Journal of Intelligent & Robotic Systems (2019) 95:511–526
https://doi.org/10.1007/s10846-018-0851-3

Physics Based Path Planning for Autonomous Tracked Vehicle in Challenging Terrain

Bijo Sebastian[1] · **Pinhas Ben-Tzvi**[1]

Received: 10 January 2018 / Accepted: 17 April 2018 / Published online: 27 April 2018
© Springer Science+Business Media B.V., part of Springer Nature 2018

Abstract

This paper describes a novel physics-based path planning architecture for autonomous navigation of tracked vehicles in rough terrain conditions. Unlike conventional path planning applications for smooth and structured environments, factors such as slip, slope of the terrain, robot actuator limitations, and dynamics of robot terrain interactions must be considered for rough terrain applications. The proposed path planning method consists of a hybrid planner/simulator, which takes into account all of the above factors by simulating the closed loop motion of the robot with a low-level controller on a realistic terrain model inside a physics engine. Once a feasible path to the goal is obtained, the same low-level closed loop controller is then used to execute the proposed path on the actual robot. The proposed architecture uses the D* Lite algorithm working on a 2D grid representation of the terrain as the high-level planner, Bullet as the physics engine and a hybrid automaton as the low-level closed loop controller. The proposed method is validated both in simulation and through experiments. Inferences based on the results from simulations and experiments show that the proposed planner is more effective in providing an optimal feasible path as compared to existing methodologies, demonstrating clear advantages for rough, unstructured terrain planning. Based on the results, possible improvements to the method are proposed for future work.

Keywords Motion planning · Mobile robot navigation · Rough terrain · Tracked vehicle · Physics engine · Hybrid automaton controller

1 Introduction

Tracked vehicles were first created to facilitate navigation over a variety of ground conditions such as snow, loose sand, mud, steep slopes, terrain cluttered with rubble or any combination of these (from here on referred to as rough terrain) that is otherwise not feasible for conventional wheeled vehicles. Such vehicles are often the best choice for applications such as hauling heavy military equipment or agricultural operations that require a significant amount of traction. The superiority of tracked locomotion over wheeled systems in such scenarios is due to its increased traction and comparatively lower ground pressure. Based on the above factors, tracked locomotion is often considered best suited for search and rescue applications [1] where terrain conditions are often treacherous and the environment very unstable. This is demonstrated in practice as well, as the majority of the search and rescue robots that have been deployed in the field over the past few decades use tracks as their primary method of locomotion [2–5].

Among other factors, an increased level of autonomy is one of the major requirements for a search and rescue platform. The ensuing conditions following natural or manmade disasters often involve poor communication channels between the affected region and the outside world, with limited bandwidth and increased latency. The use of tethered rescue systems in the past, allowing for a robust communication channel, has had varying degrees of success [6–9]. Unfortunately, the use of a tether limited the mobility of the robots and introduced the risk of the tether becoming stuck in the rubble. Owing to the critical nature of rescue missions, remote operation of a rescue robotic system in an unstructured volatile environment is a challenging task, even with a communication tether. This drives the need for higher-level autonomy in rescue robots, such as the ability

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10846-018-0851-3) contains supplementary material, which is available to authorized users.

✉ Pinhas Ben-Tzvi
bentzvi@vt.edu

[1] Robotics and Mechatronics Lab, Mechanical Engineering Department, Virginia Tech, Blacksburg, VA 24061, USA

to navigate on their own over challenging terrain conditions in a reliable manner. The recent call by the US Army [10, 11] for unmanned casualty evacuation platforms and the introduction of CasEvac/MedEvac scenarios in robotic benchmarking competitions like El-Rob [5] reinforce the growing need for such autonomous systems.

Determining a feasible path that will take the robot from the starting point to a goal location is the first step towards successful autonomous navigation. Much of the existing research towards autonomous ground robotic systems has focused on wheeled mobile robots navigating flat structured terrain conditions, mostly in indoor or urban scenarios [12, 13]. Reliable autonomy in challenging terrain conditions remains an open research problem. The two distinguishing factors that make rough terrain autonomy significantly more difficult as compared to flat/structured terrain autonomy are listed below:

1. Characteristics of terrain: For a structured terrain, a 2D occupancy grid is sufficient to plan an optimal path, but even obstacle free regions with significant slip or sinkage can prevent the robot from making progress in rough terrain. For challenging terrain conditions, it is not enough to have a 2D map of the environment. Additional factors including terrain topology, variations in slip due to changing soil makeup and terrain stability conditions need to considered in order to plan a feasible path. Terrain topology refers to the shapes and features that describe the three-dimensional nature of the surface. In this work, a 3D elevation map is used to represent terrain topology.
2. Characteristics of the vehicle: In the case of robots that are designed specifically for rough terrain locomotion such as SOLERO [14], the JPL Sojourner Rover [15], or hybrid tracked-wheeled multi-directional mobile robot like STORM [16] and HMMR [17], the vehicle characteristics need to be taken into consideration for planning. The simplest approach of assuming any terrain feature with an elevation above a threshold value is an obstacle results in an excessively conservative approach for tracked robots. Classifying terrain into traversable and non-traversable regions based on additional factors such as inclination, coefficient of friction and climbing performance of the vehicle must be performed in order to take full advantage of the vehicles' capabilities. Simultaneously, the dynamics of the vehicle must also be taken into account in order to prevent tipping while performing any traversing or climbing maneuvers.

In summary, the path planning algorithms need to take into account the dynamic interactions between the robot and the terrain in detail in order to achieve reliable path planning. For a robotic system trying to navigate in previously unseen terrain, this requires continuous estimation of terrain properties such as topology and slip while progressively exploring the terrain and updating the map. All of this should be performed while keeping in mind the limited computational power available onboard.

In this paper, we propose a novel effective path planning strategy that takes into account the above-mentioned requirements. For the purpose of this paper, we will not be discussing the pose estimation problem; instead, we will assume that the robot knows its pose information in 6D. Section 2 will provide a survey of existing path planning methods, especially for rough terrain applications. Section 3 will outline the methodology proposed and discuss in detail the various elements of the proposed path planning architecture. Section 4 talks about the simulations with a detailed discussion on the results. Section 5 includes the experimental setup, with results and inferences. Section 6 concludes the paper with directions for future research.

2 Literature Review

Robot path planning is a very well researched field, where many algorithms with specific advantages for a variety of applications have been proposed over the last few decades. A survey of recent advances in the domain of path planning techniques can be found in [12]. One of the major events that resulted in significant advances in autonomous vehicle research was the DARPA Grand Challenge (DGC) series in 2004 and 2005 [18–21]. The challenges focused for the most part on cars traversing "largely uniform and unchallenging" terrain, as mentioned in [22]. While the path planning strategies used by some of the teams considered terrain information, advanced vehicle dynamics were mostly ignored. Much of the self-driving car research that has followed draws on the foundation provided by the DGC, yet these are mostly targeted towards flat/structured terrain motion, even when focused towards driving in off-road conditions [23, 24].

Rough terrain path planning, as mentioned above, requires a more serious consideration of the robot and terrain characteristics as compared to general 3D (x,y,θ) planar robot path planning [13, 25–27]. The majority of the strategies for rough terrain use terrain maps classified into "occupied" and "unoccupied "cells based on the presence or absence of certain terrain features characteristic of the problem at hand. The work presented in [28] presents a method for estimating traversibility of unknown terrain using 3D vision sensors. The paper describes an Unevenness Point Descriptor, computed from point cloud normal vectors with Principal Component Analysis (PCA). It shows good performance in terms of ground detection but the effectiveness of the same in robot path planning has not

been demonstrated. On a similar note, [29] has proposed a method to estimate roughness of the terrain using normal vector deviation based on data obtained from a 3D LIDAR. The terrain roughness information is then integrated into a Traversable Region Model, which uses a 2D Voronoi-based map to segment XY region into cells and then assigns terrain traversibility based on roughness and slope as the cost of each cell. Others have used a terrain height map as obtained from 3D LIDAR, stereo Vision or structured light sensors to compute some form of terrain characterization based on slope, roughness and slip parameters. The above information is then used to compute artificial potential fields in order to determine the optimal path for the robot [30–33] . For instance, [32] has focused on casting the feasibility and cost of robot motion over a terrain as an optimization problem. They used the height map of the terrain along with a Fast Marching Method in order to come up with a potential field free of local minima. Using state lattice planner with primitive trajectories for path planning of a large tracked vehicle in open terrain is mentioned in [34]. However, the work assumes the terrain is assumed to be open and no consideration is provided for any terrain features like slope or actuator limitations of the robot. This allows the planner to assume that the robot is capable of executing the primitive trajectories at all times and is therefore not applicable to the challenges being addressed in this paper. Approaches that involve finding the smoothest path in a given terrain map or looking for the minimum artificial potential energy assume that the path with these characteristics to easily traversable. As mentioned previously, smoothness of a terrain does not always relate to traversibility, especially in the presence of loose sand, ice, or mud. In other cases, looking for the smoothest terrain might prove to be a conservative approach as this ignores the ability of the platform to traverse challenging terrain as offered by its mechanical design.

The one common drawback of the above approaches is not taking into account the dynamics of the robot. Even though terrain topology plays a major role, ignoring or simplifying the dynamics of the vehicle and its actuator limitations can lead to failure of the planner through collision or the vehicle's inability to execute the planned maneuvers. In contrast with the above approaches, [35–38] have used a simplified dynamic model of the vehicle along with the terrain elevation map to ensure that the vehicle does not tip over while traveling the path provided by the planner. Another similar work, [39], used a simplified model of the vehicle and terrain to check for stability at intermediate waypoints along the proposed path. Recent work by Currier and Wicks [18] proposed analytical methods for real-time estimation of Instantaneous Maneuvering Manifolds for large autonomous vehicles in order to predict their allowable dynamic operating ranges. Their method takes into consideration the uncertain and dynamic nature of payloads on autonomous vehicles as well as the varying frictional coefficient of the terrain, as applied to Ackerman steered vehicles. A two phase rough terrain path planning for actively reconfigurable robots is proposed in [40]. An initial path is obtained from a graph search, followed by identifying the rough regions on the path using vision data. Biased RRT* in the continuous state space of the robot is then used to refine the path on the rough regions, thereby taking into account the actuator limitations on the robot.

Another major factor to be taken into account for rough terrain path planning is terrain slip. Existing methods [14, 39] handle slip by means of robust trajectory tracking controllers while executing the planned trajectory. However, the above-mentioned approaches do not consider terrain slip during the planning stage. This is an inefficient strategy, as there will be cases where the robot cannot go over a slope or travel along the sides of a ravine due to significant slip. The planner then provides an un-traversable path that can result in failure, as even the best trajectory-tracking controller cannot overcome such significant slip events.

In summary, inaccuracies and simplifications in modeling the robot-terrain interactions lead to cases in which the feasible path reported by the planner results in collisions in the real world, or the robot not being able to execute the path at all. While some of the existing works may incorporate one or two of the above-mentioned factors during the planning stage, there exists no comprehensive solution to this problem.

3 Proposed Method

In order to meet the requirements as discussed above, we propose novel path planning architecture that consists of a high-level planner that takes into account the dynamic robot-terrain interactions by simulating closed loop motion of the robot with a low-level controller on a realistic terrain model inside a physics engine. Once a feasible path to goal is obtained, the same low-level controller is used to execute the proposed path on the actual robot. The overall working of the proposed planner can be explained as follows: The high-level planner starts with a 2D grid map of the region in which the terrain topology is initially flat. The robot then obtains information about the nearby static obstacles using onboard sensors such as LIDAR, and updates the 2D occupancy grid to include them. In addition, the robot obtains the terrain topology for the robot's current position and nearby cells (at minimum the immediate eight neighboring cells). Once this information is obtained, for every obstacle-free neighboring cell that the robot can move into, the feasibility of the path is further validated using the physics engine to account for the dynamics of the robot-terrain interactions.

The physics engine realistically models the terrain topology and the actuator limitations of the physical robot. The simulated robot inside the physics engine is directed to travel from the current location to the desired neighboring cell by means of the closed loop low-level controller. If the simulated robot successfully travels to the neighboring cell, the cost of the cell can be kept as a tunable combination of time taken and expended control effort. If the robot is not able to travel to the neighboring cell within a specified period, that neighboring cell is marked as unreachable. This can occur due to terrain conditions, obstacles, robot dynamics, or actuation limits of the robot. The simulated feasibility of motion is checked only for cells that already have a 3D terrain map generated, using the information obtained from the actual robot. For any cells that does not yet have this information, the high-level planner assumes the motion is feasible. This allows the system to come up with an initial path to goal extending beyond the range of the real robot's sensors.

Based on the results from the physics engine, the initial path generated by the high-level planner is passed to the physical robot as a list of waypoints. The low-level controller running on the physical robot then guides it to the next waypoint (local goal), based on the current position. Once the robot moves to the next waypoint within the vicinity of previously unexplored cells, it collects more sensory data. The updated sensor information is passed to the high-level planner, which then replans if necessary and outputs the new path. This process is iterated until the physical robot arrives at the goal or determines there is no path available. A block diagram and flow chart explaining the working of the proposed method is shown in Figs. 1 and 2 respectively. A major advantage of this method is that only intermediate goals are passed on to the physical robot, rather than the actual actuator commands. This allows the low-level controller onboard to inherently correct for any deviations in the mass, inertia or other physical parameters between the physical robot and the simulated robot, thereby providing a better guarantee of reaching the goal location without colliding with obstacles.

3.1 Novelty of the Proposed Approach

The major contribution of this paper is the introduction of the use of physics engines for state expansion of a mobile robotic system in a high-level planner**.** Following that explanation, the paper describes how to create a complete architecture for the autonomous navigation of a ground robot in rough terrain. In addition to the high-level planner and the physics engine, the proposed navigation architecture uses an additional low-level controller to execute the path proposed by planner on the physical robot. The use of a physics engine to handle the kino-dynamic aspects of the planning such as actuator limitations, slip and other terrain conditions allows the planner to come up with reliable paths while keeping the computational complexity low enough for real-time operation. In addition, the ability of the physics engine to simulate the motion of the robot under the action of the low-level controller on the simulated terrain allows the planner to take into account the effects of the hybrid controller on the motion of the robot. All of these factors demonstrate improvements on the existing work in this domain.

The use of D* Lite as the high-level planner and the hybrid automaton as the low-level controller demonstrates the approach towards creating a full navigation architecture and to demonstrate its effectiveness. As described in Section 3.4, these individual elements of the architecture can be replaced with others depending upon the application at hand, while still maintaining the overall structure and its advantages. The simulations and experiments detailed in Sections 4 and 5 clearly validate the capability of the proposed approach in meeting all of the requirements described in the introduction. While there exists works that

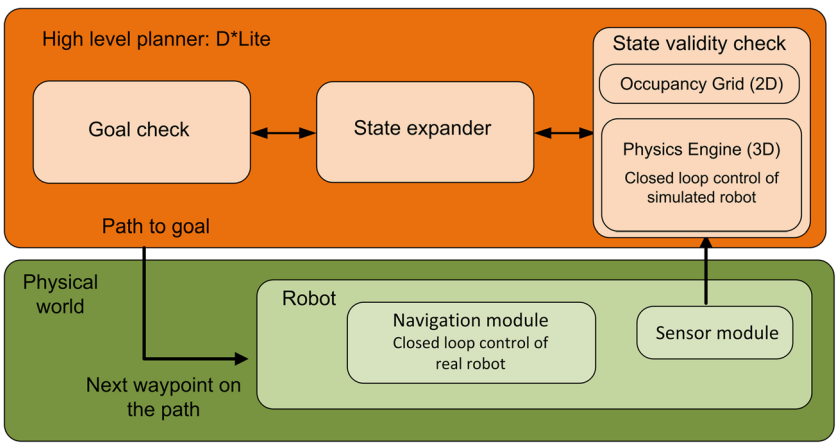

Fig. 1 Block diagram representation of the proposed planner

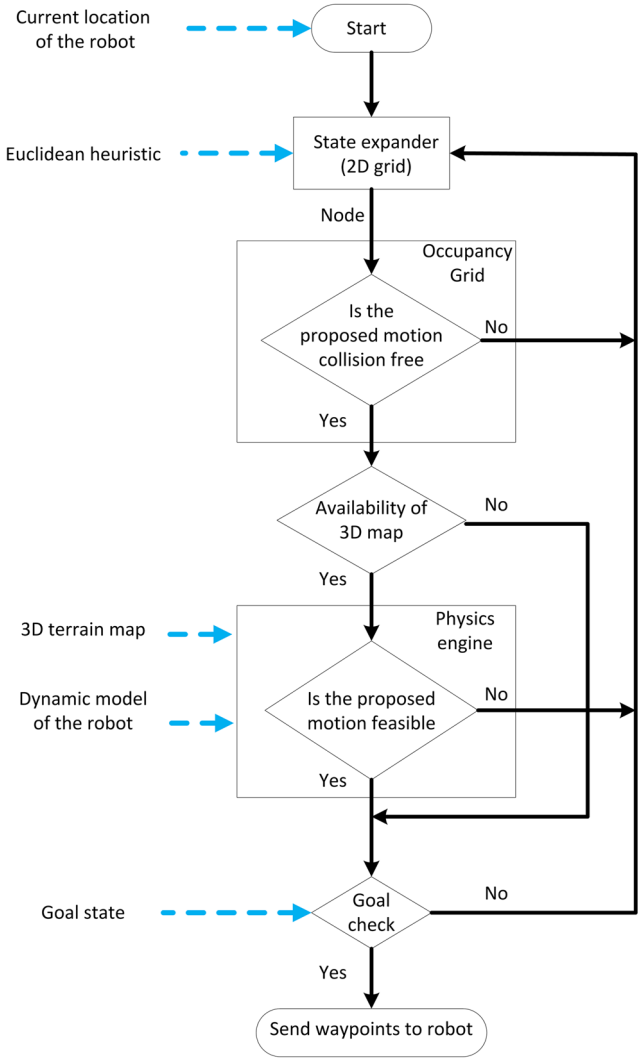

Fig. 2 Flow chart showing the working of the proposed planning algorithm

can handle one or more of the above-mentioned factors, there does not exist a method to the author's knowledge that can handle all of the above in a generalized manner.

3.2 High-level Planner

Previous approaches in path planning for robotic systems with complex dynamics, significant drift, and limited sensing mostly relied on kino-dynamic versions of stochastic methods like RRT, RRT*, or its variants [41–45]. The general approach when utilizing kino-dynamic RRT* is to plan in higher dimensional state space, where the planner randomly samples states, followed by either explicitly calculating or randomly guessing actions that will take the robot from the current state to the target state. Once a feasible path is calculated from start state to the goal state, the corresponding set of actions is then performed by the robot in the physical space. The success of these methods thereby strongly depends on the accuracy of the modelling. The presence of significant un-modeled dynamics can result in the robot being unable to reach the goal. However, performing such random sampling while using a physics engine in order to expand the states leads to high computational cost. This makes the approach unsuitable for real-time planning in dynamic environments.

Since the current application is focused towards motion planning in a previously unseen environment, a more appropriate high-level planner is D* Lite, due to its efficient re-planning capabilities that allow it to handle changes in the environment [46]. D* Lite is more computationally efficient when compared to repeated A*, while being easier to implement as compared to original D* [47]. As a grid based planner, D* Lite is made to work on the discretized work space (x, y plane in this case) rather than in the higher dimensional discretized state space (x, y, ϕ, $\dot{x}$, $\dot{y}$, $\dot{\phi}$), while using the Euclidean distance from the goal as a heuristic. The total length of the path taken by the robot from start to goal is the cost function. In order to reduce complexity, the grid-based algorithm utilizes a robot model that can move from any given cell to the eight neighboring cells, provided the neighboring cell does not have an obstacle in it. The high-level planner is initialized with a uniformly discretized 2D model of the environment (occupancy grid), with all unknown edge costs set to minimum value. This allows it to develop initial guesses on the path to the goal without having the full terrain map to begin with.

3.3 Kino-dynamic Aspect of Planning

In the case of robotic systems with complicated dynamics involving non-holonomic constraints, finding a feasible trajectory between the initial and final/desired states is a non-trivial problem. This is particularly true with systems involving significant interaction with the working environment, such as walking robots, dexterous manipulation and tracked vehicles of significant mass traversing through rough terrain while under the action of a low-level feedback controller. In order to accurately model the motion of the physical robot, the planner must also take into consideration the low-level controller that drives the physical robot [41]. Most of the existing approaches ignore these aspects or model them using simplified kinematic equations, which may result in unrealistic estimates thus leading to failures in path planning.

Existing works in this domain have tried to first model the motion of tracked vehicles on rough terrain the process as a set of analytical equations and then solve these equations. However, as the complexity of the system increases so do the computational costs of modelling, leading prior work to focus only on certain specific aspects of the challenge such

as slope of the terrain or the effect of slip. As a result, the accuracy of the prior methods is impacted by focusing on a certain specific aspect of the system and neglecting others.

In contrast with existing literature in this domain, we propose the use of physics engines to model the kino-dynamic aspect of planning. A physics engine is software that provides a simulation of physical systems, capable of simulating rigid body dynamics (including collision detection), soft body dynamics, and fluid dynamics. These are primarily used in the domains of computer graphics, video games, movies and high-performance scientific simulation. A physics engine calculates the forces that arise between bodies when they interact with each other, with the goal of preventing bodies from inter-penetrating. These forces are then used to derive the motion of the bodies, using multi-body dynamic equations. The engine also models various joints, such as revolute or prismatic joints, as sets of algebraic constraint equations. The effects of friction that arise between the bodies while they interact are also take into consideration. In order to solve for the motion of the objects within the simulation, a time stepping approach is used wherein the differential algebraic equations (DAE) from the dynamics and constraints are solved simultaneously to obtain the state of the system subsequent to the time step. For more details regarding the working of physics engines refer to [48–50].

Compared to the existing analytic modeling approaches, the use of a physics engine offers a more computationally efficient and practical solution to model the robot-terrain interactions, low-level control algorithms and the sensor models that the actual robot possesses. The use of a physics engine allows for the state evolution of these systems forward in time while taking into account motion dynamics, gravity, friction and other aspects of ground interaction. The robot model can be initialized in most existing physics engines by importing the robot in Universal Robot Description Format (URDF). The URDF consists of a set of files describing the physical attributes of the robot such as mass, inertia, and the dimensions of the various links as well as how they connect together. Provided the robot-terrain system is correctly initialized, the future states can be obtained with a reasonable degree of accuracy through simulation without having to derive and solve the equations that govern the evolution of the system.

The engine needs only the current state of the system and the control inputs in order to estimate the new state, much like the actual physical robot itself. In other words, the high-level planner can treat the physics engine as a "black box". By simulating the motion of the robot from one waypoint to another, the planner can take into account the dynamics of robot-terrain interactions, including slip, in a realistic manner during the planning stage itself and thereby accurately chart a feasible path. The above reasoning justifies the use of physics based robotic simulators for state expansion, rather than simplified dynamic equations, at the cost of marginal increase in computational load.

A comparison of some of the existing state of the art physics engines can be found in [51]. For the purposes of this work, we will be using Bullet Physics in headless mode to reduce the computational overhead [52]. Bullet was chosen due to the superior performance of its friction model and the excellent documentation available online. Moreover, Bullet falls under the category of real-time physics engine as it uses approximate calculations in order to produce accurate real time results. While Bullet is capable of modelling soft terrain such as in the case of loose sand or mud, the simulations used in this work treat both the robot and the terrain as rigid bodies. The approximation of rigid bodies allows for faster simulation based on the capabilities of current state of the art physics engines. As physics engines improve in their ability to model soft deformable objects and their interaction, simulations that are more realistic can be incorporated for better representation of the motion of a tracked vehicle on deformable terrain such as loose sand or mud, increasing the accuracy of the proposed approach. The hardware and software improvements necessary for incorporating these features are part of ongoing work.

3.4 Low-level Controller

The path computed by the high-level planner (a set of waypoints ending at the goal) will be executed on the physical robot by a low-level controller. The low-level controller, running at a higher frequency than the high-level planner, continuously monitors the state of the robot and the environment through sensors in order to generate control inputs (right and left track velocities for a differential drive robot) to ensure stable navigation from the current state to the next waypoint. For the proposed approach, a hybrid automaton is used as the low-level closed loop controller.

Navigating successfully in a dynamic environment is a challenging task, especially from a control system design perspective. The system should be capable of reacting to a variety of situations that may occur at unknown time instants. One of the ways to approach this problem is through the design of behavior-based control, where separate controllers are formulated for handling different scenarios such as going towards the goal, avoiding an obstacle, and stopping the robot upon reaching goal. A higher-level state machine is made responsible for switching between the behaviors based on the external inputs, such as sensor information from proximity sensors or laser scanners, thereby ensuring that the robot always operates within one of the finite states (the behavior-based controllers) at any given point of time.

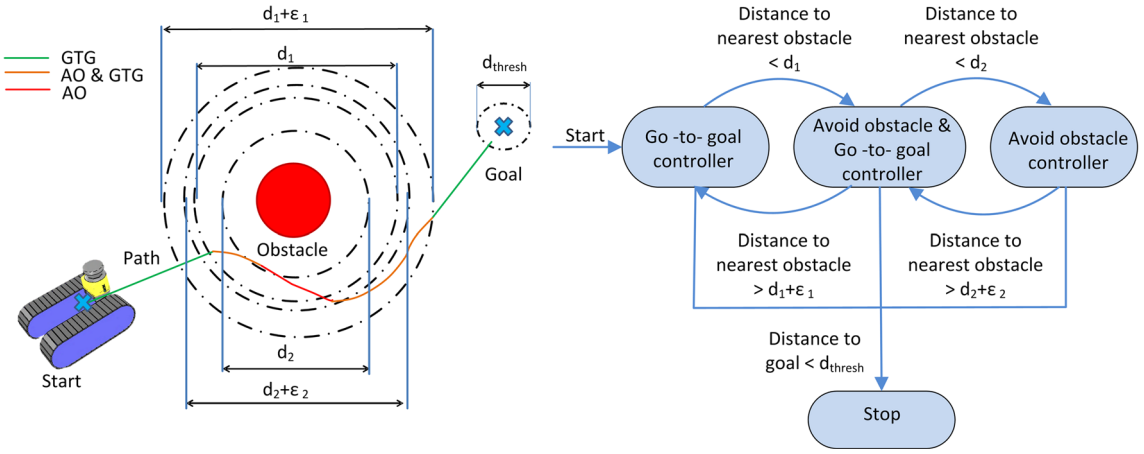

Fig. 3 **a** Working of the hybrid automaton **b** State machine representation of the hybrid automaton

Guard conditions inside the state machine designed for each behavior enable switching between the behaviors. The individual behaviors, along with the state machine and the guard conditions, together form a hybrid automaton based closed loop controller for autonomous mobile robot navigation [53].

While this approach simplifies the controller design, the presence of hard switches in the behavior could introduce chatter due to scenarios that require infinite switches in finite time (Zeno phenomenon) and thereby detrimentally affect the performance of the overall system. As mentioned in [53] this can be solved by adding regularization controllers to fuse the behavior of the two conflicting nodes in the system. The design of a hybrid automaton based closed loop controller for low-level mobile robot navigation is described in [54]. We will be using a simplified version of the same as the low-level controller for our proposed approach. The hybrid automaton used in the proposed planner essentially consists of two different behaviors, the Go-to-goal behavior and the Avoid-obstacle behavior, with an additional regularization node, Avoid-obstacle-and-Go-to-goal controller having the fused behavior of both the above nodes. Figure 3 illustrates the working of the hybrid automaton based closed loop controller.

The system starts at the Go-to-goal controller and stays in this state until an obstacle point is detected within d_1, as seen in Fig. 3. At this condition, the state machine switches to the Avoid-obstacle-and-Go-to-goal controller. Within this state, if the robot detects an obstacle point closer than d_2, where $d_2 < d_1$, it switches to the Avoid-obstacle controller. In order to prevent rapid switching on and off a particular state when the robot senses an obstacle point exactly at d_i (where, $i = 1, 2$); the exit conditions of the state have an additional ε_i term (where, $\varepsilon_i ¿ 0$). To ensure that the robot stops when it reaches within a threshold distance (d_{thresh}) of goal, an additional "Stop" state is provided.

The following section describes the modelling of the system and the design of each of the individual control

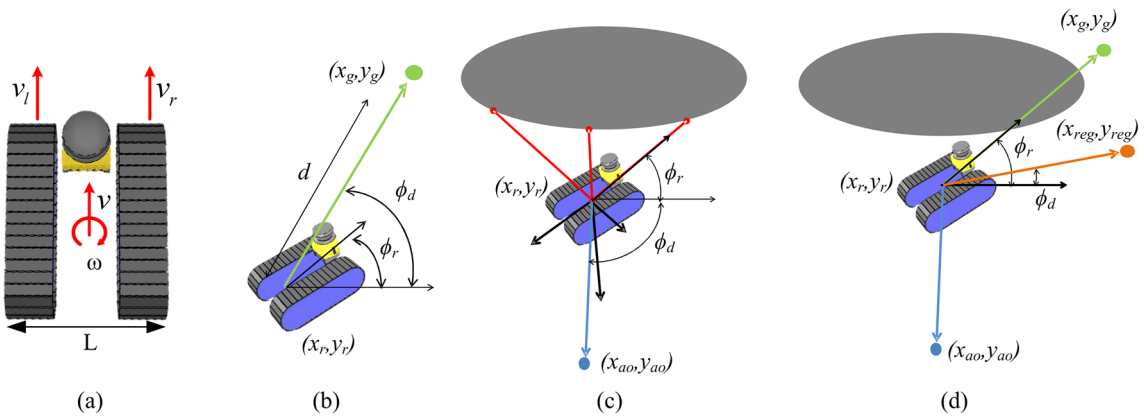

Fig. 4 **a** Kinematic model of the robot **b** Go-to-goal controller **c** Avoid-obstacle controller **d** Avoid-obstacle-and-Go-to-Goal controller

behaviors as shown in Fig. 4. The linearized kinematic model for a differential drive robot is given below:

$$\begin{aligned}\dot{x} &= \frac{1}{2}(v_r + v_l)\cos\phi \\ \dot{y} &= \frac{1}{2}(v_r + v_l)\sin\phi \\ \dot{\phi} &= \frac{1}{L}(v_r - v_l)\end{aligned} \tag{1}$$

where:

(x, y) is the 2D position and $\phi$is the orientation of the robot

L is the distance between the left and right tracks of the robot

v_r, $v_l$are the left and right track velocities, respectively

For the ease of designing a controller, the differential drive robot is modelled as a unicycle with the same three states, but different control inputs: ν (for linear velocity) and ω (for angular velocity). The unicycle robot kinematic model is given by:

$$\begin{aligned}\dot{x} &= \nu\cos\phi \\ \dot{y} &= \nu\sin\phi \\ \dot{\phi} &= \omega\end{aligned} \tag{2}$$

Once the unicycle model's control inputs (ν, ω) are determined based on the current state, desired goal state and sensor inputs, they are mapped into differential drive model's control inputs (v_r, v_l) through the following transformations:

$$\begin{aligned}v_r &= \frac{2\nu + \omega L}{2} \\ v_l &= \frac{2\nu - \omega L}{2}\end{aligned} \tag{3}$$

The Go-to-goal controller is essentially a PID controller that drives the vehicle to the nearest waypoint as provided by the high-level planner. In order to drive the robot from current location (x_r, y_r) to a goal location (x_g, y_g), the robot has to first align towards the goal and then drive forward till it reaches a location within the threshold distance to the goal point. The slope of the line connecting the robot and the goal position, ϕ_d , is the desired orientation of the robot. Based on the heading error the angular velocity control input (ω) to the robot can be determined from the equations below:

$$\begin{aligned}e &= \phi_d - \phi \\ \omega &= k_p e' + k_d \dot{e}' + k_i \int e' dt\end{aligned} \tag{4}$$

where,

k_p, k_d and k_i are the proportional , derivative and integral gains, respectively.

e' is the error limited between $[-\pi, \pi)$ in order to account for the wraparound of orientation

As for the robot's linear velocity, a proportional controller is applied with the distance from the goal (d) as the error:

$$\begin{aligned}d &= \sqrt{(x_g - x_r)^2 + (y_g - y_r)^2} \\ \nu &= kd\end{aligned} \tag{5}$$

where,

k is the proportional gain

The Avoid-obstacle controller works in a similar manner to the Go-to-goal controller, as shown in Fig. 4c. It is designed to work with real-world obstacles that have finite size, using obstacle detection sensors like ultrasonic, infrared or LIDAR. Based on the sensor information, the planar (x, y) coordinates of the obstacle point are obtained in the global coordinate frame, provided the robot's position and orientation are known. The vector leading from the robot's current location to the estimated obstacle's location is then extended in the reverse direction to obtain the coordinates of the "avoid obstacle" (x_{ao}, y_{ao}) point. The robot is then driven towards this point in the same manner as the Go-to-goal controller with (x_{ao}, y_{ao}) as the goal allowing the robot to effectively avoid the obstacle. In case of multiple sensors the points can be averaged to obtain a final "avoid obstacle" point.

The Avoid-obstacle-and-Go-to-goal controller provides for the regularization of the hybrid automaton by fusing the behavior of individual Avoid-obstacle and Go-to-goal controllers. For this behavior, the robot first computes the "avoid obstacle" point in global coordinates as described above. The normalized vectors to the "avoid obstacle" point $(\overrightarrow{AO})$ and the goal point $(\vec{G})$ are then combined using weighting factors to find the regularization point, $\vec{R}$ (x_{reg}, y_{reg}), as given below:

$$\vec{R} = \alpha\vec{G} + (1 - \alpha)\overrightarrow{AO} \tag{6}$$

where,

$\alpha \epsilon [0, 1]$ is the weighting parameter which is tuned for performance.

Once this point is obtained, the robot can be driven towards this point in the same manner as the Go-to-goal controller, with (x_{reg}, y_{reg}) as the goal. The working of the Avoid-obstacle-and-Go-to-goal controller is illustrated in Fig. 4d. For more details on the design of the hybrid automaton, individual behaviors and the guard conditions, refer to [54]. While the hybrid automaton based navigation system introduced in [54] does include other behaviors like "Follow Wall" for handling complicated environments, involving concave obstacles, the presence of a high-level planner in the proposed architecture handles these complicated cases much more efficiently, without requiring the additional behaviors.

3.5 Implementation Details

The following section provides further details on the proposed approach to provide the reader with a deeper understanding for the application of the method.

Computational time For the purpose of the simulation and the experimental validation described in the following sections, the 2D grid resolution for the high-level planner is kept equal to 0.5m along both the axis. With the fixed grid resolution, the time taken by the physics engine to simulate the motion of the robot between two adjacent cells is at max 0.6 seconds (while running on an HP laptop with a 2.6 GHz Intel processor and 8GB RAM). Once the robot reaches the neighboring cell, the simulation is terminated and the motion is reported feasible to the high-level planner along with the cost. The minor variation in time taken is caused by variations in the nature of the terrain corresponding to the cells. In cases where the robot fails to reach the goal, the simulation could run infinitely long. To prevent this, the simulation is forcibly terminated after one second and the motion is reported not feasible. Under the above approach, one second is the worst-case time taken to simulate the motion of the robot between two adjacent cells.

Optimality of path The proposed approach is capable of finding the optimum path by using an optimal high-level planner, D* Lite in this case, after a subset of feasible paths are provided through the physics engine. Like any other implementation, the definition of optimality depends on the cost function assigned to the planner. For the sake of demonstrating the functionality of the proposed implementation, the length of the path traversed by the robot to reach the goal is optimized. As indicated by previous works [45, 55, 56] the shortest path may not be optimal in terms of time taken by the robot to reach the goal or the energy spend by the robot in doing so. By modifying the cost function on the high-level planner, one can enable the proposed approach to optimize for time, energy or any combination of these, since all of the costs associated with the motion of the robot can be obtained from the physics engine.

Handling slip The kinematic model of the differential drive robot described in Section 3.3 is solely used for designing the low-level controller. This is done without taking into account the effects of slip. The physics engine in the proposed architecture is responsible for handling the longitudinal and lateral slip experienced by the robot. The coefficient of friction of the terrain modelled inside the physics engine is kept low such that the robot experiences more slip than it would in reality. By simulating robot motion on such terrain, the proposed navigation architecture can detect possible maneuvers by the robot that may fail in the real world during the planning stage itself. Even though conservative in nature, this approach allows the system to handle longitudinal and lateral slip by avoiding such maneuvers while evaluating the path to goal. This is demonstrated in the experimental trials, which were performed in a high slip condition as described in the following section (Section 5). This validates the fact that the proposed planner takes into account the slip of the robot during the planning stage itself, unlike existing approaches. Additional improvements such as estimating slip while traversing through the unknown terrain and updating the physics engine in real time will be considered as part of the future work.

4 Simulation

In order to validate the feasibility of the proposed planning architecture, it was initially tested using simulation. The simulations were performed using the V-REP robotic simulator provided by Coppelia Robotics [57]. The terrain maps were generated using the ANT landscape add on [58] in BLENDER [59], which was then imported into V-REP for simulation. The motion of the robot was simulated in two different terrain maps differentiated by increasing frequency of variations in the terrain height map, characterizing a moderate and extreme level of roughness. The simulated terrain domain in both cases was a square with sides four meters in length. For the moderate terrain case, the simulated terrain height varies from a minimum of -0.5 m to a maximum of 0.3 m. For the extreme terrain case, the simulated terrain height varies from a minimum of -0.5 m to a maximum of 0.1 m, but with high frequency variations in terrain height as compared to the moderate terrain case. For both the terrain maps, the starting position of the robot is considered as ground level or zero elevation. The robot used for the simulation is the caterpillar model provided in V-REP with a nodding SICK LMS 300 LIDAR for obtaining the terrain map and five ultrasonic sensors (two on each side and one in the front) for the low-level Avoid-obstacle controller. The simulated robot is 0.5 m in length, 0.5 m in width and 0.3m in height, with a total mass of 16Kg.

The proposed planning architecture, including the D* Lite high-level planner and the low-level hybrid automaton controller, were implemented in MATLAB which then communicates with V-REP through the remote API interface. A snapshot showing the motion of the robot perceived by the planner running in MATLAB and the corresponding position of the robot in V-REP during the extreme terrain simulation is shown in Fig. 5. During the simulation the position and orientation (all six dimensions) of the robot was read off directly from the simulator by the

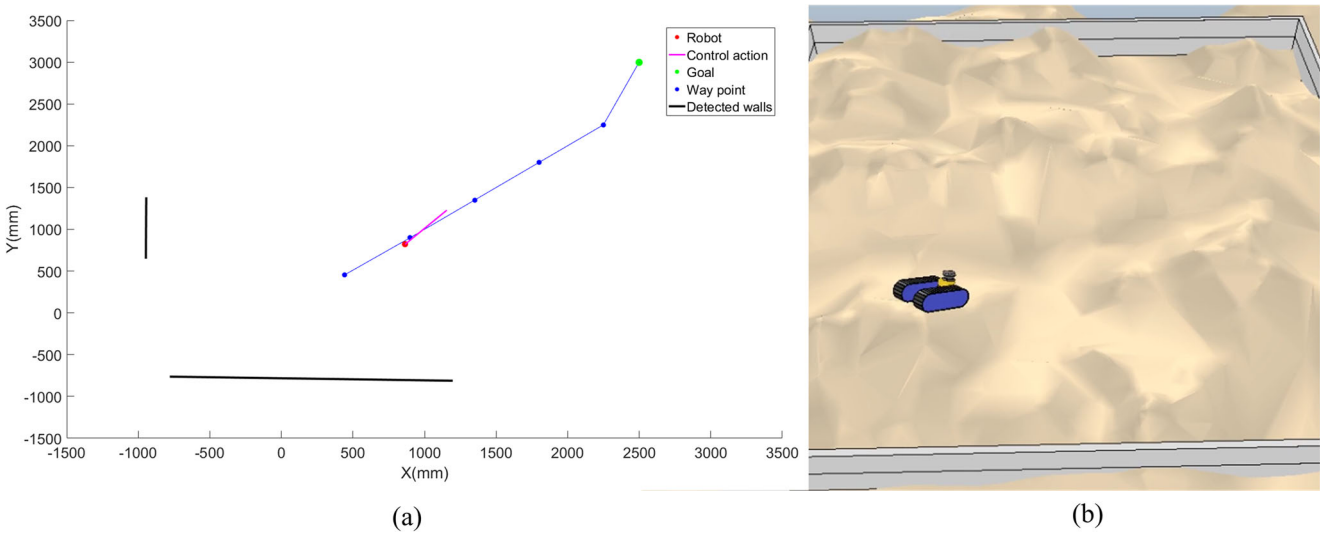

Fig. 5 **a** Motion of the robot as perceived by the planner, **b** Corresponding position of the robot in the simulation

low-level controller running in MATLAB. Bullet Physics running in headless mode was used to model the dynamic aspects of the planning problem. For both the simulation cases, the results of the proposed planner were compared with the outputs of a kinematic planner. The only difference between the proposed planner and the kinematic planner is that the kinematic version does not check the validity of the proposed path using the physics engine, ignoring the dynamic aspects of the problem. This allows the kinematic planner to produce the optimal shortest path in all cases, with no regard to the feasibility of the proposed path. Both the simulated terrains have bounding walls to prevent the robot from going outside the simulated region. Other than the bounding walls, no other static obstacles were present in both the terrain maps. For the purpose of the simulation, the low-level controller was set such that the simulated robot would not detect the terrain features as obstacles.

4.1 Results and Inferences

The proposed planning architecture was able to find feasible paths in both moderate and extreme terrain cases, whereas the kinematic planner failed in the second case. In addition, the proposed planner matched the results of the kinematic planner in producing the shortest feasible (optimal) path for the moderate terrain case. This signifies the improvements of the proposed method over the existing artificial potential field methods and others that do not consider the dynamics of the robot or terrain topography.

Based on the results, in both the simulation cases, the proposed planning architecture met the requirements that were described in the previous sections. The results of the simulation are shown in the topographical plots in Fig. 6, with the outcome of the kinematic planner in red and the proposed planner in blue. The density of the terrain lines denotes relative steepness of the terrain. For both cases, the waypoints provided by the planner are shown as dots and the path followed by the simulated robot in dashed lines. In both scenarios, the kinematic planner outputs the shortest path to the goal, which is a straight line. For the moderate terrain case, the proposed planner matched the result of the kinematic planner showing that it is optimal in terms of path length.

For the extreme terrain case, the shortest path is not feasible since at point B, as shown in Fig. 6b, the terrain is too steep for the robot to climb. The kinematic planner fails at this point whereas the proposed planner detects the failure inside the physics engine and therefore takes a right 6turn in order to ensure the feasibility of the proposed path, while still trying to minimize the overall path length. In addition, the proposed method shows better performance as compared to artificial potential field methods that use the height map of the terrain to develop with feasible paths. At point A as shown in Fig. 6b, artificial potential field methods and methods looking for smooth regions in the terrain map would have traveled in the reverse direction in order to avoid the steep climb. The proposed method instead checks whether the robot can execute the climb inside the physics engine and based on the result decides to go forward, resulting in a shorter path to goal. The deviations of the robot from the desired path at points B, C and D as shown in Fig. 6b, are due to the longitudinal and lateral slip experienced by the simulated robot in V-REP. Figure 7 shows the path followed by the simulated robot in the extreme terrain case for both the planners in 3D for better clarity.

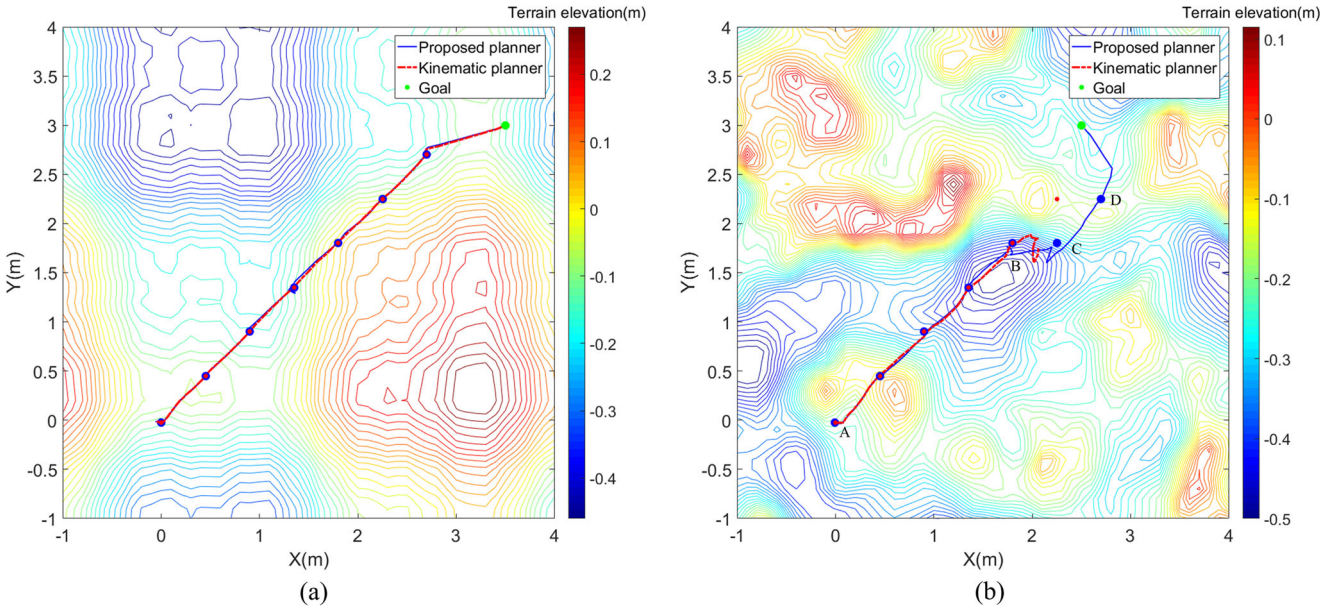

Fig. 6 Performance comparison of the proposed planner with kinematic planner: **a** Moderate terrain **b** Extreme terrain

5 Experimental Validation

The proposed planning architecture was experimentally validated using STORM, a mobile robot platform developed in the Robotics and Mechatronics Lab [16]. STORM has multi-directional mobility capabilities enabled via hybrid combination of tracks and wheels that operate independently of each other. It also has ultrasonic sensors on all four sides for obstacle avoidance applications. For the purpose of this paper, the robot was operated only in tracked differential drive mode. The robot is 0.41 m in length, 0.3 m in width and 0.12 m in height, and weighs a total of 9Kg. The robot possesses ultrasonic sensors on all four sides for obstacle avoidance and utilizes an ODROID XU4 computer for onboard processing. 3D scans of the experimental terrain (height map) were obtained using a Kinect XBOX One. Due to the limited battery power and computational capability available onboard the robot, a complete 3D scan of the

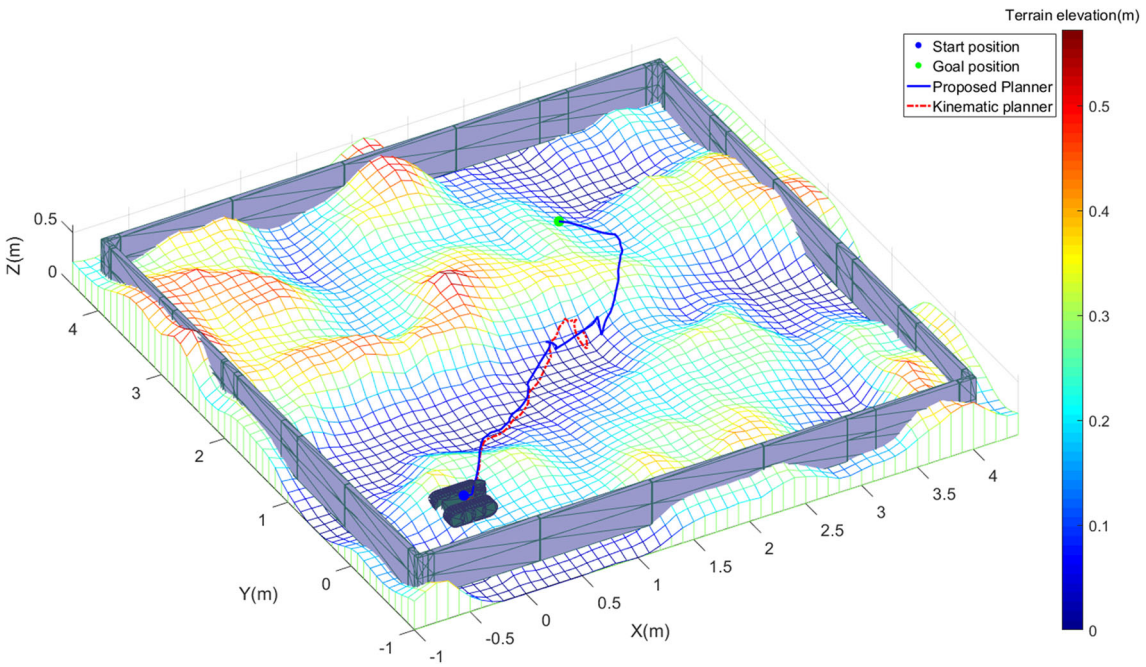

Fig. 7 Terrain topography map showing the 3D path followed by the robot under both the planners in extreme terrain simulation

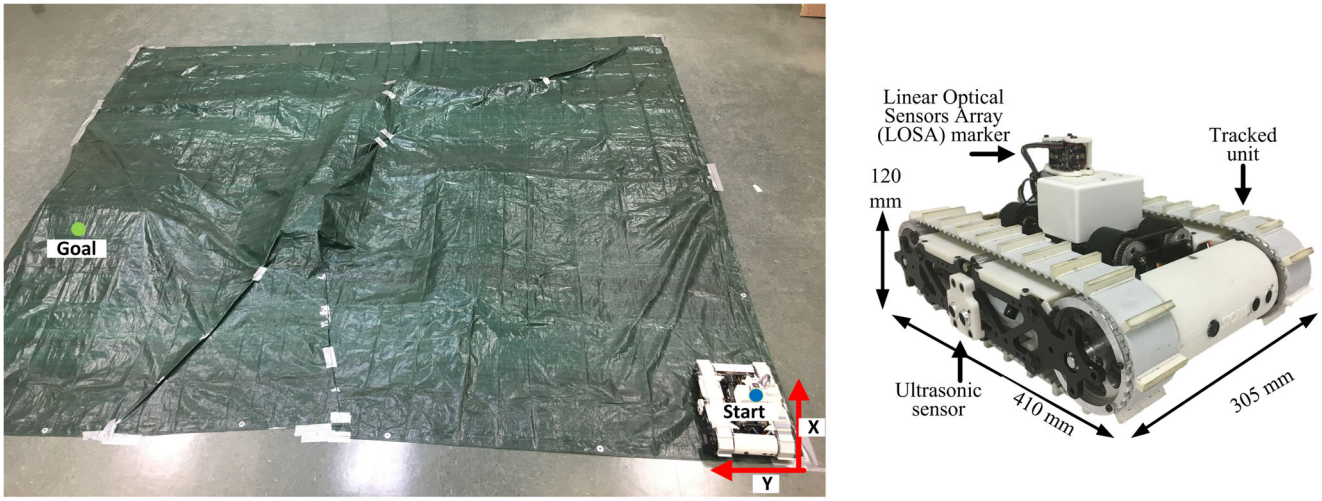

Fig. 8 **a** Experimental setup **b** STORM fitted with LOSA marker

terrain was obtained beforehand for the experiment rather than in real-time.

For the duration of the experiment, positional information of the robot was obtained using Linear Optical Sensor Arrays (LOSA) tracking system, an opto-inertial motion tracking device developed in the Robotics and Mechatronics Lab [60]. The LOSA system is capable of providing 3D position and orientation information at very high accuracy, up to 1 mm in position and 1 degree in orientation. The near-infrared LOSA marker was mounted on the robot and the LOSA tracker was connected to a laptop. Based on the current location of the robot as obtained from LOSA, the provided location of goal and the terrain map, the planner computes a feasible path to reach the goal. The proposed planner running on the laptop sends the path as waypoints to the robot's onboard computer through ROS. The low-level hybrid automaton controller running onboard the robot takes the waypoints as intermediate goals along with the positional information in order to drive the robot. As the infrared sensors on LOSA and Kinect XBOX One used in this experiment are limited to indoor use, the proposed planning architecture was tested on a rough terrain model created indoors using a tarp-covered rubble pile to create a varied, high slip environment. The experimental terrain had an overall length of 5.5 m and width of 5.5 m. The terrain height varied from 0m to a maximum of 0.25 m.

A picture of the experimental setup, showing the start and goal positions, are shown in Fig. 8a. The STORM module fitted with the LOSA marker is shown in Fig. 8b. As in the case of simulation, a virtual bounding wall was provided inside the planner to prevent the robot from going outside the limits of the experimental setup as shown in Fig. 10. In addition, the low-level controller was set such that STORM would not detect the terrain features as obstacles. For the purpose of the experiment, the 2D grid resolution for both the proposed and kinematic planner was kept equal to 0.5m along both the axis. Due to this resolution, the path proposed by the kinematic planner is not a straight line. Reducing the grid size would improve the resolution of the planner thereby making the path proposed by the kinematic planner closer to the ideal straight-line path at the cost of increased computation.

5.1 Results and Inference

For the purpose of experimental validation, the robot was required to go from one end of the terrain to the other

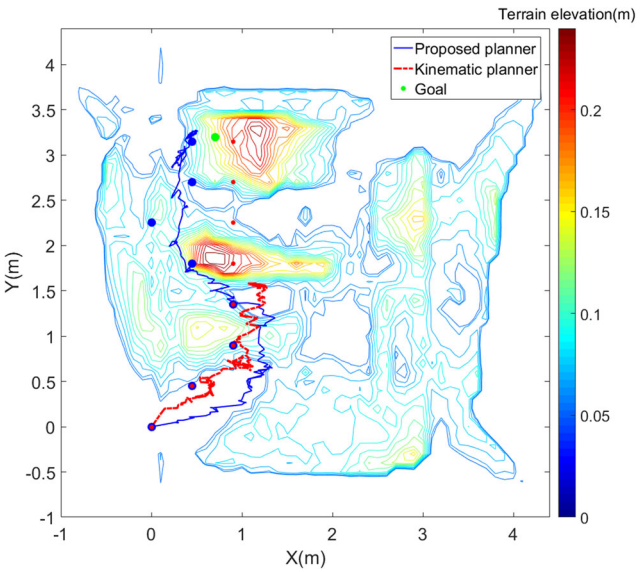

Fig. 9 Performance comparison of the proposed planner with kinematic planner over the experimental setup

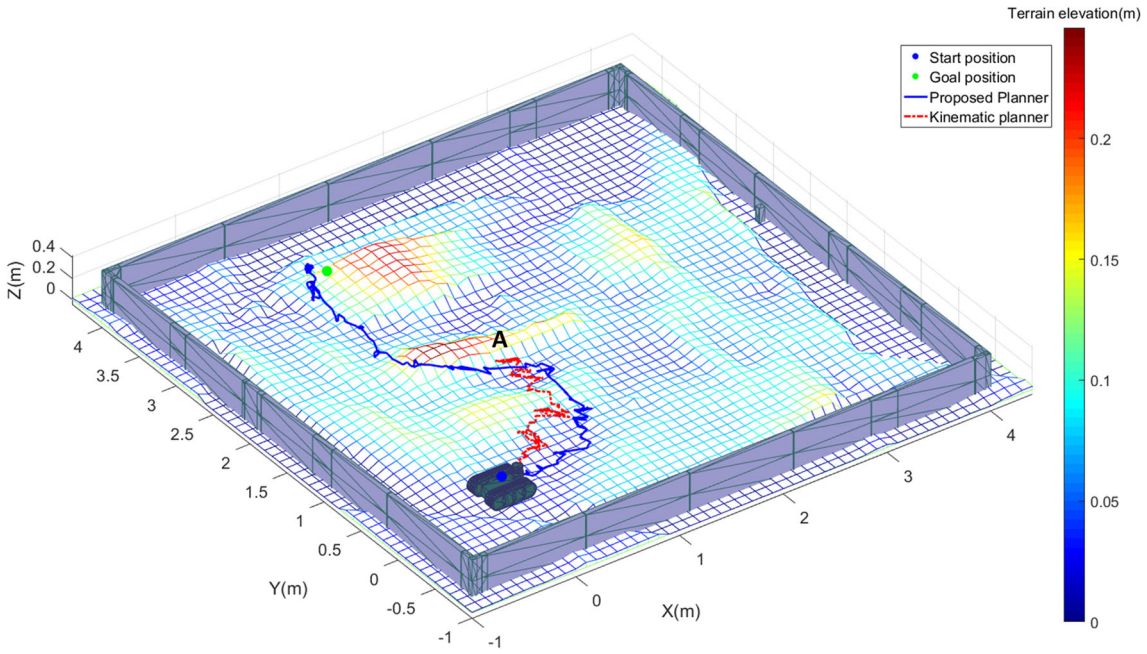

Fig. 10 Terrain topography map showing the 3D path followed by the robot under both the planners over the experimental setup

while crossing a steep ridge (denoted as point A in Fig. 9) in the middle. As in the case of the simulations, both the kinematic planner and the proposed planner were used to plan the path of the robot. The kinematic planner proposed the shortest path, which required the robot to cross the steep central ridge. The robot was unable to cross the ridge during the experiment and thus the kinematic planner failed. The proposed planner was provided with the complete map of the terrain within the physics engine, from which it evaluated that the robot would not be able to cross the central ridge. Based on this information, the proposed planner developed a longer but feasible path that successfully guides the robot to the desired goal location. The path proposed by the kinematic planner and the proposed planner are shown as waypoints in Fig. 9. The actual path followed by the robot in both cases is also shown.

The deviation of the actual path followed by the robot from the waypoints specified by the planners is mainly due to the significant slip encountered by the system on the tarp. For the purpose of the experiment, the d_{thresh} value was set to 200 mm. This means the low-level controller running on the robot assumes that the goal point is reached as soon as it is within 200 mm distance from the point. Figure 10 shows the path followed by the robot under the action of both the planners in 3D for better clarity. The high frequency oscillations in the robots motion as seen in the results are mainly due to the slipping of the robot tracks on tarp. On an outdoor terrain with less slip, the robot should be able to achieve a smoother trajectory. The experiments thus demonstrate that the proposed planner is more effective in providing a feasible path than existing planners. Coupled with the online physics check, the proposed planner presents clear advantages for rough, unstructured terrain planning.

The time taken by the robot to reach the goal under the action of the proposed navigation architecture is 1 minute and 47 seconds. As the high-level planner implements D* Lite, the proposed architecture dynamically replans at every waypoint based on the updated map. Therefore, the above-mentioned time depends not only on the planner but also on the average speed of the robot, type of terrain and even on the communication delay between the proposed planner running on the laptop and the low-level controller running on board the robot. A detailed study of the time taken by the proposed planner for different terrain conditions, including other factors such as the amount of energy expended is beyond the scope of this paper and will be explored in the future.

6 Conclusion and Future Work

This paper proposed the use of a novel planning architecture to overcome the shortcomings of existing kinematic and artificial potential field based path-planning methods for mobile ground robot navigation in high dynamic terrains. The proposed planning architecture involves the use of a grid based high-level planner along with a physics engine to take into consideration of the kino-dynamic aspects of robot-terrain interactions. The path proposed by the

high-level planner as a set of waypoints to the low-level hybrid automaton controller is checked for feasibility inside the physics engine. The same hybrid automaton was then used to execute the proposed path on the real robot. The proposed method was validated through both simulations and experimental results. In both, the performance of the proposed planner was tested against a kinematic planner that does not take into account the dynamics of the robot or terrain. In both simulation and experiment, the proposed planning architecture has shown clear improvement in performance by developing feasible paths in challenging terrain conditions where the kinematic planner failed. Furthermore, in moderate terrain conditions the proposed planner produced paths as efficient as those of the kinematic planner. These results demonstrate that the proposed planner is optimal in terms of path length, while taking into account the feasibility of the proposed path.

The proposed navigation architecture is capable of handling static obstacles by means of the D* Lite high-level planner. By assuming that all the obstacles are static, the high-level planner can take into account the possibility of collisions while the robot moves to the goal without taking into account the actions of the robot in the velocity space or adding time as a dimension to the planning space. However, there exist planners that can handle moving obstacles by adding time as a dimension to the planning space [61, 62]. By replacing D* Lite with these planners, along with providing estimated velocity of the moving obstacle inside the physics engine, the proposed approach will be able to handle moving obstacle cases efficiently. A drawback to this is that as the number of moving obstacles increases, the computational overhead of the physics engine also increases. In order to incorporate dynamic obstacle planning, additional hardware and software computational optimizations will be required.

Due to the limitations in the infrared sensing elements (Kinect and LOSA), the proposed architecture was tested under indoor conditions. In order to perform such test outdoors, the 3D scan of the terrain can be obtained from a nodding LIDAR and the position data for the robot can be obtained through a 3-D radio frequency-based positioning sensor such as POZYX [63]. To further validate the planner in future work, the proposed algorithm will be tested using a heavier robot like the HMMR [17] on more challenging outdoor terrain conditions.

The planning architecture described in this paper assumed constant slip over the full duration of navigation. The dynamic model of the robot was assumed constant as well. In real-life conditions, these assumptions may not hold. Slip can vary based on the type of terrain the robot is operating on, while factors affecting the dynamic model of the robot such as total mass of the robot and positon of the center of mass could change based on the nature of the mission performed by the robot. Future work will aim to take into account these cases as well.

The idea proposed here of using a robotic simulator to predict the behavior of an autonomous robotic system to make intelligent decisions regarding the behavior of a robot can be applied to other domains as well. Even though this work explains the application of the novel path planning method for a tracked differential drive robot, the overall approach used here can be applied to autonomous navigation of any robotic system. For instance, physics engines are capable of modeling the motion of quadcopters, including the execution of extreme maneuvers. For path planning applications involving quadcopters, a high-level planner will have to evaluate the feasibility of such maneuvers under a given environment, such as while flying inside a collapsed building structure for search and rescue operations. The proposed approach using physics engines can provide better results in such cases. The major difference in applying the proposed framework to other platforms, including quadcopters and AUVs will be in designing stable closed-loop controllers that can work at the lowest level for these highly nonlinear systems. The overall approach of using a high-level planner along with a physics based simulation, including the robot model with the low-level controller, can remain the same.

Acknowledgments This work is supported in part by the US Army Medical Research & Material Command's Telemedicine & Advanced Technology Research Center (TATRC), under Contract No. W81XWH-16-C-0062. The views, opinions, and/or findings contained in this report are those of the authors and should not be construed as an official Department of the Army position, policy, or decision unless so designated by other documentation.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

1. Odedra, S., Prior, S., Karamanoglu, M.: Investigating the mobility of unmanned ground vehicles. In: Proceedings of the international conference on manufacturing and engineering systems. Huwei, Yunlin (2009)
2. Murphy, R.R.: A decade of rescue robots. In: 2012 IEEE/RSJ international conference on intelligent robots and systems, pp. 5448–5449 (2012)
3. Murphy, R.R., et al.: Search and rescue robotics. In: Siciliano, B., Khatib, O. (eds.) Springer handbook of robotics, pp. 1151–1173. Berlin, Heidelberg (2008)
4. Nagatani, K. et al.: Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots. J. F. Robot. **30**(1), 44–63 (2013)
5. Murphy R.R.: Disaster robotics, vol. 1. The MIT Press, Cambridge (2014)
6. Davids, A.: Urban search and rescue robots: from tragedy to technology. IEEE Intell. Syst. **17**(2), 81–83 (2002)
7. Snyder, R.G.: Robots assist in search and rescue efforts at WTC. IEEE Robot. Autom. Mag. **8**(4), 26–28 (2001)

8. Murphy, R., Casper, J., Hyams, J., Micire, M., Minten, B.: Mobility and sensing demands in USAR. In: 2000 26th annual conference of the ieee industrial electronics society. IECON 2000. 2000 IEEE International Conference on Industrial Electronics, Control and Instrumentation. 21st Century Technologies, vol. 1, pp. 138–142 (2000)
9. Murphy, R.R., Kravitz, J., Stover, S.L., Shoureshi, R.: Mobile robots in mine rescue and recovery. IEEE Robot. Autom. Mag. **16**(2), 91–103 (2009)
10. U.S. Army Medical Research and Material Command, "Unmanned Systems Teaming for Semi-Autonomous Casualty Extraction," SBIR-STTR, 2017. [Online]. Available: https://www.sbir.gov/sbirsearch/detail/1319095. [Accessed: 11-Feb-2017]
11. Beebe, M.K., Gilbert, G.R.: Robotics and unmanned systems – 'Game changers' for combat medical missions. In: Proc. NATO RTO-HFM 182 Symp. Adv. Technol. New Proced. Med. F. Oper. (2010)
12. Paden, B., Cap, M., Yong, S.Z., Yershov, D., Frazzoli, E.: A survey of motion planning and control techniques for Self-Driving urban vehicles. IEEE Trans. Intell. Veh. **1**(1), 33–55 (2016)
13. LaValle, S.M.: Planning algorithms. Cambridge University Press, Cambridge (2006)
14. Lamon, P. 3D-Position tracking and control for all-terrain robots, 1st edn., vol. 43. Springer, Berlin (2008)
15. Tarokh, M., McDermott, G.J.: Kinematics modeling and analyses of articulated rovers. IEEE Trans. Robot. **21**(4), 539–553 (2005)
16. Kumar, P., Saab, W., Ben-Tzvi, P.: A hybrid tracked-wheeled multi-directional mobile robot. IEEE/ASME Trans. Mechatronics, p Under revision (2017)
17. Ben-Tzvi, P., Goldenberg, A.A., Zu, J.W.: Articulated hybrid mobile robot mechanism with compounded mobility and manipulation and on-board wireless sensor/actuator control interfaces. Mechatronics **20**(6), 627–639 (2010)
18. Currier, P.N., Wicks, A.L.: A novel method for prediction of mobile robot maneuvering spaces. J. Terramechanics **50**(2), 85–97 (2013)
19. Thrun, S. et al.: Stanley: the robot that won the DARPA grand challenge. J. F. Robot. **23**(9), 661–692 (2006)
20. Bacha, A. et al.: Odin: Team VictorTango's entry in the DARPA urban challenge. J. F. Robot. **25**(8), 467–492 (2008)
21. Urmson, C.P. et al.: High speed navigation of unrehearsed terrain?: Red Team Technology for Grand Challenge 2004 (2004)
22. Currier, P.N.: A method for modeling and prediction of ground vehicle dynamics and stability in autonomous systems. Virginia Polytechnic Institute and State University (2011)
23. Chu, K., Lee, M., Sunwoo, M.: Local path planning for Off-Road autonomous driving with avoidance of static obstacles. IEEE Trans. Intell. Transp. Syst. **13**(4), 1599–1616 (2012)
24. Goswami, A.: Hierarchical Off-Road Path planning and its validation using a scaled autonomous car. Clemson university (2017)
25. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press. (2005)
26. Siciliano, B., Khatib, O.: Springer Handbook of Robotics. Springer-Verlag New York, Inc., Secaucus (2016)
27. Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., Konolige, K.: The office marathon: robust navigation in an indoor office environment. In: 2010 IEEE international conference on robotics and automation, pp. 300–307 (2010)
28. Reina, G., Bellone, M., Spedicato, L., Giannoccaro, N.I.: 3D Traversability awareness for rough terrain mobile robots. Sens. Rev. **34**(2), 220–232 (2014)
29. Castejón, C., Boada, B.L., Blanco, D., Moreno, L.: Traversable region modeling for outdoor navigation. J. Intell. Robot. Syst. Theory Appl. **43**(2–4), 175–216 (2005)
30. Garrido, S., Moreno, L., Martín, F., Álvarez, D.: Fast marching subjected to a vector field–path planning method for mars rovers. Expert Syst. Appl. **78**, 334–346 (2017)
31. Raja, R., Dutta, A., Venkatesh, K.S.: New potential field method for rough terrain path planning using genetic algorithm for a 6-wheel rover. Rob. Auton. Syst. **72**, 295–306 (2015)
32. Amorim, D., Ventura, R.: Towards efficient path planning of a mobile robot on rough terrain, pp. 22–27 (2014)
33. Konolige, K.: A gradient method for realtime robot control. In: Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113), vol. 1, pp. 639–646 (2000)
34. Fan, X., et al.: Integrated planning and control of large tracked vehicles in open terrain. In: 2010 IEEE international conference on robotics and automation, pp. 4424–4430 (2010)
35. Kelly, A., Stentz, A.: Rough terrain autonomous mobility part 2?: an active vision , predictive control approach. Auton. Robots **5**, 163–198 (1998)
36. Seraji, H., Howard, A.: Behavior-based robot navigation on challenging terrain: a fuzzy logic approach. IEEE Trans. Robot. Autom. **18**(3), 308–321 (2002)
37. Howard, A., Werger, B., Seraji, H.: Integrating terrain maps into a reactive navigation strategy. In: 2003 IEEE international conference on robotics and automation (Cat. No.03CH37422), vol. 2, pp. 2012–2017 (2003)
38. Cherif, M., Laugier, C.: Using physical models to plan safe and executable motions for a rover moving on a terrain. In: Int. Workshop on Intelligent Robotic Systems, pp. 57–66 (1993)
39. Iagnemma, K., Dubowsky, S.: Mobile robots in rough terrain. In: Springer tracts in advanced robotics, vol. 12, p. XII, 111, no. 8, Springer, Berlin (2004)
40. Brunner, M., Bruggemann, B., Schulz, D.: Hierarchical rough terrain motion planning using an optimal sampling-based method. In: Proc. - IEEE Int. Conf. Robot. Autom., pp. 5539–5544 (2013)
41. Kuwata, Y., Fiore, G.A., Teo, J., Frazzoli, E., How, J.P.: Motion planning for urban driving using RRT. In: 2008 IEEE/RSJ international conference on intelligent robots and systems, pp. 1681–1686 (2008)
42. Pepy, R., lambert, A., Mounier, H.: Path planning using a dynamic vehicle model. 2006 2nd International Conference on Information &, Communication Tsechnologies **1**(1), 781–786 (2006)
43. LaValle, S.M., jr.J.J.K: Randomized kinodynamic planning. I. J. Robot. Res. **20**(5), 378–400 (2001)
44. Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., How, J.P.: Real-Time Motion planning with applications to autonomous urban driving. IEEE Trans. Control Syst. Technol. **17**(5), 1105–1118 (2009)
45. Webb, D.J., van den Berg, J.: Kinodynamic RRT*: optimal motion planning for systems with linear differential constraints. CoRR, vol. abs/1205.5 (2012)
46. Koenig, S., Likhachev, M. In: Proc. Eighteenth Natl. Conf. Artif. Intell., pp. 476–483 (2002)
47. (Tony) Stentz, A.: The Focussed d* algorithm for Real-Time replanning. In: Proceedings of the international joint conference on artificial intelligence (1995)
48. Haug, E.J.: Computer aided kinematics and dynamics of mechanical systems. Allyn & Bacon Inc., Needham (1989)
49. Ladd, A.M.: Motion planning for physical simulation. Rice University (2006)
50. Coutinho, M.G.: Guide to dynamic simulations of rigid bodies and particle systems london. Springer, London (2013)
51. Roennau, A., Sutter, F., Heppner, G., Oberlaender, J., Dillmann, R.: Evaluation of physics engines for robotic simulations with a special focus on the dynamics of walking robots. In: 2013 16th

International Conference on Advanced Robotics (ICAR), pp. 1–7 (2013)
52. Bullet Physics Engine Ver. 2.87. [Online]. Available: http://bulletphysics.org/wordpress/. [Accessed: 11-Feb-2017]
53. Egerstedt, M., Johansson, K., Lygeros, J., Sastry, S.: Behavior based robotics using regularized hybrid automata. In: Proceedings of the 38th IEEE conference on decision and control (Cat. No.99CH36304), vol. 4, pp. 3400–3405 (1999)
54. Egerstedt, M.: Controls for the masses [Focus on education]. IEEE Control Syst. **33**(4), 40–44 (2013)
55. Cheng, P.C.P., LaValle, S.M.: Reducing metric sensitivity in randomized trajectory design. Proc. 2001 IEEE/RSJ Int. Conf. Intell. Robot. Syst. Expand. Soc. Role Robot. Next Millenn. (Cat. No.01CH37180) **1**, 43–48 (2001)
56. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. Int. J. Rob. Res. **30**(7), 846–894 (2011)
57. Coppelia Robotics, V-REP. [Online]. Available: http://www.coppeliarobotics.com/. [Accessed: 11-Feb-2017]
58. Hazevoet, J., Anders, M., Huish, I.: ANT Landscape Extension for Blender 2.6. [Online]. Available: https://wiki.blender.org/index.php/Extensions:2.6/Py/Scripts/AddMesh/ANTLandscape. [Accessed: 01-Jan-2017]
59. Blender v2.6. [Online]. Available: https://www.blender.org/. [Accessed: 01-Jan-2017]
60. Kumar, A., Ben-Tzvi, P.: Spatial object tracking system based on linear optical sensor arrays. IEEE Sens. J **16**(22), 7933–7940 (2016)
61. Hsu, D., Kindel, R., Latombe, J.-C., Rock, S.: Randomized kinodynamic motion planning with moving obstacles. Int. J. Rob. Res. **21**(3), 233–255 (2002)
62. Tang, S.H., Kamil, F., Khaksar, W., Zulkifli, N., Ahmad, S.: Robotic motion planning in unknown dynamic environments: existing approaches and challenges. In: 2015 IEEE Int. Symp. Robot. Int.ll. Sensors, pp. 288–294 (2015)
63. POZYX positioning system. [Online]. Available: https://www.pozyx.io/. [Accessed: 01-Jan-2017]

Bijo Sebastian received his B.Tech degree in Mechanical Eng. from College of Engineering, Trivandrum, India in 2013. He received his MS in Mechatronics from Central Mechanical Engineering Research Institute, West Bengal, India in 2015. He is currently pursuing Ph.D. at the Virginia Polytechnic Institute and State University under the supervision of Prof. P. Ben-Tzvi. His research interests include autonomous mobile robots, design and control of Exo-skeletons, motion planning and computer vision.

Pinhas Ben-Tzvi received the B.S. degree (summa cum laude) in mechanical engineering from the Technion—Israel Institute of Technology, Haifa, Israel, and the M.S. and Ph.D. degrees in mechanical engineering from the University of Toronto, Toronto, Canada. He is currently an Associate Professor of Mechanical Engineering and Electrical and Computer Engineering, and the founding Director of the Robotics and Mechatronics Laboratory at Virginia Tech, Blacksburg, VA, USA. Before joining the University of Toronto in 2002, he was an R&D Engineer with General Electric Medical Systems Company, developing medical diagnostic robotic and mechatronic systems. His current research interests include robotics and intelligent autonomous systems, mechatronics, human-robot interactions, dynamic systems and control, mechanism design and system integration, and novel sensing and actuation. Application areas are varied and range from search & rescue on rough terrain to medical diagnostics, surgery, and therapy. Dr. Ben-Tzvi is the recipient of the 2013 GW SEAS Outstanding Young Researcher Award and the GW SEAS Outstanding Young Teacher Award, as well as several other honors and awards. Dr. Ben-Tzvi is a Technical Editor of the IEEE/ASME Transactions on Mechatronics, Associate Editor of IEEE Robotics and Automation Magazine, Associate Editor of ASME Journal of Mechanisms and Robotics, and Associate Editor for the Int'l Journal of Control, Automation and Systems. He is a senior member of the Institute of Electrical and Electronics Engineers (IEEE) and a member of the American Society of Mechanical Engineers (ASME).