Pose estimation-based path planning for a tracked mobile robot traversing uneven terrains

Jae-Yun Jun, Jean-Philippe Saut, Faïz Ben Amar

Pose estimation-based path planning for a tracked mobile robot traversing uneven terrains

Jae-Yun Jun[a,b,*], Jean-Philippe Saut[a,b], Faïz Benamar[a,b]

[a] *Sorbonne Universités, UPMC Univ Paris 06, UMR 7222, ISIR, F-75005, Paris, France*
[b] *CNRS, UMR 7222, ISIR, F-75005, Paris, France*

Abstract

A novel path-planning algorithm is proposed for a tracked mobile robot to traverse uneven terrains, which can efficiently search for stability sub-optimal paths. This algorithm consists of combining two RRT-like algorithms (the Transition-based RRT (T-RRT) and the Dynamic-Domain RRT (DD-RRT) algorithms) bidirectionally and of representing the robot-terrain interaction with the robot's quasi-static tip-over stability measure (assuming that the robot traverses uneven terrains at low speed for safety). The robot's stability is computed by first estimating the robot's pose, which in turn is interpreted as a contact problem, formulated as a linear complementarity problem (LCP), and solved using the Lemke's method (which guarantees a fast convergence). The present work compares the performance of the proposed algorithm to other RRT-like algorithms (in terms of planning time, rate of success in finding solutions and the associated cost values) over various uneven terrains and shows that the proposed algorithm can be advantageous over its counterparts in various aspects of the planning performance.

Keywords: Path Planning, Rough Terrain, Sampling-based Motion Planning, Mobile Robot, Tip-over Stability

1. Introduction

In path-planning problems, the definition of the domain in which a solution (i.e., a path that connects a given pair of the start and the goal configurations satisfying certain constraints that depend on the applications) may reside and the choice of algorithms to solve such problems are critical to discern whether a solution exists in a timely manner and to choose the optimal one, if multiple solutions exist.

First, the definition of the domain in which a solution may reside depends on the application types. For instance, a vertical surface (such as a wall) is prohibitive for ground mobile robots, but, for climbing robots or humanoids that use it as a mean either for creating their mobility or for increasing their stability, it forms part of the free configuration space. For ground mobile robots, any entity that protrudes from the flat surface is often seen as an obstacle. However, for the problem of planning paths over uneven terrains, such entities need to be analyzed whether they really represent obstacles for the considered robot model (e.g., cliffs, deep pits, buildings, etc.) or they are objects that a robot actually needs to interact with (sometimes being absolutely necessary) to achieve certain goals (e.g., when a goal position is located on a terrain level that can only be reached through steps, stairways, ramps or hills). In essence, a given environment for ground mobile robots can be defined as the union of traversable and non-traversable regions. And, an approach to define the environment in this manner may be by associating it to its *traversability map*.

Various definitions of *traversability* can be found in the literature. The traversability can be defined as the product between two probability values for a given position in a terrain map: the probability that the terrain slope is smaller than a chosen maximum permissible slope value and the probability that the roughness is smaller than a chosen maximum permissible roughness value [1]. Likewise, the traversability may be defined as the sum of the roughness and the curvature (or slope) of a given grid cell known as *cell impedance* [2] or *traversability index* [3]. Unfortunately, these approaches represent the traversability as a single probabilistic value that characterizes each terrain map point by only considering the terrain samples in a neighborhood of size equivalent to the robot's dimension, and they do not give a sense of how the robot actually interacts with the terrain. On the other extreme, the robot dynamics that includes the terramechanics can be considered to have a more accurate estimate on the robot-terrain interaction, and such knowledge can be used to generate an objective function such as the *dynamic mobility index* for choosing optimal paths [4, 5, 6]. However, this approach is not suitable for the design of an efficient path planner because it requires the integration of the differential equations that include the contact forces generated between the robot and irregular terrains involving rolling friction, longitudinal slippage and lateral skidding where the associated parameter values change from terrain to terrain [7]. In addition, the planning efficiency is further reduced because its domain becomes now the state space which has twice the dimension of the configuration space.

To remedy these weaknesses, other approaches can be considered. *Kubota et al.* defined the traversability as the probability that a robot pose satisfies the roll, pitch and height criteria for a wheeled microrover with the Gaussian distribution, where

*Corresponding author
Email addresses: jaeyunjk@gmail.com (Jae-Yun Jun), amar@isir.upmc.fr (Faïz Benamar)

the robot pose was approximately estimated [8]. In like manner, *Haït et al.* proposed a method that searches for a path that minimizes changes in the robot body roll and pitch angles using the A* algorithm for a six-wheeled robot [9]. However, both works deal with wheeled robots for which the contact points are known a priori. When dealing with tracked mobile robots, this assumption does not hold since the robot can make contact with the terrain along any point on the tracks, flippers and even with the main base of the robot. Moreover, they search for optimal paths using grid-based search algorithms, which might not be efficient depending on the map size and each cell size.

Recently, *Beck et al.* proposed that the robot's tip-over stability measure can be used to plan safer traversable paths for tracked mobile robots [10], and *Norouzi et al.* used a dynamic simulator (ODE [11]) to estimate the robot's uncertain height, roll and pitch values in order to estimate this stability measure, with the purpose to search the path that optimizes the stability using the A* algorithm [12]. The tip-over stability is computed as proposed by *Papadopoulos and Rey* [13] (a.k.a. *force-angle stability measure*), which considers both the distance of the robot's center of mass with respect to the terrain, and the shape and the orientation of the support polygon, which depend on the pose of the robot. In a separate study, *Roan et al.* experimentally compared (in [14]) the correspondence of various stability measures such as the force-angle stability measure [13, 15], the moment-height stability (MHS) [16], and the dynamic stability measure obtained through zero-moment point (ZMP) compensation [17]. In this study, they showed that the force-angle approach gives the best results with small negative stability measure at the time instant of tip-over with small lag time, although the number of false positives was relatively high compared to the ZMP approach due to its increased sensitivity [14].

In the present work, the problem of path planning for a ground mobile robot to traverse uneven terrains is studied, and, in particular, a tracked mobile robot is considered in the context of the ongoing FRAUDO[1] project. When navigating over off-road and over uneven terrains, tracked mobile robots are attractive for their mechanical robustness due to their reduced number of degrees of freedom, and for their large stability and traction achieved by the large contact area formed between the tracks and the terrain. For these advantages, these robots are frequently used to traverse uneven terrains for applications such as search and rescue [18, 19, 20, 21].

Concerning the definition of the interaction between the tracked mobile robot model and uneven terrains, we first assume that a set of data points that represent a terrain is given (such as a *point cloud* obtained from either a laser rangefinder or a 3D RGB sensor) and find its corresponding elevation map. Then, the elevation map is *B-splined* with the purpose to consider the terrain domain as a continuous space, as was implemented for the first time in this context by [22].

Afterwards, the terrain traversability is studied in two steps:

1) the B-splined terrain map's roughness is estimated using the *simple microrelief factor* [23, 24] to rapidly filter out the regions of the terrain that are impossible to be traversed such as walls, cliffs, steep hills or deep pits;

2) for feasible regions in the terrain map, the robot's traversability is studied using the force-angle stability measure for the quasi-static case, assuming the fact that the robot moves slowly when dealing with uneven terrains for safety. The force-angle stability measure can be computed in turn if the robot's pose is known. To this end, the pose of the tracked mobile robot, for which the location of the robot body contact points are unknown a priori, is estimated by formulating the problem as a linear complementarity problem (LCP) [25] and then by solving this problem with the Lemke's method [25].

In the literature, one can find various off-road motion-planning works, which mainly adopt either kinematic (e.g., [26]) or dynamic approaches (e.g., [22, 27]). For this work we have chosen the kinematic approach because of the difficulty to efficiently model the dynamics of the robot negotiating terrains with generic uneven terrains (often composed by stairs, steps, ramps and rough terrains). Our work differs from the work presented in [26] in the aspect that [26] uses the A* algorithm for path planning, a grid-based search algorithm, which can be inefficient depending on the map size and the cell size, whereas our work is based on the sampling-based motion planning approach (as described below).

Second, the choice of algorithms to solve path planning problems is critical to efficiently know whether a solution exists, and, if multiple solutions exist, to choose the optimal one. In path-search problems, an algorithm that finds a solution or indicates that no solution can be found in a finite amount of time is called *complete* [28, 29]. However, such completeness requirement, although a desirable property, may easily become computationally intractable (even for a simple problem such as the piano mover's problem [30]). As a result, the *probabilistic-completeness* problem, a relaxed version of the original problem, has often been addressed instead. This problem consists of finding a solution (if a solution exists) with probability one as time goes to infinity [28, 29] and can be solved using algorithms known as the *sampling-based planning algorithms*.

Among many types of sampling-based planning algorithms, the *rapidly-exploring random tree (RRT)* [31] is an efficient single-query method that does not require preprocessing like multiple-query methods do (e.g., the *Probabilistic Roadmap (PRM)* [32]). RRT is characterized by its nature of quickly exploring unexplored regions since the tree tends to grow through the nodes that have larger Voronoi regions associated [33]. Nonetheless, the RRT algorithm uniformly samples in the configuration space and does not consider any cost function, which could be used to bias the sample space and find a desirable solution with more efficiency. Recently, an RRT-based algorithm has been proposed to bias the search space to low-cost regions with a gradual incorporation of higher-cost regions as the number of rejected samples increases. This approach is known as the *Transition*-based RRT (T-RRT) algorithm [34] and can significantly increase the planning efficiency [35, 36], especially when it is used bidirectionally [36]. Although this approach

[1]FRAUDO is an acronym in French language for ***F**ranchissement **Au**tomatique **d'O**bstacles*, which means *automatic obstacle crossing*.

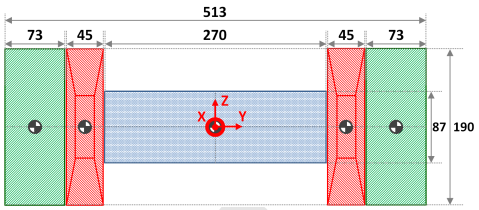
does not guarantee the solution optimality as RRT* does [37] (hence, offering most likely sub-optimal solutions), it can efficiently give good-quality solutions [36, 38]. Whereas, even though RRT* has its computational complexity within a constant factor of that required by RRT [37], it might converge slowly [36, 38].

On the other hand, the RRT-like algorithms might suffer from slow exploration if the sample domain is not well suited to a given problem, especially when the number of *frontier nodes* (i.e., nodes with the corresponding Voronoi regions growing with the size of the environment) that are also *boundary nodes* (i.e., nodes that lie in some proximity to obstacles) is significant [39]. This efficiency problem might be solved by dynamically restricting the sample domain associated to each of the boundary nodes in the tree, which is equivalent to reducing the probability that a boundary node in the tree will be chosen as the nearest node to a new sample [39]. This algorithm is known as the *Dynamic-domain-RRT* (DD−RRT) [39, 40].

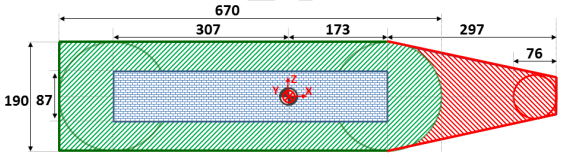
In the present work, a novel path-planning algorithm is proposed for a tracked mobile robot to traverse uneven terrains, by combining two RRT algorithms (the Transition-based RRT (T-RRT) [34] and the Dynamic-Domain RRT (DD-RRT) [39]) to increase the planning efficiency and by representing the robot-terrain interaction with the tip-over stability measure. Its efficiency can further be improved by using it bidirectionally [36]. The algorithm proposed in the present work is coined as the *bidirectional dynamic-domain transition-based RRT* (BiDDTRRT). Then, the performance of this planning algorithm is compared to other RRT-like algorithms such as the *bidirectional RRT* (BiRRT) and the *bidirectional T-RRT* (BiTRRT) algorithms in terms of the average planning time, path cost, path length and number of configurations. Moreover, the results obtained using two types of cost functions are compared: the *Minimal Work (MW)* [34] and the tip-over stability-related cost function.

The rest of the present work is structured as follows. In Section 2, we describe our robotic platform, and give details on how we solve the problem of pose estimation and compute the quasi-static tip-over stability. In Section 3, we describe our pose estimation-based motion planning algorithm for traversing uneven terrains. In Section 4, we show and analyze the results obtained using the path planning algorithm proposed in the present work. Finally, in Section 5, we provide concluding remarks and guide future work.

Figure 1: A *Cameleon EOD*, a tracked mobile robot from *Eca Robotics* [41]. It consists of one main base, two tracks and two coupled frontal flippers. The details on the robot specification are given in the text.



(a) Front view of the robot model



(b) Side view of the robot model

Figure 2: Tracked mobile robot model is shown in front and side views. The robot body consists of one main base, two tracks and two coupled frontal flippers. The center of mass is located slightly forward from its geometric center. All the values are given in mm.

2. Robot model, terrain representation, pose estimation, and tip-over stability

2.1. Description of the robotic platform

The robotic platform employed in the present work is a *Cameleon EOD* [41] (see Figure 1), a tracked mobile robot that weighs about 27 kg and has a 25 kg payload. It has as dimension 0.67 m (L) × 0.5 m (W) × 0.19 m (H), and the center of mass is slightly forward from its geometric center. It can move at up to 6 km/h with about 4 hours of autonomy. In addition, it has two flippers of about 0.4 m of length, which allow the robot to cross over obstacles with up to 0.25 m of height. It has a 3D RGB sensor (ASUS Xtion) to build maps and self-localize, two motor encoders that return the robot's odometry, and two inclinometers to measure the roll and pitch of the robot. Two DC motors attached to the front sprockets give mobility to the robot. The two front flippers are coupled and driven by the third DC motor.

In our planner, the robot is modeled as a multi-rigid body where its main frame and the tracks are modeled with their respective bounding boxes, whereas the flippers are modeled with bounding trapezoids because their dimensions are not symmetric in the frontal plane (see Figure 2).

2.2. Terrain representation

While in reality the world perceived by sensors is represented in a discrete fashion (e.g., point cloud), the employed planning algorithms work in a continuous space. Therefore, in the present work the terrain is represented as a uniform bicubic B-spline surface (which can be done offline or online with lower sampling frequency), as was implemented for the first time in

this context by [22], using the terrain map data points as control points as follows

$$s(u, v) = \mathbf{U}^{\mathrm{T}}\mathbf{M}\mathbf{D}\mathbf{M}^{\mathrm{T}}\mathbf{V}, \tag{1}$$

where $u \in [0, 1]$, $v \in [0, 1]$, $\mathbf{U}=(u^3, u^2, u, 1)^{\mathrm{T}}$, $\mathbf{V}=(v^3, v^2, v, 1)^{\mathrm{T}}$, $\mathbf{M}$ is a (4×4) constant matrix to construct bicubic B-spline patches, and $\mathbf{D}$ is a (4×4) matrix composed by 16 control points given by [42]. The resolution of control points is about 5.7% of the robot length. As a result, a B-splined terrain elevation map is obtained.

On the other hand, the roughness of the terrain is characterized using the *simple microrelief factor* [23, 24] in order to quickly discard regions with two features: large gradients and significant height changes. Some examples of these regions are walls, cliffs, steep hills and deep pits.

Hence, a binary occupancy map composed by traversable and non-traversable regions is generated by thresholding the simple microrelief factor for the entire terrain map. This binary map then goes through an erosion-and-dilation filtering procedure to reduce the noise level. Next, each non-traversable region is segmented using the *watershed transformation* [43], and the boundary of each non-traversable region is detected by using the *border following algorithm* proposed in [44]. In general, the non-traversable regions are nonconvex two-dimensional polygons, and, therefore, the convexity assumption can not be made. The detection of collision between these regions and the robot body is performed by formulating the problem as a *nonconvex polygon interior problem* and is efficiently solved by using the algorithm proposed in [45].

2.3. Robot pose estimation

As detailed in Section 1, the robot pose is estimated with the purpose to compute its tip-over stability, which in turn is used in the path planner to search for stable paths.

The pose of a tracked mobile robot depends on its interaction with the terrain and can be estimated by studying how it makes contact with the terrain. Therefore, the problem of the robot pose estimation can be considered as a contact problem between the robot body and the terrain. The contact between two objects can be mainly modeled using either compliant contact models [46, 47] or impulse-based models formulated as linear complementarity problems (LCP) [48, 49]. In the present work, we formulate the contact between the robot body and the terrain as an LCP because of its fast convergence [50] and because this method seems to be appropriate for applications that involve sustained contacts [51].

When the contact problem is formulated as an LCP, three types of complementarity constraints are typically considered: non-penetration constraint, constraint between frictional forces and tangential velocities, and constraint on the net frictional force being inside the frictional cone or on this cone [48, 52]. In the present work, only the non-penetration constraint is considered for estimating the robot pose, and no slippage is assumed.

A rigid-body robot pose in the world $\mathcal{W} \subset \mathbb{R}^3$ can be expressed as $(X, Y, Z, \alpha, \beta, \gamma)$, where (X, Y, Z) are the coordinates of the center of gravity of the robot in the terrain frame, and (α, β, γ) are the roll, pitch, and yaw angles formed from the terrain frame to the robot-body frame. In the present work (as described in Section 3), the motion planner chooses $\mathbf{c}=(X, Y, \gamma)^{\mathrm{T}}$ (i.e., the planner fixes $\mathbf{c}=(X, Y, \gamma)^{\mathrm{T}}$), and, therefore, the problem of robot pose estimation consists of finding $\mathbf{q}=(Z, \alpha, \beta)^{\mathrm{T}}$ for given $\mathbf{c}=(X, Y, \gamma)^{\mathrm{T}}$ and a terrain elevation map.

A method to solve this problem can be an approach similar to the one suggested by [53]. Initially, the robot is suspended in the air (but close enough to the terrain for a fast convergence) and falls to the terrain driven by the earth gravitational acceleration. The robot motion can be described as

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{W_n}\mathbf{A}\mathbf{f} + \mathbf{Q_{co}}, \tag{2}$$

with

$$\mathbf{M} = \mathrm{diag}(m, I_\alpha, I_\beta),$$
$$\mathbf{A} = \mathrm{diag}(a_1, a_2, \cdots, a_p),$$
$$\mathbf{f} = (f_1, f_2, \cdots, f_p)^T,$$

where: $\mathbf{M}$ is the diagonal inertial matrix; (m, I_α, I_β) are the body mass, the moments of inertia about the roll and the pitch axes, respectively; $\ddot{\mathbf{q}}$ is $(\ddot{Z}, \ddot{\alpha}, \ddot{\beta})^T$; p is the number of potential contact points uniformly distributed along the tracks and flippers; $\mathbf{W_n}$ is the $(3 \times p)$ Jacobian or Wrench matrix that maps contact forces normal to the splined surface to wrenches in the robot's body frame; $\mathbf{f}$ is the $(p\times1)$ vector of contact forces normal to the splined surface; $\mathbf{Q}_{co}$ is the $(p\times1)$ generalized conservative force (gravitational force) vector; and $\mathbf{A}$ is the $(p \times p)$ activation matrix that indicates whether each of the potential contact points is active (i.e., whether each of the potential contact points has made contact with the terrain), and its diagonal elements have the following expression

$$a_i = \begin{cases} 1, & \text{if the } i\text{-th potential contact point collided,} \\ 0, & \text{otherwise.} \end{cases}$$

The system of equations of motion (2) has two unknown vectors: $\mathbf{q}$ and $\mathbf{f}$. Hence, we have a system of three equations with $p + 3$ unknowns. $\mathbf{q}$ and $\mathbf{f}$ can be computed by noticing the complementarity relationship that exists for each potential contact point between its contact force and its distance with respect to the terrain. This relation can be formulated as p non-penetration complementarity constraints and can be expressed as

$$\mathbf{0} \leq \mathbf{d} \perp \mathbf{f} \geq \mathbf{0}, \tag{3}$$

where $\mathbf{d} = (d_1, \cdots, d_p)^T$ are the vector whose elements are the distance between the potential contact points on the robot body and the projection of these points on the terrain along the vertical direction, $\mathbf{f} = (f_1, \cdots, f_p)^T$ are the contact forces normal to the splined surface, and $\perp$ represents the complementarity operator [25]. The complementarity constraint expressed in (3) says that, for any i, either d_i is zero or f_i is zero but not both at the same time. In other words, if the distance between a potential contact point on the robot body and its projection on the terrain is non-zero, then its associated contact force must be zero. On the other hand, if this distance is zero, then the associated contact force must be non-zero.

Hence, the non-penetration LCP can be formulated from (2) and (3), and this problem can be solved for $\mathbf{q}$ and $\mathbf{f}$ as follows. First, the discrete-time version of (2) is expressed using the Euler approximation as

$$\mathbf{M}\left(\frac{\mathbf{v}^{\ell+1}-\mathbf{v}^{\ell}}{h}\right)=\mathbf{W_n}\mathbf{A}\mathbf{f}^{\ell+1}+\mathbf{Q_{co}}, \tag{4}$$

where $\mathbf{v}$ is $\dot{\mathbf{q}}$, ℓ is the iteration index, and h is the discretization time step. From (4), one can obtain the velocity expression as

$$\mathbf{v}^{\ell+1}=\mathbf{v}^{\ell}+h\mathbf{M^{-1}}\mathbf{W_n}\mathbf{A}\mathbf{f}^{\ell+1}+h\mathbf{M^{-1}}\mathbf{Q_{co}}. \tag{5}$$

On the other hand, the distance between the potential contact points on the robot body and their projection on the terrain along the vertical direction at the iteration ℓ are expressed as

$$\mathbf{d}^{\ell+1}=\mathbf{d}^{\ell}+h\dot{\mathbf{d}}^{\ell+1}=\mathbf{d}^{\ell}+h\mathbf{A^T}\mathbf{W_n^T}\mathbf{v}^{\ell+1}. \tag{6}$$

Now $\mathbf{d}^{\ell+1}$ can be related to $\mathbf{f}^{\ell+1}$ in (6) by replacing $\mathbf{v}^{\ell+1}$ with the expression given in (5), and the non-penetration LCP given by (2) and (3) can be rewritten as

$$\begin{aligned}\mathbf{d}^{\ell+1}&=\mathbf{B}\mathbf{f}^{\ell+1}+\mathbf{b},\\ \mathbf{0}\leq\mathbf{d}^{\ell+1}&\perp\mathbf{f}^{\ell+1}\geq\mathbf{0},\end{aligned} \tag{7}$$

where

$$\mathbf{B}=h^2\mathbf{A^T}\mathbf{W_n^T}\mathbf{M^{-1}}\mathbf{W_n}\mathbf{A},$$

and

$$\mathbf{b}=\mathbf{d}^{\ell}+h\mathbf{A^T}\mathbf{W_n^T}\left[\mathbf{v}^{\ell}+h\mathbf{M^{-1}}\mathbf{Q_{co}}\right].$$

Further, (7) can be solved using a pivoting method known as the Lemke's method [25].

2.4. Tip-over stability-related cost function

The path planner specific in the present work aims to search for stable paths by introducing the notion of stability in the planning problem. There are several stability definitions employed in the literature specific for ground mobile robots traversing uneven terrains (see for instance [14]). Among these definitions, the tip-over stability proposed by [13] (also known as the *force-angle stability measure*) is chosen for its good performance, as reported in [14].

The *force-angle stability measure* is defined as the product between the magnitude of the net force acting on the robot's center of mass and the smallest angle ($\min_i(\theta_i)$) formed by this net force vector ($\mathbf{f_r}$) and the tip-over-axis normals ($\mathbf{l_i}$). Therefore, the force-angle stability measure is

$$\xi=\min_i(\theta_i)\|\mathbf{f}_r\|,\quad i=\{1,\cdots,n\}, \tag{8}$$

where n is the number of tip-over axes of the support polygon as shown in Figure 3.

Notice that, according to this expression, the stability can be augmented by increasing $\min_i(\theta_i)$ and/or $\|\mathbf{f}_r\|$ values. On one hand, $\min_i(\theta_i)$ can be increased by 1) increasing the support area, 2) lowering the height of the CoM with respect to the terrain, and/or 3) shifting the force vector to the innermost point in the support polygon. On the other hand, $\|\mathbf{f}_r\|$ can be augmented by 1) increasing the mass or inertia of the robot, and/or 2) increasing the acceleration normal to the terrain.

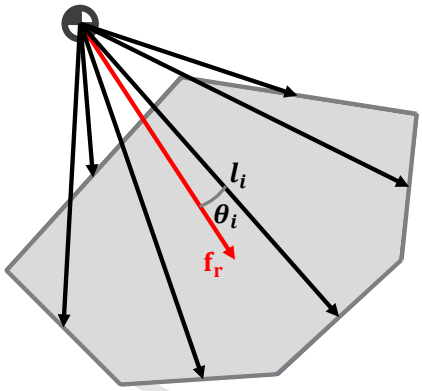


Figure 3: In the present work, the robot's tip-over stability is defined as the force-angle stability measure [13]. When the robot pose is estimated as described in Section 2.3, its support polygon can easily be found. The tip-over axes are the vectors that define the polygon boundary, and the force-angle stability measure can be computed as the product between the magnitude of the net force and the smallest angle obtained among all the angles formed between the tip-over-axis normals ($\mathbf{l_i}$'s) and the net force vector ($\mathbf{f_r}$). A tip-over-axis normal is a vector that is perpendicular to its corresponding tip-over axis, and it passes through the robot's center of mass. The details on the definition of the force-angle stability measure are given in the text.

Next, the expression given in (8) is normalized by its maximum value ($\xi_{\max}$), which is obtained when the mobile robot is in its most stable configuration. Therefore, the normalized force-angle stability measure ($\hat{\xi}$) can be expressed as

$$\hat{\xi}=\frac{\xi}{\xi_{\max}}. \tag{9}$$

This scalar is bounded, and the positive and negative values of $\hat{\xi}$ indicate stable and unstable configurations in the tip-over stability sense, respectively. The robot is critically stable when $\hat{\xi}=0$.

In the present work, the robot is assumed to move slowly over uneven terrains in order to avoid dangerous situations such as the stability loss due to slippage or sliding effects, as this is the case in other previous works (e.g., [54]). Under these circumstances, we only consider the quasi-static stability case with the net force formed by the gravitational force alone.

In Figure 4, the tip-over stability is related to changes in pitch, roll, and pitch-and-roll. First, all three graphs have the bell shape, and their respective maximum value is achieved around 0 [deg]. The graph corresponding to roll changes is shallower than that of pitch changes because the robot is long (along the x-direction) and narrow (along the y-direction). The results corresponding to roll-and-pitch changes are located in between the results corresponding to changes in pitch and roll alone. The pitch changes induce a non-symmetric graph with respect to the vertical line about the flat configuration because

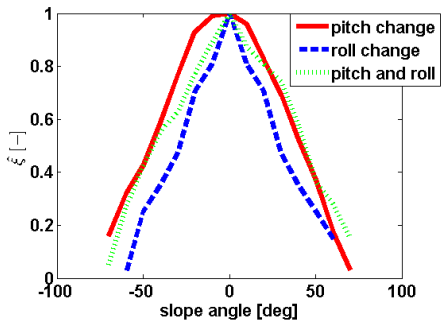


Figure 4: The normalized tip-over stability ($\hat{\xi}$) (see (9)) corresponding to changes in pitch, roll, and pitch-and-roll.

the robot's center of gravity (CoG) is located slightly forward along the robot length (as explained in Section 2.1). On the contrary, the graph corresponding to roll changes is symmetric because the CoG is located on the midline of the body along the y-direction (lateral direction). Notice that these values are obtained after estimating the robot pose, hence numerical errors are also present in the representation of the tip-over stability measures.

Finally, the tip-over stability-related cost function is defined as follows

$$\mathcal{J} = \begin{cases} 1 - \hat{\xi}, & \hat{\xi} > 0\,, \\ \infty, & \hat{\xi} \leq 0\,, \end{cases} \tag{10}$$

where $\hat{\xi}$ is the normalized tip-over stability measure defined in (9). This cost function is used in the planning algorithm (which is described in Section 3.1) in order to search paths that improve the robot stability.

3. Pose estimation-based path planning over uneven terrains

A general overview of the path-planning algorithm that is described in the subsequent subsections is first given as follows. At a given time instant in the path-planning procedure, a sample with (X,Y)-coordinates is randomly chosen within the feasible region in the terrain map. Then, a yaw angle (γ) is picked based on the motion primitive that suits the most, depending on the location of the nearest valid sample to this new sample while avoiding possible collisions to connect these two samples[2]. At this moment, $\mathbf{c}=(X, Y, \gamma)^{\mathrm{T}}$ is fixed. Then, $\mathbf{q}=(Z, \alpha, \beta)^{\mathrm{T}}$ of the robot is estimated using (2), and the associated stability-related cost value is computed using (10) to evaluate whether the connection between the two samples can be made. Details on the proposed path-planning algorithm is given in the subsequent subsections.

3.1. Problem statement

The pose estimation-based path planning problem addressed in the present work is stated as follows.

Let the world $\mathcal{W} \subset \mathbb{R}^3$ be an uneven terrain represented as an elevation map (described in Section 2.2). Let a semi-algebraic robot $\mathcal{A}$ be defined in $\mathcal{W}$ consisting of a main base, two tracks and two flippers (as described in Section 2.1).

Let $C \subset \mathbb{R}^3$ be the configuration space, and $\mathbf{c}=(X, Y, \gamma)^{\mathrm{T}} \in C$ be the configuration of $\mathcal{A}$, where (X, Y) are the Cartesian coordinates of the robot's center of gravity on the horizontal plane in the terrain frame, and γ is the yaw angle of the robot body. In this work, the first two components of $\mathbf{c}$ (i.e., $\mathbf{s}=(X, Y) \in \mathcal{S}_{\text{traversable}}$) are randomly sampled by the planner, with independent and uniform distribution, where $\mathcal{S}_{\text{traversable}} \subset \mathbb{R}^2$ is defined to be the *traversable sample space*, which in turn can be defined by using the two-dimensional occupancy map built with the simple microrelief factor, as described in Section 2.2. On the other hand, γ is chosen based on the motion primitive (defined in Section 3.3) that suits the most, depending on the location of the nearest valid sample to this new sample, while avoiding possible collisions to connect these two samples.

Let $\mathbf{c}_{\text{start}} \in C$ and $\mathbf{c}_{\text{goal}} \in C$ be some given start and goal configurations, respectively. In addition, let $\boldsymbol{\sigma}$ denote a path in C and be defined as $\boldsymbol{\sigma} : [0, N] \rightarrow C$, where $\boldsymbol{\sigma}(0)=\mathbf{c}_{\text{start}}$ and $\boldsymbol{\sigma}(N)=\mathbf{c}_{\text{goal}}$. A path is said to be *goal-reachable* if $\mathcal{J}(\boldsymbol{\sigma}(\tau))$ is finite for $\tau \in [0, N]$, where $\mathcal{J}(\cdot)$ is the tip-over stability-related cost function given in (10).

Then, the problem of pose estimation-based path planning is defined as to find a *goal-reachable* path that connects a given pair of start and goal configurations, estimating the robot pose at each sampling (i.e., by solving (2)) and then computing the cost value using (10) for each sample.

3.2. Path planning algorithms

In the present work, a novel path-planning algorithm is proposed for a tracked mobile robot to traverse uneven terrains, by first combining two RRT-like algorithms (the Transition-based RRT (T-RRT) [34] and the Dynamic-Domain RRT (DD-RRT) [39]) and by representing the robot-terrain interaction with the tip-over stability measure. As described in Section 1, the T−RRT can plan paths efficiently by biasing the search space to low-cost regions with a gradual incorporation of higher-cost regions as the number of rejected samples increases [34]. Its efficiency can further be improved by using it bidirectionally [36]. But, this approach might suffer from slow exploration when the number of *frontier nodes* (i.e., nodes with the corresponding Voronoi regions growing with the size of the environment) that are also *boundary nodes* (i.e., nodes that lie in some proximity to obstacles) is significant [39]. The DD−RRT addresses this problem by restricting the sample domain associated to each of the boundary nodes in the tree, which is equivalent to reducing the probability that a boundary node in the tree will be chosen as the nearest node to a new sample [39]. In the present work, we combine the two algorithms and adapt the resulting algorithm for planning paths for a tracked mobile robot to traverse uneven terrains constructing a cost function related to the robot's stability (shown in (10)). The algorithm proposed in the present work is coined as the *bidirectional dynamic-domain transition-based RRT* (BiDDTRRT).

[2] The concept of motion primitives is used to increase the planning efficiency and to reduce the amount of yaw motion while interpolating two nearest configurations in the tree. Details are given in Section 3.3.

Algorithm 1: For BiRRT [33] / BiTRRT [36] / BiDDTRRT
$(\mathcal{T}_a, \mathcal{T}_b, \texttt{Path}) \leftarrow \texttt{BuildTree}(\mathbf{c}_{\text{start}}, \mathbf{c}_{\text{goal}})$

$\mathcal{V}_a \leftarrow \{\mathbf{c}_{\text{start}}\}$; $\mathcal{E}_a \leftarrow \emptyset$; $\mathcal{T}_a \leftarrow (\mathcal{V}_a, \mathcal{E}_a)$;
$\mathcal{V}_b \leftarrow \{\mathbf{c}_{\text{goal}}\}$; $\mathcal{E}_b \leftarrow \emptyset$; $\mathcal{T}_b \leftarrow (\mathcal{V}_b, \mathcal{E}_b)$;
Path $\leftarrow \emptyset$; $i \leftarrow 0$;
while $i < L$ **do**
  $[\mathcal{T}_a, \mathbf{c}_{\text{new}_\mathbf{a}}, \texttt{extended}] \leftarrow \texttt{Extend}(\mathcal{T}_a)$;
  if (extended) **then**
    $[\mathcal{T}_b, \mathbf{c}_{\text{nearest}_\mathbf{b}}, \texttt{status}] \leftarrow \texttt{Connect}(\mathcal{T}_b, \mathbf{c}_{\text{new}_\mathbf{a}})$;
    if (status) **then**
      Path $\leftarrow$ ObtainPath$(\mathcal{T}_a, \mathcal{T}_b, \mathbf{c}_{\text{new}_\mathbf{a}}, \mathbf{c}_{\text{nearest}_\mathbf{b}})$;
      return $(\mathcal{T}_a, \mathcal{T}_b, \texttt{Path})$;
  Swap$(\mathcal{T}_a, \mathcal{T}_b)$;
  $i \leftarrow i + 1$;
return $(\mathcal{T}_a, \mathcal{T}_b, \texttt{Path})$;

Algorithm 2: For BiRRT [33]
$(\mathcal{T}, \mathbf{c}_{\text{new}}, \texttt{status}) \leftarrow \texttt{Extend}(\mathcal{T})$

$\mathbf{c}_{\text{rand}} \leftarrow$ RandomSample();
$\mathbf{c}_{\text{nearest}} \leftarrow$ SearchNearest$(\mathcal{T}, \mathbf{c}_{\text{rand}})$;
$[\mathbf{c}_{\text{new}}, \texttt{status}] \leftarrow$ ObtainNewConfig$(\mathcal{T}, \mathbf{c}_{\text{nearest}}, \mathbf{c}_{\text{rand}})$;
if (status) **then**
  $\mathcal{T}$.addNode$(\mathbf{c}_{\text{new}})$;
  $\mathcal{T}$.addEdge$(\mathbf{c}_{\text{nearest}}, \mathbf{c}_{\text{new}})$;
return $(\mathcal{T}, \mathbf{c}_{new}, \texttt{status})$;

With the purpose to compare the performance of this algorithm to that of other RRT-like algorithms, all the considered algorithms are described to the below. Besides the BiDDTRRT, two other RRT-like algorithms are considered: the bidirectional RRT (BiRRT) [55], and the bidirectional transition-based RRT (BiTRRT) [36]. The bidirectional algorithms are considered instead of unidirectional ones because the former ones are more efficient as shown in [36, 55].

For all the considered algorithms, the BuildTree function is the main function that builds trees and returns a path that connects a given pair of start and goal configurations, if such a path is found within L iterations. The structure of the BuildTree function is shared by the BiRRT, BiTRRT, and BiDDTRRT algorithms (Algorithm 1).

A tree $\mathcal{T}$=$(\mathcal{V}, \mathcal{E})$ is a graph that has no cycle and consists of a set of vertices ($\mathcal{V}$) and a set of edges ($\mathcal{E}$). A *vertex* contains the information of a robot configuration, the associated cost value and a pointer that points to its parent vertex. Along the presentation of the present work it will be referred as a *node* indistinguishably. The BiRRT, BiTRRT, and BiDDTRRT consist of two trees, and they are denoted as $\mathcal{T}_a$=$(\mathcal{V}_a, \mathcal{E}_a)$ and $\mathcal{T}_b$=$(\mathcal{V}_b, \mathcal{E}_b)$. $\mathcal{V}_a$ initially consists of a node with a given start configuration ($\mathbf{c}_{\text{start}}$), and $\mathcal{V}_b$ initially has a node with a given goal configuration ($\mathbf{c}_{\text{goal}}$). Both $\mathcal{E}_a$ and $\mathcal{E}_b$ are initially empty sets. A sample is randomly picked from the sample space

Algorithm 3: For BiTRRT [36]
$(\mathcal{T}, \mathbf{c}_{\text{new}}, \texttt{status}) \leftarrow \texttt{Extend}(\mathcal{T})$

$\mathbf{c}_{\text{rand}} \leftarrow$ RandomSample();
$\mathbf{c}_{\text{nearest}} \leftarrow$ SearchNearest$(\mathcal{T}, \mathbf{c}_{\text{rand}})$;
$[\mathbf{c}_{\text{new}}, \texttt{status}] \leftarrow$ ObtainNewConfig$(\mathcal{T}, \mathbf{c}_{\text{nearest}}, \mathbf{c}_{\text{rand}})$;
if (status) **then**
  $\mathcal{J}_{\text{new}} \leftarrow$ ComputeCost$(\mathbf{c}_{\text{nearest}}, \mathbf{c}_{\text{new}})$;
  if (TestTransition$(\mathcal{J}_{nearest}, \mathcal{J}_{new})$ ***and*** ControlRefinement$(\mathcal{T}, \mathbf{c}_{nearest}, \mathbf{c}_{rand})$) **then**
    $\mathcal{T}$.addNode$(\mathbf{c}_{\text{new}})$;
    $\mathcal{T}$.addEdge$(\mathbf{c}_{\text{nearest}}, \mathbf{c}_{\text{new}})$;
return $(\mathcal{T}, \mathbf{c}_{new}, \texttt{status})$;

Algorithm 4: For BiDDTRRT
$(\mathcal{T}, \mathbf{c}_{\text{new}}, \texttt{status}) \leftarrow \texttt{Extend}(\mathcal{T})$

repeat
  $\mathbf{c}_{\text{rand}} \leftarrow$ RandomSample();
  $\mathbf{c}_{\text{nearest}} \leftarrow$ SearchNearest$(\mathcal{T}, \mathbf{c}_{\text{rand}})$;
until ComputeDistance$(\mathbf{c}_{nearest}, \mathbf{c}_{rand}) < \mathbf{c}_{nearest}$.radius;
$[\mathbf{c}_{\text{new}}, \texttt{status}] \leftarrow$ ObtainNewConfig$(\mathcal{T}, \mathbf{c}_{\text{nearest}}, \mathbf{c}_{\text{rand}})$;
if (status) **then**
  $\mathcal{J}_{\text{new}} \leftarrow$ ComputeCost$(\mathbf{c}_{\text{nearest}}, \mathbf{c}_{\text{new}})$;
  if (TestTransition$(\mathcal{J}_{nearest}, \mathcal{J}_{new})$ ***and*** ControlRefinement$(\mathcal{T}, \mathbf{c}_{nearest}, \mathbf{c}_{rand})$) **then**
    $\mathcal{T}$.addNode$(\mathbf{c}_{\text{new}})$;
    $\mathcal{T}$.addEdge$(\mathbf{c}_{\text{nearest}}, \mathbf{c}_{\text{new}})$;
    $\mathbf{c}_{\text{new}}$.radius $\leftarrow \infty$
  else
    $\mathbf{c}_{\text{nearest}}$.radius $\leftarrow R$
else
  $\mathbf{c}_{\text{nearest}}$.radius $\leftarrow R$
return $(\mathcal{T}, \mathbf{c}_{new}, \texttt{status})$;

($\mathbf{s}_{\text{rand}} \in \mathcal{S}_{\text{traversable}}$) with independent and uniform distribution in the RandomSample procedure.

Then the tree goes through the Extend procedure (Algorithm 2, Algorithm 3, and Algorithm 4). For the BiRRT, BiTRRT and BiDDTRRT algorithms, every time a tree is extended the planner verifies whether the two trees can be connected in the Connect procedure (as implemented in [33]). If they are connectable, then the resulting path is returned together with the associated trees by the ObtainPath procedure. Otherwise, the trees are swapped (as proposed in [33]), and the previous steps are repeated until either a path is found or the maximum number of iterations is reached.

The tree extension procedure differs from each other for all the three algorithms. The only aspects that are shared by all the algorithms in the Extend procedure are how a new node is added to $\mathcal{V}$, and the edge between the nearest node and the new node is added to $\mathcal{E}$, when the tree can be extended to the new node from its nearest node. The algorithms BiTRRT

Algorithm 5: For BiRRT/BiTRRT/BiDDTRRT
$(\mathbf{c}_{\text{new}}, \texttt{status}) \leftarrow \texttt{ObtainNewConfig}(\mathcal{T}, \mathbf{c}_{\text{nearest}}, \mathbf{c}_{\text{rand}})$

1 $\texttt{dist} \leftarrow \texttt{ComputeDistance}(\mathbf{c}_{\text{nearest}}, \mathbf{c}_{\text{rand}})$;
2 **if** $\texttt{dist} > \eta$ **then**
3 $\quad \mathbf{c}_{\text{new}} \leftarrow \texttt{Interpolate}(\mathbf{c}_{\text{nearest}}, \mathbf{c}_{\text{rand}}, \eta/\texttt{dist})$;
4 **else**
5 $\quad \mathbf{c}_{\text{new}} \leftarrow \mathbf{c}_{\text{rand}}$;
6 $\texttt{status} \leftarrow \texttt{IsTraversable}(\mathcal{T}, \mathbf{c}_{\text{nearest}}, \mathbf{c}_{\text{new}})$;
7 **return** $(\mathbf{c}_{new}, \texttt{status})$;

Algorithm 6: For BiTRRT [36] / BiDDTRRT
$\texttt{status} \leftarrow \texttt{TestTransition}(\mathcal{J}_{\text{nearest}}, \mathcal{J}_{\text{new}})$

1 **if** $\mathcal{J}_{new} > \mathcal{J}_{max}$ **then**
2 $\quad$ **return** False;
3 **if** $\mathcal{J}_{new} \leq \mathcal{J}_{nearest}$ **then**
4 $\quad$ **return** True;
5 **if** $\exp^{\left(-\frac{(\mathcal{J}_{new}-\mathcal{J}_{nearest})}{T}\right)} > 0.5$ **then**
6 $\quad T \leftarrow \frac{T}{2^{\left(\frac{\mathcal{J}_{\text{new}}-\mathcal{J}_{\text{nearest}}}{0.1\cdot K}\right)}}$;
7 $\quad$ **return** True;
8 **else**
9 $\quad T \leftarrow T \cdot 2^{T_{\text{rate}}}$;
10 $\quad$ **return** False;

and BiDDTRRT differ from the BiRRT in that in the former algorithms not every new node from the nearest node is accepted but only if it satisfies the conditions imposed in the TestTransition and ControlRefinement functions, shown in Algorithm 6 and Algorithm 7, respectively. Further, the algorithm BiDDTRRT differs from BiTRRT in that, with the former algorithm, a boundary node can be the nearest neighbor to a new sample only if this sample is within a ball around the boundary node with some finite radius value R (Algorithm 4), whereas, with the latter algorithm, there is no distinction between boundary and non-boundary nodes. These algorithms will be described afterwards in more detail.

The search for a new node from a randomly sampled node is performed in the ObtainNewConfig function (Algorithm 5). First, the Euclidean distance is calculated between the nearest node and the randomly sampled node. If this value is larger than a planning parameter η, then a configuration that interpolates between the nearest node and the randomly sampled node with distance η from the nearest node is chosen. Afterwards, one checks whether the robot can traverse from the nearest node to the new node by using the motion primitives described in Section 3.3, avoiding non-traversable regions and using the tip-over stability measure.

As mentioned earlier, both the BiTRRT and BiDDTRRT algorithms control the acceptance of new configurations based on their associated costs through the TestTransition function and the controlRefinement function.

The transition test from the nearest node to the new node is based on the Metropolis criterion (line 5 in Algorithm 6) which is typically used in Monte Carlo optimization methods [36]. In the present work, the method proposed in [36] is used instead of the method originally proposed in [34] due to its outperformance reported in [36]. The acceptance of a new configuration depends on the cost difference between the new node and the nearest node, and the "temperature" (T), a planning parameter. The initial temperature value (T_o) is chosen to be low, hence a small cost increase from the nearest to the new node will immediately cause the rejection of the new node. However, each rejection will increase the temperature value, which will permit the acceptance of new nodes that involve larger cost differences. On the other hand, each acceptance of new nodes will cause the temperature value to decrease. The parameter K shown in line 6 in Algorithm 6 is the range of the cost values attainable as the robot interacts with a terrain. For the cost function $\mathcal{J}$, described in (10), the K is unity[3].

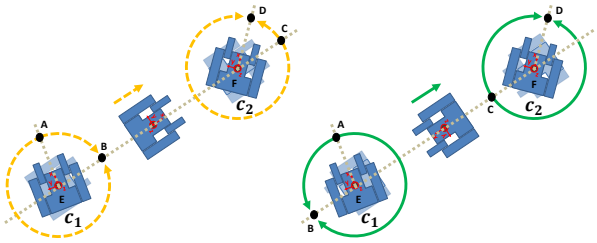
Algorithm 7: For BiTRRT [36] / BiDDTRRT
$\texttt{status} \leftarrow \texttt{ControlRefinement}(\mathcal{T}, \mathbf{c}_{\text{nearest}}, \mathbf{c}_{\text{rand}})$

1 $\texttt{dist} \leftarrow \texttt{ComputeDistance}(\mathbf{c}_{\text{nearest}}, \mathbf{c}_{\text{rand}})$;
2 **if** $\texttt{dist} < \delta$ **then**
3 $\quad$ **if** $\texttt{n\_refine\_nodes} > \rho \cdot \texttt{n\_total\_nodes}$ **then**
4 $\quad\quad$ **return** False;
5 $\quad$ **else**
6 $\quad\quad \texttt{n\_refine\_nodes} \leftarrow \texttt{n\_refine\_nodes} + 1$;
7 $\texttt{n\_total\_nodes} \leftarrow \texttt{n\_total\_nodes} + 1$;
8 **return** True;

In addition, one can bias the inherent exploration behavior that RRT-based algorithms have by using the ControlRefinement function proposed in [34]. If a randomly sampled configuration is close to any tree node, and the number of refinement nodes is above a threshold (ρ), then such nodes will be rejected.

Finally, assuming that $\mathbf{c}_{\text{start}}$ and $\mathbf{c}_{\text{goal}}$ lie in the same nonconvex, bounded, open, connected configuration space, the RRT (analogously, the BiRRT) algorithm guarantees the probabilistic completeness since the probability that the RRT (BiRRT) contains both $\mathbf{c}_{\text{start}}$ and $\mathbf{c}_{\text{goal}}$ approaches 1 as the number of vertices approaches infinity [33]. Next, the TRRT (analogously, the BiTRRT) must also guarantee the probabilistic completeness because the only difference from the RRT (analogously, the BiRRT) is that new samples may be rejected by the TestTransition and ControlRefinement functions, and the transition-success probability is strictly positive since the cost function is finite with subsequent bounded cost variations [34]. Further, the DDRRT (analogously, the BiDDRRT) also guarantees the probabilistic completeness because the lower bound (R) on the dynamic-domain radius value always ensures the possibility for a node to be extended [40]. For the same reasons that the BiTRRT and BiDDRRT are probabilistically complete,

[3] In the present problem, K is the range of the cost defined in (10) and ranges from 0 to 1.



(a) Forward motion (b) Backward motion

Figure 5: Two motion primitives are defined for the tracked mobile robot: linear and turn-in-place motions. With these primitives, the robot can move from $\mathbf{c_1}$ to $\mathbf{c_2}$ either forwardly or backwardly with clockwise or counterclockwise rotational motions about **E** and/or **F** to achieve the desired $\mathbf{c_1}$ and $\mathbf{c_2}$ configurations.

the `BiDDTRRT` must also be probabilistically complete.

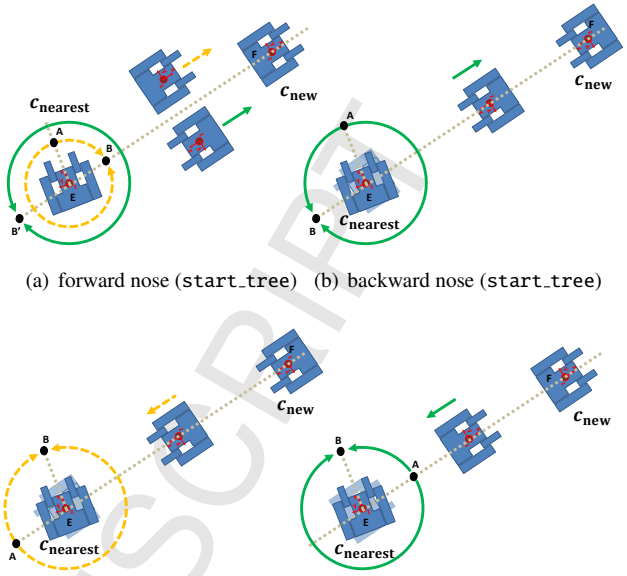
3.3. Motion primitives

As indicated in Section 3.1, the yaw angle is not included in the definition of samples picked in the path-planning procedure. This is because when the sampling-space dimension is increased by including the yaw angle, the paths found using the algorithms described in Section 3.2 generally involve large variations in yaw motions. To remedy this drawback, the concept of motion primitives composed by forward/backward linear motions and clockwise/counterclockwise angular motions is incorporated in the planning algorithm, which significantly reduces the variations in yaw motions and also increases the efficiency due to the reduction of the sampling-space dimension.

The set of the simplest motion primitives for a tracked mobile robot to move between two given configurations is the linear and turn-in place motions. As shown in Figure 5, for a given pair of configurations ($\mathbf{c_1}$, $\mathbf{c_2}$), the robot can move either forwardly (Figure 5(a)) or backwardly (Figure 5(b)) from $\mathbf{c_1}$ to $\mathbf{c_2}$. In each case, the robot can rotate clockwisely or counterclockwisely about the points **E** and/or **F** to achieve the desired robot poses while avoiding collision with non-traversable regions.

For the tree extension, the yaw angle of the $\mathbf{c}_{\text{nearest}}$ configuration is given from the previous tree extension, while the yaw angle for the $\mathbf{c}_{\text{new}}$ configuration needs to be chosen. The choice will be made based on the one that allows the robot to avoid collision with non-traversable regions along its whole motion between $\mathbf{c}_{\text{nearest}}$ and $\mathbf{c}_{\text{new}}$, giving priority to the one that involves the smallest motion range. If a tree is `start_tree`, the robot moves from the $\mathbf{c}_{\text{nearest}}$ to $\mathbf{c}_{\text{new}}$. At $\mathbf{c}_{\text{new}}$, the robot can stay along the same motion direction (Figure 6(a)) or opposite to it (Figure 6(b)). Between these two possibilities, the first one is preferable for subsequent tree extensions having its nose forwardly. Further, the first case (Figure 6(a)) can even be achieved with either forward motion or backward motion with a rotational motion about the point **F**.

On the other hand, if the considered tree is `goal_tree`, the robot moves from the $\mathbf{c}_{\text{new}}$ to $\mathbf{c}_{\text{nearest}}$ (that is, in the opposite direction done in `start_tree`). Once again, the yaw angle of $\mathbf{c}_{\text{new}}$ can be either along the same motion direction (Figure 6(c)) or opposite to it (Figure 6(d)), and the choice is made based on the one that avoids collisions with non-traversable regions



(a) forward nose (`start_tree`) (b) backward nose (`start_tree`)

(c) forward motion (`goal_tree`) (d) backward motion (`goal_tree`)

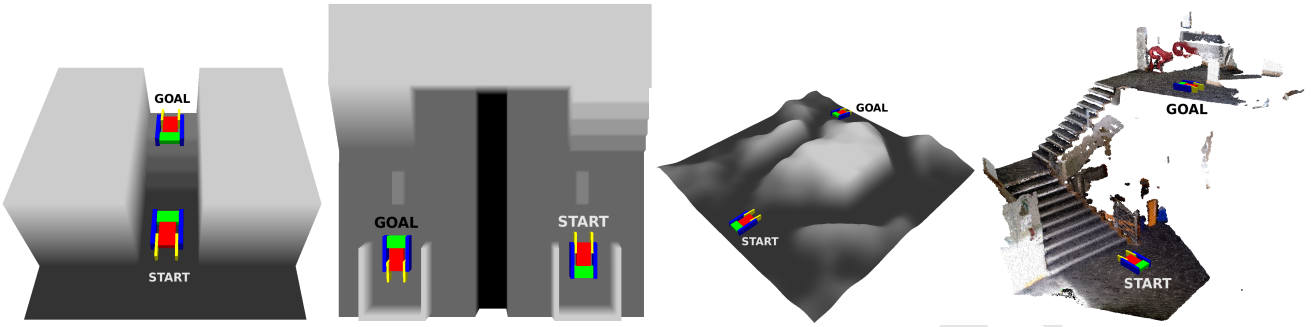
Figure 6: In tree extension, the robot moves from $\mathbf{c}_{\text{nearest}}$ to $\mathbf{c}_{\text{new}}$, if the tree is `start_tree` ((a) and (b)). During the tree extension, the yaw angle for the $\mathbf{c}_{\text{new}}$ is subject to be determined. For the case of `start_tree`, the robot can move forwardly or backwardly with clockwise or counterclockwise turn-in-place motion (depending on the results of collision checking), while the forward-nose case is preferred. If it moves backwardly, then it rotates about **F** to have the forward yaw angle for $\mathbf{c}_{\text{new}}$, if possible. If this is not possible (due to collision with non-traversable regions), then it stays in the back-nose configuration (shown in (b)). In its next extension, the planner will try to assign the forward-nose configuration (shown in (a)). On the other hand, for the case of `goal_tree`, the robot moves from $\mathbf{c}_{\text{new}}$ to $\mathbf{c}_{\text{nearest}}$ ((c) and (d)). This can be achieved by moving either forwardly (c) or backwardly (d) with clockwise or counterclockwise turn-in-place motion (depending again on the results of collision checking), while the forward motion is preferred.

along the whole motion from $\mathbf{c}_{\text{new}}$ to $\mathbf{c}_{\text{nearest}}$. Between these two possibilities, the former one is preferable for the same reason given for the case with the `start_tree`.

3.4. Comments on the flipper motion planning

The ability of tracked mobile robots to cross over obstacles is limited by their track sprocket radius, and this limitation can be improved by attaching flippers in the frontal side of the robot. Moreover, when tracked mobile robots interact with uneven terrains, they might experience sudden downward pitch motion which might cause large body impacts with possible damages. In these situations, the frontal flippers can be used to mitigate the body impacts by lowering them before the body pitch velocity becomes large.

Although it is true that the flippers' position that optimizes the tip-over stability measure might be different from when they are parallel to the main base, its value should not be too far from the flat configuration, especially when the length gained by the inclusion of the flippers is not large. In the view of the fact that the incorporation of the flippers to the robot is not mainly to increase the robot stability but to enlarge the robot's ability to cross over taller obstacles and to mitigate the robot body impact, the flipper position is not included in the sample space defined in the present path planning procedure, with the purpose



(a) Terrain I (4m × 4m) (b) Terrain II (7m × 7m) (c) Terrain III (8m × 8m) (d) Terrain IV (8.4m × 10.1m)

Figure 7: Four terrain maps are considered in the present work in order to evaluate the performance of the pose-estimation based path planner: (a) a narrow passage with a stair on one extreme and a crossing on the other (Terrain I); (b) a two-level flat terrain connected by a stair and a ramp, with a cliff, a deep pit, two crossable small steps and two parking lots (Terrain II); (c) a rough terrain with two passages: one being wide and the other, narrow (Terrain III); and, (d) a real 3D map of a two-level flat terrain connected by a stair, obtained by merging several point clouds captured using a 3D RGB sensor (Terrain IV).

to ease the path search and increase the planning efficiency (by the reduction of the sample space dimensionality).

Therefore, the initial setting of the flippers is parallel to the main base of the tracked mobile robot if possible. When the robot needs to cross objects that are higher than the sprocket's radius but within the reachable region with flippers, or when significant pitch motion is expected in the planning process, then the flippers' motion can be planned accordingly. This can be done by relating the desired flipper motion to the robot's body pitch angle change. Notice that the roll change is not considered because the flippers are coupled, and, therefore, their main function is to influence on the body pitch motion and not the roll motion. If the objective is also to affect the robot's roll motion, then the flippers must be independent from each other but this is not the case with *Cameleon* (see Figure 1).

4. Results and discussion

The pose estimation-based path-planning framework described in Section 3 is implemented in C++ using a PC with an Intel Xeon CPU (E5607) 2.27GHz with 24GB of memory RAM.

Four terrains (Figure 7) are considered to study the influence of the pose-estimation parameters, the `BiTRRT` and `BiDDTRRT` parameters, and the choice of the algorithm (between the `BiRRT`, `BiTRRT` and `BiDDTRRT`) on the planning performance. The search for a solution is terminated either when the first path that connects a given pair of the start and goal configurations is found or if no such path is found within 10^6 iterations.

The first terrain consists of a narrow passage with a stair and a crossing in its extremes, with the robot start configuration being in the narrow passage facing backwardly to the stair and its goal configuration being at the top level of the stair (Terrain I (Figure 7(a))). The passage width is not wide enough for the robot to make turn while it is in the passage. The aim for the consideration of such a terrain is to show that our planner can find solutions by using the crossing with its orthogonal passage.

The second terrain consists of a two-level flat terrain connected by a stair and a ramp, with a cliff, a deep pit, two crossable small steps and two parking lots (Terrain II (Figure 7(b))). The robot is initially parked in the right parking lot, and the goal is to park the robot in the parking lot of the left side in the map. This terrain map is considered to see whether the planner can generate a path that connects the start and the goal configurations through the stair and the ramp, while avoiding the pit, cliff, parking-lot walls and crossable small steps.

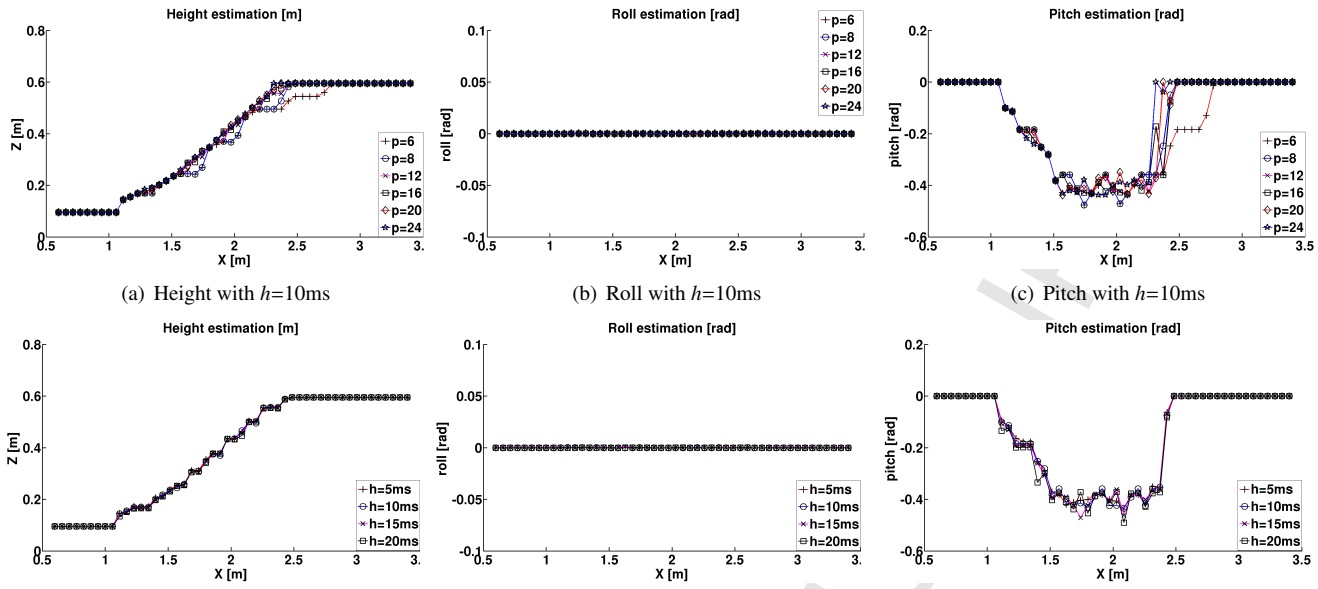
The third scenario is a rough terrain with two passages: one being wide and the other, narrow (Terrain III (Figure 7(c))). Initially, the robot starts from the right-down position in the map, and it has to reach the left-top position in the map. The aim of this study is to see whether the planner can plan a path that connects the start and the goal configurations through the longer-but-wider passage and not through the shorter-but-narrower passage (that involves changes in the robot's roll and pitch), while avoiding the dangerous mountainous regions.

Finally, the fourth case consists of a real 3D map built from a series of point cloud data captured using a 3D RGB sensor. This map consists of a two-level flat terrain connected by a stair (Terrain IV (Figure 7(d))). The robot starts from a position in the low flat level, and it has to reach a position on the high flat level. The goal of this study is to show that the planner is robust even with a real 3D map.

4.1. *Influence of the robot pose-estimation parameters on the planning performance*

In this section, the influence of the robot pose-estimation parameters on the planning performance is studied. The aim of this study is to find the smallest number of potential contact points (p) and the largest LCP discretization time step (h) with which the planner can quickly return a solution while guaranteeing an acceptable accuracy of the robot pose estimation.

First, the influence of the number of potential contact points (p) on the robot pose estimation is studied for a fixed LCP discretization time step (h=10ms). In this study, the robot pose is estimated as the robot model is asked to climb a stair following a piecewise-linear path with a uniform length step along the longitudinal direction of the stair (Figure 9). The height, roll and pitch angles are respectively represented in Figure 8(a), Figure 8(b), and Figure 8(c), for various numbers of potential



(a) Height with h=10ms (b) Roll with h=10ms (c) Pitch with h=10ms

(d) Height with p=12 (e) Roll with p=12 (f) Pitch with p=12

Figure 8: The results of solving the robot pose estimation problem are shown when the robot model climbs the stair shown in Figure 9. (a) - (c) are the height, roll and pitch angles against the X-direction for various numbers of potential contact points (p) and for h=0.01s, as of the LCP discretization time step. (d) - (f) are the height, roll and pitch angles against the X-direction for various LCP discretization time steps and for p=12, as of the number of potential contact points.

contact points ($p \in \{6, 8, 12, 16, 20, 24\}$). Among these potential contact points, two are attributed to the flippers (one per flipper on the mid-point of their respective longitudinal mid-axis), and the rest, to the two tracks.

Figure 8(a) shows that the progress of the robot's CoM height is similar for all the p values except for p=(6, 8). While the progress of the roll angle is invariant from p (Figure 8(b)), significant pitch angle difference can be observed for p=6, especially at the end of the stair. More detailed results are presented in Table 1. The results shown in this table are the root mean square (RMS) of the difference between the reference height-roll-pitch (p=24) and the height-roll-pitch of the remaining numbers of potential contact points. These results indicate that in effect the RMS errors are significantly larger for the height when p=(6, 8) than for the rest of the cases. On the other hand, the errors along the roll direction are nearly zero because in this study the robot model is asked to move along the longitudinal direction of the stair.

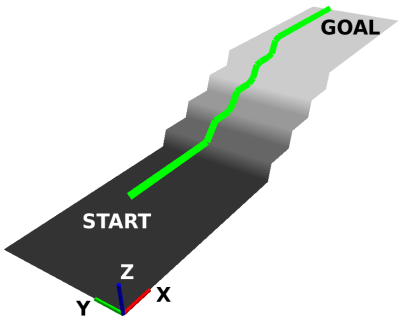


Figure 9: Stair climbing following a piecewise linear path for evaluating the performance of robot pose estimation for various p and h values.

In addition, the influence of the LCP discretization time step (h) on the robot pose estimation is studied for a fixed number of potential contact points (p=12) while the robot is asked to climb the stair. The corresponding height, roll and pitch angles are respectively shown in Figure 8(d), Figure 8(e), and Figure 8(f), for various LCP discretization time step values ($h \in \{5, 10, 15, 25\}$ms). Contrary to the previous study (in which the number of potential contact points is varied having the LCP discretization time step fixed), when the discretization time step is varied having the number of potential contact points fixed, the progress of height-roll-pitch values seems to be very similar for all the considered time steps. In effect, Table 2 shows that the RMS errors of the height-roll-pitch values for $h \in \{10, 15, 25\}$ms computed with respect to the reference height-roll-pitch values (h=5ms) are small in general. These results indicate that the robot pose estimation is more robust in the changes of the LCP discretization time steps than in the changes of the number of potential contact points.

Next, the influence of the combination of the number of potential contact points (p) and the LCP discretization time step (h) on the path-planning time is studied. For this study, $p \in \{6, 8, 12, 16, 20, 24\}$ and $h \in \{5, 10, 15, 20\}$ms are consid-

Table 1: Root mean squared error of the robot pose estimation with respect to the case of (p=24, h=10ms)

Error \ p	**6**	**8**	**12**	**16**	**20**
$\text{height}_{\text{RMS}}$ [m]	0.034	0.027	0.012	0.006	0.005
roll_{RMS} [rad]	0.000	0.000	0.000	0.000	0.000
$\text{pitch}_{\text{RMS}}$ [rad]	0.099	0.066	0.072	0.056	0.058

Table 2: Root mean squared error of the robot pose estimation with respect to the case of (p=12, h=5ms)

Error \ h	10ms	15ms	20ms
$\text{height}_{\text{RMS}}$ [m]	0.0029	0.0031	0.0044
roll_{RMS} [rad]	0.0000	0.0002	0.0000
$\text{pitch}_{\text{RMS}}$ [rad]	0.0098	0.0153	0.0212

ered over Terrain II shown in Figure 7(b). The path search is performed using the `BiTRRT` algorithm with T_{rate} = 0.05 and $T_o = 1e^{-6}$, and the search is repeated one-hundred times with different seeds for random number generation for each p and h values. The results shown in Figure 10 indicate that the planning time increases as the number of potential contact points increases for all the considered LCP discretization time steps. They also manifest that the planning time decreases as the LCP discretization time step increases for all the considered numbers of potential contact points. Further, these results suggest once more that the planning time is more sensitive to the changes on the number of potential contact points than to the LCP discretization time step. Moreover, they show that for smaller h values the planning time increases more quickly as the number of potential contact points increases.

After all, the results shown in Figure 8, Figure 10, Table 1, and Table 2 seem to suggest that p=12 and h=10ms are acceptable pose-estimation parameter values for quickly finding a solution with an acceptable accuracy of the robot pose estimation.

4.2. *Influence of the* `BiTRRT` *algorithm parameters and of the cost functions on the planning performance*

In this section, the influence of the `BiTRRT` parameters and of the cost functions on the planning performance such as the planning time, number of trees' nodes, path cost, path length and the success rates[4], with p=12 and h=10ms (as suggested from Section 4.1).

First, as presented in Algorithm 6 (following the approach proposed by [36]), the `testTransition` function uses one parameter and one initial value: $\text{T}_{\texttt{rate}}$ and T_o, respectively. The initial value for the "temperature" (T) is typically set to be a small value with the purpose to initially restrict the sample region to be of low cost. $T_o = 1e^{-6}$ is used for the present study. On the other hand, the parameter related to the `controlRefinement` (ρ) is fixed with the value suggested by [34] and [36] (i.e., $\rho = 0.1$). Thus, the parameter T_{rate} is left to study its influence on the planning performance, as shown posteriorly.

Second, two types of cost functions are used in the present study: the *Minimal Work (MW)* proposed by [34] and the tip-over stability-related cost function proposed in the present

[4]The success rate is defined as the ratio of the number of trials in which a solution is found within L=10^6 iterations to the total number of trials.

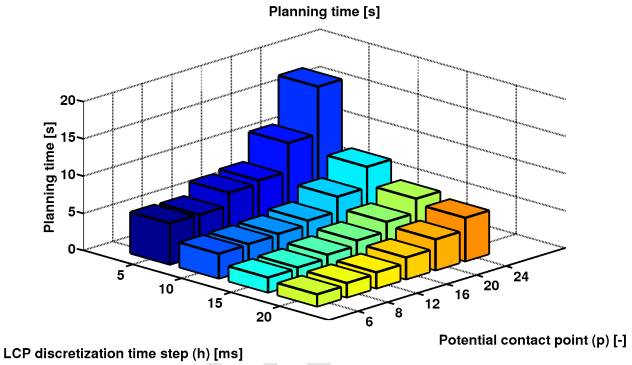


Figure 10: Influence of the pose-estimation parameters (the number of potential contact points (p) and the LCP discretization time step (h)) on the planning time.

work. Recall from [34] that a path that minimizes the mechanical work for a given query is called the MW path. The mechanical work of a path is defined in [34] as

$$W(\mathcal{P}) = \sum_i \Delta v_i^+ + \epsilon l, \tag{11}$$

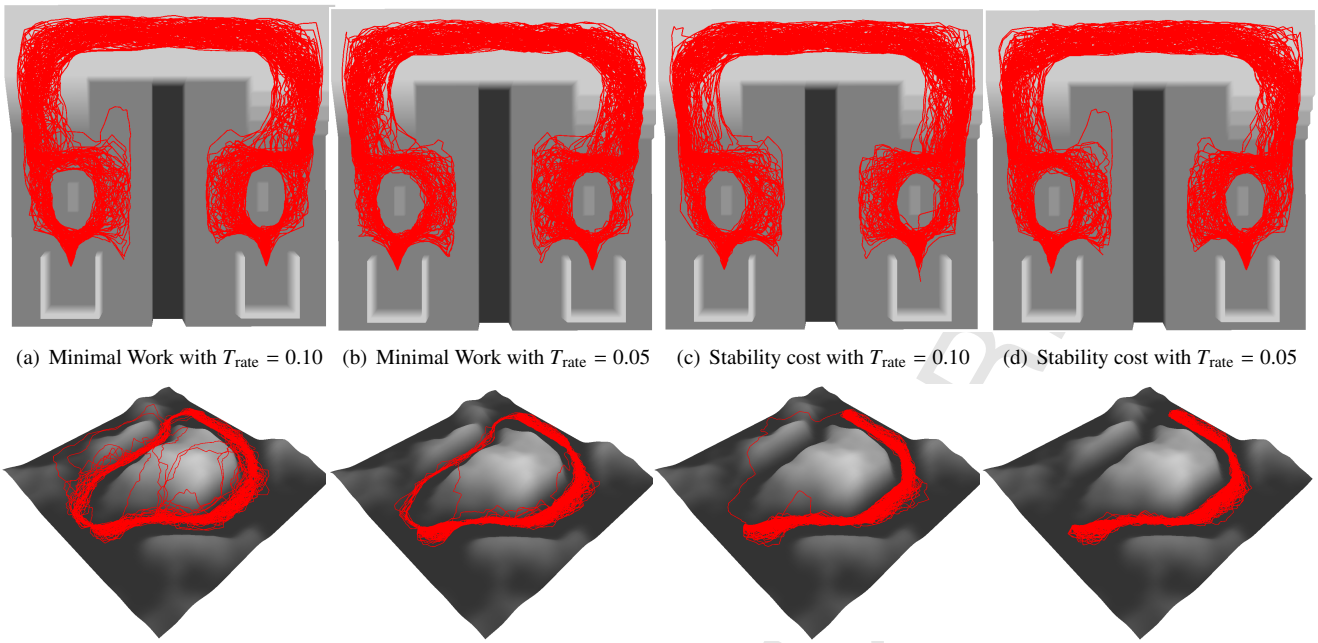
where Δv_i^+ is the positive cost difference between two configurations in the path $\mathcal{P}$. ϵl is an additive term considered to favor shortest paths, where ϵ is a small positive real number and l is the distance between two configurations in the path. In this study, v_i is considered as the robot height, and $\epsilon = 10^{-5}$ is used.

Therefore, in the present study, solutions are searched using the `BiTRRT` algorithm over Terrain II (Figure 7(b)) and Terrain III (Figure 7(c)) with the Minimal Work (11) and the stability-related cost function (10), and with two T_{rate} values: 0.1 and 0.05. One-hundred solution paths are found over Terrain II and Terrain III for the following four scenarios (11): a) Minimal Work with T_{rate} = 0.10; b) Minimal Work with T_{rate} = 0.05; c) stability cost with T_{rate} = 0.10; and d) stability cost with T_{rate} = 0.05.

For both Terrain II and Terrain III, the planner is able to plan paths to connect the start and the goal configurations while avoiding the crossable steps for all one-hundred simulations.

For Terrain II, at the first glance, the T_{rate} does not seem to affect the stability results for both cost functions (see the first row of Figure 11 and Table 3). However, when one gives a special attention around the corners of the stair and the ramp in Terrain II, one can realize that the paths obtained using the stability cost are further away from these corners than when the Minimal Work is used. And, this difference is more accentuated with T_{rate} = 0.05 than with T_{rate} = 0.10. This observation becomes more clear by the difference of the stability cost values indicated in Table 3 for each of the cost functions and of the T_{rate} values. All the values given in Table 3 are the average of the one-hundred solution paths for each case. The best stability result is achieved when paths are planned using the stability cost with T_{rate} = 0.05, and the worst stability result is obtained with the Minimal Work and T_{rate} = 0.10.

For Terrain III, the difference of results obtained using the Minimal Work and the stability cost becomes more evident.



(a) Minimal Work with $T_{\text{rate}} = 0.10$ (b) Minimal Work with $T_{\text{rate}} = 0.05$ (c) Stability cost with $T_{\text{rate}} = 0.10$ (d) Stability cost with $T_{\text{rate}} = 0.05$

(e) Minimal Work with $T_{\text{rate}} = 0.10$ (f) Minimal Work with $T_{\text{rate}} = 0.05$ (g) Stability cost with $T_{\text{rate}} = 0.10$ (h) Stability cost with $T_{\text{rate}} = 0.05$

Figure 11: One-hundred solution paths obtained using the `BiTRRT` algorithm (with different seeds for random number generation) are shown over Terrain II and Terrain III, using both the *Minimal Work (MW)* and the *tip-over stability-related cost* function with two *temperature rate* (T_{rate}) values: 0.1 and 0.05.

Table 3: Influence of T_{rate} and of the cost functions on the planning performance

Terrain	**Cost**	T_{rate}	**Iterations**	**# Nodes**	**Time[s]**	**Path length[m]**	$\mathcal{J}(=(\mathbf{1}-\hat{\xi}))$ **[-]**	**MW**	**Success rate [%]**[a]
Hybrid	MW	0.10	9068.744	766.611	1.086	18.931	4.202	0.506	100.0
	MW	0.05	14404.867	1065.144	1.671	19.210	4.356	0.503	100.0
	Stability	0.10	12179.644	815.789	1.343	18.872	3.353	0.546	100.0
	Stability	0.05	16929.478	1293.067	2.104	19.042	3.360	0.548	100.0
Rough	MW	0.10	941.422	387.278	0.662	11.421	20.437	0.845	100.0
	MW	0.05	1255.356	351.778	0.748	11.801	8.433	0.538	100.0
	Stability	0.10	1145.278	375.756	0.868	12.778	3.800	1.034	100.0
	Stability	0.05	1387.611	328.867	0.929	12.732	2.610	0.921	100.0

[a] The success rate is defined as the ratio of the number of trials in which a solution is found within L=10^6 iterations to the total number of trials.

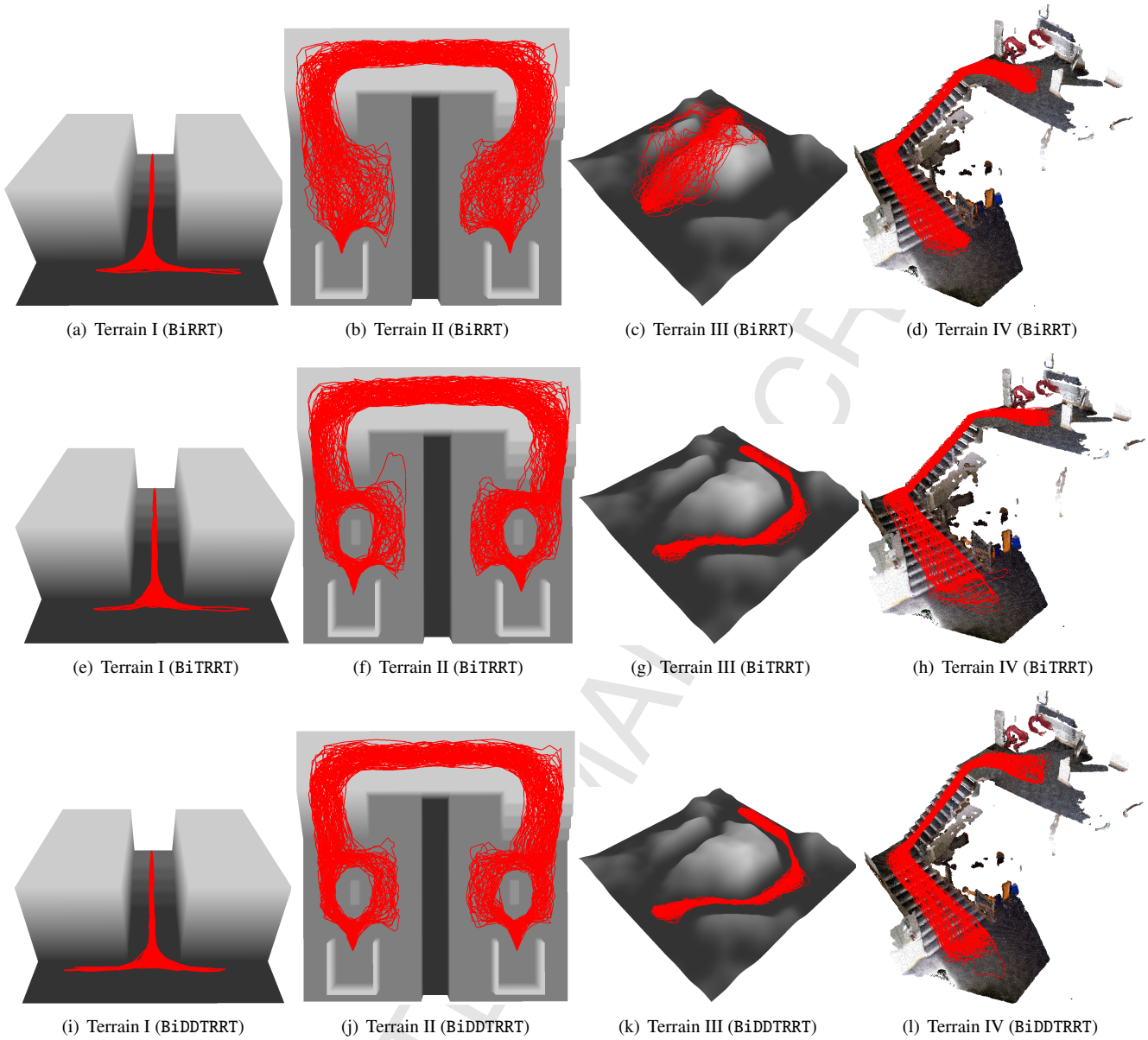
When the planner uses the Minimal Work as its cost function, it seems to generally choose the paths along both the wide and the narrow passages, whereas when it uses the stability cost function, the planner generally chooses the paths along the wide passage. The robot suffers from roll and pitch changes when it passes through the narrow passage. On the other hand, the planning performance is clearly sensitive to the choice of T_{rate}. For both types of cost function, the smaller T_{rate} is, the lower Minimal Work corresponds to the solution path and the more stable is the solution path (see Figure 11(e) - 11(h) and Table 3).

In addition, notice that the smaller T_{rate} is, the more refined is the path search. This is because, as indicated in Algorithm 7, the smaller T_{rate} is, the more slowly the temperature raises, and, therefore, overconstrains the growth of the path search area in the traversable sample space than when the T_{rate} is larger. The solutions found with a lower T_{rate} value are stabler (but longer in length in this case) requiring longer time to plan than those found with a higher T_{rate} value.

Finally, the results obtained using the proposed stability cost function seem to give better planning-performance results than using the Minimal Work, and this aspect becomes more clear with the smaller T_{rate} value.

4.3. Comparison between the results obtained using the `BiRRT`, `BiTRRT` *and* `BiDDTRRT` *algorithms*

In this section, the planning-performance results obtained using the `BiRRT`, `BiTRRT` and `BiDDTRRT` algorithms are compared for the considered four terrain maps in terms of the planning time, number of trees' nodes, path cost, path length and the success rates. While the `BiRRT` algorithm does not consider cost function in the planning procedure, both the `BiTRRT` and `BiDDTRRT` algorithms use the tip-over stability-related cost function to instantaneously restrict the sample region for the solution search. These algorithms are employed to plan the robot's motion path over the four terrains shown in Figure 7.



(a) Terrain I (BiRRT) (b) Terrain II (BiRRT) (c) Terrain III (BiRRT) (d) Terrain IV (BiRRT)

(e) Terrain I (BiTRRT) (f) Terrain II (BiTRRT) (g) Terrain III (BiTRRT) (h) Terrain IV (BiTRRT)

(i) Terrain I (BiDDTRRT) (j) Terrain II (BiDDTRRT) (k) Terrain III (BiDDTRRT) (l) Terrain IV (BiDDTRRT)

Figure 12: The solution paths obtained using the BiRRT, BiTRRT and BiDDTRRT algorithms are represented over each of the four terrain elevation maps. The results shown in each row correspond to those obtained using three different RRT-based algorithms: (a) - (d) BiRRT, (e) - (h) BiTRRT, and (i) - (l) BiDDTRRT.

Due to the fact that all these algorithms do not guarantee the optimality of the solutions, for each algorithm the path search is performed one-hundred times with different random seeds, and the results shown in Table 4 represent the averaged values. For the BiDDTRRT algorithm, $R{=}\lambda\varepsilon$ is used as suggested by [39, 40], where ε is the *interpolation step* with $\lambda{=}10$.

First, the illustrations shown in each row of Figure 12 correspond to the solution paths found over the considered four terrain maps using the BiRRT, BiTRRT and BiDDTRRT algorithms, respectively. Figure 12 does not show a clear difference between the solutions found by the three algorithms on Terrain I and Terrain IV due to the fact that the path-search regions on these terrain maps are restricted (i.e., narrow passages and stairs)[5]. However, the results obtained for Terrain II and Terrain III show a clear difference in terms of the solution quality depending on the employed path-planning algorithm.

On one hand, the study with Terrain II shows that the BiTRRT and BiDDTRRT algorithms can always find paths that avoid crossable steps (i.e., steps with height values that are within the reachability range using the flippers). Whereas, the BiRRT algorithm can not guarantee to obtain paths that avoid these steps. On the other hand, the study on Terrain III shows that both the BiTRRT and BiDDTRRT are again always able to avoid dangerous regions for the choice made for the parameter values (indicated in Section 4.2), while the BiRRT always fails to avoid the dangerous regions since this algorithm does not take into account any cost function.

[5]Nonetheless, the planning-performance results obtained for Terrain I and Terrain IV are clearly different, as shown in Table 4 and in Figure 13

Table 4: Comparison of the planning performance results between the BiRRT, BiTRRT, and BiDDTRRT

Terrain	Algorithm	Iterations [-]	# Nodes [-]	Time [s]	Path length [m]	$\mathcal{J}(=(1-\hat{\xi}))$ [-]	Success rate [%][a]
Terrain I	BiRRT	67352.106	291.847	2.500	4.361	1.801	95.0
	BiTRRT	107083.180	346.281	4.443	4.264	1.782	99.0
	BiDDTRRT	19336.600	132.000	1.852	4.290	1.031	100.0
Terrain II	BiRRT	5214.678	444.356	0.545	17.689	6.311	100.0
	BiTRRT	16929.478	1293.067	2.104	19.042	3.360	100.0
	BiDDTRRT	10809.400	1887.400	4.813	18.091	2.506	100.0
Terrain III	BiRRT	339.522	311.989	0.335	9.981	34.752	100.0
	BiTRRT	1387.611	328.867	0.929	12.732	2.610	100.0
	BiDDTRRT	1326.400	242.000	2.209	12.250	0.907	100.0
Terrain IV	BiRRT	7875.644	1214.611	1.193	15.042	31.674	100.0
	BiTRRT	103657.221	1389.426	15.704	15.054	26.152	78.0
	BiDDTRRT	8821.800	373.400	5.683	14.023	13.329	100.0

[a] The success rate is defined as the ratio of the number of trials in which a solution is found within L=10^6 iterations to the total number of trials.

Table 4 shows the planning-performance results obtained using the BiRRT, BiTRRT, and BiDDTRRT algorithms. The results show that the BiTRRT requires a larger number of iterations, with a smaller ratio between the accepted nodes and the sampled nodes (i.e., (number of nodes)/Iterations) to find a solution. This is mainly due to the node-rejection mechanism that the BiTRRT algorithm has by restricting the sample region (using the TestTransition function) and by controlling the refinement of the search tree (using the controlRefinement function). As a result, all the paths obtained by the BiTRRT have better tip-over stability-related cost values than those obtained using the BiRRT, even though in some cases the length of the obtained paths are larger than those obtained from the BiRRT algorithm. The success rates obtained within L=10^6 iterations are nearly 100% for all the cases, except for the case of Terrain IV with the BiTRRT algorithm (see Table 4 and Figure 13). For this last case, even the planning time is significantly larger than when the BiRRT algorithm is employed. These poor results may be due to the fact that many new samples have boundary nodes as their nearest neighbors (which have associated large Voronoi regions) and that the mechanism of restricting the sample region with the TestTransition function does not suit well to this type of problems. This observation motivated the authors to consider the dynamic-domain aspect [39, 40], which reduces the probability that a boundary node is chosen to be the nearest neighbor to a new sample, by restricting its corresponding sample region. The incorporation of this aspect to the BiTRRT originated a new algorithm called hereby as BiDDTRRT, and it significantly improved both the success rate and the planning time for the Terrain IV.

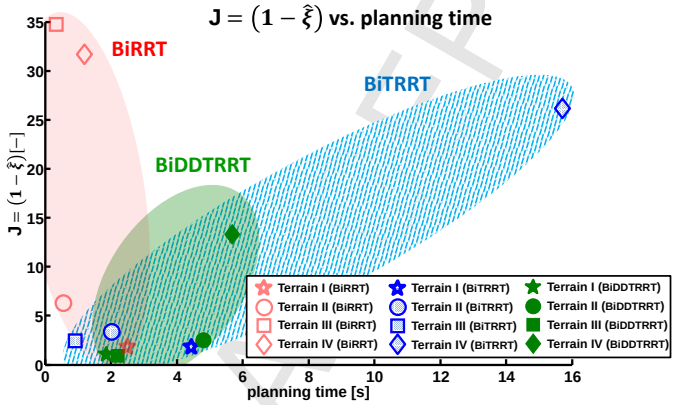


Figure 13: The tip-over stability-related cost values are plotted against the planning time for all the four scenarios using the BiRRT, BiTRRT and BiDDTRRT algorithms. These values are precisely those shown in Table 4, which are averaged values over one-hundred different simulations.

In fact, both Table 4 and Figure 13 show that, in terms of the average cost, the BiDDTRRT outperforms both the BiRRT and BiTRRT. On the other hand, in terms of the planning time, BiDDTRRT performs better than BiTRRT on Terrain I and Terrain IV and even better than BiRRT on Terrain I. The relatively small values for the average number of nodes for both the Terrain I and Terrain IV clearly indicate the effect of the dynamic-domain. This effect is more accentuated for Terrain I, where most of the nodes in the corridor are in proximity with the non-traversable regions (the walls).

Finally, Figure 13 shows three clustering regions for each planning algorithm in terms of the stability-related cost and the planning time. This is another way to visualize the key results shown in Table 4. The results show that the BiRRT algorithm returns solutions at lower planning-time values but with higher stability-related cost values, while the BiTRRT algorithm returns solutions with lower stability-related cost but at the price of larger planning time in some cases. On the other hand, the results obtained using the BiDDTRRT correspond to the lowest cost values with the highest success rates, while compromising the planning time.

5. Conclusion and future work

A novel path-planning algorithm is proposed in the present work for a tracked mobile robot to traverse uneven terrains by combining two RRT-like algorithms (the *transition-basd*

RRT (T-RRT) [34] and the *dynamic-domain RRT* (DD-RRT) [39]) and by representing the robot-terrain interaction with the robot's tip-over stability. In order to increase the planning efficiency, the resulting algorithm is used bidirectionally, and it is called as the *bidirectional dynamic-domain transition-based RRT* (`BiDDTRRT`). On the other hand, the robot's tip-over stability is computed quasi-statically by first solving the problem of the robot pose estimation over uneven terrains (assuming that the robot's speed is low for safety), which is interpreted as a contact problem, formulated as a linear complementarity problem (LCP) and solved using the Lemke's method.

Then, the performance of the `BiDDTRRT` is compared to other RRT-like algorithms such as the bidirectional RRT (`BiRRT`) and bidirectional transition-based RRT (`BiTRRT`) algorithms over various uneven terrains. The comparison study between these three algorithms manifests that the `BiRRT` algorithm quickly finds solutions, but the corresponding stability-related cost values are high, since it does not use the knowledge of the associated cost. On the other hand, the `BiTRRT` can find solutions with better stability-related cost values than the `BiRRT`, but its success rate can be poor if the number of *boundary nodes* is large, requiring significant amount of planning time in some cases. The drawbacks of the `BiTRRT` can be mitigated by incorporating the dynamic-domain aspect to the `BiTRRT` (i.e., `BiDDTRRT`), as we suggest in the present work, in order to improve the success rate, the solution quality (in terms of the stability of the overall path) and the planning time.

Further, the performance of planning paths with a stability-related cost function is compared to that of planning with the cost consisting of the Minimal Work, and the results show that the solutions found with the stability-related cost function have a stronger bias towards safer regions.

In the near future, the framework described in the present work will be implemented on a tracked mobile robot (Figure 1) to traverse uneven terrains with the purpose to experimentally validate the results shown hereby.

ACKNOWLEDGMENT

This work is partially supported by the RAPID-FRAUDO project (Num. 112906242) funded by the DGA (French Defence Agency). The authors deeply appreciate Anis Sahbani for the discussions maintained during the early phase of the present work, and Andrew Comport and Maxime Meilland, for sharing the point cloud data used in Figure 7(d), Figure 12(d), Figure 12(h), and Figure 12(l). Jae-Yun Jun appreciates Alexandre Eudes for his several comments on the low-level implementation of the present work.

[1] D. B. Gennery, Traversability analysis and path planning for a planetary rover, Autonomous Robots 6 (2) (1999) 131–146.

[2] M. Tarokh, Z. Shiller, S. Hayati, A comparison of two traversability based path planners for planetary rovers, in: Proc. of the Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), Noordwijk, Netherlands, 1999, pp. 151–7.

[3] H. Seraji, Traversability index: a new concept for planetary rovers, in: Proc. of IEEE International Conference on Robotics and Automation, Vol. 3, Detroit, MI, USA, 1999, pp. 2006–2013.

[4] G. Ishigami, A. Miwa, K. Nagatani, K. Yoshida, Terramechanics-based model for steering maneuver of planetary exploration rovers on loose soil, Journal of Field Robotics 24 (3) (2007) 233–250.

[5] G. Ishigami, K. Nagatani, K. Yoshida, Path planning for planetary exploration rovers and its evaluation based on wheel slip dynamics, in: Proc. of IEEE International Conference on Robotics and Automation, Rome, Italy, 2007, pp. 2361–2366.

[6] G. Ishigami, K. Nagatani, K. Yoshida, Path planning and evaluation for planetary rovers based on dynamic mobility index, in: Proc. of IEEE/RSJ International Conferference on Intelligent Robots and Systems, 2011.

[7] Y. Liu, G. Liu, Modeling of tracked mobile manipulators with consideration of track-terrain and vehicle-manipulator interactions, Robotics and Autonomous Systems 57 (11) (2009) 1065–1074.

[8] T. Kubota, Y. Kuroda, Y. Kunii, T. Yoshimitsu, Path planning for newly developed microrover, in: Proc. of IEEE International Conference on Robotics and Automation, Seoul, South Korea, 2001, pp. 3710–3715.

[9] A. Hait, T. Siméon, M. Taix, Algorithms for rough terrain trajectory planning, Advanced Robotics 16 (8) (2002) 673–699.

[10] C. Beck, J. Miro, G. Dissanayake, Trajectory optimisation for increased stability of mobile robots operating in uneven terrains, in: Proc. of IEEE International Conference on Control and Automation, 2009, pp. 1913–19.

[11] Open Dynamics Engine, http://www.ode.org/.

[12] M. Norouzi, J. Valls Miro, G. Dissanayake, A statistical approach for uncertain stability analysis of mobile robots, in: Proc. of IEEE International Conference on Robotics and Automation, 2013.

[13] E. Papadopoulos, D. Rey, A new measure of tipover stability margin for mobile manipulators, in: Proc. of IEEE International Conference on Robotics and Automation, Minneapolis, MN, USA, 1996, pp. 3111–16.

[14] P. R. Roan, A. Burmeister, A. Rahimi, K. Holz, D. Hooper, Real-world validation of three tipover algorithms for mobile robots, in: Proc. of IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 2010, pp. 4431–4436.

[15] E. Papadopoulos, D. A. Rey, Force-angle measure of tipover stability margin for mobile manipulators, Vehicle System Dynamics 33 (1) (2000) 29–48.

[16] S. A. A. Moosavian, K. Alipour, Stability evaluation of mobile robotic systems using moment-height measure, in: Proc. IEEE Conference on Robotics, Automation and Mechatronics, Bangkok, Thailand, 2006.

[17] J. Kim, W. K. Chung, Y. Youm, B. H. Lee, Real-time ZMP compensation method using null motion for mobile manipulators, in: Proc. IEEE International Conference on Robotics and Automation, Washington, DC, United states, 2002, pp. 1967–1972.

[18] J. Casper, R. R. Murphy, Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 33 (3) (2003) 367–385.

[19] W. Lee, S. Kang, M. Kim, M. Park, ROBHAZ-DT3: teleoperated mobile platform with passively adaptive double-track for hazardous environment applications, in: Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004.

[20] B. M. Yamauchi, PackBot: a versatile platform for military robotics, in: Proc. of SPIE 5422, Unmanned Ground Vehicle Technology VI, 228, September 2, 2004.

[21] M. Guarnieri, P. Debenest, T. Inoh, E. Fukushima, S. Hirose, Development of HELIOS VII: an arm-equipped tracked vehicle for search and rescue operations, in: Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004.

[22] Z. Shiller, Y.-R. Gwo, Dynamic motion planning of autonomous vehicles, IEEE Transactions on Robotics and Automation 7 (2) (1991) 241–249.

[23] R. O. Stone, J. Dugundji, A study of microrelief: its mapping, classification, and quantification by means of a fourier analysis, Engineering Geology 1 (2) (1965) 89–187.

[24] R. Hoffman, E. Krotkov, Terrain roughness measurement from elevation maps, in: Proc. of SPIE - Int. Soc. Opt. Eng., Vol. 1195, 1990, pp. 104–14.

[25] R. Cottle, J. Pang, R. Stone, The Linear Complementarity Problem, Society for Industrial and Applied Mathematics, 2009.

[26] T. Siméon, B. Dacre-Wright, A practical motion planner for all-terrain mobile robots, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 1993, pp. 1357–63.

[27] A. Kelly, A. Stentz, Rough terrain autonomous mobility – part 2: an active vision, predictive control approach, Autonomous Robots 5 (2) (1998)

163–198.
[28] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, Principles of Robot Motion: Theory, Algorithms, and Implementations, MIT Press, 2005.
[29] S. LaValle, Planning Algorithms, Cambridge University Press, 2006.
[30] J. H. Reif, Complexity of the movers problem and generalizations, in: The 20th Annual IEEE Conference on Foundations of Computer Science, 1979.
[31] S. M. LaValle, Rapidly-exploring random trees: a new tool for path planning, Tech. Rep. TR 98-11, Department of computer science, Iowa State University (1998).
[32] L. E. Kavraki, P. Svestka, J.-C. Latombe, M. H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Transactions on Robotics and Automation 12 (4) (1996) 566–580.
[33] S. M. LaValle, J. J. Kuffner, Randomized kinodynamic planning, The International Journal of Robotics Research 20 (5) (2001) 378–400.
[34] L. Jaillet, J. Cortés, T. Siméon, Sampling-based path planning on configuration-space costmaps, IEEE Transactions on Robotics 26 (4) (2010) 635–646.
[35] R. Iehl, J. Cortés, T. Siméon, Costmap planning in high dimensional configuration spaces, in: Proc. of IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 2012.
[36] D. Devaurs, T. Siméon, S. Cortés, Enhancing the transition-based RRT to deal with complex cost spaces, in: Proc. of IEEE International Conference on Robotics and Automation, 2013.
[37] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, The International Journal of Robotics Research 30 (7) (2011) 846–894.
[38] B. Akgun, M. Stilman, Sampling heuristics for optimal motion planning in high dimensions, in: Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011.
[39] A. Yershova, L. Jaillet, T. Siméon, S. M. La Valle, Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain, in: Proc. of IEEE International Conference on Robotics and Automation, 2005.
[40] L. Jaillet, A. Yershova, S. M. LaValle, T. Siméon, Adaptive tuning of the sampling domain for dynamic-domain RRTs, in: Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005.
[41] Cameleon EOD, Eca Robotics: http://www.eca-robotics.com/en/robotic-vehicle/robotics-terrestrial-unmanned-ground-vehicles-(ugv)-cameleon-eod-lightweight-eod-ugv/22.htm.
[42] M. Mortenson, Geometric Modeling, John Wiley and Sons, 1985.
[43] S. Beucher, C. Lantuejoul, Use of watersheds in contour detection, in: Proc. of Int. Workshop Image Processing, Real-Time Edge and Motion Detection/Estimation, 1979.
[44] S. Suzuki, Topological structural analysis of digitized binary images by border following, Computer Vision, Graphics, and Image Processing 30 (1985) 32–46.
[45] T.-Y. Chyou, Country interior: Two solutions to the nonconvex polygon interior problem, Wolfram Demonstrations Project.
[46] P. R. Kraus, V. Kumar, Compliant contact models for rigid body collisions, in: Proc. of IEEE International Conference on Robotics and Automation, Albuquerque, NM, USA, 1997, pp. 1382–1387.
[47] D. Marhefka, D. Orin, A compliant contact model with nonlinear damping for simulation of robotic systems, IEEE Transactions on Systems, Man & Cybernetics, Part A (Systems & Humans) 29 (6) (1999) 566–72.
[48] D. Stewart, J. Trinkle, An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and Coulomb friction, International Journal for Numerical Methods in Engineering 39 (15) (1996) 2673–91.
[49] J. Trinkle, S. Berard, J. Pang, A time-stepping scheme for quasistatic multibody systems, in: Proc. of IEEE International Symposium on Assembly and Task Planning, Montreal, QC, Canada, 2005, pp. 174–181.
[50] J. E. Lloyd, Fast implementation of Lemke's algorithm for rigid body contact simulation, in: Proc. of IEEE International Conference on Robotics and Automation, Barcelona, Spain, 2005, pp. 4538–4543.
[51] A. T. Miller, P. K. Allen, GraspIt!: A versatile simulator for robotic grasping, IEEE Robotics and Automation Magazine 11 (4) (2004) 110–122.
[52] M. Anitescu, F. Potra, Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems, Nonlinear Dynamics 14 (3) (1997) 231–247.
[53] K. Egan, S. Berard, J. Trinkle, Modeling nonconvex constraints using linear complementarity, Tech. Rep. 03-13, Rensselaer Polytechnic Institute, Department of Computer Science (December 2003).
[54] K. Iagnemma, S. Dubowsky, Mobile Robots in Rough Terrain: Estimation, Motion Planning, and Control with Application to Planetary Rovers, Springer, 2004.
[55] J. J. Kuffner Jr., S. M. LaValle, RRT-connect: an efficient approach to single-query path planning, in: Proc. of IEEE International Conference on Robotics and Automation, 2000, pp. 995–1001.

Jae-Yun JUN's short biography

Jae-Yun Jun received the B.S. degree in Telecommunications in 2004 from Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona (ETSETB) at Universitat Politècnica de Catalunya (UPC), Spain. He received the M.S. degree in Electrical Engineering in 2005 from the University of Pennsylvania, USA, and the Ph.D. degree in Mechanical Engineering in 2011 from the Florida State University, USA. He is currently a Posdoctoral Researcher in the Institut des Systèmes Intelligents et de Robotique (ISIR) at Université Pierre et Marie Curie (UPMC), Paris, France. His research interests include robot motion planning, trajectory tracking, legged locomotion and dynamic modeling.

Jean-Philippe SAUT's short biography

Jean-Philippe Saut received the M.S. degree in robotics in 2003 from University Pierre et Marie Curie (UPMC), Paris, France. He received the Ph.D. degree in 2007 from UPMC. He continued his work in the field of robotic manipulation during a postdoctoral fellowship at the Laboratoire d'Analyse et d'Architecture des Systemes-Centre National de la Recherche Scientifique (LAAS-CNRS), France, and another at the Institut des Systemes Intelligents et de Robotique (ISIR) in UPMC. His research interests include dexterous manipulation, grasp planning, manipulation planning and assistive robotics.

Faïz BEN AMAR's short biography

Faïz Ben Amar was born in Sfax, Tunisia in 1965. He is graduate from the Ecole Nationale Supérieure des Arts et Métiers (France) and he received PhD degree in 1994 from the Université Pierre et Marie Curie - UPMC. He is currently full professor at UPMC. Dr. Faïz Ben Amar developed his researches in the field of design and control of high mobility locomotion systems and self-reconfigurable modular systems. He is interested in modeling and simulation of multibody systems interacting with complex environment.

Photo of Jae Yun JUN KIM

Photo of Jean-Philippe SAUT

Photo of Faiz BEN AMAR