Path planning for terrain of steep incline using Bézier curves

Georgios Kamaras, Panagiotis Stamatopoulos
Department of Informatics and Telecommunications
National and Kapodistrian University of Athens
Athens, Greece
georgioskamaras@gmail.com, takis@di.uoa.gr

Stasinos Konstantopoulos
Institute of Informatics and Telecommunications
NCSR "Demokritos"
Athens, Greece
konstant@iit.demokritos.gr

***Abstract*—In order to navigate rugged, sloped, slippery, or, in general, challenging terrains, a robot needs to occasionally adapt its path plan in order to accommodate particular circumstances. Steep inclines are a prime example of situations where a non-shortest path plan can dramatically increase a mobile platform's ability to operate. In this paper, we present a path planning method that produces smooth curves that are longer than a straight line to the goal, but are designed to be optimal (not longer than necessary) for a given maximum inclination that the robot platform is capable of, while simultaneously taking into consideration the obstacles in the field.**

***Keywords*-Unmanned Ground Vehicle, Offline Path Planning, Incline Terrain, Bézier Curves, Evolutionary Algorithm**

I. Introduction

Traversing challenging terrains is one of the core goals of field robotics research and can greatly improve the scope of application of outdoors rovers. Traversing steep inclines is well suited to research in robot autonomy and intelligent behavior: improving path planning can offer a great increase in the maximum inclination that can be navigated without modifying the underlying mechatronics and motor control.

Traversing steep inclines has received considerable research attention regarding the hardware design of wheeled [1], [2], legged [3] and crawling [4] platforms, and the motor control [5]–[7]. To the best of our knowledge, there is no literature on path planning that is more sophisticated than shortest-path optimization for increased steep incline traversability. Non-shortest path planning has only been explored for skid-steered rovers traversing loose or slippery terrain, where paths constructed from circular arcs have been proven more energy-efficient than the shortest path [8], [9].

We explore the idea of using Bézier curves to construct paths that follow the zig-zag pattern that is characteristic of mountain roads. Being constructed by concatenating curves, these paths avoid sharp turns that exceed skid-steered rovers' capabilities. At the same time, these paths are optimized for a given field configuration, taking into account both the inclination and the obstacles present in the field. In this paper, we first provide the background on path optimization and on using Bézier curves for path planning (Section II), and then we present our method (Section III), experimental results (Section IV), and conclusions (Section V).

This research was conducted by G. Kamaras, while at NCSR "Demokritos" as a part of his Undergraduate Thesis, under the supervision of S. Konstantopoulos and P. Stamatopoulos.

II. Background

A. Optimization algorithms

In path planning, the area between the current position and the goal is typically seen as a graph of waypoints (vertices) connected with edges labelled with the cost of moving between these two waypoints. This graph can derive from the grid map typically used to represent traversability: first connect adjacent grid cells when not occupied by obstacles, then weight the connection using the traversability value of the landing cell. To plan a path from its current position to the goal, the robot needs to find a path through this graph that minimizes the total cost of the edges along the path.

Steepest Ascent Hill Climbing (SAHC) algorithms are widely used to reach a goal state from an initial state due to their simplicity and computational efficiency. As pure SAHC may be unable to backtrack from solutions that lead to collision with an obstacle, we also experiment with its *N-Best* variant. N-Best is able to plan paths around smaller obstacles, but may fail for large obstacles where it cannot backtrack deep enough.

Obstacles are discontinuities in the cost heuristic that SAHC algorithms do not take into account, until they are forced to backtrack. So, a natural choice for an algorithm to compare against SAHC is *Evolutionary Algorithms (EA)* that evaluate and iteratively improve complete solutions. Mapping EA strengths and limitations [10] to our specific problem, we note that the crossover and mutation operations often produce paths that are impossible to realise for a given platform's dynamics and their detection and elimination adds a computational overhead by comparison to SAHC. EAs also evaluate a far larger set of candidates than SAHC; although this is mitigated by the fact that in out setup candidate solutions are very efficiently computed.

B. Producing smooth curves

Our path planning problem is similar to the problem of designing a smooth curve between two endpoints, such that it balances between being smooth and closely following a

2375-0197/20/$31.00 ©2020 IEEE
DOI 10.1109/ICTAI50040.2020.00026

given list of *control points* that outline the curve. The most prominent approach in the computer graphics literature is to use Bézier curves [11].

Complex multi-control point curves can be produced either by higher-order curves, or by stitching together lower-order curve segments. The robot motion planning literature has explored their behavior, although not very extensively. Miller et al. [12] use cubic Bézier curves in a sensorimotor loop, in order to generate paths that are consistent with a vehicle's dynamics. Choi et al. [13] present two path planning algorithms, leveraging cubic Bézier curve properties in order to efficiently generate optimized smooth paths satisfying corridor constraints. They also discuss the further optimization of the resulting paths for user-defined cost functions. Lau et al. [14] showcase quintic Bézier splines effectiveness for time-optimal motion planning in an environment populated with obstacles, through the optimization of a straight-line path. Qian et al. [15] adopt a path-velocity decomposition approach and present a motion planning pipeline in which quintic Bézier curves are used to smoothen a collision-free piecewise linear path. Velocity planning follows path planning to accommodate user preferences and task-specific requirements. Tharwat et al. [16] recognise the influence of Bézier curve control points in the overall optimally of a path planning model based on Bézier curves. Assuming a two-dimensional static environment, they propose two variants of the Chaotic Particle Swarm Optimization evolutionary algorithm in order to search for an optimal path from a start to a goal position, avoiding lethal obstacles along the way.

In the work described here, we explore stitching *quadratic Bézier curves*, the simplest Bézier curves that are defined by their endpoints and a single control point. By doing this, we aim at a behavior that is more appropriate for physical motion: high-order Bézier curves require a larger number of more abrupt changes in direction, which is very impractical when planning the motion of robot platforms. Our approach explores moving the burden of optimization from configuring fewer, longer, and more complex segments to stitching a larger number of shorter and simpler segments. Our approach also makes the application of EA viable, as the low cost for computing candidate solutions mitigates the larger number of candidates needed by these algorithms.

III. Optimizing Curve Parameters

A. Assumptions and Representation

We consider a robot able to estimate the inclination, an ability well-established in the traversability estimation literature [17], and to identify and localize obstacles. We also assume that these obstacles are static, so the full path is decided before the robot starts moving. Should new obstacles appear or become visible, the path will need to be re-computed. Finally, we assume a standard navigation system that implements paths represented as series of waypoints and that, by nature, any quadratic Bézier segment can be implemented by skid-steered rovers.

In this setup, our method processes a representation of the terrain and the obstacles in it, an admissible *initial state*, and an admissible *goal state*; and produces a *path* (i.e., a series of waypoints) starting from the initial state and leading to the goal state.

We diverge from the path planning literature by not searching for an optimal series of waypoints, but rather for an optimal series of Bézier segments, each defined by the endpoints and the control point. That is, the search space is the space of the parameters of Bézier segments and not the space of waypoints. The actual path passed to the navigation system is then computed from the quadratic Bézier formula.

B. Path Cost

Regarding the heuristic that guides the search for an optimal solution:

Bias towards shorter paths: We include a term that depends on the distance d_x between point x and the straight line connecting the initial position and the goal, normalized to the length of this straight line. Steeper slopes should tolerate greater deviations from the shortest path, so the term is $k_1 d_x/s$, where k_1 is a constant and s is the overall slope.

Bias towards smooth paths: We prefer solutions that make smooth transitions between curves, minimize abrupt changes of direction when changing from one curve into the next, and also minimize the number of zig-zags needed to reach the goal. For each waypoint, this can be expressed as maximizing the angle formed by connecting three consecutive waypoints, so that it is as close as possible to $180°$. However, this is risky for steeper slopes, so the term expressing this bias is $k_2 s/a_x$ where k_2 is a constant and a_x is the angle formed by connecting x with the two previous waypoints.

Pitch and roll: Note how higher pitch values make a path riskier, while higher roll values make it safer: When a wheeled platform moves in an incline terrain, higher pitch values translate into moving straight towards the top, whereas high roll values translates into moving along a horizontal traverse. Bias towards low-pitch high-roll routes is necessary to balance the shortest-path bias above. We express the above reasoning as $k_3(p_x - r_x)$, where k_3 is a constant and p_x, r_x the pitch and roll at x.

Summing the above over all points in a path P, we define:

$$\text{cost}(P) = \sum_{x \in P} k_1 \frac{d_x}{s} + k_2 \frac{s}{a_x} + k_3(p_x - r_x)$$

We have empirically found $k_1 = 157.5, k_2 = 0.1, k_3 = 10$ to behave well with s, a_x, p_x, r_x expressed in degrees. The summation can be made over denser or sparser point samples. We have experimentally found that as few as 5 points per Bézier curve converge to a cost value that is practically identical to what a denser calculation would yield.

C. Optimal path generation

As discussed in Section II, generating a path can be formulated as searching for the optimal traversal of the graph that connects the cells of the grid map that are reachable from each other. The start position is P_0 of the first local Bézier curve. We search the incline area's grid by taking two rows at each step. In the SAHC generator, we greedily try to place the control point P_c, which is not part of the local path, in the first row and the end point P_1 in the second row. First, we calculate each local Bézier curve's total cost. Then, we examine all the possible local Bézier curves, we select the locally optimal Bézier curve and we add it to the global Bézier path. In the next step, we assign the P_1 of the latest local Bézier curve to be the P_0 of the next local Bézier curve and we repeat the same process. When we reach the last row of the grid, we assign P_1 to the goal position. This way, we efficiently create a globally "good-enough" Bézier path, by combining locally optimal decisions.

We expect SAHC generator to be the most time efficient of our three generators, but also the most naive, as it fails in the presence of a lethal obstacle in a locally optimal solution. N-Best's time efficiency is expected to be close to the SAHC. However, the N-Best generator is less naive, as it can successfully handle simple lethal obstacles. Our EA generator terminates either when our search has reached a designated threshold of examined generations, or when our search has succeeded in producing an admissible path from start to goal, or when a designated threshold of consecutive stagnated generations has been reached. A valid smooth curved path from start to goal is an individual, and each path's waypoints are the individual's chromosomes.

In Figure 1, we see the logic level results of using the N-Best and EA optimization methods for producing a path in order to climb a $45°$ and a $43°$ - 45m long slope respectively. The colored lines represent the paths that each algorithm considered. Some of them were deemed inadmissible because they coincide with an obstacle, others because they are not locally optimal. The path that our method finally selected is annotated with small yellow spheres. Each one of these yellow spheres is a waypoint of the selected path.

IV. Validation and Demonstration

A. Experimental setup

For the experimental validation of our work we used the Clearpath Husky UGV in simulation and we created three simulation environments: A 5.9m-long ramp at an inclination of $35°$, a 5.8m/$45°$ ramp (Figure 2), and a 45m/$43°$ ramp.[1]

The $35°$ slope has a reduced (< 1) friction coefficient in order to represent slippery terrain conditions, whereas the $43°$ and $45°$ slopes have a friction coefficient of 1.0. The $35°$ slope demonstrates in simulation, whether our method can currently help a UGV in a challenging real-world scenario to go further than it would have gone if it was climbing straight upward. The $45°$ slope is ideal for testing our methods, being steep enough for a wheeled UGV with no special equipment to be unable to traverse it in a straight path without tipping over, while not too steep so that the UGV will be unable to manoeuvre on it. The 45m/$43°$ slope environment was created having in mind a maximum range of about 25m for the localization sensors available in the market at the time of writing this; in this context, this environment tests our method in cases where the goal (and some of the obstacles) are beyond the robot's perception range.

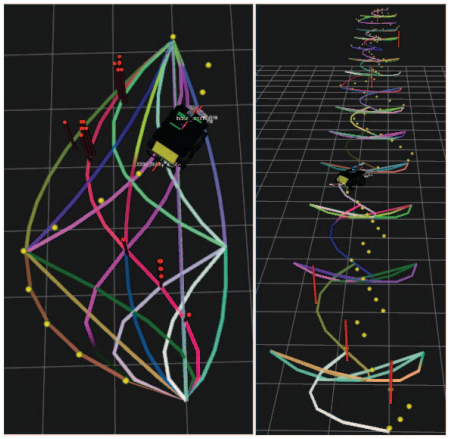

Figure 1: Candidates and selected path (dotted) for a $45°$ slope (left) and a longer $43°$ slope (right).

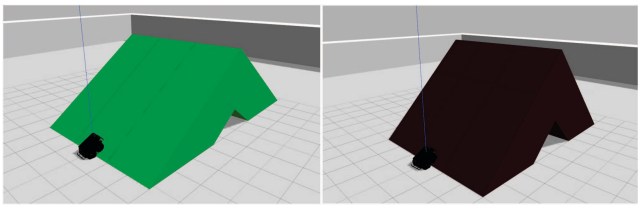

Figure 2: $35°$ (left) and $45°$ (right) slopes set-up.

For each slope, three scenarios have been created: one without obstacles, one with point obstacles, and one with sizeable obstacle formations. We expect (a) SAHC to be the most efficient method in the first scenario and often fail to find a solution in the other two; (b) N-Best to be able to backtrack out of the point obstacles in the second scenario but not all of the obstacles in the third scenario; and (c) EA will be able to find a path in all scenarios.

The *search resolution* or *search step* is kept at *1.5 meters* for all the simulation scenarios. This resolution was experimentally determined to be the best for our generators to search for a solution in our simulation scenarios, which are characterized by the narrowness of the area in which our robot can move. It is a small resolution considering that our reference UGV has a length of *0.3 meters*.

[1]The complete experimental setup is available at https://github.com/yorgosk/ugv_navigation_goals

Table I: Generated paths details.

Slope and distance	Generator	Visited States	Path Length	$\sum d_x$	Cost
$35°$, $5.9m$	SAHC	7	$8.03m$	$3.68m$	6627
	N-Best	7	$8.03m$	$3.68m$	6627
	EA	13	$7.52m$	$2.74m$	6649
$45°$, $5.8m$	SAHC	7	$7.00m$	$3.28m$	11107
	N-Best	7	$7.00m$	$3.28m$	10437
	EA	208	$6.15m$	$2.30m$	9575
$43°$, $45.0m$	SAHC	85	$57.08m$	$17.93m$	79481
	N-Best	85	$56.31m$	$21.69m$	81581
	EA	1508	$53.27m$	$28.32m$	60279

B. Simulation results

The robot fails to traverse the $35°$ slope in a straight upward path, never reaching its goal due to wheel slippage. The robot fails to traverse the $45°$ and 45m/$43°$ slopes going straight upward as it tips over. Now, we will present the results of simulations, demonstrating our generators' capabilities in helping a UGV traverse these slopes.

1) Generated paths quality: Table I presents sample details of the paths generated by our SAHC, N-Best and EA Bézier path generators for each simulation scenario. The $\sum d_x$ column presents the values of the total deviation of a planned path from its respective straight upward path.

The robot's effort to climb the $35°$ slope with a SAHC generated path is not successful compared to climbing straight upward. While the robot is manoeuvering to follow the curved path, it loses a substantial part of its achieved progress. From the attempt with an EA generated path, we further understand that the success of the robot traversing a slippery incline terrain will depend on the amount of time that it spends turning and moving away from a straight upward path. Too much turning will cause our robot to lose a substantial part of its progress and provide us with a result much worse than desired.

In the $45°$ slope environment, in all cases the robot manages to climb the incline terrain following a relatively smooth path, while maintaining high stability and showing no wheel slippage. It is also evident that in a logic level the robot manages to escape the obstacles on its way. However, we also see an error accumulating and creating a loss of precision when it comes to the actual obstacle avoidance. This is a localization error created by improper odometry whenever the UGV's front wheels, momentarily loose contact with the terrain. An EA generator's path can have almost thirty times the visited states compared to the SAHC and the N-Best generators (Table I). This is expected, as the crossovers that take place between generations are considered as visiting a new state. The path length is smaller than the length of our other two generators paths and its cost is considerably improved. The path's size can be smaller than the SAHC or N-Best generators or larger, depending on the route that will derive from the evolutionary process.

In each 45m/$43°$ slope scenario, the robot manages to climb the slope with its overall generated path quality prevailing over the accumulating localization error. Again, in a logic level, the robot avoids every lethal obstacle, however this is not visible while observing the plan's execution because of the accumulating localization error. As desired, there are no signs of wheel slippage. For the N-Best and EA generated paths the observations made for the $45°$ slope scenario, still hold true.

Table II: EA consistency in path production for the $45°$ slope scenario under different configurations (*Limited*, *Modest* and *Extensive* search).

Configuration	Limited	Modest	Extensive
Init. Gen. Size	25	50	100
Max. Generations	50	100	800
Similarity	$9/20$	$14/20$	$18/20$
Consistency	45%	70%	90%

Table III: Comparison of SAHC's, N-Best's and EA's average execution times under the same conditions, with no obstacles (no obst.), simple point obstacles (pnt. obst.), or complex obstacle formations (obst. for.), and in the case of EA for modest (mod.) and extensive (ext.) search.

Slope & Length	$35°$, $5.9m$	$45°$, $5.8m$	$43°$, $45.0m$
Start-Goal Distance	$5.80m$	$5.71m$	$44.0m$
SAHC	$0.71ms$	$0.88ms$	$57.52ms$
EA mod. no obst.	$5.33ms$	$5.24ms$	$20.85ms$
EA ext. no obst.	$28.08ms$	$29.15ms$	$48.73ms$
N-Best	$0.70ms$	$0.75ms$	$31.17ms$
EA mod. pnt. obst.	$5.37ms$	$5.56ms$	$22.17ms$
EA ext. pnt. obst.	$27.51ms$	$28.48ms$	$33.12ms$
EA mod. obst. for.	$6.18ms$	$6.22ms$	$29.76ms$
EA ext. obst. for	$34.12ms$	$58.38ms$	$79.47ms$

2) EA performance: A common issue of the EAs is their instability. Being probabilistic by nature, EAs may be inconsistent in their output when run given a certain input. However, Tables II and III demonstrate that a well-designed EA can have configurations for which it performs steadily while being time efficient. It does so by producing *Similarly* "viable" paths *Consistently*, i.e. for a high percentage of consecutive runs. In all three configurations, the two *Best Fit* paths of each generation survive to the next one. The evolutionary process is also set to terminate after six *Stagnated Generations*. We consider a generation to be stagnated if its fittest individual, which is its minimum-cost path, represents a *Stagnation Rate* of 0.01, meaning a $\leq 1\%$ improvement compared to the fittest individual of the previous generation.

3) Time efficiency: Table III presents the average execution times recorded by sample runs of our three path generators under certain conditions. All runs were executed on an Intel© Core™ i7-4710MQ CPU at 2.50GHz, 5.7 GiB RAM and 64-bit Ubuntu 16.04 LTS with ROS Kinetic.

We run the SAHC generator considering the absence of lethal obstacles, the N-Best considering the existence of simple obstacles and the EA considering the existence

of complex obstacles. The EA generator can also be run with no obstacles and with simple obstacles and, being our most advanced generator, it is useful to examine how it performs under such conditions. Furthermore, as the EA is probabilistic, it is important to examine its performance both for a modest intensity and for an extensive search.

Evidently, each path generator's execution time is determined by the combination of the straight-line distance between start and goal with the presence of obstacles and the algorithm's search configurations. For the N-Best generator, the presence of obstacles can indirectly result to the "trimming" of the search space, reducing the execution time. The EA's execution time depends on its probabilistic evolutionary process and the size of the search space. However, its execution time remains stable for similar scenarios. Lastly, we understand that the EA scales up much better than the Hill Climbing and N-Best algorithms, because, as the complexity of its task grows, the computing time required grows in a decreasing rate. Moreover, if safety is not an immediate concern, bounding the size of the EA's search space can result in an even faster path generation, which, for complex problems, can be much faster than the generation process of a SAHC or an N-Best algorithm.

In sum, for less complex scenarios the greedy SAHC and N-Best generators are recommended, because of their precision and efficiency. As the complexity of the scenarios grows, the probabilistic EA generator becomes increasingly preferable, because it scales up better. This can be attributed to the EA's probabilistic nature, which, in large scales, is much more efficient than exhaustive searches.

V. Conclusion

Our main contribution is proposing quadratic Bézier curves as the building component of a smooth path for a three-dimensional path planning problem, like traversing incline terrains. Following the state of the art in path planning with Bézier curves [16], we propose that the optimization algorithm is not applied to the path waypoints, but to the search space of the Bézier lines' control points, which determine the line's curvature. This search space exploits the inherent properties of Bézier curves, ensuring that all candidate paths are implementable by skid-steered rovers, and avoid obstacles while progressing towards the goal. A further contribution is testing prominent algorithms from the optimization literature, and providing results which strongly suggest that the EAs operate best in this search space.

As further research, we plan to explore the interaction between inclination and other characteristics of challenging terrains: rugged and uneven terrains might capsize a platform even when the inclination is relatively small. To address this, we will involve all three traversability parameters that can be remotely sensed (slope, height, ruggedness) and not only slope, so that the planned path is safe with respect to the overall traversability [17]. We also plan to extend the scope of planning to output the speed of the movement, so a shorter/slower path is more accurately compared against a longer/faster one.

References

[1] P. McGarey, D. Yoon, T. Tang, F. Pomerleau, and T. D. Barfoot, "Developing and deploying a tethered robot to map extremely steep terrain," *J Field Robotics*, vol. 35, no. 8, 2018.

[2] C.-T. Chen, C.-C. Feng, and Y.-A. Hsieh, "Design and realization of a mobile wheelchair robot for all terrains," *Journal of Advanced Robotics*, vol. 17, no. 8, pp. 739–760, 2003.

[3] K. Inoue and M. Kaminogo, "Steep slope climbing using feet or shins for six-legged robots," in *Proceedings of the 10th Asian Control Conference (ASCC 2015)*, 2015.

[4] J. Ge, A. Calderón, and N. Pérez-Arancibia, "An earthworm-inspired soft crawling robot controlled by friction," in *Proc. IEEE Intl Conf. on Robotics and Biomimetics (ROBIO)*, 2017.

[5] D. Dunlap, W. Yu, E. G. Collins, and C. V. Caldwell, "Motion planning for steep hill climbing," in *Proc. 2011 IEEE Intl Conf. Robotics and Automation (ICRA)*. IEEE, 2011.

[6] C. Ordonez, N. Gupta, O. Chuy, and E. G. Collins, "Momentum based traversal of mobility challenges for autonomous ground vehicles," in *Proc. 2013 IEEE Intl Conf. Robotics and Automation (ICRA)*. IEEE, 2013.

[7] N. Gupta, C. Ordonez, and E. G. Collins, "Dynamically feasible, energy efficient motion planning for skid-steered vehicles," *Autonomous Robots*, vol. 41, no. 2, 2017.

[8] M. Effati and K. Skonieczny, "Circular ARC-based optimal path planning for skid-steer rovers," in *Proc. IEEE Canadian Conf. Electrical and Computer Engineering (CCECE)*, 2018.

[9] M. Effati, J.-S. Fiset, and K. Skonieczny, "Considering slip-track for energy-efficient paths of skid-steer rovers," *Journal of Intelligent & Robotic Systems*, pp. 1–14, 2020.

[10] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *Proc. 2016 Intl Conf. on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*. IEEE, 2016, pp. 261–265.

[11] Wolfram Mathword, "Bézier Curve," http://mathworld.wolfram.com/BezierCurve.html, last accessed July 2020.

[12] I. Miller, S. Lupashin, N. Zych, P. Moran, B. Schimpf, A. Nathan, and E. Garcia, "Cornell University's 2005 DARPA Grand Challenge entry," in *The 2005 DARPA Grand Challenge*. Springer, 2007, pp. 363–405.

[13] J. Choi, R. Curry, and G. Elkaim, "Path planning based on Bézier curve for autonomous ground vehicles," in *Advances in Electrical and Electronics Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*. IEEE, 2008, pp. 158–166.

[14] B. Lau, C. Sprunk, and W. Burgard, "Kinodynamic motion planning for mobile robots using splines," in *IEEE/RSJ Intl Conference on Intelligent Robots and Systems*. IEEE, 2009.

[15] X. Qian, I. Navarro, A. de La Fortelle, and F. Moutarde, "Motion planning for urban autonomous driving using Bézier curves and MPC," in *Proc. 19th IEEE Intell. Transportation Systems Conf. (ITSC)*. IEEE, 2016, pp. 826–833.

[16] A. Tharwat, M. Elhoseny, A. E. Hassanien, T. Gabel, and A. Kumar, "Intelligent Bézier curve-based path planning model using chaotic particle swarm optimization algorithm," *Cluster Computing*, vol. 22, no. 2, pp. 4745–4766, 2019.

[17] M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krüsi, R. Siegwart, and M. Hutter, "Navigation planning for legged robots in challenging terrain," in *Proc. 2016 IEEE/RSJ Intl Conference on Intelligent Robots and Systems (IROS 2016), Daejeon, South Korea*, Oct. 2016.