Rapidly Exploring Random Tree Star (RRT*) that generates a vector field for specifying motion between nodes; a sub-goal finder when the final goal is outside of the current map; and a finite-state machine to handle high-level mission decisions. The system also includes a reactive thread to obviate the non-smooth motions that arise with traditional RRT* algorithms when performing path following. The reactive thread copes with robot deviation while eliminating non-smooth motions via a vector field (defined by a closed-loop feedback policy) that provides real-time control commands to the robot's gait controller as a function of instantaneous robot pose. The system is evaluated on various challenging outdoor terrains and cluttered indoor scenes in both simulation and experiment on Cassie Blue, a bipedal robot with 20 degrees of freedom. All implementations are coded in C++ with the Robot Operating System (ROS) and are available at https://github.com/UMich-BipedLab/CLF_reactive_planning_system.**

I. Introduction

Motion planning as a central component for autonomous navigation has been extensively studied over the last few decades. Algorithms such as RRT*, A*, and their variants focus on finding an (asymptotically) optimal path as computationally efficiently as possible [1]–[9]. The application of these algorithms relies on designing a control policy to track the planned path, resulting in waypoint following or pathway tracking. In turn, the tracking of path segments (between waypoints) leads to non-smooth motion of the actual robot, due to abrupt acceleration or heading changes when transitioning between waypoints/pathways.

This paper seeks to develop a reactive planning system for bipedal robots on unexplored, unmapped, challenging terrains and to provide high-rate (directional) velocity and heading commands to be realized by the robot's low-level feedback-control gait-generation algorithm. For this application, the non-smooth aspects of the planned motions arising from waypoints/pathways transitions are detrimental to stability of the overall system.

Several approaches have been developed to address the non-smooth aspects of paths produced by motion planning, such as reactive motion planning [10]–[18] and feedback motion planning [19]–[21]. Fundamentally, these approaches replace paths to be followed with smooth vector fields whose solutions guide the robot's evolution in its configuration space.

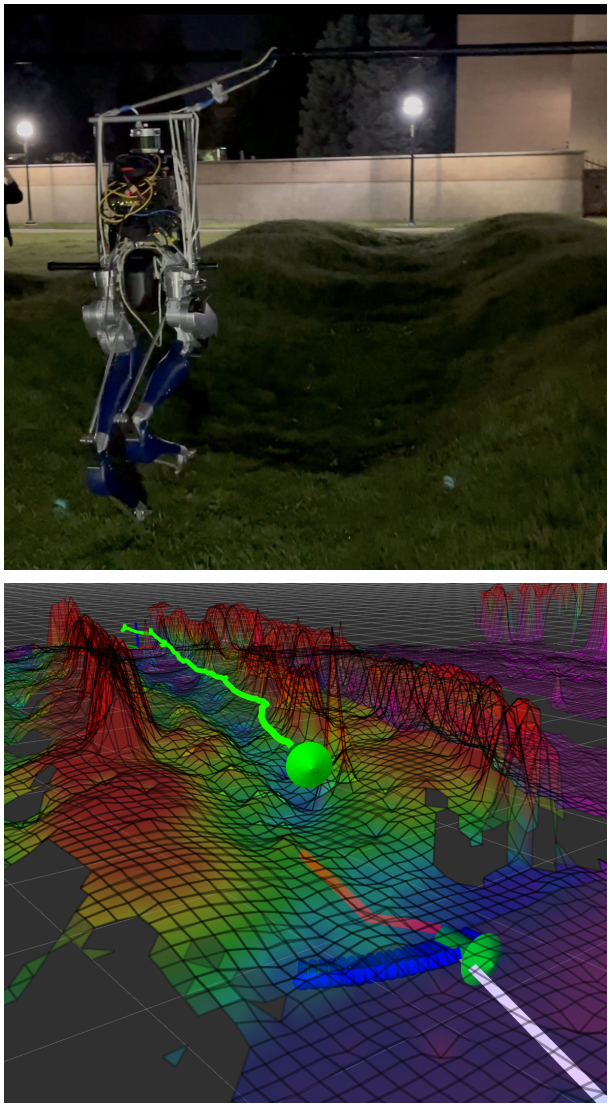


Fig. 1: In this figure Cassie Blue autonomously traverses the Wave Field via the proposed reactive planning system, which consists of a planning thread and a reactive thread. The planning thread involves a multi-layer local map to compute traversability, a sub-goal finder, and an omnidirectional Control Lyapunov Function RRT*. Instead of a common waypoint-following or path-tracking strategy, the reactive thread copes with robot deviation while eliminating non-smooth motions via a vector field (defined by a closed-loop feedback policy) that provides real-time control commands to the robot's gait controller as a function of instantaneous robot pose.

Jiunn-Kai Huang and J. Grizzle, are with the Robotics Institute, University of Michigan, Ann Arbor, MI 48109, USA. {bjhuang, grizzle}@umich.edu.

We are most inspired by the work of [20], [21], which proposes a Control Lyapunov Function (CLF) to realize reactive planning for a non-holonomic differential-drive wheeled robot. While their underlying model is not applicable to a Cassie bipedal robot, their basic concept is applicable. As part of our work, we design an appropriate CLF for robots capable of walking in any direction with any orientation. Moreover, we take into account features specific to bipeds, such as the limited lateral leg motion that renders lateral walking more laborious than sagittal plane walking.

The feedback motion planning algorithm in [20], [21] has not yet been evaluated on hardware. In general, there is a significant chasm between a planning algorithm and autonomous navigation on real robots. Most planning algorithms assume not only that a fully-explored, noise-free, perfect map is given but also that the robot's destination will always lie within this map. Moreover, the algorithms also assume a perfect robot pose and a perfect robot with ideal actuators that can execute an arbitrary trajectory. These assumptions are not practical. Therefore, utilizing a planning algorithm for autonomous navigation with real robots remains challenging. We propose and demonstrate experimentally an autonomous navigation system for a Cassie bipedal robot that is able to handle a noisy map in real-time, a distant goal that may not be in the initial map when the user decides where to send the robot, and importantly, a means to smoothly handle robot deviation. Additionally, a rudimentary finite-state machine is integrated to handle actions such as where to turn at intersections.

II. Related Work and Contributions

Motion planning, an essential component of robot autonomy, has been an active area of research for multiple decades with an accompanying rich literature. In this section, we review several types of planning algorithms and summarize our main contributions.

A. Sampling-Based Motion Planning

Rapidly exploring Tree (RRT) [1] stands out for its low complexity and high efficiency in exploring unknown configuration spaces. Its asymptotically optimal version — RRT* [5] — has also gained much attention and has contributed greatly to the spread of the RRT family. RRT, RRT*, and variations on the basic algorithms, generate a collision-free path comprised of piece-wise linear paths between discrete poses of the robot [1]–[9], [22], [23]. However, abrupt (non-differentiable) transitions between waypoints/pathways are an inherent issue with this family of planning algorithms and in addition, the generated trajectories do not account for control constraints. Therefore, to ensure the produced trajectories are feasible, additional expensive computations such as trajectory smoothing or optimization are often involved. A great deal of attention has been directed to this area, resulting in versions of RRT* [24]–[29] that utilize different smoothing techniques or steering functions.

Trajectory smoothing (B-spines, Dubins, or other parametric curves) is often designed independently of robot dynamics [30]–[32], which can lead to unbounded turning rate, acceleration, or jerk. Therefore, additional computations are necessary to validate the resulting smoothed trajectory. Furthermore, these methods are often ambiguous about how they treat robot deviations about the planned path and in the end provide open-loop control laws for tracking.

B. Reactive Planning

Reactive planning contributes another significant concept to the motion planning literature [10]–[18], namely potential fields. The method of reactive planning controls the motion of a robot by covering the configuration space with a potential field, creating a single attractive equilibrium around the target point and repulsive actions around obstacles. In other words, the reactive planning replaces the concept of trajectory with that of a vector field arising as the gradient of a potential function. The method of potential fields seems to address all the issues raised in Sec. II-A for sampling-based methods. However, most of the experimental work has been carried out on flat ground and it is unclear how extensions to undulating terrain can be performed. Maybe the biggest drawback for real-time applications is that these algorithms require a complete map to construct a potential field.

The concept of combining sampling-based algorithms with reactive planning was developed in [19]–[21], which not only provides a feasible path to follow from RRT* , but also a smooth feedback control law that instantaneously replans a path to the next goal as the robot deviates due to model imperfections in the robot's hardware or terrain. The feedback laws greatly ameliorate the issue of non-smooth paths. The feedback motion planning in [19] is based on a family of CLFs designed via linearization of the robot's model around a sufficiently large set of points in the robot's state space, LQR, and Sum of Squares (SoS), whereas the feedback motion planning of [20], [21] uses a single CLF and varies the associated equilibrium to set sub-goal poses.

The feedback motion planning of [20], [21] is a form of non-holonomic RRT* for differential-drive wheeled robots, where a CLF is utilized as the steering function in the RRT* algorithm and results in a system where robot control and motion planning are tightly coupled. This method is closest to ours. The CLF in [20], [21] is designed for differential-drive wheeled robots and hence is not suitable for bipeds, see Sec. III-A. Additionally, their version of RRT* assumes the robot is on flat ground and thus is not appropriate for undulating terrains. To date, the work has not been evaluated on hardware.

In this paper, we propose a CLF that is suitable for bipeds or omnidirectional robots (Sec. III) and utilize the proposed CLF in the RRT* algorithm; see Sec. IV. We integrate the omnidirectional RRT* into a full reactive planning system consisting of both a planning thread and a reactive thread. The planning thread includes a multi-layer, robot-centric local map, an anytime omnidirectional CLF RRT*, a sub-goal finder, and a finite-state machine. The reactive thread utilizes the CLF as a reactive planner to handle robot deviations; see Sec. V.

C. Contributions

In particular, the present work has the following contributions:

1) We propose a novel smooth Control Lyapunov Function (CLF) with a closed-form solution to the feedback controller for omnidirectional robots. The CLF is designed such that when a goal is far from the robot position, the CLF controls the robot orientation to align with the goal while moving toward the goal. On the other hand, the robot walks to the goal disregarding its orientation if the goal is close. Additionally, we study the behaviors of the CLF under different initial conditions and parameters.
2) We define a closed-form distance measure from a pose (position and orientation) to a target position for omnidirectional robots under a pose-centric polar coordinate. This distance metric nicely captures inherent features of Cassie-series robots, such as the low-cost of longitudinal movement and high-cost of lateral movement.
3) We utilize the proposed CLF and the distance measure to form a new variation of RRT* (omnidirectional CLF RRT*) to tackle undulating terrains, in which both distance and traversability are included in the cost to solve the optimal path problem.
4) We further develop a reactive planner where the motion of the robot is generated by a vector field depending on a closed-loop feedback policy giving control commands to the robot in real-time as a function of the robot's instantaneous pose. In other words, this reactive planner automatically adjusts the control commands to handle any deviations the robot may incur.
5) We integrate all the above components together as a reactive planning system for challenging terrains/cluttered indoor environments. It contains a planning thread to guide Cassie to walk in high traversable areas toward a distant goal on the basis of a multi-layer map being built in real-time and a reactive thread to handle robot deviation via a closed-loop feedback control instead of a commonly used waypoint-following or path-tracking strategy.
6) We perform three types of evaluation of the reactive planning system: **1)** A simplified biped pendulum model (inputs are piece-wise constant, similar to Cassie-series robots) navigates various synthetic, noisy, challenging outdoor terrains and cluttered indoor scenes. The system guides the robot to its goals in various scenes, both indoors and outdoors, with or without obstacles. The system also guides the robot to completion of several high-level missions, such as turning left at every intersection. **2)** To verify that the outputs of the control commands are feasible for Cassie-series robots, the system gives commands to a Cassie whole-body dynamic simulator [33], which simulates 20 degrees of freedom (DoF) of Cassie in Matlab Simmechanics on a 3D terrain. **3)** Lastly, the reactive planning system successfully allows Cassie Blue to traverse through the Wave Field on the North campus of the University of Michigan, as shown in Fig 1.
7) We provide all the simulated environments, the experimental data, and the implementations for the reactive planning system for challenging outdoor terrains/cluttered indoor environments in C++ and Robot Operating System (ROS) [34]; see https://github.com/UMich-BipedLab/CLF_reactive_planning_system [35].

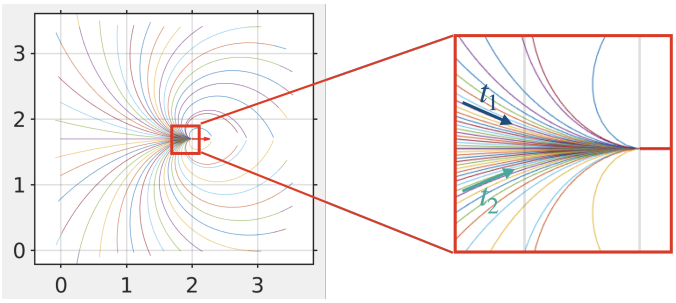


Fig. 2: The plots show paths in 2D generated by the CLF of [20], [21] for Dubins cars. At each point, the tangent to a path (t_1 and t_2 in the blowup) is the heading angle for the robot. These paths clearly fail to account for a biped's ability to move laterally. Moreover, in practice, an underactuated robot such as Cassie Blue would experience chattering in the heading angle when approaching the goal (red arrow). Moreover, if the robot overshoots the goal, it would have to walk along a circle to return to the goal. For these reasons, a new CLF is needed.

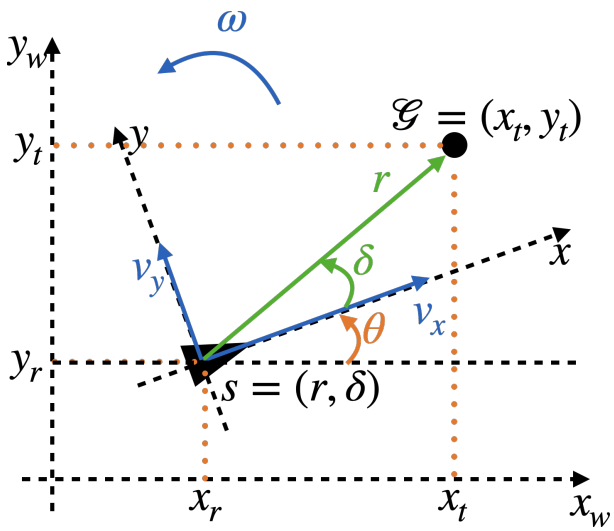


Fig. 3: Illustration of the robot pose-centric polar representation ($s = (r, \delta)$) for robot pose (x_r, y_r, θ) and target position $\mathcal{G} = (x_t, y_t)$. Here, r is the radial distance to the target and δ is the angle between the heading angle θ of the robot and the line of sight from the robot to the goal. The longitudinal velocity, lateral velocity, and angular velocity are v_x, v_y and ω, respectively.

The remainder of this paper is organized as follows. Section III constructs the new CLF for bipeds and omnidirectional robots. The omnidirectional CLF RRT* is introduced in Sec. IV. Section V integrates all the above components as a reactive planning system. Simulated and experimental evaluations of the proposed reactive system is presented in Sec. VI. Finally, Sec. VII concludes this paper and provides suggestions for future work.

III. Construction of a Control Lyapunov Function

This section describes the reasons for creating a new CLF function, the construction of the CLF, and an analysis of its parameters.

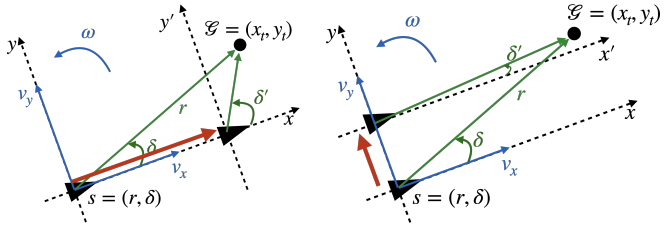


Fig. 4: This figure explains the signs in (2). On the left, δ increases when the robot moves parallel to the x-axis. Therefore, $\frac{v_x}{r}\sin\delta$ is positive. Similarly, $\frac{v_y}{r}\cos\delta$ is negated because δ decreases when the robot moves toward the y-axis.

A. Deficiency of Existing CLFs

The existing 2D CLF planners [20], [21] are designed for differentially driven non-holonomically constrained robots, whose dynamics and control laws are inappropriate for bipedal robots. Like most robot models, this work assumes that the robot is able to continuously change its velocity and heading. However, this is not possible for underactuated bipeds such as Cassie Blue. According to the ALIP model used for low-level feedback control of Cassie Blue [36]–[40], the heading angle and the longitudinal and lateral velocity commands can only be updated at the initiation of a step and not within a step. In other words, bipedal robots such as Cassie are not able to update their control commands during the swing phase and will execute the received control commands for the entire swing phase. When piece-wise constant commands are applied to the existing 2D CLF of [20], [21], built around a Dubins car model, the closed-loop system will oscillate about the discrete heading directions as the robot approaches the goal pose, as explained in Fig. 2. This oscillation is undesirable as it can affect the robot's balance.

With a Dubins car model as used in [20], [21], the linear velocity is always aligned with the heading angle of the vehicle, and hence this is also true as the vehicle approaches an equilibrium pose. Consequently, a CLF for a target position must also include a target heading, therefore, a target pose. The vehicle must steer and align itself as it approaches the target. Cassie Blue, on the other hand, similar to an omnidirectional robot, is able to move laterally with zero forward velocity, which allows the robot to start with an arbitrary pose and arrive at a goal position with an arbitrary heading (i.e., start with a pose and end with a position). Lateral walking, however, requires more effort due to the limited workspace of the lateral hip joints on the robot and this should be taken into account when designing a CLF.

To avoid undesirable oscillating movement and account for lateral walking, a new candidate CLF is designed on the basis of an appropriate kinematics model for underactuated bipeds and other omnidirectional robots.

B. State Representation

As mentioned in Sec. III, Cassie Blue is able to walk in any direction. Therefore, we model Cassie Blue as an omnidirectional robot and reduce it to a directional point mass. We will account for the increased effort required to walk laterally when we design the CLF.

Denote $\mathcal{P} = (x_r, y_r, \theta)$ the robot pose and $\mathcal{G} = (x_t, y_t)$ the goal position in the world frame. Let s be the state of an omnidirectional robot represented in a *robot pose-centric polar coordinate*:

$$s = \{(r, \delta) | r \in \mathbb{R}, \text{ and } \delta \in (-\pi, \pi]\}, \tag{1}$$

where $r = \sqrt{(x_t - x_r)^2 + (y_t - y_r)^2}$ and δ is the angle between the heading angle of the robot (θ) and the line of sight from the robot to the goal, as shown in Fig. 3.

Remark 1. *References [20], [21] used target pose-centric polar coordinates because the wheelchair robot needed to arrive at a target position with a target heading angle. In our case, we can use robot pose-centric coordinates because we have the freedom to arrive at the target position with any heading angle.*

C. Construction of Control Lyapunov Function

The kinematics of an omnidirectional robot is defined as

$$\begin{bmatrix} \dot{r} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} -\cos(\delta) & -\sin(\delta) \\ \frac{1}{r}\sin(\delta) & -\frac{1}{r}\cos(\delta) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} + \begin{bmatrix} 0 \\ \omega \end{bmatrix}. \tag{2}$$

In the above expression, we view v_x, v_y and ω as control variables. Because the matrix

$$\begin{bmatrix} -\cos(\delta) & -\sin(\delta) \\ \frac{1}{r}\sin(\delta) & -\frac{1}{r}\cos(\delta) \end{bmatrix}$$

is negative definite (and hence invertible) for all $r > 0$, the model (2) is over actuated for $r > 0$.

Remark 2. *Observe that $\delta < \delta'$ when the robot moves along the x-axis, as shown in Fig. 4a. Therefore, $\frac{v_x}{r}\sin\delta$ is positive. Similarly, $\frac{v_y}{r}\cos\delta$ is negated because $\delta < \delta'$ when the robot moves toward the y-axis, as shown in Fig. 4b.*

We next note that the change of control variables

$$\begin{bmatrix} v_r \\ v_\delta \end{bmatrix} := \begin{bmatrix} \cos\delta & \sin\delta \\ \frac{\sin\delta}{r} & -\frac{\cos\delta}{r} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} + \begin{bmatrix} 0 \\ \omega \end{bmatrix},$$

allows us to feedback linearize the model to a pair of integrators

$$\begin{bmatrix} \dot{r} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} -v_r \\ v_\delta \end{bmatrix}.$$

We note that for this model, any positive definite quadratic function is automatically a CLF. For later use, we note that for all $r > 0$,

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} := \begin{bmatrix} \cos(\delta) & r\sin(\delta) \\ \sin(\delta) & -r\cos(\delta) \end{bmatrix} \begin{bmatrix} v_r \\ v_\delta - \omega \end{bmatrix}. \tag{3}$$

As mentioned in Sec. III, lateral walking is more expensive than longitudinal walking because movement in the lateral hip joint is limited. A candidate control Lyapunov function[1] ℓ, in

[1]In polar coordinate, the function ℓ is positive definite in the sense that $\ell = 0 \implies r = 0$, and when $r = 0$, the angle δ is arbitrary or undefined.

terms of the robot's current pose and target (end) position, is defined as

$$\ell = \frac{r^2 + \gamma^2 \sin^2(\beta\delta)}{2}, \tag{4}$$

where γ is a weight on the orientation and the role of $\beta > 0$ will be described later. We next check that ℓ is a Control-Lyapunov function. The derivative of ℓ is

$$\begin{aligned} \dot{\ell} &= r\dot{r} + \frac{\beta\gamma^2}{2} \sin(2\,\beta\,\delta)\,\dot{\delta} \\ &= r(-v_r) + \frac{\beta\gamma^2}{2} \sin(2\beta\delta) v_\delta. \end{aligned} \tag{5}$$

The feedback

$$\begin{aligned} v_r &= k_{r1} \frac{r}{k_{r2} + r} \\ v_\delta &= -\frac{2}{\beta} k_{\delta 1} \frac{r}{k_{\delta 2} + r} \sin(2\beta\delta) \end{aligned} \tag{6}$$

results in

$$\dot{\ell} = -\frac{k_{r1}}{k_{r2} + r} r^2 - k_{\delta 1} \beta \gamma^2 \frac{r}{k_{\delta 2} + r} \sin(2\beta\delta), \tag{7}$$

which is negative for all $r > 0$, $\beta > 0$, $k_{r1} > 0$, $k_{r2} > 0$, $k_{\delta 1} > 0$, and $k_{\delta 2} > 0$.

Remark 3. *From* (6)*, it follows that* $\dot{\delta} = 0$ *for* $2\beta\pi \in \{0, \pm\pi\}$. *Therefore, the manifolds*

$$M_\delta := \{(r, \delta) \mid r \geq 0, \delta \in \{0, \frac{\pi}{\beta}, \pm\frac{\pi}{2\beta}\}\}$$

are invariant for the closed-loop system. From (7)*, the manifold* M_δ *is locally attractive for* $\delta \in \{0, \frac{\pi}{\beta}\}$ *and repulsive for* $\delta = \pm\frac{\pi}{2\beta}$. *By selecting* $\beta > 0$*, the repulsive invariant manifold can be placed outside the field of view (FoV) of Cassie, as shown in Fig. 6. In practice, a finite-state machine (FSM) is needed so that the robot will initially turn in place so that it starts with the goal located within the FoV of its sensor suite.*

The next step is to set up an optimization such that the control variables (v_x, v_y, ω) satisfy (6) and take into account that walking sideways takes more effort than walking forward, for Cassie. Because the camera faces forward, walking backward is only selected if the robot is localized into an already built portion of the map.

D. Closed-form Solution

Taking (3) as a constraint, we propose to select ω so as to keep v_y small (limit lateral walking) by optimizing

$$J = \min_{v_y, \omega} \quad (v_y)^2 + \alpha\omega^2. \tag{8}$$

The parameter $\alpha > 0$ allows us to penalize aggressive yaw motions ω, as will be illustrated in Sec. III-E. Plugging in the constraint (3), (8) leads to

$$\begin{aligned} J &= \min_{\omega} \quad \{[\sin(\delta) v_r - r\cos(\delta)(v_\delta - \omega)]^2 + \alpha\omega^2\} \\ &= \min_{\omega} \quad \{(\sin(\delta) v_r)^2 + [r\cos(\delta)(v_\delta - \omega)]^2 \\ &\qquad - 2\sin(\delta) v_r (r\cos(\delta)(v_\delta - \omega)) + \alpha\omega^2\}. \end{aligned}$$

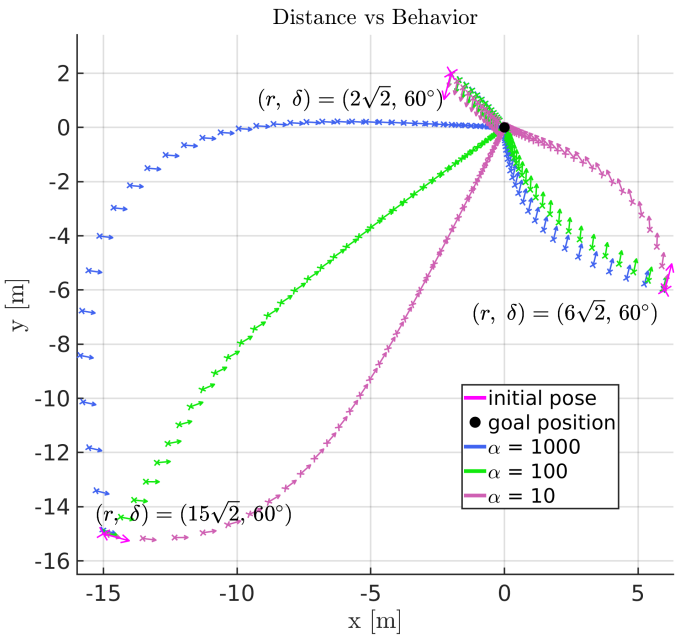


Fig. 5: The distance to the target and the penalty on yaw motion in (8) both affect closed-loop behavior arising from the CLF. The arrows indicate the robot's absolute heading. In each solution of the closed-loop system, the robot's heading relative to the target is initialized at 60°. When the robot is distant from the goal and the heading does not point toward the goal, it will align its relative heading to the target while approaching the goal. The level of alignment depends on the yaw motion penalty, α. On the other hand, when the robot is close to the goal, the closed-loop controller no longer adjusts the heading angle and employs a lateral motion to reach the goal.

TABLE I: The default values of parameters.

α	β	γ	k_{r1}	k_{r2}	$k_{\delta 1}$	$k_{\delta 2}$
10	1.2	1	1	5	0.1	10

A few algebraic calculations and the dropping of "constant terms" lead to

$$\begin{aligned} \omega^* = \arg\min_{\omega} \quad & \{r^2 \cos^2(\delta) (v_\delta - \omega)^2 + \\ & 2 r v_r \sin(\delta)\cos(\delta)\omega + \alpha\omega^2\}, \end{aligned}$$

which implies that

$$\left(\alpha + r^2\cos^2(\delta)\right)\omega^* + r\cos(\delta)\left[v_r \sin(\delta) - r v_\delta \cos(\delta)\right] = 0$$

. The final result is

$$\omega^* = \frac{r\cos(\delta)\left[r v_\delta \cos(\delta) - v_r \sin(\delta)\right]}{\alpha + r^2\cos^2(\delta)}, \tag{9}$$

and then

$$\begin{aligned} v_y^* &= \frac{\alpha\ (v_r\ \sin(\delta) - r\, v_\delta\ \cos(\delta))}{r^2 \cos(\delta)^2 + \alpha} \\ v_x^* &= \frac{v_r\ \cos(\delta)\ r^2 + \alpha\, v_\delta\ \sin(\delta)\ r + \alpha\, v_r\ \cos(\delta)}{r^2 \cos(\delta)^2 + \alpha}. \end{aligned} \tag{10}$$

E. Qualitative Analysis of the Closed-loop Trajectories

The default parameters applied in this analysis are shown in Table. I. Figure 5 shows how the closed-loop trajectories vary as a function of heavy, medium, and light penalties on yaw motion, and three different initial distances from the target,

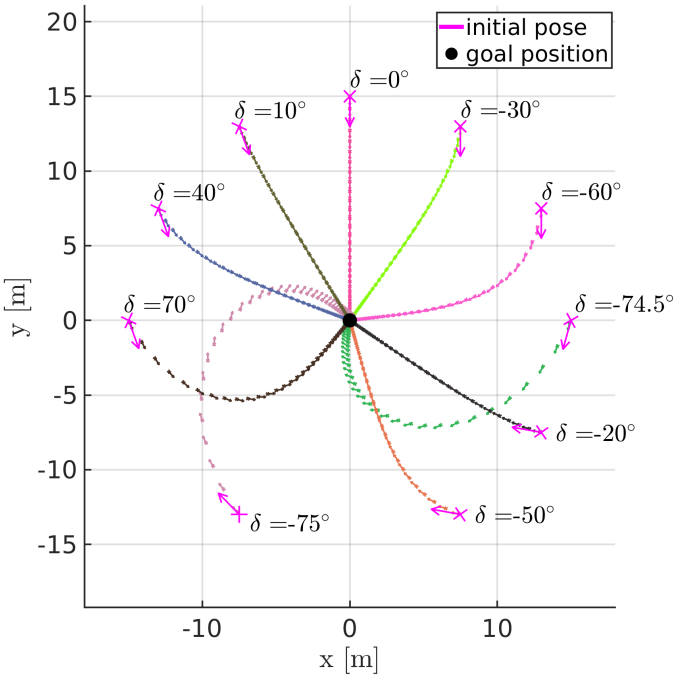


Fig. 6: This figure illustrated how the closed-loop trajectories generated by the CLF in (4) vary as a function the initial relative heading to the target, when starting at a fixed distance of $r = 15$ m, and $\alpha = 10$. The arrows indicate the robot's heading. As shown in Table I, we are using $\beta = 1.2$, which yields an FoV of $\pm 75°$. For relative heading "errors" less than $40°$, the robot aligns quickly to the target and longitudinal walking dominates. These motions should be compared to those in Fig. 5

with δ, the robot's heading relative to the target, fixed at $60°$. We observe that with $r = 2\sqrt{2}$, the robot walks laterally to achieve the goal for all values of the penalty on yaw motion. With $r = 15\sqrt{2}$ and $\alpha = 10$, the robot aligns its heading to the target while walking to reduce its lateral movement, whereas with $\alpha = 100$, it maintains its heading and combines lateral and longitudinal motion as needed to reach the goal.

Figure 6 shows how the closed-loop trajectories vary as a function the initial relative heading to the target, when starting at a fixed distance of $r = 15$ m, and $\alpha = 10$. As indicated in Table I, we are using $\beta = 1.2$, which yields FoV of $\pm 75°$. For relative heading "errors" less than $40°$, the robot aligns quickly to the target and longitudinal walking dominates. If quicker zeroing of the heading error is desired, a smaller value of α could be used or the robot could turn in place before starting a new segment.

IV. Omnidirectional CLF-RRT*

This section integrates the CLF proposed in Sec. III into the original RRT* algorithm. The resulting omnidirectional CLF RRT* provides feasible paths for (2) while (i) accounting for relative heading, (ii) the asymmetry in roles of target position and current pose induced by the CLF, and (iii) the fact that walking laterally is more challenging than walking in the longitudinal direction for robots such as Cassie.

A. Standard RRT* Algorithm

The original RRT* [3] is a sampling-base, incremental planner with guaranteed asymptotic optimality. In configuration space, RRT* grows a tree where leaves are states connected by edges of linear path segments with the minimal cost. Additionally, RRT* considers nearby nodes of a sample to choose the best parent node and to rewire the graph if shorter path is possible to guarantee asymptotic optimality.

B. Omnidirectional CLF-RRT* Algorithm

The omnidirectional CLF-RRT* differs from the original RRT* in four aspects. First, the distance between two nodes is defined by the CLF in (4), which takes relative heading into account. Second, the steering/extending functions use the closed-loop trajectories generated by (10) to define paths between nodes. Third, because the cost (4) between two nodes i and j is not symmetric (i.e, a different cost is assigned if node i is the origin versus it is the target), a distinction must be made between near-to nodes and near-from nodes. The above three aspects are common to the CLF-RRT* variant introduced in [20], [21]. Finally, when connecting, exploring, and rewiring the tree, an additional term is added to the cost (4) to account for the relative ease or difficulty of traversing the path.

Our proposed RRT* modification is summarized below with notation that generally follows [4]. Let $\mathcal{X} = \{(x, y, \theta) \mid x, y \in \mathbb{R} \text{ and } \theta \in (-\pi, \pi]\}$ be the configuration space and let $\mathcal{X}_{\text{obs}}$ denote the obstacle region, which together define the free region for walking $\mathcal{X}_{\text{free}} = \mathcal{X} \backslash \mathcal{X}_{\text{obs}}$. The omnidirectional CLF RRT* solves the optimal path planning problem by growing a tree $\mathcal{T} = (V, E)$, where $V \in \mathcal{X}_{\text{free}}$ is a vertex set of poses connected by edges E of feasible path segments. Briefly speaking, the proposed RRT* (Algorithm 1) explores the configuration space by random sampling and extending nodes to grow the tree (explore the configuration space), just as in the classic RRT [1]. Considering nearby nodes of a sample to choose the best parent node and rewiring the graph guarantee asymptotic optimality (Algorithm 2 and 3), as with the classic algorithm. As emphasized previously, a key difference lies in how the paths between vertices are generated.

1) Sampling: This step randomly samples a pose $n_{\text{rand}} = (x, y, \theta) \in \mathcal{X}_{\text{free}}$. To facilitate faster convergence and to find better paths, we use sampling with a goal bias, limited search space, and Gaussian sampling. For more details, see our implementation on GitHub [35].

2) Distance: The distance $d(n_i, n_k)$ from node n_i to node n_k in the tree $\mathcal{T}$ is defined by (4), which takes relative heading into account. Note that when computing the distance, n_i is a pose (x_i, y_i, θ_i) and the heading of n_k is ignored and only its (x_k, y_k) values are used.

Remark 4. *As mentioned in Sec. III-C, the robot will rotate in place if the target point is outside the FoV. If rotating in place is laborious, one can also consider the following distance function:*

$$d(n_i, n_k) = \ell + k_\delta \max\left(|\delta| - |U|, 0\right),$$

where ℓ is defined in (4), k_δ is a positive constant, and U corresponds to a repulsive point (i.e., $\pm\frac{\pi}{2\beta}$) in Remark 3.

Algorithm 1: $\mathcal{T} = (V, E) \leftarrow$ Omnidirectional CLF RRT*

$\mathcal{T} \leftarrow$ InitializeTree();
$\mathcal{T} \leftarrow$ InsertNode($\emptyset, n_{\text{init}}, \mathcal{T}$);
for *i=1* **to** N **do**
  $n_{\text{rand}} \leftarrow$ Sample(i)
  $n_{\text{nearest}} \leftarrow$ Nearest($\mathcal{T}, n_{\text{rand}}$)
  $(n_{\text{new}}, \mathscr{T}') \leftarrow$ Extend($n_{\text{nearest}}, n_{\text{rand}}, \kappa$)
  if *ObstacleFree($\mathscr{T}'$)* **then**
    $\mathcal{N}_T \leftarrow$ NearTo($\mathcal{T}, n_{\text{new}}, |V|$)
    $n_{\text{min}} \leftarrow$ ChooseParent($\mathcal{N}_T, n_{\text{nearest}}, n_{\text{new}}$)
    $\mathcal{T} \leftarrow$ InsertNode($n_{\text{min}}, n_{\text{new}}, \mathcal{T}$)
    $\mathcal{N}_F \leftarrow$ NearFrom($\mathcal{T}, n_{\text{new}}, |V|$)
    $\mathcal{T} \leftarrow$ ReWire($\mathcal{T}, \mathcal{N}_F, n_{\text{min}}, n_{\text{new}}$)
return $\mathcal{T}$

3) Traversability of a path: Let $\mathcal{P} = (x_r, y_r, z_r, \theta)$ be the current robot pose and denote $\mathscr{T}(\mathcal{P}, n_i, n_j)$ the path[2] connecting n_i and n_j. We also denote $C_e(x_t, y_t), C_s(x_t, y_t)$ as the elevation and the magnitude of the gradient at a point (x_t, y_t) in a local map $\mathcal{M}$, see Sec. V-B. Finally, let $\mathbb{T}(\mathcal{P}, \mathscr{T})$ be the cost of the path traversability, defined as

$$\mathbb{T}(\mathcal{P}, \mathscr{T}) = \sum_{\forall x_t, y_t \in \mathscr{T}} C_e(x_t, y_t) + k_s C_s(x_t, y_t) + k_r(C_e(x_t, y_t) - z_r), \tag{11}$$

where k_s and k_r are the corresponding positive coefficients.

Remark 5. *Traversability varies among different types of robots. Additional elements can be added as needed to account for different aspects for traversability computation.*

4) Cost between Nodes: Let $c(n_i, n_k)$ be the cost from n_i to n_k in the tree $\mathcal{T}$, defined as

$$c(n_i, n_k) = d(n_i, n_k) + k_t \mathbb{T}(\mathcal{P}, \mathscr{T}), \tag{12}$$

where k_t is the weight of traversability.

5) Nearby Nodes: Due to the use of the CLF function, the distinction between near-to nodes $\mathcal{N}_T$ and near-from nodes $\mathcal{N}_F$ is necessary.

$$\mathcal{N}_T(n_i, \mathcal{T}, \mathcal{M}, m) := \{n \in V \mid d(n, n_i) \le L(m) \ \ \& \ |\mathbb{T}(n, \mathcal{P}) - \mathbb{T}(n_i, \mathcal{P})| \le T_k\}, \tag{13}$$

where $|\cdot|$ is the absolute value, m is the number of nodes in the tree $\mathcal{T}$, and $L(m) = \eta \left(\log(n)/n\right)^{(1/\xi)}$ with the constant η and dimension of space ξ (3 in our case) [6] and T_k is a positive constant. Similarly, the near-from nodes $\mathcal{N}_F$ are determined by

$$\mathcal{N}_F(n_i, \mathcal{T}, \mathcal{M}, m) := \{n \in V \mid d(n_i, n) \le L(m) \ \ \& \ |\mathbb{T}(n, \mathcal{P}) - \mathbb{T}(n_i, \mathcal{P})| \le T_k\}. \tag{14}$$

6) Nearest Node: Given a node $n_i \in \mathcal{X}$, the tree $\mathcal{T}$, and the local map $\mathcal{M}$, the nearest node is any node $n_* \in \mathcal{T}$ in the tree where the cost from n_* to n_i is minimum.

[2]The path is generated from the CLF in Sec. III.

Algorithm 2: $n_{\text{parent}} \leftarrow$ ChooseParent($\mathcal{N}_T, n_{\text{nearest}}, n_{\text{new}}$)

$n_{\text{parent}} \leftarrow n_{\text{nearest}}$
$c_{\text{parent}} \leftarrow$ Cost(n_{nearest}) + c($n_{\text{nearest}}, n_{\text{new}}$)
for $n_{near} \in \mathcal{N}_T$ **do**
  $\mathscr{T}' \leftarrow$Steer($n_{\text{near}}, n_{\text{new}}$)
  if *ObstacleFree($\mathscr{T}'$)* **then**
    $c' =$ Cost(n_{near}) + $c(n_{\text{near}}, n_{\text{new}})$
    if $c' <$ *Cost*(n_{new}) *and* $c' < c_{parent}$ **then**
      $n_{\text{parent}} \leftarrow n_{\text{near}}$
      $c_{\text{parent}} \leftarrow c'$
return n_{parent}

7) Steering and Extending: The steering function generates a path segment $\mathscr{T}$ that starts from n_i and ends exactly at n_k. The extending function extends the path from n_i toward n_k until n_k is reached or the distance traveled is κ in which case it returns a new sample n_{new} at the end of the extension.

8) Parent Choosing and Graph Rewiring: Choosing the best parent node (Algorithm 2) and rewiring the graph (Algorithm 3) guarantee asymptotic optimality. Let Cost(n_i) be the cost from the root of the tree $\mathcal{T}$ to the node n_i. The parent n_{parent} of a node n_{new} is determined by finding a node $n_i \in \mathcal{N}_T$ with smallest cost from the root to the node:

$$n_{\text{parent}} = \underset{n_{\text{near}} \in \mathcal{N}_T}{\arg\min} \text{Cost}(n_{\text{near}}) + c(n_{\text{near}}, n_{\text{new}}).$$

After a parent node is chosen, nearby nodes $\mathcal{N}_F$ are rewired if shorter paths are found. In our experiments, we used the extending function for exploration, and the steering function to find the best parent node and to rewire the graph.

9) Collision Check: This step verifies whether a path $\mathscr{T}$ lies within the obstacle-free region of the configuration space. Note that additional constraints, such as curvature bounds and minimum clearance, can also be examined in this step.

10) Node Insertion: Given the current tree $\mathcal{T} = (V, E)$ and a node $v \in V$, this step inserts the node n to V and creates an edge e_{nv} from n to v.

V. Reactive Planning System

The previous section provides a sparse set of paths from a robot's initial location to a goal. The degree of optimality depends on how long the planning algorithm is run. A typical update rate may be 5 Hz for real-time applications. When the robot is perturbed off the nominal path, one is left with deciding how to reach the goal, say by tracking the nominal

Algorithm 3: $\mathcal{T} \leftarrow$ ReWire($\mathcal{T}, \mathcal{N}_F, n_{\text{min}}, n_{\text{new}}$)

for $n_{near} \in \mathcal{N}_F \backslash \{n_{min}\}$ **do**
  $\mathscr{T}' \leftarrow$Steer($n_{\text{new}}, n_{\text{near}}$)
  if *ObstacleFree($\mathscr{T}'$) and*
    Cost(n_{new}) + $c(n_{new}, n_{near})$ < *Cost*(n_{near}) **then**
    $\mathcal{T} \leftarrow$ Re-Connect($n_{\text{new}}, n_{\text{near}}, \mathcal{T}$)
return $\mathcal{T}$

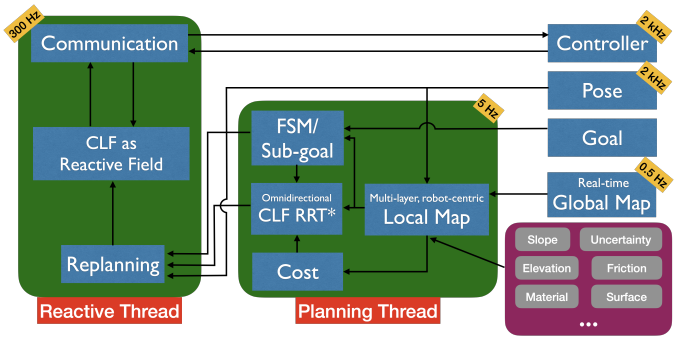


Fig. 7: This figure summarizes the proposed reactive planning system. The planning thread is built around RRT* and an omnidirectional CLF that is used to assign distances, define locally optimal path segments, search radius, and linking conditions for re-wiring and choosing a parent. In addition, the planning thread contains a multi-layer, robot-centric local map for computing traversability, a sub-goal finder, and a finite-state machine (FSM) to choose sub-goal locations guiding the robot to a distant goal. Instead of a common waypoint-following or path-tracking strategy, the reactive thread copes with robot deviation while eliminating non-smooth motions via a vector field (defined by a closed-loop feedback policy arising from the CLF). The vector field provides real-time control commands to the robot's gait controller as a function of instantaneous robot pose.

path with a PID controller. Important alternatives to this, called a high-frequency reactive planner or a feedback motion planner, were introduced in [10]–[19]. A version based on the work of [20], [21] will be incorporated into our overall planning system. In addition, we take into account features in a local map.

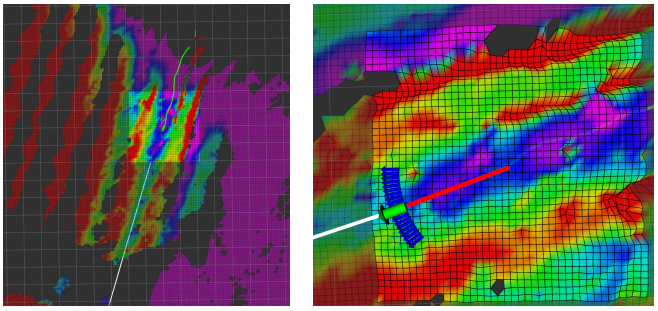
A. Elements of the Overall Planning System

The overall objective of the planner system is to replace the commonly used waypoint-following or path-tracking strategies with a family of closed-loop feedback control laws that steer the robot along a sequence of collision-free sub-goals leading to the final goal. In simple terms, as in [19]–[21], we populate the configuration space with a discrete set of feedback control laws that steer the robot from local chart about a sub-goal to the sub-goal itself. The collision free property is handled by the low-frequency planner at the current time. Others have used CBFs for this purpose [41]–[46]. A finite-state machine (FSM) is integrated into the low-frequency planner to handle high-level mission requirements such as turning left at every intersection.

The planner assumes the initial robot pose, a final goal, and real-time map building are provided. It is assumed that the initial robot pose and final goal are initialized in an otherwise featureless metric map, with the robot's initial pose as the origin. The featureless map is filled in by the real-time mapping package [47]–[49] based on collected LiDAR and/or camera data.

B. Planning Thread

The planning thread deals with short-range planning (less than 20 meters) at a frequency of 5 to 10 Hz. It includes a robot-centric local map, our omnidirectional CLF-RRT* algorithm of Section III, cost computation, a sub-goal finder, and a finite-state machine.



(a) (b)

Fig. 8: The elevation map was built online while Cassie was autonomously traversing the Wave Field on the North Campus of the University of Michigan. The highlighted area is the smoothed, robot-centric local map. The blue arc and the green arrow are the sub-goal finder and the chosen sub-goal for the omnidirectional CLF RRT*, respectively. The red shows the locally optimal path.

1) Robot-centric Local Map and Cost Computation: Figure 8a shows the robot-centric multi-layer local map (highlighted area), which crops a sub-map centered around the robot's current position from the global map provided by the mapping algorithm. The local map computes additional useful information such as terrain slope (local gradient) which is useful for assigning cost. The computations with the local map are efficient compared to processing the full map.

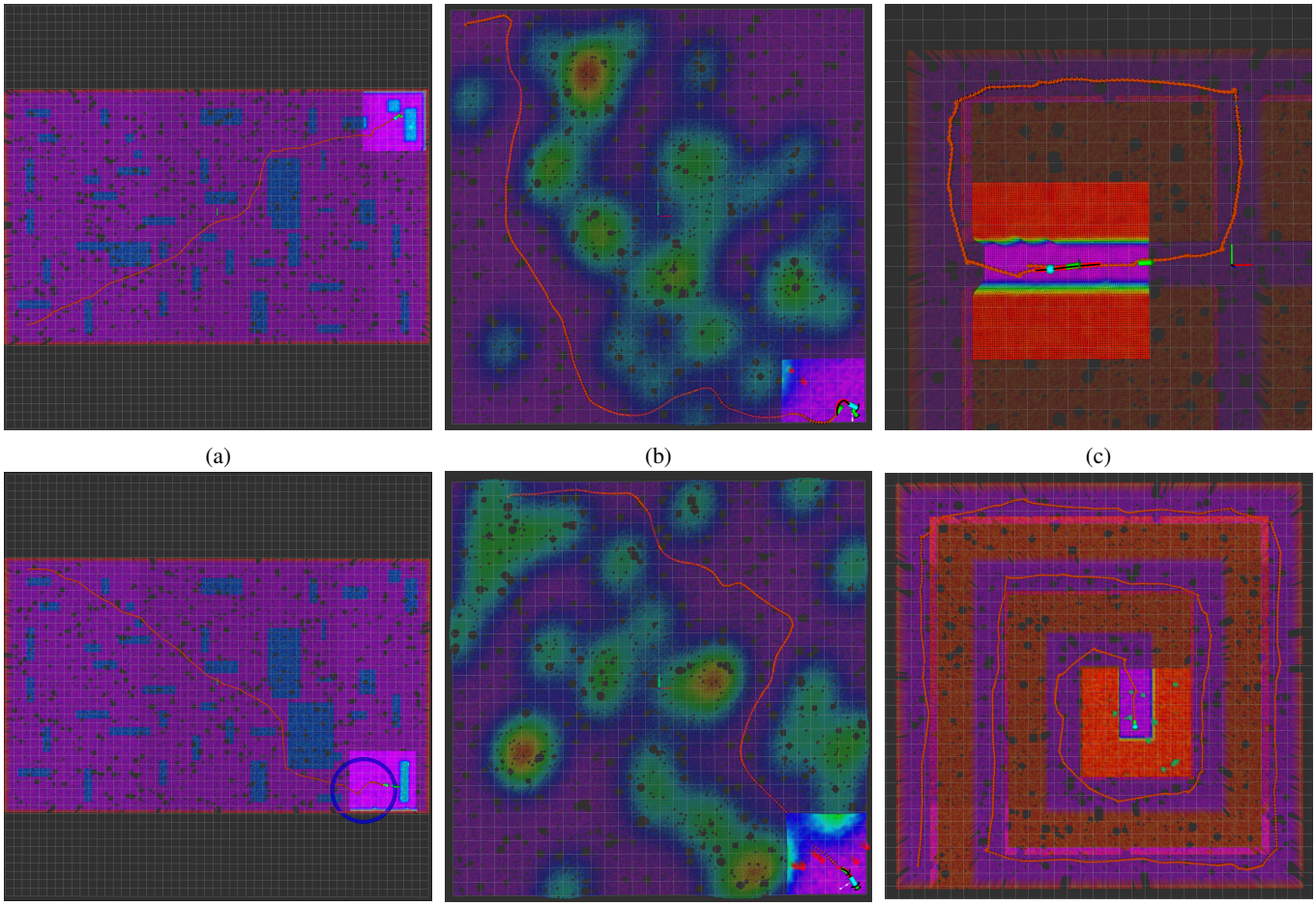
2) Anytime Omnidirectional CLF-RRT Planner:* The anytime feature is a direct result of using RRT* as a planner. The algorithm can be queried at anytime to provide a suboptimal path comprised of wayposes, which the CLF (4) turns into real-time feedback laws for anytime replanning.

3) Sub-goal Finder and Finite-State Machine: Ideally, a global planner is present to guide the robot to a distant goal, which may not be viewable at the time of mission start [50]. In relatively simple situations such as that shown in Fig. 9 and Fig. 8b, it is sufficient to complete many of short-term missions by positioning a sub-goal (green arrow) at the lowest cost (cost-to-come + cost-to-goal) on an arc (blue arrows) to guide the robot to the final goal. This sub-goal finder is also used as a finite-state machine to handle high-level missions such as making turn selections at intersections. In the future, the sub-goal finder will be replaced with a global planner.

C. Reactive Thread

The work in [10]–[18] provided a significant alternative to the standard path tracking. Their *high frequency reactive planners* create a vector field on the configuration space whose integrals curves (i.e., solutions of the vector field) provide alternative paths to the goal. When the robot is perturbed, it immediately starts following the new path specified by the vector field, instead trying to asymptotically rejoin the original path. The vector field is in essence an instantaneous re-planner.

In the reactive planner of [11], [13], the vector field arises from the gradient of a potential function defined on the configuration space. Here, we use the solutions of the closed-loop system associated with the CLF in (4) to define alternative paths in the configuration space. In essence, our feedback functions (10) and (9) provide instantaneous re-planning of



(a) (b) (c)

(d) (e) (f)

Fig. 9: Simulated scenes and results obtained with the proposed reactive planning system. On the left are cluttered indoor scenes with obstacles and holes, in the middle are noisy undulating outdoor terrains, and on the right, are high-level missions. Each grid in a map is 1×1 meter. The simulated robot is based on the ALIP model and accepts piece-wise constant inputs at the beginning of each step, is used in all simulation. The robot's initial pose and position of the final goal were hand selected. The highlighted areas show the local maps being provided to the robot 8×8 for left and middle columns and 9×9 for the right column. In each case, the planner guided the robot to the goal. Animations of the simulations are available at [35].

the control commands for the omnidirectional model (2). This reactive planner can be run at 300 Hz in real-time.

The reactive thread is a reactive planner, in which the motion of the robot is generated by a vector field that replies on a closed-loop feedback policy giving controller commands in real-time as a function of the instantaneous robot pose. In other words, the reactive planner utilizes the proposed Control Lyapunov Function described in Sec. III to adjust controller commands automatically when the robot deviates from the optimal path. This thread steers the robot to the optimal path at 300 Hz.

Remark 6. *The "timing" of Cassie's foot placement is inherently event-driven and stochastic. Even though a step cycle may be planned for 300 ms, variations in the terrain and deviations of the robot's joints from nominal conditions result in foot-ground contact being a random variable, with a mean of roughly 300 ms. Running the reactive planner at anything over 100 Hz essentially guarantees that Cassie's gait controller, which runs at 2 kHz, is accepting the most up-to-date commands from the planner, even if a few messages are lost over UDP transmission.*

VI. Simulation and Experimental Results

The proposed reactive planning system integrates a local map, the omnidirectional CLF RRT*, and fast replanning from the reactive thread. We performed three types of evaluation of the reactive planning system.

A. Angular Linear Inverted Pendulum (ALIP) Robot with Simulated Challenging Outdoor Terrains and Indoor Cluttered Scenes

We first ran the reactive planning system on several synthetic environments, in which an ALIP robot model [36], [37] navigated several simulated noisy, patchy, challenging outdoor terrains as well as cluttered indoor scenes. The ALIP robot successfully reached all the goals in different scenes. We tested the system on more than 10 different environments, both indoor and outdoor with and without obstacles. Due to space limitations, we only show the results of six simulations in Fig 9; see our GitHub [35] for videos and more results.

Remark 7. *The ALIP robot [36], [37] takes piece-wise constant inputs from the reactive planning system. Let* g, H, τ

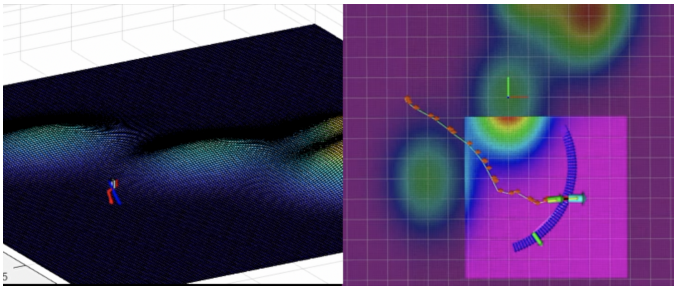


Fig. 10: A simulation of a C++-implementation of the reactive planner on full-dynamic model of Cassie, which accounts for all 20 degrees of freedom of Cassie in Matlab-Simmechanics and a 3D terrain model. The reactive planning system receives the pose of the simulated Cassie via User Datagram Protocol (UDP). Cassie's simulator receives and executes the resulting control commands via UDP. The planning system successfully takes the simulated Cassie to the goal without falling. An animation is available at [35]

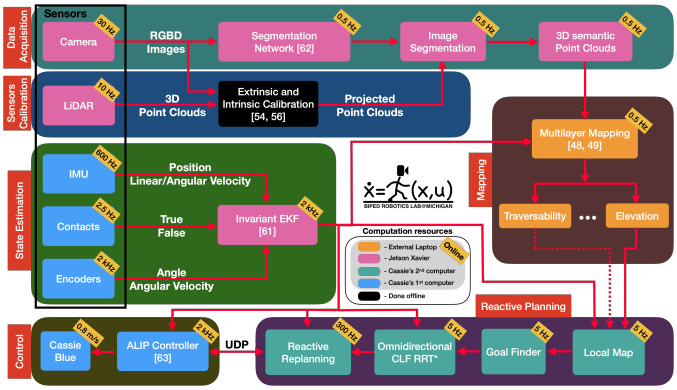


Fig. 11: Illustration of how the various processes in the overall autonomy system are distributed and their computation frequencies. The larger boxes indicate various modules such as Data Acquisition, Planning, and Control. The smaller boxes are colored according to the processor that runs them.

be the gravity, the robot's center of mass height, and the time interval of a swing phase, respectively. The motion of an ALIP robot on the x-axis is defined as

$$\begin{bmatrix} x_{k+1} \\ \dot{x}_{k+1} \end{bmatrix} = \begin{bmatrix} \cosh(\xi) & \frac{1}{\rho}\sinh(\xi) \\ \rho\sinh(\xi) & \cosh(\xi) \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix} + \begin{bmatrix} 1-\cosh(\xi) \\ -\rho\sinh(\xi) \end{bmatrix} p_x, \tag{15}$$

where p_x is the center of mass (CoM) on the x-axis of the robot, $\xi = \rho\tau$ and $\rho = \sqrt{g/H}$. Similarly, the motion of the robot on the y-axis can be defined.

Remark 8. *Even though a full global map is given in each simulation environment, only the information in the local map is given to the planning system at each timestamp. The path generated from omnidirectional RRT* is asymptotically optimal within the local map, for the given time window. It is emphasized that no global information is provided to the planner which is why the resulting trajectory from the initial point to the goal may not be the shortest path.*

B. Validation of Control Command Feasibility via a Whole-body Cassie Simulator

To ensure the control commands from the reactive planning system are feasible for Cassie-series bipedal robots, we sent the commands via User Datagram Protocol (UDP) from ROS C++ to Matlab-Simmechanics, which simulates a 20 DoF of Cassie, using footfalls on the specified terrain. The simulator then sent back the pose of the simulated Cassie robot to the planning system to plan for the optimal path via UDP. The planner system successfully took the simulated Cassie to the goal without falling, as shown in Fig. 10.

C. System Integration for Real-time Deployment

System integration is critical for real-time use. Figure 11 shows the integrated system, distribution, and frequency of each computation. In particular, the sensor calibrations are performed via [51]–[59]. The invariant Extended Kalman Filter (InEKF) [60], [61] is used to estimate the state of Cassie Blue at 2 kHz. Images are segmented via MobileNets [62] and a LiDAR point cloud is projected back to the segmented image to produce a 3D segmented point cloud. The resulting point clouds are then utilized to build a multi-layer map (MLM) [47]–[49]. The reactive planning system then crops the MLM around the robot position to create a local map and performs several operations to acquire extra information, as described in Sec. V-B. Additionally, the reactive planner receives the robot poses from the InEKF at 300 Hz to adjust the control commands that guide the robot to the nominal sub-poses via the proposed CLF; see Sec. III and Sec. V-C. The control commands are then sent to Cassie Blue's gait controller [36], [63], [64] via UDP.

D. Full Autonomy Experiments with Cassie Blue

We achieved full autonomy with Cassie Blue on the Wave Field, located on the North campus of the University of Michigan, an earthen sculpture designed by Maya Lin [65]; see Fig. 12a. The Wave Field consists of sinusoidal humps with a depth of approximately 1.5 m from the bottom of the valleys to the crest of the humps; there is a second sinusoidal pattern running orthogonal to the main pattern, which adds 25 cm ripples peak-to-peak even in the valleys. Figure 12b shows the top-view of the resulting trajectory of the reactive planning system. The planning system guided Cassie Blue to walk in the valley (the more traversable area), as shown in Fig. 12c. The planning system navigated Cassie Blue around a hump that protrudes into one of the valleys, as shown in Fig. 12d. Figure 13 shows the control commands sent to Cassie Blue. This experiment was presented in the Legged Robots Workshop at ICRA 2021; the video can be viewed at [66]. The video of the Wave Field experiment is uploaded and can be found at [67]. More experiments are still being conducted and this paper will be updated accordingly.

VII. Conclusion and Future Work

We presented a novel reactive planning system that consists of a 5-Hz planning thread to guide a robot to a distant goal and a 300-Hz CLF-based reactive thread to cope with robot deviations. In simulation, we evaluated the reactive planning system on ten challenging outdoor terrains and cluttered indoor scenes, and performed fully autonomy experiments with Cassie Blue, a bipedal robot with 20 DoF, on sinusoidally varying terrain.

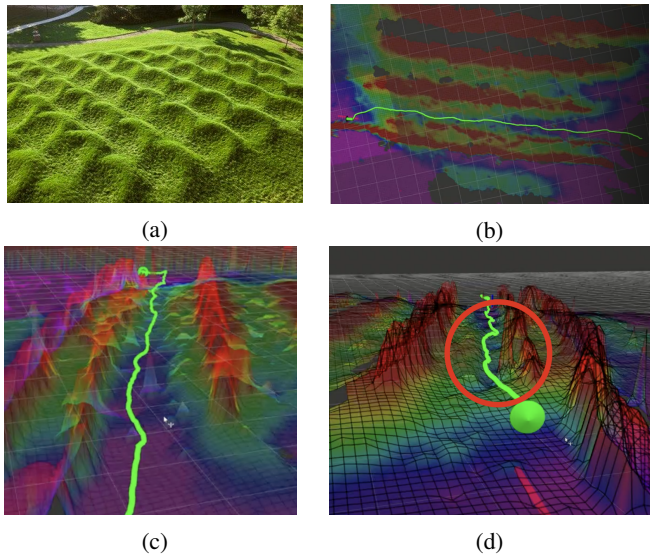


Fig. 12: Experimental results on the Wave Field. The top-left shows the experiment terrain, the Wave Field, on the North Campus of the University of Michigan. The top-right shows a bird's-eye view of the resulting trajectory from the reactive planning system. The bottom-left shows a back-view of the trajectory produced by the planning system as Cassie Blue walks in a valley (highly traversable area) of the Wave Field. The bottom-right demonstrates the planning system avoiding areas of higher cost.

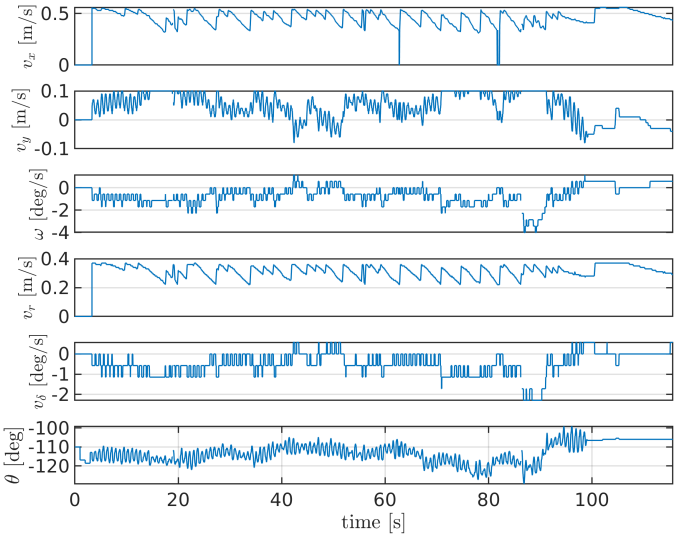


Fig. 13: Control Commands sent to Cassie Blue.

The planning thread uses a multi-layer, robot-centric local map to compute traversability for challenging terrains, a sub-goal finder, and a finite-state machine to choose a sub-goal location as well as omnidirectional CLF RRT* to find an asymptotically optimal path for Cassie to walk in a traversable area. The omnidirectional CLF RRT* utilizes the newly proposed Control-Lyapunov function (CLF) as the steering function and the distance measure on the CLF manifold in the RRT* algorithm. Both the proposed CLF and the distance measure have a closed-form solution. The distance measure nicely accounts for the inherent "features" of Cassie-series robots, such as high-cost for lateral movement. The robot's motion in the reactive thread is generated by a vector field depending on a closed-loop feedback policy providing control commands to the robot in real-time as a function of instantaneous robot pose. In this manner, problems typically encountered by waypoint-following and pathway-tracking strategies when transitioning between waypoints or pathways (unsmooth motion, sudden turning, and abrupt acceleration) are resolved.

In the future, we shall combine control barrier functions [41]–[46] with the CLF in the reactive thread to handle dynamic obstacles. Additionally, the current local map is a 2.5D, multi-layer grid map with fixed resolution; it is also interesting to see how to efficiently represent a continuous local map. Furthermore, how to extend the CLF to 3D is another interesting area for future research.

Acknowledgment

Toyota Research Institute provided funds to support this work. Funding for J. Grizzle was in part provided by NSF Award No. 1808051 and 2118818. This article solely reflects the opinions and conclusions of its authors and not the funding entities. The authors thank Lu Gan and Ray Zhang for their assistance in the development of the autonomy package used on Cassie Blue in these experiments and Yukai Gong and Dianhao Chen for the low-level gait controller used in Cassie. They also thank Lu Gan, Ray Zhang, Yukai Gong, Dianhao Chen, Oluwami Dosunmu-Ogunbi, Jinze Liu, Jianyang Tang, and Minzhe Li for their assistance in the Wavefield Experiment. The first author thanks to Jong Jin Park, Collin Johnson, Peter Gaskell, and Prof. Benjamin Kuipers for kindly providing insightful discussion for their work. The first author thanks Wonhui Kim for useful conversations.

References

[1] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[2] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[3] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[4] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt*," in *Proc. IEEE Int. Conf. Robot. and Automation*, 2011, pp. 1478–1483.

[5] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, vol. 104, no. 2, 2010.

[6] ——, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Proc. IEEE Conf. Decision Control*. IEEE, 2010, pp. 7681–7687.

[7] Y. Li, W. Wei, Y. Gao, D. Wang, and Z. Fan, "Pq-rrt*: An improved path planning algorithm for mobile robots," *Expert Systems with Applications*, vol. 152, p. 113425, 2020.

[8] L. Palmieri, S. Koenig, and K. O. Arras, "Rrt-based nonholonomic motion planning using any-angle path biasing," in *Proc. IEEE Int. Conf. Robot. and Automation*, 2016, pp. 2775–2781.

[9] J. Wang, M. Q.-H. Meng, and O. Khatib, "Eb-rrt: Optimal motion planning for mobile robots," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 2063–2073, 2020.

[10] F. Golbol, M. M. Ankarali, and A. Saranli, "Rg-trees: trajectory-free feedback motion planning using sparse random reference governor trees," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6506–6511.

[11] O. Arslan and D. E. Koditschek, "Sensor-based reactive navigation in unknown convex sphere worlds," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 196–223, 2019.

[12] S. Paternain, D. E. Koditschek, and A. Ribeiro, "Navigation functions for convex potentials in a space with convex obstacles," *IEEE Transactions on Automatic Control*, vol. 63, no. 9, pp. 2944–2959, 2017.

[13] O. Arslan and D. E. Koditschek, "Exact robot navigation using power diagrams," in *Proc. IEEE Int. Conf. Robot. and Automation*, 2016, pp. 1–8.

[14] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances in applied mathematics*, vol. 11, no. 4, pp. 412–442, 1990.

[15] E. Rimon, "Exact robot navigation using artificial potential functions," Ph.D. dissertation, Yale University, 1990.

[16] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.

[17] D. Koditschek, "Exact robot navigation by means of potential functions: Some topological considerations," in *Proc. IEEE Int. Conf. Robot. and Automation*, vol. 4. IEEE, 1987, pp. 1–6.

[18] J. V. Gómez, A. Lumbier, S. Garrido, and L. Moreno, "Planning robot formations with fast marching square including uncertainty conditions," *Robotics and Autonomous Systems*, vol. 61, no. 2, pp. 137–152, 2013.

[19] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sums-of-squares verification," *Int. J. Robot. Res.*, vol. 29, no. 8, pp. 1038–1052, 2010.

[20] J. J. Park and B. Kuipers, "A smooth control law for graceful motion of differential wheeled mobile robots in 2d environment," in *Proc. IEEE Int. Conf. Robot. and Automation*, 2011, pp. 4896–4902.

[21] ——, "Feedback motion planning via non-holonomic rrt* for mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, 2015, pp. 4035–4040.

[22] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International journal of robotics research*, vol. 34, no. 7, pp. 883–921, 2015.

[23] M. Otte and E. Frazzoli, "Rrtx: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 797–822, 2016.

[24] G. Vailland, V. Gouranton, and M. Babel, "Cubic bézier local path planner for non-holonomic feasible and comfortable path generation," in *ICRA 2021-IEEE International Conference on Robotics and Automation*, 2021.

[25] C. Lau and K. Byl, "Smooth rrt-connect: An extension of rrt-connect for practical use in robots," in *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, 2015, pp. 1–7.

[26] W. G. Aguilar, S. Morales, H. Ruiz, and V. Abad, "Rrt* gl based optimal path planning for real-time navigation of uavs," in *International Work-Conference on Artificial Neural Networks*. Springer, 2017, pp. 585–595.

[27] X. Lan and S. Di Cairano, "Continuous curvature path planning for semi-autonomous vehicle maneuvers using rrt," in *2015 European Control Conference (ECC)*, 2015, pp. 2360–2365.

[28] H.-T. L. Chiang and L. Tapia, "Colreg-rrt: An rrt-based colregs-compliant motion planner for surface vehicle navigation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2024–2031, 2018.

[29] T. T. Enevoldsen, C. Reinartz, and R. Galeazzi, "Colregs-informed rrt* for collision avoidance of marine crafts," *arXiv preprint arXiv:2103.14426*, 2021.

[30] V. Parque and T. Miyashita, "Smooth curve fitting of mobile robot trajectories using differential evolution," *IEEE Access*, vol. 8, pp. 82 855–82 866, 2020.

[31] A. Zdevsar and I. vSkrjanc, "Optimum velocity profile of multiple bernstein-bézier curves subject to constraints for mobile robots," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 9, no. 5, pp. 1–23, 2018.

[32] T. Jusko and E. Stoll, *Scalable Trajectory Optimization Based on Bézier Curves*. Deutsche Gesellschaft für Luft-und Raumfahrt-Lilienthal-Oberth eV, 2016.

[33] A. Robotics, "Cassie Simulators," http://www.agilityrobotics.com/sims/, 2018.

[34] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, 2009.

[35] J.K. Huang and Jessy W. Grizzle, "omni-directional CLF Reactive Planning System for tough terrains," 2020. [Online]. Available: https://github.com/UMich-BipedLab/CLF_motion_planning

[36] Y. Gong and J. Grizzle, "Angular momentum about the contact point for control of bipedal locomotion: Validation in a lip-based controller," *arXiv preprint arXiv:2008.10763*, 2020.

[37] S. Kajita and K. Tani, "Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 1991, pp. 1405–1411 vol.2.

[38] R. Blickhan, "The spring-mass model for running and hopping," *Journal of Biomechanics*, vol. 22, no. 11, pp. 1217–1227, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0021929089902248

[39] J. Grizzle, G. Abba, and F. Plestan, "Asymptotically stable walking for biped robots: analysis via systems with impulse effects," *IEEE Transactions on Automatic Control*, vol. 46, no. 1, pp. 51–64, 2001.

[40] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 295–302.

[41] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[42] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Proc. IEEE Conf. Decision Control*, 2014, pp. 6271–6278.

[43] Y. Chen, H. Peng, and J. Grizzle, "Obstacle avoidance for low-speed autonomous vehicles with barrier function," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 1, pp. 194–206, 2018.

[44] Q. Nguyen, A. Hereid, J. W. Grizzle, A. D. Ames, and K. Sreenath, "3d dynamic walking on stepping stones with control barrier functions," in *Proc. IEEE Conf. Decision Control*, 2016, pp. 827–834.

[45] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015.

[46] Q. Nguyen, X. Da, J. Grizzle, and K. Sreenath, "Dynamic walking on stepping stones with gait library and control barrier functions," in *Algorithmic Foundations of Robotics XII*. Springer, 2020, pp. 384–399.

[47] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart, "Robot-centric elevation mapping with uncertainty estimates," in *International Conference on Climbing and Walking Robots (CLAWAR)*, 2014.

[48] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3019–3026, 2018.

[49] L. Gan, R. Zhang, J. W. Grizzle, R. M. Eustice, and M. Ghaffari, "Bayesian spatial kernel smoothing for scalable dense semantic mapping," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 790–797, April 2020.

[50] F. Blochliger, M. Fehr, M. Dymczyk, T. Schneider, and R. Siegwart, "Topomap: Topological mapping and navigation based on visual slam maps," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3818–3825.

[51] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, "Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4304–4311.

[52] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1280–1286.

[53] L. Oth, P. Furgale, L. Kneip, and R. Siegwart, "Rolling shutter camera calibration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1360–1367.

[54] J. Huang and J. W. Grizzle, "Improvements to Target-Based 3D LiDAR to Camera Calibration," *IEEE Access*, vol. 8, pp. 134 101–134 110, 2020.

[55] J. K. Huang, S. Wang, M. Ghaffari, and J. W. Grizzle, "LiDARTag: A Real-Time Fiducial Tag System for Point Clouds," *IEEE Robotics and Automation Letters*, pp. 1–1, 2021.

[56] J.-K. Huang, C. Feng, M. Achar, M. Ghaffari, and J. W. Grizzle, "Global Unifying Intrinsic Calibration for Spinning and Solid-State LiDARs," *arXiv preprint arXiv:2012.03321*, 2020.

[57] J.K. Huang and Jessy W. Grizzle, "Extrinsic LiDAR Camera Calibration," 2019. [Online]. Available: https://github.com/UMich-BipedLab/extrinsic_lidar_camera_calibration

[58] Jiunn-Kai Huang, Shoutian Wang, Maani Ghaffari, and Jessy W. Grizzle, "LiDARTag ROS Package," 2020. [Online]. Available: https://github.com/UMich-BipedLab/LiDARTag

[59] J.K. Huang, C Feng, M. Achar, M. Ghaffari and Jessy W. Grizzle, "Intrinsic LiDAR Calibration," 2019. [Online]. Available: https://github.com/UMich-BipedLab/LiDAR_intrinsic_calibration

[60] R. Hartley, M. G. Jadidi, J. Grizzle, and R. M. Eustice, "Contact-aided invariant extended Kalman filtering for legged robot state estimation," in *Proc. Robot.: Sci. Syst. Conf.*, Pittsburgh, Pennsylvania, June 2018.

[61] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle, "Contact-aided invariant extended kalman filtering for robot state estimation," *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 402–430, 2020.

[62] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[63] Y. Gong and J. Grizzle, "Zero dynamics, pendulum models, and angular momentum in feedback control of bipedal locomotion," 2021.
[64] Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, "Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 4559–4566.
[65] "The Wave Field on the North Campus of the University of Michigan," https://arts.umich.edu/museums-cultural-attractions/wave-field/.
[66] "ICRA 2021 Workshop on Legged Robots (Towards Real-World Deployment of Legged Robots)," https://youtu.be/0Gg8BTs6HLY, accessed: 2021-06-10.
[67] J. Huang. (2021) Fully Autonomous on the Wave Field 2021. https://youtube.com/playlist?list=PLFe0SMV3hBCAYMpChjW_-LQ71PNfieWAe.