Performance-Based Rough Terrain Navigation for Nonholonomic Mobile Robots

Yi Guo , Lynne E. Parker , David Jung and Zhaoyang Dong

***Abstract*— This paper addresses path planning and control of mobile robots in rough terrain environments. Previous research separates path planning and control into two different problems and addresses them in different contexts. Instead, we formulate these issues in connected modules with performance requirement considerations in each module. We advocate the idea that by incorporating criterion-optimizing design in each module and organizing them in a behavior-based architecture, performance issues (e.g., robot safety, or geometric, time-based, and physics-based criteria) are adequately addressed. A feedback control strategy is used for trajectory tracking, and closed-loop stability of error dynamics is granted. Simulation results show that the trajectory controller is robust with respect to initial conditions and model uncertainties.**

I. Introduction

There are considerable research efforts towards solving the mobile robot navigation problem in different applications in indoor or outdoor environments (see [12] and the survey paper [20]). For some navigation tasks, such as planetary exploration ([18]), robots are required to travel long distances within constrained resources (energy, etc.), and a prior map exists at certain degree of accuracy (see [10], [13]). In such cases, effective path planning and motion control algorithms are needed to achieve the goal while meeting certain performance requirements, such as robot safety, or geometric, time-based, and physics-based criteria. However, where uncertainties exist (e.g., local map inaccuracy, robot sensor and perception errors), theoretically optimal solutions may not be achievable. How to adequately address the performance issue while maintaining safe robot operations is not straightforward. Rather than address all of the issues which arise in the complex navigation problem, we focus on autonomous path planning and control.

Although path planning and control are closely related in the robot navigation problem, they are usually treated as two separate problems in much of the existing literature. Planning is the determination of the geometric path points for the mobile robots to track, and control is the determination of the physical input to the robot motion components. These issues are typically discussed using methods in different areas such as those in artificial intelligence and control theory. Such a separation makes it difficult to address robot performance in a complete application, since the discrete geometric path points planned in the first step may not be efficiently tracked in the second step, thus losing the meaning of optimization in each step. For example, in a typical steering arbiter method ([23]), discrete steering commands are generated for goal acquisition using the search method D^*. However, the discontinuity of the control commands may not produce a satisfactory path tracking result in practice, and will not be applicable to high speed traveling. Also the feedforward control method ([9]) has disadvantages from the robot motion point of view, as it needs a relatively accurate model and is not robust to uncertainties. In this paper, we use feedback control for robot motion, which prevents the increase of trajectory errors from the reference trajectory.

Planning and control with optimal distance or time strategies are discussed in [2], [5]. Physics-based rough terrain navigation is discussed in [4], where static model-based safety evaluation is studied assuming the robot moves slowly and dynamic effects are negligible. Although these kinds of analyses are necessary for performance-based navigation in rough terrain environments, they are limited in their application. For example, in cases where high speed, long distance travel is required, static stability evaluation ([4]) is not enough.

This paper presents algorithms for mobile robot path planning and motion control. The approach comprises three modules: path searching, trajectory generation, and trajectory tracking. In each module, we explicitly address the performance considerations, which are also uniform with the other component modules. In implementation, deliberate planning can be implemented before the robot moves, and locally revised path re-planning can be achieved dynamically while the robot runs. A feedback control scheme is applied, which improves system robustness with respect to uncertainties, and guarantees convergence of the closed-loop system to the reference trajectory.

The rest of the paper is organized as follows. In Section 2, the navigation problem is defined with the description of a nonholonomic model of a mobile robot, and performance criteria are proposed. Then in Section 3, the performance-based navigation algorithms are described in three partitioned and connected modules. A software flowchart is presented in Section 4, which was partly implemented in a behavior-based vehicle planning and control simulation system. An example path on a 3D Mars-like terrain is shown; also simulation results of the trajectory controller are demonstrated with good robustness. Finally, the paper is concluded with brief remarks in Section 5.

Yi Guo is with the School of Electrical Engineering and Computer Science, University of Central Florida, USA. Work done while working at the Oak Ridge National Laboratory (ORNL), Oak Ridge, TN 37831, USA.

Lynne E. Parker is with the Department of Computer Science, University of Tennessee, USA. Work done while working at ORNL.

David Jung is with the Computer Science and Mathematics Division, Oak Ridge National Laboratory, USA.

Zhaoyang Dong is with the School of Information Technology and Electrical Engineering, University of Queensland, Australia.

II. Problem Statement

We consider a nonholonomic mobile robot $\mathcal{A}$ driven by two differential wheels, whose kinematics is governed by:

$$\begin{aligned} \dot{x} &= v\cos(\theta) \\ \dot{y} &= v\sin(\theta) \\ \dot{\theta} &= \omega \end{aligned} \tag{1}$$

where vector $q = (x, y, \theta) \in \Re^2 \times \mathcal{S}$ is a configuration specifying the horizontal position and heading of the robot in the global frame; v and ω are the translational and angular velocities respectively. The relationship between (v, ω) and the left and right wheel velocities $(\omega_{left}, \omega_{right})$ is described by:

$$\begin{aligned} v &= \frac{r_w}{2}(\omega_{right} + \omega_{left}) \\ \omega &= \frac{r_w}{d_w}(\omega_{right} - \omega_{left}) \end{aligned} \tag{2}$$

where r_w is the radius of the wheels, and d_w is the azimuth length between the wheels.

The robot has the following dynamic limitations on velocities and accelerations:

$$\begin{aligned} |v| \le v_{\max} \qquad & |\omega| \le \omega_{\max} \\ |\dot{v}| \le a_{\max} \qquad & |\dot{\omega}| \le \gamma_{\max} \end{aligned} \tag{3}$$

where a, γ are translational and angular accelerations respectively.

The robot operates in a rough terrain, which is described by surface patches defined from an elevation map in z associated with a regular grid in (x, y). We assume the robot knows its initial configuration q_{init} and goal configuration q_{goal}. The navigation problem is to decide wheel velocity inputs $(\omega_{left}, \omega_{right})$ within constraint (3), so that the robot achieves q_{goal} starting from q_{init} according to certain performance criteria.

We define performance as:

- Safety: The robot should avoid any dangerous terrain which causes static instability of the robot by certain margins;
- Geometry-based criteria: criteria that relate to the geometry, such as shortest distance;
- Time-based criteria: criteria that are a function of time, such as shortest time;
- Physics-based criteria: criteria depending on the physical configuration of the robot, such as optimal fuel or energy.

We define two concepts which will be used later:

- Path: A path is a sequence of geometric points (x, y) which the robot is to pass through;
- Trajectory: A trajectory is a set of valid robot configurations q_r and velocity profiles (v_r, ω_r) depending on time t.

III. Performance-Based Navigation Algorithms

The navigation algorithm comprises three modules:

- Path searching: To search for a practical path according to the existing map and given performance measurements;
- Trajectory generation: To determine a reference trajectory which is a function of time;
- Trajectory tracking: To calculate current wheel velocity inputs $(\omega_{left}, \omega_{right})$ according to the reference trajectory generated in the previous step, and form the feedback control loop which takes current measurements of the robot configuration (x_c, y_c, θ_c) as feedback variables.

In the next subsection, we discuss each module in detail.

A. Path Searching

Assume robot $\mathcal{A}$ moves on a planar workspace $\mathcal{W} \subset \Re^2$, and the prohibited space is $\mathcal{W}_{prohibit} \subset \Re^2$. The prohibited space defines regions in which the robot should not travel due to mission constrains (e.g., minefields) or undetectable navigation challenges (e.g., quicksand). We define a safety margin by a positive constant ε, and $\mathcal{W}^{\varepsilon}_{prohibit}$ is the space that is within ε distance of $\mathcal{W}_{prohibit}$. Denote the valid search space to be $\mathcal{W}^{\varepsilon}_{free} = \mathcal{W} \setminus \mathcal{W}^{\varepsilon}_{prohibit}$. The path searching task is to find a sequence of geometric points from the start to the goal in $\mathcal{W}^{\varepsilon}_{free}$ according to the existing elevation map defined on the regular grid.

A^* is a widely used search algorithm to find a path through the terrain grid from the start position to the goal. Its efficiency is highly determined by the cost function ([16]). The performance issue in this module is reflected by carefully choosing the cost function.

We first define the following parameters:

Terrain roughness: At each grid (x, y), terrain roughness ϕ is defined as the variance of elevation z of all points on a circular domain centered at (x, y) with a radius R related to the size of the robot, that is,

$$\phi(x, y) = \text{var}\sqrt{z(R)}.$$

Terrain slope: Terrain slope k from the grid point (x_1, y_1) to (x_2, y_2) is defined to be

$$k_{12} = \frac{z_2 - z_1}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}$$

where z_1 and z_2 are the elevations at (x_1, y_1) and (x_2, y_2) respectively.

To decide which node is to be expanded first in A^*, the following elements are taken into consideration for choosing the cost function:

- Distinguishing between challenging and benign terrain: A primary requirement for static stability of the robot is the need for ground contact support on the path. The distinction between challenging and benign terrain can be defined through parameters terrain roughness ϕ and slope k. The solution depends on the physical configuration of the robot.
- Cost to travel over current patch: Power consumption is related to the motor torques and the tractive forces. It is a function of the parameters terrain roughness ϕ and terrain slope k, denoted by $P(\phi, k)$, which also depends on the physical configuration of the robot.
- Pre-set performance requirement: Geometry based criteria such as shortest path are formulated in metrics: the travel distance from grid point (x_1, y_1) to (x_2, y_2) is:

$$D_{12} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}.$$

The search algorithm finds a path with minimized cost function, which is defined as:

$$f_{pp} \quad = \quad \alpha_1 \sum P(\phi, k) + \alpha_2 \sum D \tag{4}$$

where $\sum P(\phi, k)$ is the total cost to travel on the designated path, $\sum D$ is the total distance to travel from the start position to the goal position, and α_1, α_2 are weighting factors.

In the standard A^* algorithm ([16]), there is one more step to test when expanding a node during the search: if the node to expand is within the challenging terrain, remove it from the expanding list. More explicitly, if (x, y) is within τ (a positive constant) distance of a challenging terrain, then remove node (x, y) from the successors list. Here the positive constant τ is a design parameter that reflects a safety margin.

B. Trajectory Generation

After a sequence of path points is generated in the previous step, the task in this step is to smooth the path and generate a reference trajectory (x_r, y_r, θ_r) and (v_r, ω_r) associated with time t for the robot to track. It consists of the following two sub-steps:

Generating a smooth curve: With the input (output of the first step) of a sequence of path points, the first task is to use interpolation to fit a smooth curve though the data points. Interpolation methods such as Lagrange interpolation or splines are applicable. For long trajectories, it is possible to set waypoints such that there are smooth curve connections with simple continuous curvature between two adjacent waypoints. Examples of smooth curve connections with simple curvature are straight lines, arcs, clothoids, *etc.* (see [7], [15], [14], [21] and references therein), which are easy for trajectory tracking in the next step. The output of this sub-step is a *piecewise* smooth curve with valid robot configurations (x_r, y_r, θ_r).

Generating reference velocity profiles: It has been proved in [19] that time-optimal trajectories are bang-bang; that is, the translational and angular accelerations are either zero or maximum in absolute value. However, we have to consider the following:

- In areas that are δ (a positive constant) distance from $\mathcal{W}^{\varepsilon}_{prohibit}$, we expect the robot to keep a relatively slow speed, that is,

$$|v| \leq v_I$$
when $\mathcal{A}$ is within δ distance from $\mathcal{W}^{\varepsilon}_{prohibit}$

  where v_I is a certain safe translational velocity value which is dependent on the robot physics.
- On rough terrain, the robot may not be able to make point turns due to high terrain-induced transversal forces. Therefore, there should be robot turning constraints on each terrain patch; that is,

$$|\omega| \leq \omega_I$$
when $\mathcal{A}$ is in terrain patch centered at (x, y)

  where ω_I is a certain safe angular velocity value, dependent on the robot physics.

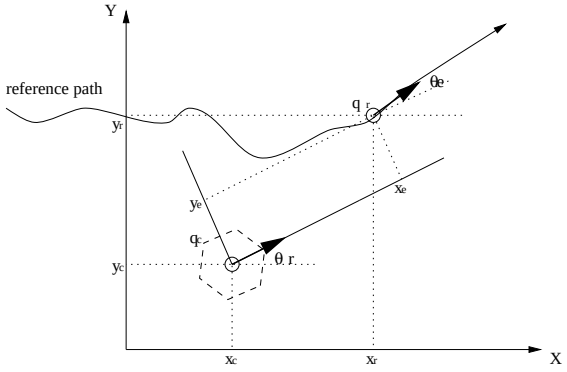

Fig. 1. Error posture $q_e = (x_e, y_e, \theta_e)$: q_r is the reference configuration, q_c is the current robot configuration.

The output of this sub-step is a velocity profile (v_r, ω_r) associated with time t, which are bounded by (3).

The generated trajectory data is stored in the following table format for use at the next step:

segment number
segment type
segment parameters
ending waypoint (x, y, θ)
velocity (v, ω)

where segment type can be a straight line, a piece of an arc, or a spline, with their parameters defined by the curvature or other parameters uniformly describing the curve.

C. Trajectory Tracking

The task in this step is to construct the wheel velocities input such that the robot tracks the reference trajectory generated in the previous step.

First, we need to set up an error posture. Suppose the reference posture is $q_r = (x_r, y_r, \theta_r)^T$, and the current posture of the robot is $q_c = (x_c, y_c, \theta_c)^T$. The error posture $q_e = (x_e, y_e, \theta_e)^T$ is a transformation of the reference posture q_r in a local coordinate system with an origin of (x_c, y_c) and an X-axis in the direction of θ_c, which is defined as (see Figure 1):

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} \quad = \quad \begin{bmatrix} \cos\theta_c & \sin\theta_c & 0 \\ -\sin\theta_c & \cos\theta_c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x_c \\ y_r - y_c \\ \theta_r - \theta_c \end{bmatrix} \tag{5}$$

Differentiating both sides of (5), using (1), and after simplifying, it turns to be (see [8]):

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} \quad = \quad \begin{bmatrix} \omega y_e - v + v_r \cos\theta_e \\ -\omega x_e + v_r \sin\theta_e \\ \omega_r - \omega \end{bmatrix} \tag{6}$$

where (v, ω) is the velocity vector to be designed.

Equation (6) is called *error dynamics*. Now the trajectory tracking is transferred to a conventional control problem, which is to determine the control inputs v, ω as functions of $(x_e, y_e, \theta_e, v_r, \omega_r)$ such that the closed-loop system is asymptotically stable; that is, the error configuration q_e tends to its equilibrium 0 eventually.

It should be pointed out such a control system (6) cannot be stabilized by continuously differentiable, time-invariant, state feedback control laws. Complex-structured controllers have been proposed in various literature (see [3], [6], [11] and the references therein). We apply the following nonlinear feedback control law:

$$\begin{aligned} v &= \xi(\omega_r, v_r)x_e + v_r \cos\theta_e \\ \omega &= \omega_r + \xi(\omega_r, v_r)v_r \frac{\sin\theta_e}{\theta_e} y_e + c_2\theta_e \end{aligned} \tag{7}$$

where $\xi(\omega_r, v_r)$ is a nonlinear function of (ω_r, v_r) as:

$$\xi(\omega_r, v_r) = 2c_1\sqrt{\omega_r^2 + c_2 v_r^2};$$

and $c1, c2$ are positive design constants.

Note that the velocity input (v, ω) should be bounded by (3), which corresponds to a saturation effect on (7).

It has been proved in [1] that controller (7) asymptotically stabilizes (6). That is, the closed-loop system of (6) with (v, ω) taking the form of (7) is asymptotically stable.

From (2), it is straightforward to obtain the wheel velocities:

$$\begin{aligned} \omega_{left} &= \frac{2v - d_w\omega}{2r_w} \\ \omega_{right} &= \frac{2v + d_w\omega}{2r_w} \end{aligned} \tag{8}$$

Figure 2 shows the feedback control diagram governing robot motion.

IV. Implementation

The proposed navigation algorithms have been partially implemented in a vehicle planner and control simulation for a cooperative autonomous mining system (CAMS). CAMS is a behavior-based multi-robot control system that adopts the ALLIANCE architecture ([17]) for multi-vehicle action selection. The proposed navigation algorithm is denoted as the path planner (PP) behavior. The diagram in Figure 3 shows the software flow-chart of PP.

In Figure 3, the local path and trajectory replanning is composed of local path re-searching and local trajectory segment re-generating. D^* ([22]) can be used as a dynamic path planning method which is good at real time replanning of the path. As the trajectory is defined by piecewise continuous trajectory segments, unexpected obstacles detected by the onboard sensors can immediately trigger the local path planner to insert a stop segment at the current path location, and then resume the move by generating a startup segment to continue the robot along the revised path. Unlike [9], where feedforward control is used for trajectory searching, feedback control is used in the proposed method to track the previously generated trajectory. The reason behind this is that vehicle dynamics is more reliable these days, and feedback control improves system robustness. Therefore, in the current implementation, local path replanning deals with local incorrectness in the existing map, and feedback control deals with uncertainties arising from robot motion such as sensor noise.

An example terrain in the 3D simulator environment is shown in Figure 4 with a path planned, which is typical from applications such as site preparation in space-based robot colonies. A shortest path is planned according to the elevation map of the terrain. The generated trajectory data is shown in Table I for a time optimal trajectory, where segment type 0=line, 1=arc, 2=spline; segment parameters for line are the slope and displacement in the Cartesian coordinate, segment parameters for spline are polynomial coefficients; and 1 is the maximum acceleration scale in the simulator. Velocity (v, ω) can be calculated by the supplied acceleration in the program to produce a smooth ramp startup and a smooth ramp-down stop on each trajectory segment.

V. Conclusions

In this paper, we have discussed the autonomous path planning and trajectory control problem. We partition the design into three modules: path searching, trajectory generation, and trajectory tracking. In each module, performance issues are addressed in the context of robot safety considerations, which are summarized as:

- In path searching, a geometry-based optimal criterion is considered by choosing a terrain condition and robot physics dependent cost function in the searching algorithm;
- In trajectory generation, a time-based criterion is considered by optimally distributing reference velocities;
- In trajectory tracking, nonlinear feedback control laws are applied so that the motion of the robot is robust with respect to position or measurement uncertainties.

From the system point of view, the second module prepares the smooth simple curvature curves to ease the third module for robot tracking. This design is considered necessary for long distance traveling of vehicle-like systems. Alternative discrete representations such as the steering arbiter method ([23]) do not allow this type of high speed, long-distance frame, due to the discontinuity of control commands.

The algorithms can be implemented in a behavior-based architecture, where the local replanning of the trajectory is plugged in. A software flow-chart is presented, which was partly implemented in a 3D simulator environment for CAMS. We show in the simulation a typical Mars-like terrain with a path planned on it. In future work, we plan to run the proposed algorithms on our ATRV-mini all-terrain mobile robots in an outdoor testing arena featuring challenging Mars-like terrain such as hills, rocks and ditches. A coarse elevation map will be produced before the navigation starts. The equipped onboard GPS will return real-time robot location, and a laser rangefinder will detect local obstacles.

References

[1] C. C. de Wit, B. Siciliano, and G. Bastin. *Theory of Robot Control*. Springer, 1996, New York.

[2] P. Graglia and A. Meystel. Planning minimum time trajectory in the traversability space of a robot. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 82–87, 1987.

[3] Y. Guo, Z. P. Jiang, and D. J. Hill. Decentralized robust disturbance attenuation for a class of large-scale nonlinear systems. *Systems & Control Letters*, 37:71–85, 1999.

[4] K. Iagnemma, F. Genot, and S. Dubowsky. Rapid physics-based rough-terrain rover planning with sensor and control uncertainty. In *Proceedings of IEEE International Conference on Robotics and Automation*, Detroit, MI, May 1999.

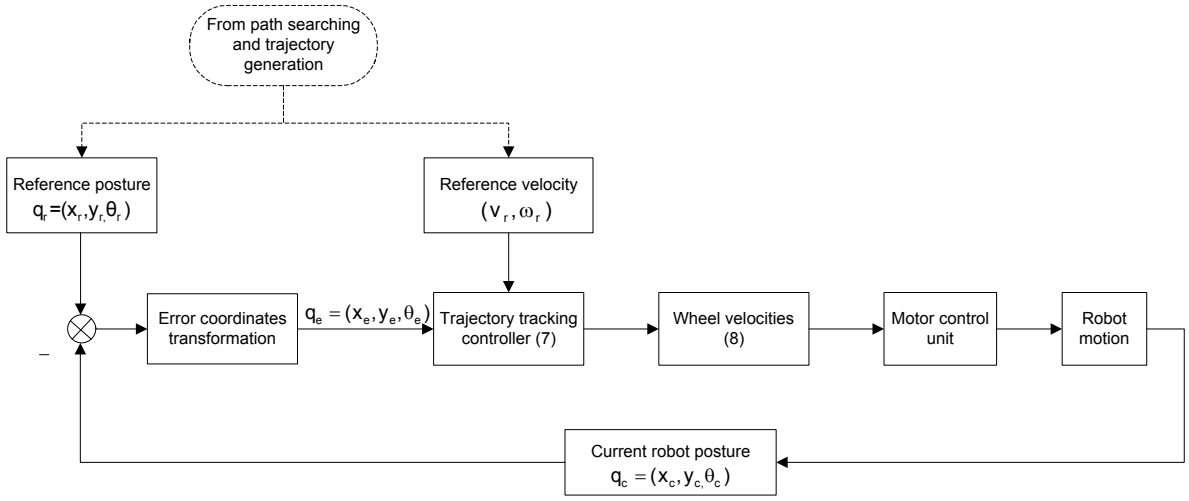

Fig. 2. Feedback control loop.

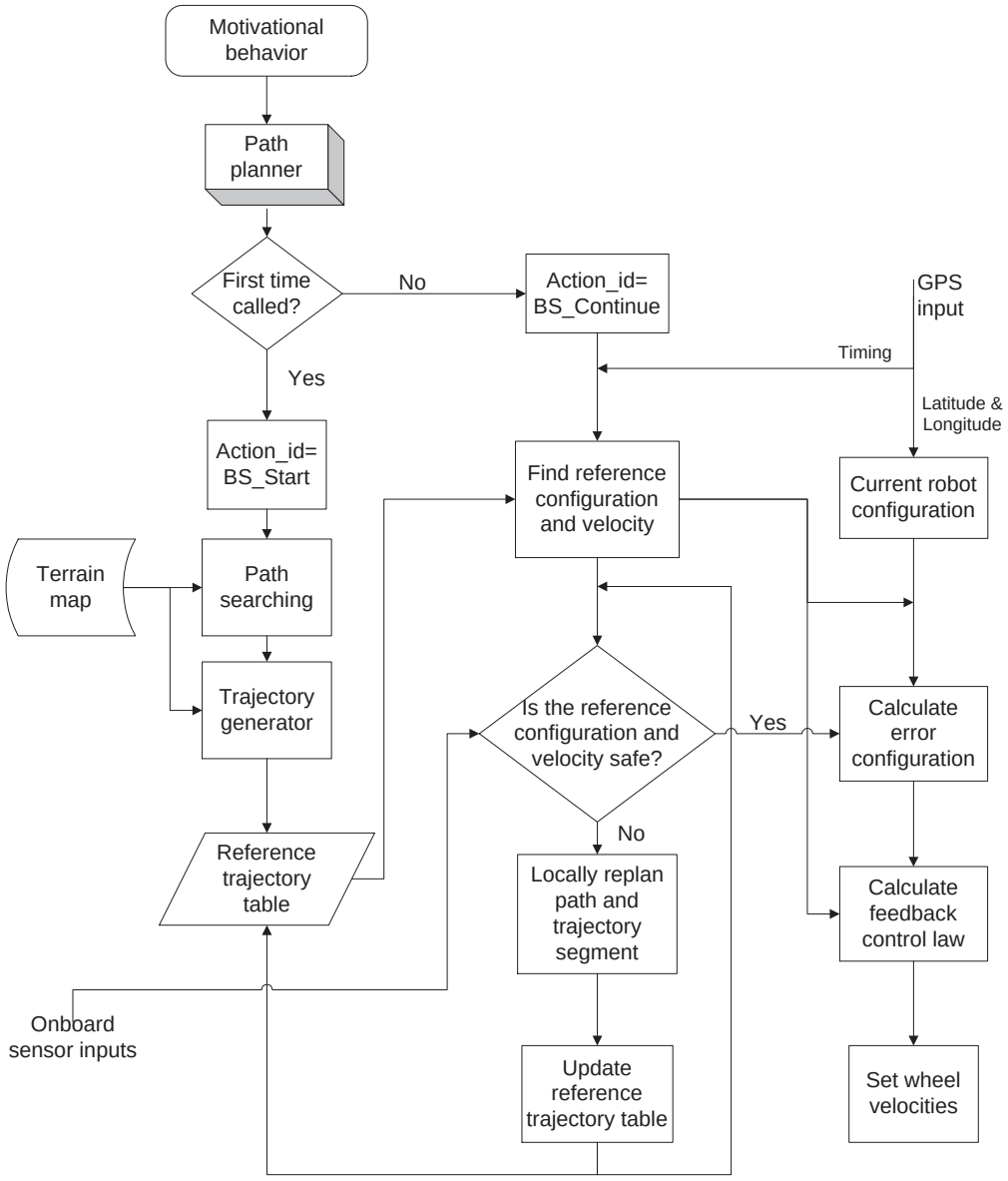

Fig. 3. Flow-chart of implementation.

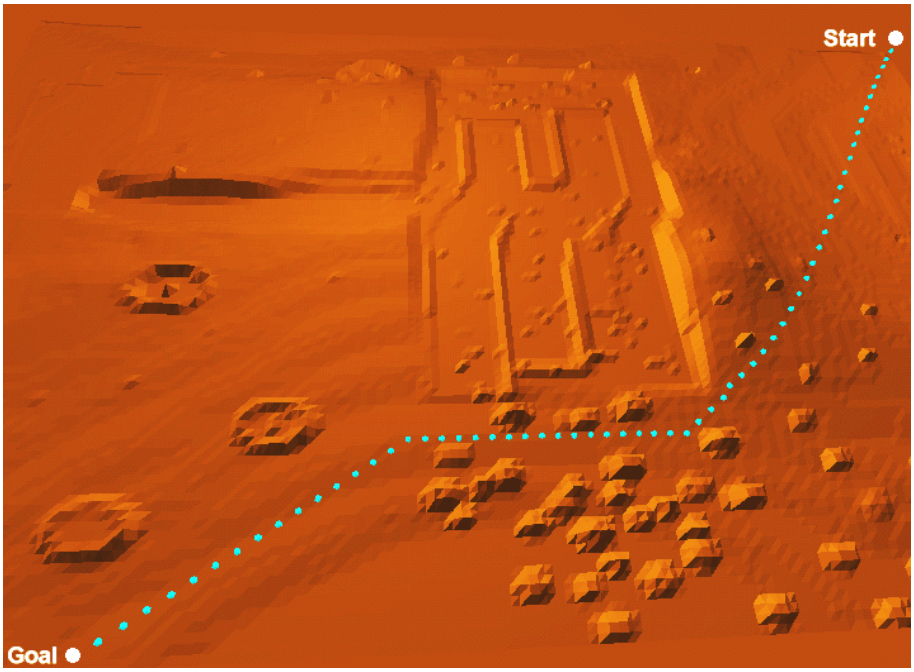

Fig. 4. A path on the terrain in the 3D simulator environment of CAMS.

TABLE I
REFERENCE TRAJECTORY DATA

segment number	segment type	segment parameters	ending waypoint	acceleration
1	2	(-0.0012,0.0430,1.0336,0.0)	(32.0,37.78,0.86)	1
2	0	(0.0,38.9550)	(49.0,38.9550,0.67)	1
3	0	(0.65,7.77)	(65.0,50.0,0.65)	1

[5] K. Jiang, L. D. Seneviratne, and S. W. E. Earles. A shortest path based path planning algorithm for nonholonomic mobile robots. *Journal of Intelligent and Robotic Systems*, 24:347–366, 1999.

[6] Z. P. Jiang and H. Nijmeijer. Tracking control of mobile robots: A case study in backstepping. *Automatica*, 33:1393–1399, 1997.

[7] Y. Kanayama and B. I. Hartman. Smooth local path planning for autonomous vehicles. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1265–1270, 1989.

[8] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi. A stable tracking control method for an autonomous mobile robot. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 384–389, 1990.

[9] A. Kelly and A. T. Stentz. An approach to rough terrain autonomous mobility. *Autonomous Robots*, April 1998.

[10] A. Kelly and A. T. Stentz. Rough terrain autonomous mobility - Part 1: A theoretical analysis of requirements. *Autonomous Robots*, (5):129 – 161, May 1998.

[11] K. C Koh and H. S. Cho. A smooth path tracking algorithm for wheeled mobile robots with dynamic constraints. *Journal of Intelligent and Robotic Systems*, 24:367–385, 1999.

[12] J-C. Latombe. *Robot motion planning*. Kluwer Academic Publishers, 1991.

[13] S. L. Laubach and J. W. Burdick. An autonomous sensor-based path-planner for planetary microrovers. In *Proceedings of IEEE International Conference on Robotics and Automation*, Detroit, Mich., 1999.

[14] W. Nelson. Continuous-curvature paths for autonomous vehicles. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1260–1264, 1989.

[15] W. L. Nelson. Local path control for an autonomous vehicle. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1504–1510, 1988.

[16] N. J. Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann Publishers, 1998, San Francisco.

[17] L. E. Parker. ALLIANCE: An architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14:220–240, 1998.

[18] L. E. Parker, D. Jung, T. Huntsberger, and P. Pirjanian. Opportunistic adaptation in space-based robot colonies: Application to site preparation. In *Proceedings of World Automation Congress (WAC)*, Maui, Hawaii, June 2000.

[19] D. Reister and F. G. Pin. Time-optimal trajectories for mobile robots with two independently driven wheels. *The International Journal of Robotics Research*, 13:38–54, 1994.

[20] M. A. Salichs and L. Moreno. Navigation of mobile robots: open questions. *Robotica*, 18:227–234, 2000.

[21] A. Scheuer and Th. Fraichard. Continuous-curvature path planning for car-like vehicles. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, pages 1260–1264, 1997.

[22] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3310–3317, 1994.

[23] A. T. Stentz and M. Hebert. A complete navigation system for goal acquisition in unknown environments. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, August 1995.