Motion Planning for Multi-legged Robots using Levenberg-Marquardt Optimization with Bézier Parametrization

David Valouch   Jan Faigl

***Abstract*— This paper presents a novel formulation of motion planning for multi-legged walking robots. In the proposed method, a single-step motion is formulated as a nonlinear equation problem (NLE): including kinematic, stability, and collision constraints. For the given start and goal configurations, the robot's path is parametrized as Bézier curve in the configuration space. The resulting NLE is solved using Levenberg-Marquardt optimization implemented using a sparse matrix solver. We propose handling the trigonometric kinematic constraints with the polynomial path parametrization. A relaxation of the constraint is used while guaranteeing a desired tolerance along the planned path. Although the proposed method does not explicitly optimize any criterion, it produces high-quality paths. The method is deployed in transforming a sequence of discrete configurations produced by a step sequence planner into a valid path for a multi-legged walking robot in challenging planning scenarios where a regular locomotion gait cannot be used because of sparse footholds.**

I. Introduction

Motion planning for multi-legged robots [1], [2] can be considered challenging because of many controllable degrees of freedom. In addition to collision constraints, multi-legged robots are subject to kinematic and stability constraints [3], [4]. Moreover, the constraints change as the robot takes steps resulting in a multimodal constrained planning problem [5], [6]. Although popular approaches to motion planning are based on a random sampling of the configuration space or control space [7]–[9], optimization-based approaches have also been successfully deployed [2], [10], [11] that are reminiscent of the early motion planning using potential functions [12].

We focus on motion planning for multi-legged walking robots, such as the SCARAB II hexapod robot [13], depicted in Fig. 2a. In particular, we investigate scenarios where a correct sequence of steps is necessary for traversing challenging structured terrain with limited footholds as depicted in Fig. 3. In our previous work [14], [15], we address finding the sequence of steps to cross a gap with limited footholds. The sequence of steps is represented by discrete configurations that subsequently need to be connected with continuous motions that satisfy the required constraints. Since single-step motions do not require solving maze-like obstacle avoidance, we opt for a local optimization technique as motion planning.

The authors are with the Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, Czechia. {valoudav, faiglj}@fel.cvut.cz

The presented work has been supported by the Czech Science Foundation (GAČR) under the research project No. 21-33041J.

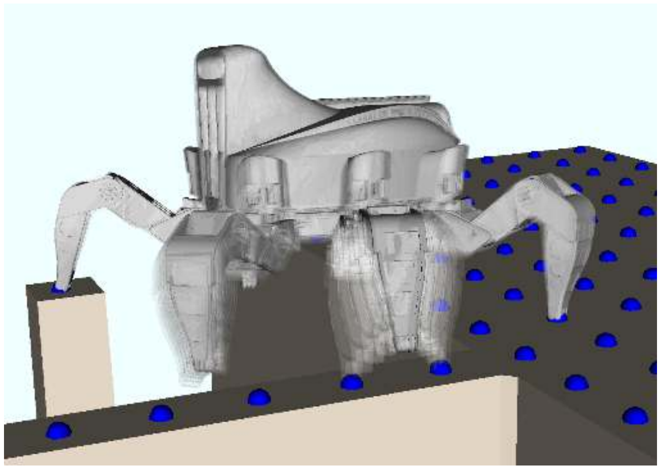


Fig. 1. Visualization of the planned motion of a legged robot.

The former work [14] is based on the Levenberg-Marquardt method to find valid configurations validating the feasibility of individual steps. In this work-in-progress report, we present an extension of the method to determine a valid path connecting two configurations while satisfying the motion constraints. Although we focused solely on constraint satisfaction, the developed motion planner yields relatively high-quality, smooth paths. Our results support a similar observation by [8] that a well-formulated simple method can produce near-optimal solutions without explicitly optimizing any criterion.

The rest of the paper is organized as follows. The studied problem is formulated in the following section. The proposed method is described in Section III. The validation results are reported in Section IV. The paper is concluded in Section V.

II. Problem Statement

Let $\mathcal{C}$ be the configuration space of the multi-legged robot. Specifically for a robot with only revolute joints, $\mathcal{C}$ can be defined as

$$\mathcal{C} = SE(3) \times SO(2)^{\text{CDOF}}\,, \tag{1a}$$

$$SE(3) \equiv \mathbb{R}^6\,,\ SO(2) \equiv \mathbb{R}\,, \tag{1b}$$

where CDOF is the number of controllable degrees of freedom. The six-legged walking robot SCARAB II has three revolute joints on each leg. Hence, with the pose of the base link, it has 24 degrees of freedom (DOF) in total, and its configuration is isomorphic with $\mathbb{R}^{24}$.

Now assume we have obtained a sequence of discrete configurations

$$q_1, q_2, \ldots, q_k \in \mathcal{C} \tag{2}$$

as a result of step-sequence planning, such as [4], [6], [14], [15]. In our case, the sequence consists of configurations in which the robot's leg switches between swinging and supporting state. Thus, the sequence represents the steps of the walking robot.

Then, for every pair of two consecutive configurations $q_i, q_{i+1} = q_{\text{start}}, q_{\text{end}}$, we aim to find a path π connecting them that can be defined as

$$\pi : [0,1] \to \mathcal{C} \,, \tag{3a}$$
$$\pi(0) = q_{\text{start}} \,,\; \pi(1) = q_{\text{end}} \,. \tag{3b}$$

The motion is subject to holonomic constraints that can be expressed as

$$f(\pi(t)) = \mathbf{0} \;:\; \forall t \in [0,1] \,, \tag{4a}$$
$$g(\pi(t)) \leq \mathbf{0} \;:\; \forall t \in [0,1] \,, \tag{4b}$$
$$g, f : \mathcal{C} \to \mathbb{R}^* \,. \tag{4c}$$

The equality constraint function f represents the kinematic constraints for the supporting legs that have to stay at the assigned footholds during the whole motion. The inequality constraint g enforces stability and collision-freeness. Note that we do not specifically define the collision-free subset $\mathcal{C}_{\text{free}}$, as collisions are covered by the inequality constraint (4b).

Further, we request the path π to be optimal under some quality criterion $\mathcal{Q} : \Pi \to \mathbb{R}$; here it is expressed using derivatives of π:

$$\mathcal{Q}(\pi) = \sum_{i=1}^{\infty} \int_{t=0}^{1} w_i \left\| \pi^{(i)}(t) \right\| dt \,, \tag{5}$$

where $w_i \geq 0$ are arbitrary weights, and $\pi^{(k)}$ represents the k-th derivative of π. The rationale behind the criterion is to minimize the length of the path: $\int \left\| \pi^{(1)}(t) \right\| dt$, and the complexity or 'roughness' of the path: contributing to the integral of the higher derivatives.

III. Planning Method

The proposed motion planning follows the problem formulation (3–5). We present the used parametrization of the path (3). Then, we formulate the utilized representation of the constraints (4). Finally, we describe the proposed algorithmic solution to the problem. Although the method is applied to a particular SCARAB II robot [13], which determines the used motion constraints, the method can be adapted to other planning scenarios with different robots.

A. *Bézier Curve Parametrization*

The proposed path π representation is based on Bézier curve [16] already used for robot motion parametrization [17], [18]. Bézier curve of the degree d is parametrized by $d+1$ control points $P = (p_0, p_1, p_2, \ldots p_d)$. Each point on the curve is a linear combination of the control points

$$\pi_d(P, t) = \sum_{i=0}^{d} \mathrm{B}_i^d(t)\, p_i \,, \tag{6}$$

where B_i^d is the i-th Bernstein polynomial of the degree d. In our case, the control points are vector representations of the robot's configurations $P = (q_{\text{start}}, q_1, \ldots, q_{d-1}, q_{\text{end}})$.

Further, we use the property that for any Bézier curve, an equivalent Bézier curve of a higher degree can be constructed as

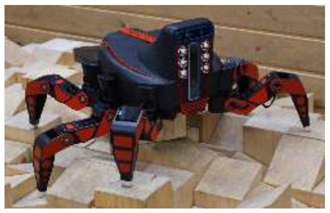
$$\pi_d(P_d, t) = \pi_{d+1}(P_{d+1}, t) \;;\; \forall t \in [0,1] \,, \tag{7a}$$
$$P_{d+1} = (p_0, \tilde{p_1}, \ldots, \tilde{p_d}, p_d) \,, \tag{7b}$$
$$\tilde{p_i} = \frac{i}{d+1} p_{i-1} + \left(1 - \frac{i}{d+1}\right) p_i \;;\; \forall i \in 1 \ldots d \,. \tag{7c}$$

Here, it is worth noting that the degree of Bézier curve can be used to limit the path complexity, an important rationale behind the proposed planning algorithm.

B. *Constraints*

The constraints considered for planning the steps of the SCARAB II walking robot are as follows.

1) The kinematic constraint ensures that the legs in the support phase remain in their assigned footholds during the planned step. The constraints function is formulated using the forward kinematics of the supporting legs using standard methods [19].
2) Stability constraint ensures the stability of the robot that we constrain the robot's center of mass to be above the support polygon, defined as the convex hull of the footholds. The constraint is a simple affine inequality constraint.
3) Collisions are distinguished as self-collisions and collisions with the terrain. For the SCARAB II robot, the motion ranges of the joints avoid self-collisions. The collisions with the terrain are modeled using a signed distance field (SDF) [20]. In particular, using implementation [21]. A simplified collision model consisting of spherical primitives is used (see Fig. 2b) to efficiently check robot collisions using the SDF. A sphere is in a collision if and only if the signed distance of its center is less than its radius. The collision constraint is relaxed around the footholds as in [14], allowing the foot tips to reach the terrain.


(a) The SCARAB II robot

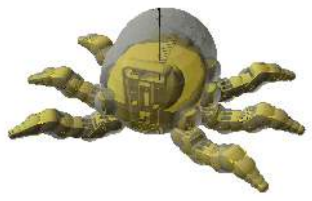

(b) Collision model

Fig. 2. The SCARAB II robot used for the evaluation of the proposed method, and its collision model using sphere primitives.

Note that, in our implementation, the inequality constraint (4b) is replaced by an equality constraint

$$\tilde{g}(\pi(t)) = \mathbf{0} \;;\; \forall t \in [0,1] \,, \tag{8a}$$
$$\tilde{g}(q) = \min(g(q), 0) \,. \tag{8b}$$

C. Path Constraint

The constraints (4a) and (8a) are used to form a path constraint

$$\mathcal{F}(\pi) = \int_{t=0}^{1} \|f(\pi(t))\|^2 + \|\tilde{g}(\pi(t))\|^2 \, dt = 0 \,. \qquad (9)$$

Computing (9) directly would be computationally intractable. However, in practice, the constraint functions are "well-behaved", with bounded derivatives. Therefore, the integral is approximated by a Riemann sum with a uniform partition of the size N:

$$\mathcal{F}(\pi) \approx \sum_{i=0}^{N} \|f(\pi(t_i))\|^2 + \|\tilde{g}(\pi(t_i))\|^2 \;,\; t_i = \frac{i}{N} \,. \qquad (10)$$

It is clear that (10) is 0, if and only if $\|f\|$ and $\|\tilde{g}\|$ are both 0 in all sample points, which is exploited in the proposed planning algorithm.

The function f represents the kinematic constraint and thus contains trigonometric terms. Therefore, a curve represented by a polynomial spline, such as Bézier curve, cannot fully satisfy it. Hence, we allow a tolerance $\|f\| \leq \varepsilon$ for a small $\varepsilon > 0$ to ensure compatibility with Bézier curve parametrization. It is achieved by introducing slack vectors ξ_i such that

$$\tilde{f}(\pi(t_i), \xi_i) = f(\pi(t_i)) - \xi_i = \mathbf{0} \,, \qquad (11a)$$

$$\|\xi_i\| \leq \varepsilon \,. \qquad (11b)$$

The inequality (11b) is included in $\tilde{g}$ in the same way as the inequality constraint (4b) is included in (8b).

Now, we can formulate the constraint for the Bézier parametrized path with the slack ε and sampling N as

$$\tilde{\mathcal{F}}_{\varepsilon,N}(P, \boldsymbol{\xi}) = \begin{bmatrix} \tilde{f}(\pi(P, t_1), \xi_1) \\ \vdots \\ \tilde{f}(\pi(P, t_N), \xi_N) \\ \tilde{g}_\varepsilon(\pi(P, t_1), \xi_1) \\ \vdots \\ \tilde{g}_\varepsilon(\pi(P, t_N), \xi_N) \end{bmatrix} = \mathbf{0} \,. \qquad (12)$$

D. Levenberg-Marquardt Method

Levenberg-Marquardt (LM) method, also called *damped least squares method* [22], is a more numerically stable version of the Newton-Raphson method for finding roots of nonlinear equations. It is used to solve nonlinear systems of equations in the form $h(\boldsymbol{\theta}) = \mathbf{0}$ by iteratively improving an initial guess for $\boldsymbol{\theta}$ by a step $\Delta\boldsymbol{\theta}$:

$$\Delta\boldsymbol{\theta} = (J_h(\boldsymbol{\theta})^T J_h(\boldsymbol{\theta}) + \lambda I)^{-1} J_h(\boldsymbol{\theta})^T h(\boldsymbol{\theta}) \,, \qquad (13)$$

where J_h is the Jacobian of h and $\lambda > 0$ is the damping parameter. The process is repeated until the solution error $\|h\|$ is reduced below numeric precision, $\|h\| < \epsilon_{\text{num}}$. Note that (13) minimizes the criterion

$$\|J_h(\boldsymbol{\theta})\Delta\boldsymbol{\theta} - h(\boldsymbol{\theta})\| + \sqrt{\lambda}\, \|\Delta\boldsymbol{\theta}\| \,. \qquad (14)$$

Furthermore, for $\lambda \to 0$, the LM method becomes equivalent to the Newton-Raphson method using the pseudo-inverse of the Jacobian.

The choice of the damping parameter λ regulates the convergence speed and numeric stability of the method. A dynamic damping strategy [23] is used, which updates λ at each iteration for better performance. The damping is reduced and increased by multiplying it by constants $\lambda_{\text{drop}} \in (0, 1)$ and $\lambda_{\text{boost}} > 1$, respectively. The LM algorithm is described in the SolveLM subroutine of Algorithm 1.

E. Planning Algorithm

The proposed step planning is summarized in Algorithm 1. It finds a path parametrized by the Bézier control points P satisfying the constraint $\tilde{\mathcal{F}}_{\varepsilon,N}$ for the given ε and N. Note that the path quality criterion (5) is tackled indirectly as we aim for a similar effect as in [8], where a "cost-indifferent" sampling-based path-planning algorithm produced paths of competitive quality to cost-aware algorithms. The proposed planning algorithm works as follows.

Algorithm 1: Plan Step

```
PlanStep($q_{\text{start}}, q_{\text{end}}, \varepsilon, N, d_{\max}$)
1  $\mathcal{P} \leftarrow (q_{\text{start}}, q_{\text{end}})$
2  for $d \leftarrow \{2, \dots, d_{max}\}$ do
3      $\mathcal{P} \leftarrow P_d$ (7) s.t. $\pi_d(P_d, t) = \pi_{d-1}(P, t)$ ; $\forall t \in [0, 1]$
4      $\boldsymbol{\xi} \leftarrow \mathbf{0}$
5      $\mathcal{P}, \boldsymbol{\xi} \leftarrow$ SolveLM($\tilde{\mathcal{F}}_{\varepsilon,N}, \mathcal{P}, \boldsymbol{\xi}$)
6      if $\left\|\tilde{\mathcal{F}}_{\varepsilon,N}(\mathcal{P}, \boldsymbol{\xi})\right\| \leq \epsilon_{num}$ then
7          END SUCCESS – return $\mathcal{P}$
8  END FAILURE

SolveLM($\tilde{\mathcal{F}}, \mathcal{P}, \boldsymbol{\xi}$)
1  $\lambda \leftarrow \lambda_{\text{init}}$
2  $\mathcal{P}^\star, \boldsymbol{\xi}^\star \leftarrow \mathcal{P}, \boldsymbol{\xi}$
3  for $i \leftarrow \{1, \dots,$ MaxIt$\}$ do
4      $\Delta\mathcal{P}, \Delta\boldsymbol{\xi} \leftarrow (J_{\tilde{\mathcal{F}}}^T J_{\tilde{\mathcal{F}}} + \lambda I)^{-1} J_{\tilde{\mathcal{F}}}^T \tilde{\mathcal{F}}(\mathcal{P}, \boldsymbol{\xi})$
5      if $\left\|\tilde{\mathcal{F}}(\mathcal{P}^\star + \Delta\mathcal{P}, \boldsymbol{\xi}^\star + \Delta\boldsymbol{\xi})\right\| < \left\|\tilde{\mathcal{F}}(\mathcal{P}^\star, \boldsymbol{\xi}^\star)\right\|$
        then
6          $\lambda \leftarrow \lambda\lambda_{\text{drop}}$
7          $\mathcal{P}^\star, \boldsymbol{\xi}^\star \leftarrow \mathcal{P} + \Delta\mathcal{P}, \boldsymbol{\xi} + \Delta\boldsymbol{\xi}$
8      else
9          $\lambda \leftarrow \lambda\lambda_{\text{boost}}$
10     if $\left\|\tilde{\mathcal{F}}(\mathcal{P}^\star, \boldsymbol{\xi}^\star)\right\| < \epsilon_{num}$ then
11         BREAK for
12 return $\mathcal{P}^\star, \boldsymbol{\xi}^\star$
```

The algorithm starts with a naive linear interpolation $P = (q_{\text{start}}, q_{\text{end}})$. We increase the degree d of Bézier curve at each step as in (7). The current P is then used as an initial guess for the solution to $\tilde{\mathcal{F}}_{\varepsilon,N}(P, \boldsymbol{\xi}) = \mathbf{0}$ in the LM method. If the constraint is satisfied, we find a valid path; otherwise, we continue with the increased d. By iterative increasing

d, the procedure ensures we minimize the degree of Bézier parametrization. Bézier curve of the degree d satisfies

$$\pi^{(i)} = \mathbf{0}\ ;\ \forall i \geq d\,. \tag{15}$$

Thus, we eliminate unnecessary terms from the path quality criterion (5).

Also, note that the LM algorithm only accepts steps that improve the objective. Even if the final iteration does not fully satisfy the constraint, we obtain a local optimum for the current d. The LM method minimizes the constraint function "*efficiently*," without unnecessary changes to the parameters. The pseudo-inverse method used in (13) produces the smallest step minimizing (14), and the term $\lambda \|\Delta\boldsymbol{\theta}\|$, in the criterion (14), further penalizes the change in the parameters. Our intuition is that, in the d-th iteration, Algorithm 1 makes a minimal necessary change mostly to the d-th derivative $\pi^{(d)}$ of the path. Therefore, it is indirectly optimizing (5). The resulting performance of the proposed method is reported in the following section.

IV. Results

The proposed planning method has been validated on a testing scenario with sequences of configurations generated by our step-sequence planner [14]. A sequence is planned for a challenging scenario, illustrated in Fig. 3, which is selected for demonstrating the ability of the planner to plan the individual steps. The robot is requested to cross a wide gap in the terrain using only a narrow beam and a single additional support. The modeled constraints are derived for the SCARAB II robot [13] depicted in Fig. 2a.

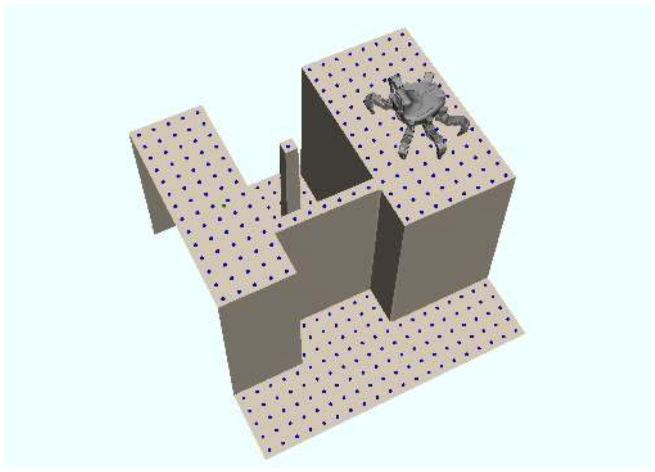


Fig. 3. A gap-crossing evaluation scenario, where only a singular pillar and a narrow beam can be used to cross the wide gap.

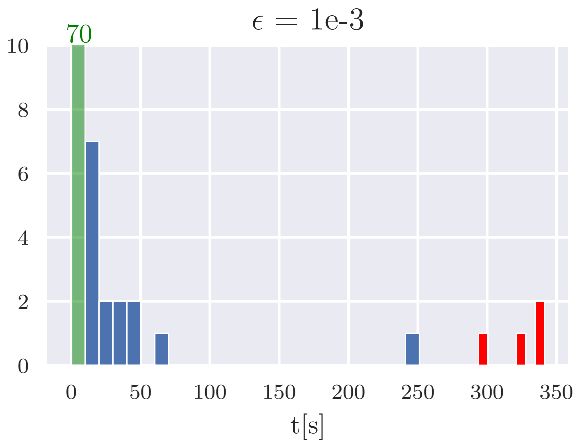
The proposed algorithm has been implemented in C++, compiled with GNU C++ compiler v9.4.0 with `-O3` level optimization. For computing the collision function, an implementation of the SDF provided with the *GridMap* library [21] is used. The Jacobian of the path constraint is constructed as a sparse matrix in the compressed column format using the *Eigen 3* linear algebra library [24]. The inversion of the matrix $J^T J + \lambda I$ in (13) is computed using LL^T Cholesky decomposition provided in *Intel® MKL Pardiso* sparse solver [25]. The method has been run on a computer with the Intel® i7-10700 processor running at 4.8 GHz and 64 GB RAM.

TABLE I

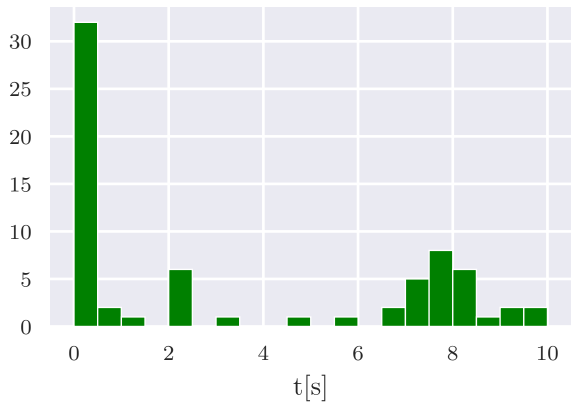
Parameters of the planning algorithm.

N	100
$d_{\max}$	10
`MaxIt`	100
λ_{init}	1
λ_{drop}	0.1
λ_{boost}	1.5
ε	1×10^{-3} m

The chosen values of the algorithm parameters are summarized in Table I. The selection of the values of the damping strategy parameters $\lambda_{\text{init}}, \lambda_{\text{drop}}, \lambda_{\text{boost}}$ is based on [23]. Values of the other parameters were chosen based on our practical experience.



(a) Histogram of the overall planning times; runs marked red failed to find a path satisfying the given tolerances. The green column represents runs under 10 s shown in detail in Fig. 4b.



(b) Detailed histogram of the planning times under 10 s

Fig. 4. Histograms of planning times for $\varepsilon = 10^{-3}$.

The results are summarized in Fig. 4 and Table II. In total, 89 steps have been planned. Most results have been obtained in less than 50 s, with most results in less than 10 s. The results found under 10 s are plotted in a separate histogram depicted in Fig. 4b to show the details of their distribution.

TABLE II

SUMMARY OF RESULTS

Steps planned	89
Steps succeeded	85
Worst constraint violation	$1.73 \times 10^{-3}\,\mathrm{m}$
Average planning time	$24.3\,\mathrm{s}$
Median planning time	$6.7\,\mathrm{s}$
Average, $t < 50\,\mathrm{s}$	$6.7\,\mathrm{s}$
Average, $1\,\mathrm{s} < t < 50\,\mathrm{s}$	$11.1\,\mathrm{s}$

The concentration of results near $0\,\mathrm{s}$ is caused by the steps that can be solved within the desired tolerance by a simple linear interpolation of $q_{\text{start}}, q_{\text{end}}$.

The average achieved planning time is $24.3\,\mathrm{s}$; however, the median planning time is only $6.7\,\mathrm{s}$, which corresponds to the average time if outliers above $50\,\mathrm{s}$ are excluded. Also, excluding the ultra-short planning times shorter than $1\,\mathrm{s}$, we obtain an average planning time $11.1\,\mathrm{s}$ for most non-trivial steps. It can be highlighted that the worst constraint violation across all configurations on a planned path is $1.73 \times 10^{-3}\,\mathrm{m}$, including the failed attempts. That is below the real-world accuracy of the robot's mechanical precision and the utilized localization using only the onboard sensors. We, therefore, believe that with further optimization, and relaxation of the tolerances, the method is viable for practical deployments.

An example of the generated motion is visualized in Fig. 1. Note that the path taken by the robot's leg is not produced by any motion primitive. It results from the numeric solver satisfying the collision constraint with a $2\,\mathrm{cm}$ collision margin locally relaxed around the footholds [14]. The motion results from satisfying the constraints by iteratively increasing the degree of the polynomial representation and alternating with a run of the LM solver.

V. CONCLUSION

We present our work-in-progress on optimization-based motion planning that targets single-step motions for multi-legged walking robots. The planner produces paths fitted to the submanifold defined by the kinematic constraint of the supporting legs up to a defined tolerance while satisfying stability and collision-freeness criteria. The method has been validated in a challenging planning scenario to connect a sequence of discrete preplanned configurations. Even with a relatively tight tolerance of $1\,\mathrm{mm}$, the method successfully finished in 85 out of 89 cases. The worst constraint violation in the remaining four cases remained under $2\,\mathrm{mm}$. Besides, the motions are planned with the median planning time of $6.7\,\mathrm{s}$. The method produces smooth natural looking motions without using any motion primitives. Thus, based on the reported results, the proposed optimization-based planning is a viable approach to motion planning for multi-legged walking robots.

REFERENCES

[1] D. Belter, P. Łabęcki, and P. Skrzypczyński, "Adaptive motion planning for autonomous rough terrain traversal with a walking robot," in *Journal of Field Robotics*, vol. 33, no. 3, 2016, pp. 337–370.

[2] A. Winkler, D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," in *IEEE Rpbotics and Automation Letters*, vol. 3, no. 3. IEEE, 2018, pp. 1560–1567.

[3] Z. Kingston, M. Moll, and L. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual review of control, robotics, and autonomous systems*, vol. 1, pp. 159–185, 2018.

[4] K. Hauser and J.-C. Latombe, "Multi-modal motion planning in non-expansive spaces," *The International Journal of Robotics Research*, vol. 29, no. 7, pp. 897–915, 2010.

[5] Z. Kingston, A. Wells, M. Moll, and L. Kavraki, "Informing multi-modal planning with synergistic discrete leads," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[6] A. Escande, A. Kheddar, and S. Miossec, "Planning contact points for humanoid robots," *Robotics and Autonomous Systems*, vol. 61, no. 5, pp. 428–442, 2013.

[7] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[8] R. Luna, M. Moll, J. Badger, and L. E. Kavraki, "A scalable motion planner for high-dimensional kinematic systems," *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 361–388, 2019.

[9] Z. Kingston, M. Moll, and L. Kavraki, "Decoupling constraints from sampling-based planners," in *Robotics Research*, 2020, pp. 913–928.

[10] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[11] B. Magyar, N. Tsiogkas, J. Deray, S. Pfeiffer, and D. Lane, "Timed-Elastic Bands for Manipulation Motion Planning," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 4, pp. 3513–3520, 2019.

[12] E. Rimon, *Exact robot navigation using artificial potential functions*. Yale University, 1990.

[13] M. Forouhar, P. Čížek, and J. Faigl, "Scarab ii: A small versatile six-legged walking robot," in *5th Full-Day Workshop on Legged Robots at IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 1–2.

[14] D. Valouch and J. Faigl, "Gait-free planning for hexapod walking robot," in *European Conference on Mobile Robots (ECMR)*, 2021, pp. 1–8.

[15] ——, "Caterpillar heuristic for gait-free planning with multi-legged robot," *IEEE Robotics and Automation Letters*, 2023. [Online]. Available: https://doi.org/10.1109/LRA.2023.3293749

[16] R. H. Bartels, J. C. Beatty, and B. A. Barsky, *An introduction to splines for use in computer graphics and geometric modeling*. Morgan Kaufmann, 1995.

[17] A. A. Saputra, N. N. W. Tay, Y. Toda, J. Botzheim, and N. Kubota, "Bézier curve model for efficient bio-inspired locomotion of low cost four legged robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4443–4448.

[18] S. Tonneau, "Convex strategies for trajectory optimisation: application to the polytope traversal problem," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 3335–3340.

[19] F. C. Park and K. M. Lynch, *Modern Robotics Mechanics, Planning and Control*. Cambridge University Press, 2017.

[20] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of computing*, vol. 8, no. 1, pp. 415–428, 2012.

[21] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) – The Complete Reference (Volume 1)*, A. Koubaa, Ed., 2016, ch. 5.

[22] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 2004.

[23] M. Lampton, "Damping–undamping strategies for the levenberg–marquardt nonlinear least-squares method," *Computers in Physics*, vol. 11, no. 1, pp. 110–115, 1997.

[24] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," https://eigen.tuxfamily.org, 2010, accessed: 2023-05-01.

[25] Intel®, "oneAPI Math Kernel Library," http://intel.com, 2022, accessed: 2023-05-01.