A Recursive Receding Horizon Planning for Unmanned Vehicles

Bin Zhang, *Senior Member, IEEE*, Liang Tang, *Member, IEEE*, Jonathan DeCastro, Michael J. Roemer, and Kai Goebel, *Member, IEEE*

***Abstract*—This paper proposes a recursive receding horizon path planning algorithm for unmanned vehicles in nonuniform environments. In the proposed algorithm, the map is described by grids in which nodes are defined on corners of grids. The planning algorithm considers the map as four areas, namely, implementation, observation, explored, and unknown. The Implementation area is a subset of the Observation area, whereas the Explored area is the union of all the previous Observation areas. The path is planned with a receding horizon planning strategy to generate waypoints and in-between map updates. When a new map update occurs, the path is replanned within the current Observation area if necessary. If no such path exists, the search is extended to the Explored area. Paths can be planned by recursively searching available nodes inside the Explored area that can be connected to available nodes on the boundary of the Explored area. A robot platform is employed to conduct a series of experiments in a laboratory environment to verify the proposed path planning algorithm.**

***Index Terms*—Nonuniform environment, path planning, receding horizon planning (RHP), recursive searching, unmanned robot.**

I. Introduction

PATH and mission planning is a fundamental enabling technique for vehicle autonomy that has been widely used on vehicles from unmanned ground vehicles (UGVs), unmanned surface vessels, to unmanned air vehicles, micro air vehicles, unmanned underwater vehicles, and others [1]–[7]. It plays an important role in improving vehicle's availability, sustainability, survivability, and safety; reducing operating cost; guaranteeing mission success; and enabling tasks in harsh, restricted, and remote environments. Research in this area has drawn extensive interest from military and industrial communities [6], [8]–[18].

In the past few decades, many planning algorithms have been developed and utilized on various robots. Algorithms such as Dijkstra's algorithm and the A* algorithm [19] apply graph search to find the least cost path from a given starting point to a goal point. A heuristic function usually consisting of distance is used to determine the order in which searching algorithm investigates nodes. The heuristic function includes two parts: the distance from the starting node to the current location and the estimated distance from the current location to the goal. The advantage of these methods is that they are complete, i.e., they will always find a solution if one exists. There are many variants of A* algorithms [20]–[23]. The limitation is that they require a full map of area under exploration. This may not hold for applications that require field exploration itself and, therefore, cannot be used in unknown environments.

Another branch of searching algorithms is so-called "dynamic A*," or D* in short [24]–[27]. It is an incremental search algorithm, which makes assumption about the unknown area and finds the shortest path from the robot's location to its goal under this assumption. When the robot observes new areas, it adds the observed area to the map and replans a new path accordingly. This process is repeated until the robot reaches the goal. D* and its variants have been widely used for autonomous robots, including the Mars rovers Opportunity and Spirit [28]. Field D* is an interpolation-based algorithm [26]. Different from its predecessors in which nodes are defined as the centers of grids, field D* defines nodes on corners of grids. It uses linear interpolation to enable waypoints to be located on any position on edges of grids. This way, it can generate direct, low-cost, and smooth paths in nonuniform environments [26].

It is worth noting that planning problems are multiobjective in nature, i.e., the planner's cost function includes several individual objectives. The cost functions for optimality are typically terrain, mission time, energy consumption, or the combination of these factors. When these factors are summed up, the overall cost may not be linear and, in certain scenarios, may make the robot get stuck when facing complicated environments, as in [29]. In this reference, a scenario shows that Spirit was unable to autonomously navigate to a location on the other side of a cluster of rocks. To avoid such a livelock situation, it requires that the robot is able to navigate through a big obstacle or move out from blocked areas [30]–[32]. This requirement motivates our research into a recursive receding horizon planning (RHP) method.

In field D*, the interpolation is carried out in the grids next to the robot's current location. Since the robot can observe a

Manuscript received March 4, 2014; revised July 26, 2014 and September 7, 2014; accepted September 16, 2014. Date of publication October 17, 2014; date of current version April 8, 2015. This work was supported by NASA under Grant NNX09CB61C.

B. Zhang is with the Department of Electrical Engineering, University of South Carolina, Columbia, SC 29208 USA (e-mail: zhangbin@cec.sc.edu).

L. Tang is with Pratt & Whitney, East Hartford, CT 06118 USA (e-mail: mr.tangliang@gmail.com).

J. DeCastro is with the Department of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853 USA (e-mail: jdc1177@gmail.com).

M. J. Roemer is with Impact Technologies, LLC, A Sikorsky Innovations Company, Rochester, NY 14623 USA (e-mail: mike.roemer@impact-tek.com).

K. Goebel is with the NASA Ames Research Center, Moffett Field, CA 94035 USA (e-mail: kai.goebel@nasa.gov).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2014.2363632

number of grids, it is desirable to include all grids in the robot's observation range in planning. To this end, an RHP scheme is developed, which is able to use runtime information to plan a smooth path. In this RHP scheme, a series of waypoints are planned to connect the robot to the goal. It is important that only the first waypoint is executed when the robot moves. The map is updated as more grids are explored, and the path is replanned if necessary. The benefit is that this planning algorithm is able to plan a smooth path where waypoints can be located on any position on the edge of grids without linear interpolation, which may not work for a cost function that includes nonlinear factors.

In a complex environment, it is still possible that RHP scheme exhibits undesirable behaviors, such as cycling between visited waypoints or deadlocking (a situation where the robot can no longer make progress toward its goal). To address this, recursive searching is superimposed to RHP to make the planning algorithm more robust in workspaces.

If a confounding situation arises in the Observation area, the planner uses a recursive search to extend the planning to regions of the workspace that have been already explored. The objective of recursive searching is to find segmented paths in the Explored area to lead the robot to an available node defined on the boundary of the Explored area. With this extension, the algorithm is able to guide the robot in a complicated environment and avoid livelock or getting stuck.

The remaining parts of this paper are organized as follows. Section II presents a system overview of the proposed path planning algorithm. Section III introduces the RHP and how it is implemented. Recursive searching is discussed in Section IV. Two experiments on a robot are presented in Section V, which is followed by concluding remarks in Section VI.

II. Algorithm Overview

Before discussing the path planning algorithm, several assumptions are introduced.

Assumption 1: The robot is operated in a 2-D area, which is decomposed into a set of square grids.

Assumption 2: Each grid has an associated terrain value indicating the difficulty to traverse. The area consists of non-uniform grids.

Assumption 3: Onboard sensors and processing algorithms are able to calculate the grids' terrain values in the sensing scope, i.e., the grids' terrain values in the robot's sensing range are known.

Assumption 4: The robot travels a straight line in grids [24].

As illustrated in Fig. 1, the map is composed of square grids, and the proposed algorithm considers the map as four areas: Implementation area $\mathbb{I}$, which contains grids adjacent to the current location (inner box area), Observation area $\boldsymbol{O}$, which consists of grids that can be detected by the robot's onboard sensors (top box area), Unknown area $\mathbb{U}$, whose information remains unknown (area beyond boxes), and Explored area ($\mathbb{E}$), which is the union of all the previous Observation areas. Note that the Implementation area $\mathbb{I}$ is a subset of the Observation area $\boldsymbol{O}$ and is a subset of the Explored area $\mathbb{E}$, i.e., $\mathbb{I} \subset \boldsymbol{O} \subseteq \mathbb{E}$. The Unknown area is limited to the boundary of the map.

Fig. 2 illustrates the structure of the proposed planning strategy. At the robot's location C, sensor data are used to

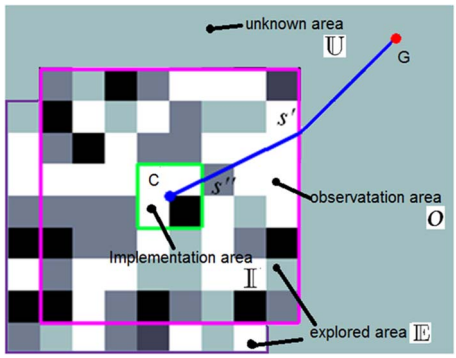


Fig. 1. Illustration of areas in the map.

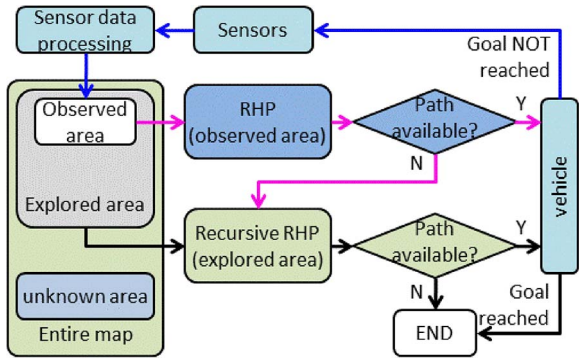


Fig. 2. Structure of the proposed mission planning algorithm.

construct an Observation area. For $\mathbb{U}$, an assumption of terrain is given in advance [26]. With this Observation area, RHP is implemented to plan a path. If there are available paths $\mathbb{A}P$, the optimal one will be selected, and the robot will move to the next waypoint following this optimal path. When the robot moves, sensor data update the Observation area. If no available path can be planned in the Observation area, searching is extended to the Explored area, and recursive RHP is activated. This process is repeated until the robot reaches the Goal G. If there is no path in the Explored area, the robot is in an unconnected workspace, it cannot reach goal, and the planning terminates.

As mentioned earlier, planning is a multiobjective problem with the cost function being a weighted sum of individual objectives. In our paper, the cost function is defined as [25]

$$\min_{s' \in \boldsymbol{O}} \left(\boldsymbol{c}(C, s') + \boldsymbol{g}(s') \right). \tag{1}$$

In this cost function, $\boldsymbol{c}(C, s')$ is the cost on path segment $C \to s'$, whereas $\boldsymbol{g}(s')$ is the cost of a path segment on $s' \to G$. The two path segments are illustrated in Fig. 1 as $C \to s' \to G$.

At the robot's location C, sensor data are processed to obtain the terrain values in newly observed grids. Based on the terrain values, the Observation area $\boldsymbol{O}$ is built, which contains nodes having high priority to the search path. An RHP strategy is introduced to search the optimal path in $\boldsymbol{O}$ based on the cost function (1) defined above. The path is defined by line segments connecting the robot to goal G via waypoints (as s' and s'' in Fig. 1) on corners or edges of grids. This cost is evaluated for all available nodes in $\boldsymbol{O}$, and the node that has the least cost defines the optimal path. When the optimal path $\mathbb{O}P$ is obtained, the robot moves from current location C to the adjacent waypoint

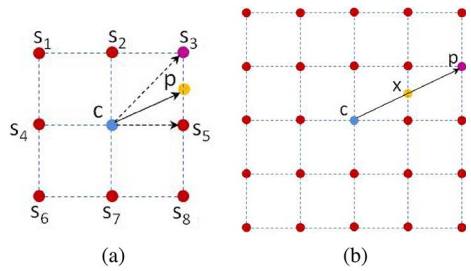


Fig. 3. (a) Nodes and interpolation in field D^*. (b) RHP (dots are nodes, C is the robot current location, P is the node indicating the optimal path, and X is a waypoint on the edge of a grid).

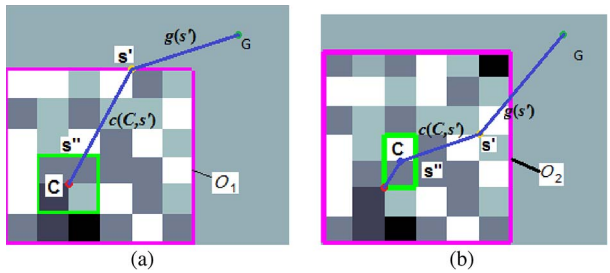


Fig. 4. Illustration of RHP. (a) Planning at step 1 from initial position C. (b) Planning at step 2 from new C [s'' in (a)].

s''. At this waypoint with map update, the path is replanned if necessary.

When no path can be planned in the Observation area $\boldsymbol{O}$, the planning is extended to the Explored area $\mathbb{E}$, which is the union of all the Observation areas at the waypoints that the robot has observed. The Explored area $\mathbb{E}$ in Fig. 1 is illustrated as the union of the Observation areas in two waypoints. Then, recursive RHP is proposed to find available paths in $\mathbb{E}$, which are able to guide the robot from its current location to an available node on the boundary of $\mathbb{E}$. This process is repeated until the goal is reached or it turns out that no available path can be found to reach goal.

III. RHP

One limitation of traditional search algorithms is that they constrain a robot's motion to a small set of possible headings, as dashed lines from C to nodes s_1 to s_8 shown in Fig. 3(a) [21], [24]. This often results in unnatural suboptimal paths with unnecessary turns and awkward directions. Therefore, it is desirable that the robot can have more flexibility in heading direction, such as shown in solid line in which waypoint p is located on the boundary of a grid. Field D^* achieves this by using a linear interpolation scheme [25], [26].

Field D^*, however, only considers neighboring grids to the robot and does not use full knowledge of the observed area. Since motion is planned over grids, it often requires the size of the grid to be small to maintain path optimality. In this case, the robot can observe an area beyond neighboring grids. If this full information about observation is utilized, the optimality of planning can be improved.

An RHP strategy borrowed from control area [33]–[35] is proposed here to overcome these limitations. The basic idea is that, at any time instant, a path is planned from the robot's location C to goal G by searching all available nodes in $\boldsymbol{O}$. The path costs for all available nodes in $\boldsymbol{O}$ are calculated, and the optimal node P is selected, as illustrated in Fig. 3(b). The path planned in $\boldsymbol{O}$ is $C \to P$. This path generates a waypoint X on the adjacent grid, and the path is described by $C \to X \to P$.

When the robot moves, however, it only executes the path segment in the neighboring grid. That is, the robot moves to X and ignores the remaining path segments $X \to P$. When the robot reaches X, $\boldsymbol{O}$ is updated, and the path is replanned if necessary.

The RHP is carried out iteratively in the following steps.

1) At waypoint p_i, a fixed horizon optimization problem over $[p_{i+1}, p_{i+L}]$ is solved.
2) Execute the first waypoint p_{i+1} generated by step 1.
3) Update the map at waypoint p_{i+1}.
4) Repeat steps 1–3 at waypoint p_{i+1} over $[p_{i+2}\ p_{i+L+1}]$.

The implementation of the RHP is further elaborated in Fig. 4. In this figure, the inner box is the Implementation area $\mathbb{I}$, the outer box is the Observation area $\boldsymbol{O}_1$ at step 1, and all other areas beyond that are the Unknown areas $\mathbb{U}$. Note that the robot C is located at the corner of four grids; the Implementation area in this case consists of four grids that share the node C. Each available node N in $\boldsymbol{O}_1$ indicates a possible path defined by two straight line segments from C to node N and to goal G. The costs of all possible paths are calculated from cost function (1). Then, the optimal path $\mathbb{O}P$ with the lowest cost is selected from all available paths $\mathbb{A}P$. In Fig. 4, the node that defines the optimal path $\mathbb{O}P$ is given by s'. That is, the cost of path $\boldsymbol{c}(C, s') + \boldsymbol{g}(s')$ is the least for all nodes in the Observation area $\boldsymbol{O}_1$. The optimal path $\mathbb{O}P$ is denoted by $C \to s' \to G$.

When the optimal path $\mathbb{O}P$, $C \to s' \to G$, is planned, a waypoint s'' is generated on the boundary of the Implementation area $\mathbb{I}$, which is located on the edge of a grid adjacent to the robot's location C. According to RHP, the robot moves to the waypoint s'' when the optimal path is implemented. After the robot reaches s'' and the map gets updated, the path is planned again if necessary. A replanning example is shown in Fig. 4(b).

In Fig. 4(b), the waypoint s'' in Fig. 4(a) becomes the robot location C. In addition, the Observation area $\boldsymbol{O}_2$ at step 2 is larger than $\boldsymbol{O}_1$ at step 1. This is a result of increased map exploration. Moreover, the Implementation areas (inner boxes) in these two figures are also different, which indicates that the Implementation areas change as the robot moves. In this figure, the robot C is now located on the edge of two grids; the implementation area is composed of the two grids that share the edge where C is located on. Then, a new optimal path with s' in $\boldsymbol{O}_2$ is planned. This path generates a new s'' on the boundary of the Implementation areas $\mathbb{I}$, and the robot will move to s''.

This example also shows that RHP is able to locate waypoints on the edge of a grid without interpolation, as shown by two s'' in Fig. 4(a) and (b). The proposed algorithm can generate smooth paths with flexible heading directions that reduce unnecessary turns and fit better the requirements of a robot. This is an advantage because the factors in the cost function could be nonlinear, and a linear interpolation may not work well.

Note that, with current map information at step k, the optimal path from an available node in $\boldsymbol{O}_k$ has lower cost than the optimal path from an available node in $\boldsymbol{O}_i$, $i = 1, \ldots, k-1$.

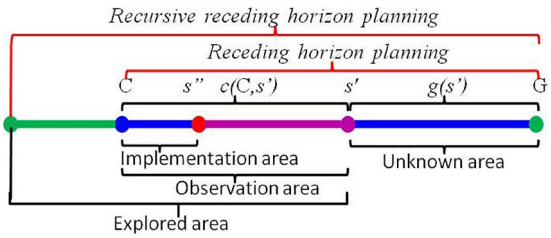


Fig. 5. Illustration of areas and planning horizons.

Therefore, the nodes in the current observation area $\boldsymbol{O}_k$ have a higher priority in searching.

The RHP is able to plan a smooth path. However, it is possible that the robot enters into livelock or deadlock. If all paths in $\boldsymbol{O}$ are blocked, the robot is deadlocked in this area. In this case, no path can be planned directly from C to an available node $N \in \boldsymbol{O}$ and to goal G. To move the robot forward, recursive searching is introduced in the next section.

IV. Recursive Searching

When the recursive searching is activated, it is superimposed on RHP, and the planning is extended to the Explored area. This is illustrated in Fig. 5, in which recursive planning is conducted in a larger area. The Explored area $\mathbb{E}$ is the union of all the Observation areas at previous steps, i.e., $\mathbb{E} = \cup \boldsymbol{O}_i$, with $i = 1, \ldots, k$ (refer to Fig. 1). The map is then composed of $\mathbb{E}$ and $\mathbb{U}$, i.e., $\mathbf{MAP} = \mathbb{E} \cup \mathbb{U}$. It is clear that $\mathbb{I}_k \subset \boldsymbol{O}_k \subseteq \mathbb{E}_k$, where subscript k is the indication of current step, and this relationship is also illustrated in Fig. 5. Note that the costs on path segments are also shown in this figure.

The objective is to recursively search available nodes in $\mathbb{E}$ to find out a path that guides the robot from its location to the goal. To this end, all nodes of $\mathbb{E}$ are divided into two categories, namely, available nodes $\mathbb{N}_{\boldsymbol{a}}$ and unavailable nodes $\mathbb{N}_{\boldsymbol{u}}$. Available nodes $\mathbb{N}_{\boldsymbol{a}}$ are those nodes that can be a waypoint, whereas unavailable nodes $\mathbb{N}_{\boldsymbol{u}}$ are those that cannot. Available nodes $\mathbb{N}_{\boldsymbol{a}}$ are further divided into those on the boundary of $\mathbb{E}$, denoted by $\mathbb{N}_{\boldsymbol{a},\boldsymbol{e}}$, and those inside $\mathbb{E}$, denoted by $\mathbb{N}_{\boldsymbol{a},\boldsymbol{i}}$, i.e., $\mathbb{N}_{\boldsymbol{a}} = \mathbb{N}_{\boldsymbol{a},\boldsymbol{i}} \cup \mathbb{N}_{\boldsymbol{a},\boldsymbol{e}}$.

An available node $N \in \mathbb{N}_{\boldsymbol{a},\boldsymbol{e}}$ could indicate available paths, denoted by $\mathbb{A}P$. That is, if there is an available node on the boundary of the Explored area $\mathbb{E}$, i.e., $\mathbb{N}_{\boldsymbol{a},\boldsymbol{e}} \neq \emptyset$, an available path can be planned. It is worth noting that whether $\mathbb{N}_{\boldsymbol{a},\boldsymbol{e}}$ is empty is known because $\mathbb{E}$ is the union of the Observation areas. That is, all the nodes $\mathbb{N}_{\boldsymbol{a},\boldsymbol{i}}$ and $\mathbb{N}_{\boldsymbol{a},\boldsymbol{e}}$ inside and on the boundary of $\mathbb{E}$ have been previously observed.

For an available node $N_e \in \mathbb{N}_{\boldsymbol{a},\boldsymbol{e}}$, a path is planned if a number of available nodes inside $\mathbb{E}(N_i \in \mathbb{N}_{\boldsymbol{a},\boldsymbol{i}})$ can be found to connect the robot from its current location C to N_e. The available path therefore is described by C, followed by a number of available node $N_{i,j}$, i, $j = 1, \ldots, n$, an available node N_e, and goal G, i.e., $C \to N_{i1} \to \cdots \to N_{in} \to N_e \to G$. The final path can be a complicated one, containing turns that avoid collisions and guide the robot in complicated environments.

The robot then moves following this path until map information is updated. If, on the other hand, $\mathbb{N}_{\boldsymbol{a},\boldsymbol{e}} = \emptyset$ or the robot is in an unconnected workspace that does not contain goal G, the search is terminated, and no path is planned.

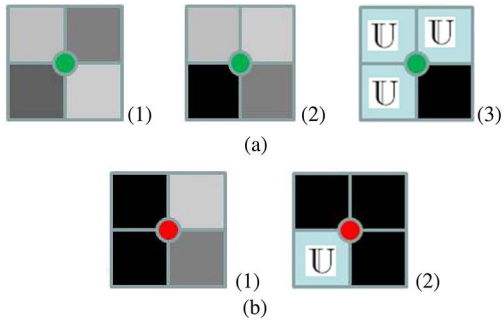


Fig. 6. Definition of available nodes and unavailable nodes. (a) Available nodes. (b) Unavailable nodes.

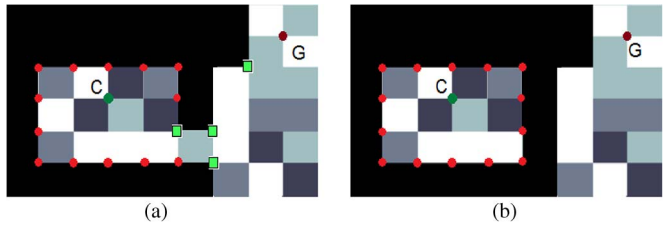


Fig. 7. Definition of connected and unconnected workspaces. (a) Connected. (b) Unconnected.

A. Available and Unavailable Nodes

As mentioned earlier, when recursive searching is invoked, all nodes in the Explored areas are divided into two sets: "available nodes" $\mathbb{N}_{\boldsymbol{a}}$ and "unavailable nodes" $\mathbb{N}_{\boldsymbol{u}}$.

Fig. 6 illustrates the definition of available and unavailable nodes. In this figure, grids with letter U are in the Unknown area with unknown terrains; grids with gray shades are different terrain difficulties; darker shades are more difficult than lighter ones, and black grids are obstacles. From Fig. 6, a simple rule to classify nodes can be obtained:

$$\text{Node is} \begin{cases} \text{available} & \text{if no. of obstacles in neighboring grid} < 2 \\ \text{unavailable} & \text{if no. of obstacles in neighboring grid} \geq 2. \end{cases}$$

B. Region Connectivity

By defining available and unavailable nodes, the algorithm is able to determine whether the workspace is connected to the goal or not. Fig. 7(a) shows an example of a connected workspace. It is clear that, with current knowledge, if unavailable nodes cannot form a closed area around the robot, it remains unobstructed. In this case, there are paths that are able to guide the robot from current location C to goal G via some available nodes (square-shaped nodes).

Fig. 7(b) shows an example of an unconnected workspace. In this case, the robot is located in a zone in which no available nodes can connect C to goal G. In other words, if G is unconnected from C, then no path can be planned.

C. Recursive Searching Strategy

Fig. 8(a) shows a scenario in which the robot cannot plan a path in the Observation area $\boldsymbol{O}$, formed by $[s_1, s_2, s_3, s_4]$. In this figure, G is the goal, and the robot location is indicated

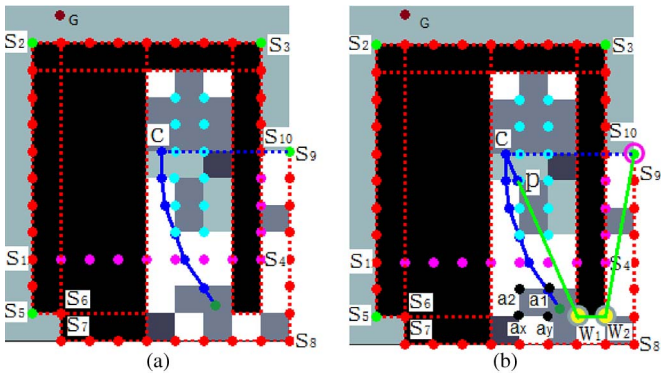


Fig. 8. Illustration of recursive searching. (a) Robot cannot find an available path in the observed area. (b) Robot finds an available path in the Explored area via recursive searching.

by waypoint C. The Explored area $\mathbb{E}$ is formed by $[s_2, s_3, s_{10}, s_9, s_8, s_7, s_6, s_5]$. The red dots are unavailable nodes on the boundary of the Explored area $\mathbb{E}$. The green dots, s_2, s_3, s_5, and s_9, are those available nodes $\mathbb{N}_{\boldsymbol{a},\boldsymbol{e}}$ on the boundary of $\mathbb{E}$; the cyan dots are the available nodes inside $\boldsymbol{O}$; the blue dots and the blue lines are the waypoints visited and the path segment traveled, respectively.

From Fig. 8(a), it is clear that there is no available path in $\boldsymbol{O}$, i.e., $\mathbb{A}P_{s' \in \boldsymbol{O}} = \emptyset$. Since there are available nodes on the boundary of $\mathbb{E}$, $\mathbb{N}_{\boldsymbol{a},\boldsymbol{e}} = [s_2, s_3, s_5, s_9] \neq \emptyset$. That means that the robot is able to avoid obstacle and moves to at least one of the nodes in $[s_2, s_3, s_5, s_9]$. To find a path in $\mathbb{E}$, all $N_i \in \mathbb{N}_{\boldsymbol{a},\boldsymbol{i}}$ are recursively investigated to build a path for all $N_e \in \mathbb{N}_{\boldsymbol{a},\boldsymbol{e}}$. The resulting path can be described as $C \to N_i \in \mathbb{N}_{\boldsymbol{a},\boldsymbol{i}} \to N_e \in \mathbb{N}_{\boldsymbol{a},\boldsymbol{e}} \to G$.

For a selected node $N_e \in \mathbb{N}_{\boldsymbol{a},\boldsymbol{e}}$, such as s_9 in Fig. 8(b), all available nodes $N_i \in \mathbb{N}_{\boldsymbol{a},\boldsymbol{i}}$ are sorted according to their cost to node $N_e(s_9)$, and the sorted nodes are denoted by $\mathbb{S}N_{\boldsymbol{a},\boldsymbol{i},1}$. At the beginning, an available path $\mathbb{A}P$ is empty, and we put $N_e(s_9)$ as the first node in $\mathbb{A}P$. Then, the first node N_1 in $\mathbb{S}N_{\boldsymbol{a},\boldsymbol{i},1}$ is checked to see if it can lead to $N_e(s_9)$. If so, N_1 is appended to $\mathbb{A}P$, and N_1 is removed from $\mathbb{S}N_{\boldsymbol{a},\boldsymbol{i},1}$. In Fig. 8(b), N_1 is found as W_2, and $\mathbb{A}P = [s_9, W_2]$. After $N_1(W_2)$ is appended to $\mathbb{A}P$ and $\mathbb{S}N_{\boldsymbol{a},\boldsymbol{i},1}$ is updated, the set of available nodes in $\mathbb{S}N_{\boldsymbol{a},\boldsymbol{i},1}$ is sorted again according to cost to $N_1(W_2)$ and denoted by $\mathbb{S}N_{\boldsymbol{a},\boldsymbol{i},2}$. Then, the first node $N_2 \in \mathbb{S}N_{\boldsymbol{a},\boldsymbol{i},2}$ is checked to see if it can be connected to s_9 in $\mathbb{A}P$. If so, the last node in $\mathbb{A}P$ is removed, and N_2 is appended in $\mathbb{A}P$. If not, append N_2 in $\mathbb{A}P$ without removing the last node. In Fig. 8(b), W_1 is checked, and since W_1 cannot be connected to s_9 directly, it is appended to $\mathbb{A}P$ without removing W_2. Now, $\mathbb{A}P = [s_9, W_2, W_1]$. Again, $\mathbb{S}N_{\boldsymbol{a},\boldsymbol{i},2}$ is updated and sorted again according to cost to W_1 to get $\mathbb{S}N_{\boldsymbol{a},\boldsymbol{i},3}$.

Suppose that, if the first node to be checked in $\mathbb{S}N_{\boldsymbol{a},\boldsymbol{i},3}$ is node a_1, it is found that a_1 cannot be connected to the second last node W_2 in $\mathbb{A}P$ directly. Therefore, a_1 is appended to $\mathbb{A}P$, and $\mathbb{A}P = [s_9, W_2, W_1, a_1]$. Then, suppose that node a_2 at the left-hand side of a_1 is checked. Since a_2 can be connected to the second last node W_1 in $\mathbb{A}P$ directly, $\mathbb{A}P$ is updated with a_1 replaced by a_2, and $\mathbb{A}P = [s_9, W_2, W_1, a_2]$. Now, all remaining available nodes in $\mathbb{N}_{\boldsymbol{a},\boldsymbol{i}}$ can be connected to W_1 directly. This checking and replacing are repeated until the last node becomes the robot's location C and $\mathbb{A}P = [s_9, W_2, W_1, C]$. That is, this path follows $C \to W_1 \to W_2 \to G$.

Similarly, other available paths can be planned as $[s_9, W_2, a_x, C]$ and $[s_9, W_2, a_y, C]$. In the process of searching each available path, its associated $\mathbb{S}N_{\boldsymbol{a},\boldsymbol{i}}$ is updated every time when a new node is checked. This newly checked node is either appended to $\mathbb{A}P$ or used to replace the last node in $\mathbb{A}P$. This is a recursive search process allowing available paths to be computed in complicated environments without becoming trapped by a big obstacle or livelock or deadlock.

In the preceding example, recursive searching is able to find all available paths, and by comparing the costs of these paths, the optimal path $C \to W_1 \to W_2 \to s_9$ is selected to guide the robot out of the blocked area. When this path is implemented, the robot drives to the first waypoint p. Once at waypoint p, if there is a map update, the recursive searching process is carried out again. Note that the previous planning result can be reused to reduce computation. In the example shown in Fig. 8(b), since the map does not have an update at p, the searching results in the previous step (when the robot is at C) are used at this step (when the robot is at p). That is, the previously planned path is followed until the map has an update. In addition, when map information is updated, only the updated part is replanned. Fig. 9 shows the pseudocode of the algorithm.

V. Experiments

A. Experimental Mobile Robot Platform

The experiments are conducted on a Pioneer 3-AT robotic UGV platform, as shown in Fig. 10. The platform hosts an onboard computer that supports a built-in vehicle controller, serial communications, sonar sensors, encoders, and other autonomous functions. The built-in controller drives the robot at a commanded speed up to 0.8 m/s and calculates its position.

The onboard computer communicates with the built-in controller through a serial port. The path planning algorithm can be run either onboard or on a client remote computer that sends waypoints to the robot via Internet or wireless communication. After a path is planned, the first waypoint is sent to the robot and executed. As the robot moves, its onboard sensors measure vehicle's position, heading angle, and velocity and send back to computer for planning.

A safety constraint is added at each waypoint received from planning. When the robot moves to a waypoint, if this waypoint is next to obstacle, it moves to a location $d = 0.5$ m away from obstacle to avoid collision. The value of d can be adjusted to accommodate the actual size of the robot and leave more safety margin distance.

B. Cost Function

As mentioned in Section II, planning is a multiobjective problem, which is often based on several competing criteria. Typical criteria include terrain, distance, energy consumption, and number of turns, among others. When the robot travels at variable velocities, the mission time can be included as well.

In this paper, the cost function is a weighted sum of path distance and terrain cost criteria. The distance criterion is meant

ALGORITHM: Recursive Receding Horizon Planner

Inputs: Nodes$\in O$; Nodes$\in \mathbb{E}$; Goal G, Robot location C
Outputs: Waypoints
00:WHILE $C \neq G$
01: IF G$\in O$
02: **SUBROUTINE1: Searching**
03: BREAK \\(Execute commands at *IMPLEMENTATION*)
04: ELSE \\ $G \notin O$
05: IF there is $\mathbb{A}P$ in O
06: CALCULATE costs for all $\mathbb{A}P$
07: SELECT the optimal path
08: BREAK \\(Execute commands at *IMPLEMENTATION*)
09: ELSE
10: FOR $N_a \in \mathbb{E}$
11: IF C is in an unconnected workspace \\ No path can be found
12: TERMINATE searching
13: ELSE
14: LIST available nodes on edge of $\mathbb{E}$ in set $\mathbb{N}_{a,e}$
15: LIST available nodes inside $\mathbb{E}$ in set $\mathbb{N}_{a,i}$
16: FOR $N_e \in \mathbb{N}_{a,e}$
17: SET $\mathbb{A}P\_N_e$ as an empty set
18: $\mathbb{A}P\_N_e = [N_e]$
19: **SUBROUTINE2: Recursive searching**
20: END
22: END
23: FOR all $\mathbb{A}P$, CALCULATE path costs
24: SELECT the optimal path $\mathbb{O}P$
25: BREAK \\(Execute commands at *IMPLEMENTATION*)
26: END
27: END
28: END
29: *IMPLEMENTATION*:
30: MOVE the robot to the next waypoint and UPDATE map
31: WHILE $\mathbb{E}$ does not have update, MOVE the robot to the next waypoint
32:END

SUBROUTINE1: Searching
Inputs: Nodes$\in O$; Robot Location C; Goal G
Outputs: Optimal path $\mathbb{O}P$
01: IF C is in a connected workspace
02: FOR all $N \in O$ and $N \neq C$ build path $C \rightarrow N \rightarrow G$
03: CALCULATE path cost
04: ELSE \\ No path can be planned
05: TERMINATE searching
06: END
07: SELECT the optimal path $\mathbb{O}P$

SUBROUTINE2: Recursive searching

Inputs: $\mathbb{A}P\_N_e$; Robot Location C; Goal G
Outputs: Nodes of available path $\mathbb{A}P\_N_e$
01: SORT $\mathbb{N}_{a,i}$ according to the cost to the last node in $\mathbb{A}P\_N_e$ and get $\mathbb{S}N_{a,i}$
02: APPEND the first node N_1 in $\mathbb{S}N_{a,i}$ to $\mathbb{A}P\_N_e$, $\mathbb{A}P\_N_e$=[$\mathbb{A}P\_N_e$, N_1]
03: REMOVE N_1 from $\mathbb{S}N_{a,i}$
04: SORT nodes in $\mathbb{S}N_{a,i}$ according to cost to N_1
05: FOR $N_j \in \mathbb{S}N_{a,i}$
06: FIND the least cost N_j connects to the second last node in $\mathbb{A}P\_N_e$
07: IF N_j can be connected to the 2nd last node in $\mathbb{A}P\_N_e$ without collision
08: REPLACE the last node in $\mathbb{A}P\_N_e$ with N_j
09: IF N_j=C, TERMINATE searching
10: ELSE
11: APPEND N_j to $\mathbb{A}P\_N_e$, $\mathbb{A}P\_N_e$=[$\mathbb{A}P\_N_e$ N_j]
12: REMOVE N_j from $\mathbb{S}N_{a,i}$
13: **SUBROUTINE2: Recursive searching**
14: END
15: END

Fig. 9. Pseudocode of recursive RHP.

to allow the robot to seek the shortest path, whereas the terrain criterion is meant to allow the robot to seek the path that minimizes wear on its components. The cost is as follows:

$$\min_{s'} J = w_{\text{Path}} \left(t_o(C, s') + t_u(s') \right) + w_{\text{Terrain}} \left(d_o(C, s') + d_u(s') \right)$$

where $t_o(C, s')$ and $d_o(C, s')$ are the terrain estimates and path distance for path segment $C \rightarrow s'$, respectively; whereas $t_u(s')$ and $d_u(s')$ are the corresponding terrain and distance measures on path segment $s' \rightarrow G$, respectively; w_{Path} and w_{Terrain} are weighting factors on each cost factor; and $w_{\text{Path}} + w_{\text{Terrain}} = 1$. In the following experiments, $w_{\text{Path}} = w_{\text{Terrain}} = 0.5$.

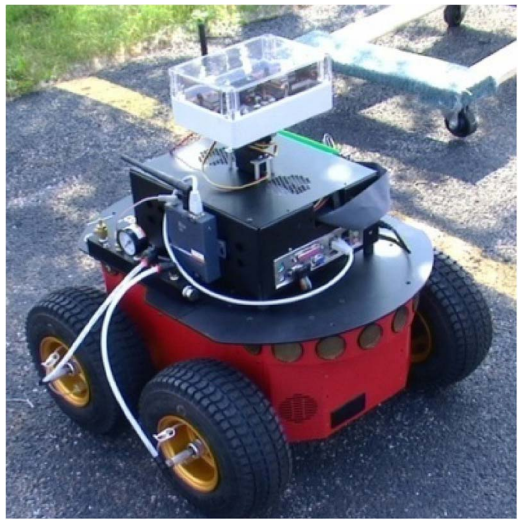


Fig. 10. Robot platform.

C. Experimental Scenarios

Experiments are conducted on a robot platform to verify the proposed recursive RHP algorithm. When a path is planned, the first waypoint is sent to the robot and is executed. As the robot moves, the terrain in the robot's sensing range becomes available. With the updated map information, the path is replanned if necessary.

Scenario 1: The map consists of 19 $\times$ 17 grids. Terrain values are given in five discrete values [0.1, 0.35, 0.55, 0.75, 1], in which 0.1 is the easiest terrain to traverse, and 1 represents an obstacle. For scenarios in this work, terrain values for grids are generated manually to simulate a map of desired complexity.

The robot is assumed to have a sensing range of four grids. That is, the robot can detect the terrain in the 8 $\times$ 8 square surrounding the robot. For the Unknown areas, the terrain value is assumed to be a uniform value of 0.5.

Fig. 11 shows a mission scenario that starts from A at [16.4, 1.2] and ends at G at [1, 16]. This figure is a snapshot when the robot reaches G and the map has been explored. In the figure, the square dots are the available nodes in the Observation area $\boldsymbol{O}$, the black areas define the obstacles, and the dots on the edge of obstacles or boundary of area are the unavailable nodes. The planned path is shown in the traversable area, and the dots on the path are the waypoints. The mark of available and unavailable nodes in other areas beyond the Observation area $\boldsymbol{O}$ has been removed.

The trajectory planned in Fig. 11 is as follows.

1) The robot moves toward goal G by path segment L_1 before obstacle $s_2 - s_3$ is detected.
2) At waypoint T_1, the robot finds that forward progress is blocked by obstacles $s_2 - s_3$. Because of this, it extends planning to the Explored area $\mathbb{E}$ and finds that available nodes at s_4 connect the robot from T_1.
3) With this new path, the robot moves backward, following path segments L_2 and L_3.

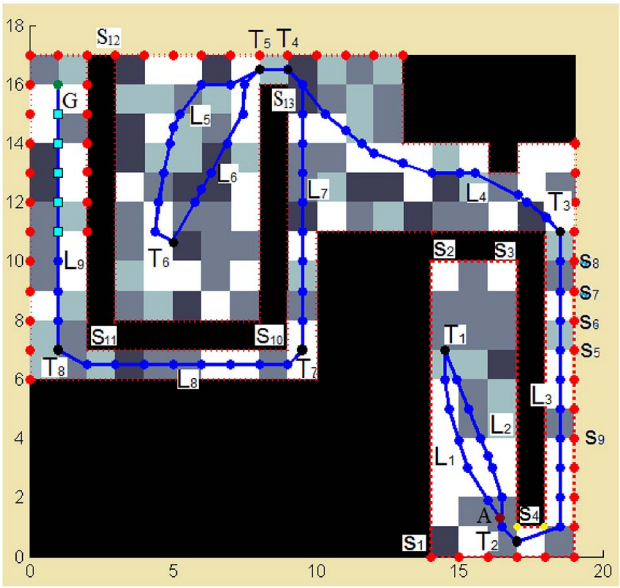


Fig. 11. Planning scenario 1.

4) At waypoint s_9, s_5 becomes an unavailable node, and at the same time, s_6 becomes an available node. Similarly, s_7 and s_8 are explored in sequence until T_3 is reached.
5) After T_3, the robot moves toward the upper left corner following path segment L_4 since it has the lowest cost.
6) At T_4, the robot moves left toward goal G, before it detected that this area is blocked.
7) Following L_5, it finds that it is a blocked area at T_6. Then, it moves back to T_5 following path segment L_6.
8) When it moves out of the blocked area, the robot is able to plan a path leading it to the goal G.

Note that, at T_1, the robot goes back to T_2 because it finds that $s_2 - s_3$ is blocked. When the robot visited T_4 and T_5 for the first time, it did not know $s_{13} - s_{10} - s_{11} - s_{12}$ is blocked. At T_6, it found obstacle $s_{13} - s_{10} - s_{11} - s_{12}$; thus, it went back to T_5 and T_4 to move out.

The experiment demonstrates the following.

1) The proposed planning algorithm is able to drive the robot out of blocked areas, as described by $[s_1, s_2, s_3, s_4]$ and $[s_{10}, s_{11}]$, to avoid livelock or deadlock.
2) The proposed algorithm is able to find an alternative path from a previously visited set of nodes (avoiding being blocked by obstacles). For instance, when the robot visited T_4 for the first time, it moved to left following L_5. When the robot moved out of the blocked area via L_6 to T_4 again, it moved via L_7 to s_{10}. It did not go back to following L_4 because nodes on the edge of $\mathbb{E}$ in that direction had been declared as unavailable.
3) The proposed algorithm is able to guide the robot within a narrow corridor, as shown by path segments L_3, L_7, and L_8. It shows that our algorithm works in complicated environments.
4) Waypoints can be located along grid boundaries, leading to smooth paths.

Scenario 2: Using satellite images of NASA Mars Yard in Fig. 12(a), a workspace 37 m $\times$ 20 m in size is scaled,

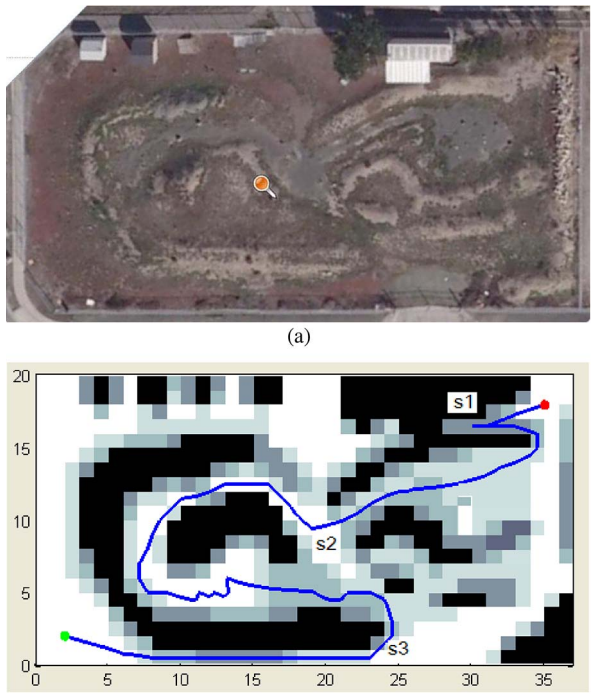


Fig. 12. Path planning in Mars Yard. (a) Satellite image of the NASA Mars Yard. (b) Path planning in scaled Mars Yard map.

as shown in Fig. 12(b). The map has eight different terrain levels indicated by different colors in which black grids indicate obstacles, as before.

The robot navigates the simulated Mars Yard terrain from a starting point [16], [34] (red dot) to the goal [2], [2] (green dot). The blue lines in the figure are the planned path. The result shows that, in a real environment, the proposed planning algorithm can plan the optimal path. The planner plans the path according to the defined cost function and can find the path with the least cost based on the available map information. At the same time, the planner can avoid livelock and deadlock. For instance, at the starting point, the robot moves into a narrow dead-end path at s_1. After the dead end is detected, the planner is able to guide the robot moving backward and continue the planning. It also shows that the robot can move around a big obstacle, as illustrated in the middle of this figure from s_2 to s_3. Note that, at s_2, the grids on the upper direction have lower terrain value and indicate an optimal path on that direction. The optimal path is therefore on that direction, and the robot moves upward.

It is worth noting that, if different weighting factors are used in the cost function, the optimal path could be different from the path shown in this figure.

Comparing with existing approaches, the proposed planning algorithm works well in complex environments. The A* algorithm requires that a full map is known, which is not suitable for mission with map exploration. The D* algorithm defines nodes in the center of grids, and therefore, the path segments between two nodes often involve two different terrain values. Another limitation of the D* algorithm is that it can only generate a path with eight orientations. Field D* defines nodes on the corner of grids, and this is convenient for cost calculation.

It is an interpolation-based algorithm that generates smooth path, in which waypoints can be located on any position on edges. However, it is not valid for planning with cost function that includes nonlinear factors. Note that all of these existing approaches only plan path in the range of the neighboring grids.

The proposed path planning algorithm has advantages over the existing methods in the following aspects: Same as the field D* algorithm, it defines nodes on corners of grids. Rather than using interpolation, the proposed approach plans path in a larger Observation area, which includes all grids in the sensing range of the robot's onboard sensors. This planning, as shown above, can generate a smooth path in which waypoints can be located on any points on the edge of grids. Since the proposed approach does not involve linear interpolation, it is valid for cost function with nonlinear factors [36]–[40].

The limitation of the proposed approach is that the recursive searching in the Explored area could be time consuming because it needs to search all available nodes in the Explored area to connect the robot to all available nodes on the edge of the Explored area. The benefits are that the proposed algorithm trades off computation time with capabilities to plan path in complex environment and avoids deadlock and livelock.

VI. Conclusion

In this paper, a path planning algorithm has been developed to find optimal path in nonuniform environments according to a defined cost. One advantage of the proposed algorithm is that it utilizes the full information of the robot's observation in an RHP framework. This enables the planner to plan a smooth path without interpolation operation. The second one is that recursive searching in the Explored area is able to avoid the robot getting stuck or livelock or deadlock. Some experimental results on simulated maps are presented to verify the proposed algorithm.

For the next step of this research, the planning algorithm will be optimized and compared quantitatively with other algorithms. The planning performance such as planning time should be also compared. Field tests will be conducted to test the algorithm in real-world applications. The robot dynamics, such as velocity and orientation control, and terrain classification and identification will be integrated with the planning algorithm developed in this paper to form a comprehensive robot design and navigation.

References

[1] Y. Lu, X. Huo, P. Arslan, and P. Tsiotras, "Incremental multi-scale search algorithm for dynamic path planning with low worst-case complexity," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 6, pp. 1556–1570, Dec. 2011.

[2] C. Petres *et al.*, "Path planning for autonomous underwater vehicles," *IEEE Trans. Robot.*, vol. 23, no. 2, pp. 331–341, Apr. 2007.

[3] A. Willms and S. Yang, "An efficient dynamic system for real-time robot-path planning," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 4, pp. 755–766, Aug. 2006.

[4] C.-C. Tsai, H.-C. Huang, and C.-K. Chan, "Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation," *IEEE Trans. Ind. Electron.*, vol. 58, no. 10, pp. 4813–4821, Oct. 2011.

[5] L. Capisani and A. Ferrara, "Trajectory planning and second-order sliding mode motion/interaction control for robot manipulators in unknown environments," *IEEE Trans. Ind. Electron.*, vol. 59, no. 8, pp. 3189–3198, Aug. 2012.

[6] Y. Chen, B. Wu, H. Huang, and C. Fan, "A real-time vision system for nighttime vehicle detection and traffic surveillance," *IEEE Trans. Ind. Electron.*, vol. 58, no. 5, pp. 2030–2044, May 2011.

[7] T. Sato, S. Sakaino, E. Ohashi, and K. Ohnishi, "Walking trajectory planning on stairs using virtual slope for biped robots," *IEEE Trans. Ind. Electron.*, vol. 58, no. 4, pp. 1385–1396, Apr. 2011.

[8] J. Tisdale, Z. Kim, and J. Hedrick, "Autonomous UAV path planning and estimation," *IEEE Robot. Autom. Mag.*, vol. 16, no. 2, pp. 35–42, Jun. 2009.

[9] N. Sudha and A. Mohan, "Hardware-efficient image-based robotic path planning in a dynamic environment and its FPGA implementation," *IEEE Trans. Ind. Electron.*, vol. 58, no. 5, pp. 1907–1920, May 2011.

[10] E. DiGiampaolo and F. Martinelli, "A passive UHF-RFID system for the localization of an indoor autonomous vehicle," *IEEE Trans. Ind. Electron.*, vol. 59, no. 10, pp. 3961–3970, Oct. 2012.

[11] V. Roberge, M. Tarbouchi, and G. Labonte, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *IEEE Trans. Ind. Electron.*, vol. 9, no. 1, pp. 132–141, Feb. 2013.

[12] K. Zhang, E. Collins, and D. Shi, "Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction," *ACM Trans. Auton. Adapt. Syst.*, vol. 7, no. 2, pp. 21:1–21:22, Jul. 2012.

[13] W. Wang and G. Xie, "Online high-precision probabilistic localization of robotic fish using visual and inertial cues," *IEEE Trans. Ind. Electron.*, vol. 62, no. 2, pp. 1113–1124, Feb. 2015.

[14] C. Moon and W. Chung, "Kinodynamic planner dual-tree RRT (DT-RRT) for two-wheeled mobile robots using the rapidly exploring random tree," *IEEE Trans. Ind. Electron.*, vol. 62, no. 2, pp. 1080–1090, Feb. 2015.

[15] H. Shin and B. Kim, "Energy-efficient gait planning and control for biped robots utilizing vertical body motion and allowable ZMP region," *IEEE Trans. Ind. Electron.*, vol. 62, no. 4, pp. 2277–2286, Apr. 2015.

[16] H. Kim and B. K. Kim, "Online minimum-energy trajectory planning and control on a straight-line path for three-wheeled omnidirectional mobile robots," *IEEE Trans. Ind. Electron.*, vol. 61, no. 9, pp. 4771–4779, Sep. 2014.

[17] J.-S. Hu, J.-J. Wang, and D. M. Ho, "Design of sensing system and anticipative behavior for human following of mobile robots," *IEEE Trans. Ind. Electron.*, vol. 61, no. 4, pp. 1916–1927, Apr. 2014.

[18] R. Luo and C. Lai, "Enriched indoor map construction based on multisensor fusion approach for intelligent service robot," *IEEE Trans. Ind. Electron.*, vol. 59, no. 8, pp. 3135–3145, Aug. 2012.

[19] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[20] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*," *Artif. Intell. J.*, vol. 155, no. 1/2, pp. 93–146, May 2004.

[21] X. Sun, S. Koenig, and W. Yeoh, "Generalized adaptive A*," in *Proc. Int. Joint Conf. Auton. Agents Multiagent Syst.*, 2008, pp. 469–476.

[22] X. Sun and S. Koenig, "The fringe-saving A* search algorithm—A feasibility study," in *Proc. Int. Joint Conf. Artif. Intell.*, 2007, pp. 2391–2397.

[23] R. E. Korf, "Depth-first iterative-deepening: An optimal admissible tree search," *Artif. Intell.*, vol. 27, no. 1, pp. 97–109, Sep. 1985.

[24] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1994, pp. 3310–3317.

[25] D. Ferguson and A. Stentz, "Field D*: An interpolation-based path planner and replanner," in *Proc. Int. Symp. Robot. Res.*, 2005, pp. 239–253.

[26] D. Ferguson and A. Stentz, "The field D* algorithm for improved path planning and replanning in uniform and non-uniform cost environments," Carnegie Mellon Univ., Pittsburgh, PA, USA, Rep. CMU-RI-TR-05-19, Jun. 2005.

[27] S. Koenig and M. Likhachev, "Fast re-planning for navigation in unknown terrain," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 354–363, Jun. 2005.

[28] S. Singh *et al.*, "Recent progress in local and global traversability for planetary rovers," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2000, pp. 1194–1200.

[29] J. Carsten, A. Rankin, D. Ferguson, and A. Stentz, "Global path planning on board the Mars exploration rovers," in *Proc. IEEE Aerosp. Conf.*, Big Sky, MT, USA, Mar. 2007, pp. 1–11.

[30] H. Qu, S. Yang, A. Willms, and Y. Zhang, "Real-time robot path planning based on a modified pulse-couple neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 11, pp. 1724–1739, Nov. 2009.
[31] G. Jan, K. Chang, and I. Parberry, "Optimal path planning for mobile robot navigation," *IEEE/ASME Trans. Mechatronics*, vol. 13, no. 4, pp. 451–460, Aug. 2008.
[32] J. Yang, Z. Qu, J. Wang, and K. Conrad, "Comparison of optimal solutions to real-time path planning for a mobile vehicle," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 40, no. 4, pp. 721–731, Jul. 2010.
[33] J. Bellingham, A. Richards, and J. How, "Receding horizon control of autonomous vehicles," in *Proc. Amer. Control Conf.*, Anchorage, AK, USA, 2002, pp. 3741–3746.
[34] T. Schouwenaars, B. DeMoor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *Proc. Eur. Control Conf.*, Porto, Portugal, Sep. 2001, pp. 2603–2608.
[35] T. Schouwenaars, J. How, and E. Feron, "Receding horizon path planning with implicit safety guarantees," in *Proc. Amer. Control Conf.*, Boston, MA, USA, 2004, pp. 5576–5581.
[36] L. Tang, B. Zhang, J. DeCastro, and E. Hettler, "An integrated health and contingency management case study on an autonomous ground robot," in *Proc. IEEE Int. Conf. Control Autom.*, 2011, pp. 584–589.
[37] B. Zhang, L. Tang, J. DeCastro, and K. Goebel, "Prognostics enhanced receding horizon mission planning for field unmanned vehicles," presented at the AIAA Guidance, Navigation, Control Conf., Portland, OR, USA, 2011, Paper AIAA 2011-6294.
[38] L. Tang, E. Hettler, B. Zhang, and J. DeCastro, "A testbed for real time autonomous vehicle PHM and contingency management applications," in *Proc. Annu. Conf. Prognost. Health Manage. Soc.*, 2011, pp. 1–11.
[39] B. Zhang, L. Tang, J. DeCastro, M. Roemer, and K. Goebel, "Autonomous vehicle battery state-of-charge prognostics enhanced mission planning," *Int. J. Prognost. Health Manage.*, vol. 5, no. 2, pp. 1–12, 2014.
[40] B. Zhang, L. Tang, and M. Roemer, "Probabilistic weather forecasting analysis for unmanned aerial vehicle path planning," *J. Guid., Control Dyn.*, vol. 37, no. 1, pp. 309–312, 2014.

Bin Zhang (S'03–M'06–SM'08) received the B.E. and M.E. degrees from Nanjing University of Science and Technology, Nanjing, China, in 1993 and 1999, respectively, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2007.

He is currently with the Department of Electrical Engineering, University of South Carolina, Columbia, SC, USA. Before that, he was with General Motors Research and Development, Detroit, MI, USA; Impact Technologies, Rochester, NY, USA; and Georgia Institute of Technology, Atlanta, GA, USA. His research interests are prognostics and health management, resilient systems, and intelligent systems and controls.

Liang Tang (M'04) received the Ph.D. degree from Shanghai Jiaotong University, Shanghai, China, in 1999.

He then worked with Georgia Institute of Technology, Atlanta, GA, USA, as a Postdoctoral Fellow prior to joining United Technologies Corporation (UTC). In 2004, he joined Impact Technologies, a Sikorsky Innovations company, where his work focused on research and development of innovative prognostics and health management (PHM) and automated fault accommodation solutions for a wide range of aerospace systems and components. He was the Principal Investigator on several government-funded Small Business Innovation Research/Small Business Technology Transfer/NASA Research Announcement programs. He is currently a Principal Engineer with Pratt & Whitney, East Hartford, CT, USA, with over 15 years of experience in the development of PHM systems. He has authored or coauthored over 50 technical papers.

Dr. Tang is a member of the PHM Society and serves as an Associate Editor of the *International Journal of Prognostics and Health Management*.

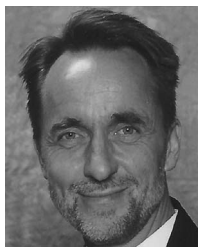
Jonathan DeCastro received the B.S. and M.S. degrees in mechanical engineering from Virginia Polytechnic Institute and State University, Blacksburg, VA, USA. He is currently working toward the Ph.D. degree in mechanical and aerospace engineering at Cornell University, Ithaca, NY, USA.

After receiving the B.S. and M.S. degrees and prior to Cornell University, he spent several years in the industry as an Aerospace Engineer with the National Aeronautics and Space Administration and Impact Technologies (now Sikorsky). His current research focuses on synthesis of provably correct controllers for complex robotics.

Michael J. Roemer received the B.Sc. degree in electrical engineering and the Ph.D. degree in mechanical engineering from the State University of New York, Buffalo, NY, USA.

He is currently a Technical Fellow with Sikorsky Aircraft, Stratford, CT, USA, with over 20 years of experience in the development of automated health monitoring, diagnostic, and prognostic systems. He was previously the Cofounder of Impact Technologies prior to its acquisition by Sikorsky/UTC. He is also currently an Adjunct Professor of mechanical engineering with Rochester Institute of Technology, Rochester, NY, USA. His experience includes a wide range of integrated vehicle health management system implementations to detect and predict system faults in real time and perform automated troubleshooting and maintenance planning. He has coauthored the book *Intelligent Fault Diagnosis and Prognosis for Engineering Systems* (Wiley, 2006) and has authored or coauthored over 100 technical papers.

Mr. Roemer is a Member of The American Society of Mechanical Engineers Controls and Diagnostics Committee. He has been a Cofounder and the Vice-President of the Prognostics and Health Management Society, the Chairman of the SAE HM-1 Integrated Vehicle Health Management Committee, and the Chairman of the Machinery Failure Prevention Technology Society.



Kai Goebel (M'08) received the Dipl.Ing. degree from the Technische Universitat München, Munich, Germany, in 1990 and the Ph.D. degree from the University of California, Berkeley, CA, USA, in 1996.

He is currently with the Ames Research Center, National Aeronautics and Space Administration, Moffett Field, CA, where he is the Area Lead for Discovery and Systems Health. He is also currently a Guest Professor with the University of Cincinnati, Cincinnati, OH, USA. He has been with General Electric's Corporate Research Center in upstate New York. He has also been an Adjunct Professor with Rensselear Polytechnic Institute (RPI), Troy, NY, USA. He has been on the dissertation committee of seven Ph.D. students at RPI; Syracuse University, Syracuse, NY; University of Cincinnati; Vanderbilt University, Nashville, TN, USA; Georgia Institute of Technology, Atlanta, GA, USA; and Stanford University, Stanford, CA. He has authored or coauthored over 250 technical papers and is the holder of 18 patents.