Hierarchical Incremental Path Planning and Situation-Dependent Optimized Dynamic Motion Planning Considering Accelerations

Xue-Cheng Lai, Shuzhi Sam Ge, *Fellow, IEEE*, and Abdullah Al Mamun, *Senior Member, IEEE*

***Abstract*—This paper studies a hierarchical approach for incrementally driving a nonholonomic mobile robot to its destination in unknown environments. The A* algorithm is modified to handle a map containing unknown information. Based on it, optimal (discrete) paths are incrementally generated with a periodically updated map. Next, accelerations in varying velocities are taken into account in predicting the robot pose and the robot trajectory resulting from a motion command. Obstacle constraints are transformed to suitable velocity limits so that the robot can move as fast as possible while avoiding collisions when needed. Then, to trace the discrete path, the system searches for a waypoint-directed optimized motion in a reduced 1-D translation or rotation velocity space. Various situations of navigation are dealt with by using different strategies rather than a single objective function. Extensive simulations and experiments verified the efficacy of the proposed approach.**

***Index Terms*—Collision avoidance, forward kinematics, incremental path planning, optimization, robot dynamics, velocity space.**

I. Introduction

IT IS ESSENTIAL for mobile robots to utilize partial knowledge of the environment in order to accomplish tasks, such as autonomous exploration and path planning, as *a priori* model maps are not always available. To find a path between two points in a known environment, graph search algorithms, such as Dijkstra's, the A* algorithm [1], Bellman–Ford's algorithms [2], the wavefront algorithm [3], or visibility graph approaches [4], can be used. The A* search algorithm is an effective heuristic improvement over Dijkstra's algorithm and yields a better average performance when one only needs the optimum path between two grid cells in a directed graph with nonnegative weights. As a dynamic version of the A* algorithm, the D* algorithm [5], [6] plans optimal traverses by incrementally repairing paths to the robot's state as new information is discovered. In the field of path planning, there are a number of search-based methods [7]–[9] which exhaustively explore the system's configuration space (C-space) to capture the global connectivity of a robot's free space.

Manuscript received April 8, 2007. This paper was recommended by Associate Editor W. Dixon.

The authors are with the Electrical and Computer Engineering Department, National University of Singapore, Singapore 117576 (e-mail: laixuecheng@alumni.nus.edu.sg; elegesz@nus.edu.sg; eleaam@nus.edu.sg).

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/TSMCB.2007.906577

If a series of subgoals/waypoints connected with their adjacent ones is provided (e.g., by a high-level path planner), the remaining problem is to generate motion commands for the robot to sequentially trace them (and perform collision avoidance at the same time). To accomplish this task, a number of approaches use local sensory information in a purely reactive fashion for robustness to uncertainties and incorporate collision avoidance in the meantime. One category of such approaches is the directional approach, and the other is the velocity space (translational–rotational velocity space) approach. The directional approach computes a direction for the robot to head in, either in Cartesian or configuration space. For instance, potential field methods [10]–[12] use the vector summation of an attractive force representing the goal and a number of repulsive forces that are associated with obstacles to compute a desired robot direction. By computing a 1-D polar histogram from detected obstacles, the vector field histogram approach [13] improves over traditional potential field methods, in the sense that it can achieve a smoother navigation and has more chances to successfully find paths through narrow openings. The nearness diagram algorithm [14], [15] and instant goal approach [16], [17] also belong to this category of approaches. The path deformation method [18] iteratively deforms the current path using potential fields over the C-space in order to achieve collision-free smooth motions.

Although simple and efficient in producing a direction command for a collision-free motion, the direction approach is inadequate to take the robot dynamics into account, which may result in slow or jerky movements. The velocity space approach, on the other hand, incorporates vehicle dynamics and decides both rotation and translation velocities at the same time. For example, dynamic window approaches [19]–[21] search the velocity space for a heading close to the goal direction, without hitting obstacles within several command intervals. The curvature-velocity method [22] searches the velocity space for a point that satisfies the velocity and acceleration constraints and maximizes an objective function. Although it produces reliable, smooth, and speedy navigation in office environments, it has shortcomings such as passing some collision-free paths with a high turning angle. In addition, to find the best motion command, it is insufficient to simply define a single objective function for the constrained optimization problem [23].

It is believed that gradual learning about the surroundings, with the capability of simultaneously planning a path, often results in better plans. On the other hand, each of deliberative planning and reactive control compliments the other and

compensates for the other's deficiencies. This paper studies a hierarchical approach that is used to incrementally plan paths while the robot acquires more knowledge about the surroundings. The A* algorithm is modified to suit the needs of this research (i.e., to handle a map with unknown information), based on which a high-level planner incrementally provides a series of optimal paths robustly. Next, modeling of forward kinematics is presented for differential steering mobile robots, and dynamic constraints and accelerations are considered to track a motion command. Then, a situation-dependent approach is proposed for optimized dynamic motion planning which searches for dynamic admissible and collision-free motions in a reduced velocity space. Finally, experimental results are included, with extensive discussions. Compared with traditional path planning approaches, the main contributions of this paper include the following:

1) Accelerations in both translation and rotation are considered in order to better predict the robot pose and the robot trajectory resulting from a certain motion command. The obstacle constraints are transformed to suitable velocity limits for the robot to move as fast as possible while avoiding collision when needed.
2) An optimized motion is obtained by searching in a reduced 1-D translation or rotation velocity space with situation-dependent object functions rather than a single objective function to cater to different situations of navigation.
3) A hierarchical two-stage approach robustly plans optimal paths incrementally by using partial map information. By ensuring connectivity and a decrease in its distance/orientation angle to a waypoint, the robot is thus able to converge to it and, eventually, to the goal.

II. Incremental Dynamic Path Planning With Partial Map

This section presents an incremental search algorithm for replanning robot paths with a periodically updated map.

A. *Modified* A* *Search for Partially Known Environments*

1) A Star Search: The A* search employs a "heuristic estimate" that ranks each node with an estimate of the best route that goes through that node. It visits the nodes in order of this heuristic estimate. In each step of a search, a weight function f is used to order the queue. Two cost functions are defined as follows first:

1) $g(s)$—the actual cost of going from the initial "state" (a grid cell of the map) to the current state s;
2) $h(s)$—a heuristic estimate of the cost of going from the current state s to the solution (goal) state.

The potential of reaching the goal via state s is then evaluated by the estimated total cost given by

$$f(s) = g(s) + h(s) \tag{1}$$

where $g(s)$ can be taken as the length of the path linking the geometric centers of the traversed regions (cells), and $h(s)$ as the straight-line distance from state s to the goal position.

Open and closed lists are maintained to keep track of the progress of the A* search. The open list is initialized, with the start state as the first node, and the closed list is initialized to be empty. The following steps are repeated until the open list is empty.

Step 1) The node with the lowest f-value in the open list is removed and labeled as the current node. If the removed node is the goal node, the A* search will exit and return a complete chain of nodes for the path to the goal.
Step 2) A neighboring node s (in the eight directions) of the current node is added to the open list at a place according to its f-value if it is not in the open list or it is matched to a node inside the open list which has a higher f-value.
Step 3) If node s is matched to a node inside the closed list (which consists of nodes that have been checked) which has an equal or lower f-value, no further processing will be done on node s. Otherwise (i.e., if node s has a lower cost than the stored node), the stored node is replaced by node s.

A discrete search (made by the A* algorithm or other planners) may result in jagged across grids. Straight paths typically look more plausible than jagged paths, particularly through open spaces. Therefore, after the discrete path sequence is obtained, redundant points are removed, and only a set of waypoints that are connected by straight line segments are left.

2) Dynamic Searching in Partially Known Environments: The robot may have only partial information about the environment before it begins its A* search, as *a priori* model maps are rarely available. D* algorithms are capable of planning paths in partially known and changing environments. However, D* search is much harder to implement because of the added dimension characteristic to the problem of attempting to navigate with real-time replanning in partially known environments.

Being similar to the work in [24], this paper uses the A* algorithm for search, considering its robustness in finding an optimal path. To handle a map containing unknown information, the condition of finding a path is accordingly defined as follows [25]: if the removed node is the goal node, the successful path is reconstructed, and the algorithm ends; if the removed node is unknown, a possible path is found, and the algorithm ends. In the latter case, no sufficient knowledge is available to ensure that there exists a feasible path planned between the second last node and the goal node.

B. *Obstacle Enlarging and Addition of Obstacle Cost*

In this paper, occupancy grid map is used since it provides more detailed information about the environment (compared with geometric or topological map) and can easily be updated when there is a sensor input due to the probabilistic nature of grids. The planner first creates a configuration space from the grid map, considering the robot dimensions: The grid map is processed by enlarging each occupied grid to its neighboring

free grids with a value bigger than the size of the robot (or r_{rob}, which is the radius of a circular robot):

$$r_{\mathrm{enlarge}} = c_{\mathrm{enlarge}} \cdot r_{\mathrm{rob}} \tag{2}$$

where c_{enlarge} (larger than one and typically in the range [1.2, 1.5]) is a coefficient about how conservative the safety margin should be to ensure the safety of the robot.

If the A^* algorithm is applied onto such a configuration space, it can be found that some parts of the obtained optimal path may be located very close to the obstacles. "Obstacle cost" is thus assigned to each free cell based on d_{nearest}, which is its distance from its nearest obstacle cell(s) in the C-space constructed. Obstacle cost is defined as follows, such that it will be infinitely big if the distance is within r_{enlarge}, zero if the distance reaches the obstacle-influencing distance, $r_{\max}(> r_{\mathrm{enlarge}})$, and, otherwise, proportional to the inverse of the distance:

$$c_{\mathrm{obs}} = \begin{cases} \infty, & d_{\mathrm{nearest}} < r_{\mathrm{enlarge}} \\ \alpha_{\mathrm{obs}}/d_{\mathrm{nearest}}, & r_{\mathrm{enlarge}} \le d_{\mathrm{nearest}} \le r_{\max} \\ 0, & d_{\mathrm{nearest}} > r_{\max} \end{cases} \tag{3}$$

where α_{obs} is a predefined coefficient to determine the significance of obstacle cost.

When the search was subsequently carried out, both occupied and unknown cells are treated as occupied ones, and each free grid node in the constructed C-space is assigned with two values: a g-value and an h-value, with an addition of obstacle cost to $g(s)$, i.e.,

$$g(s) = g_a(s) + c_{\mathrm{obs}}(s) \tag{4}$$

where $g_a(s)$ is the cost of traveling a cell length or $\sqrt{2}$-fold of a cell length, depending on whether node s is directly or diagonally neighboring to the current node that is just removed from the open list as the root node.

C. Incremental Planning Algorithm

Fig. 1 shows the flowchart of the proposed incremental planning algorithm. Initially, it sets up basic settings such as the robot's initial pose, the goal, and the information (size and resolution) of the grid map. After a partial map is created with laser and odometry sensory data, the planner is able to plan/replan a path with the following procedure.

Step 1) Search and Planning: Based on the C-space and the obstacle cost information, an optimal path, in the form of a series of nodes, is planned from the current state to the goal using the modified A^* search algorithm. As only a partial map is available, such a path can be either a path to the goal node, which is denoted as "Path_To_Goal," or a possible path to it (i.e., the path from the second last node to the goal node contains unknown areas), which is denoted as "Possible_Path."

Step 2) Moving and Sensing: The robot moves toward its target node while collecting information about its surroundings. When the target node is reached (i.e., within a certain distance from the robot), the robot will decide its next action as follows.

a) If the target node is the goal, the path planning task is accomplished.
b) If the target node is the second last node in the series of nodes and the search result is "Possible_Path," the procedure will go to Step 3).
c) Otherwise, the robot will continue this step to approach the remaining nodes in the plan.

Step 3) Relocalization for Next Round of Planning: Before the robot is able to go to Step 1) to replan a new path, the robot sends requests for the best estimate of the relative position between it and the goal, as well as the updated map information of the world.

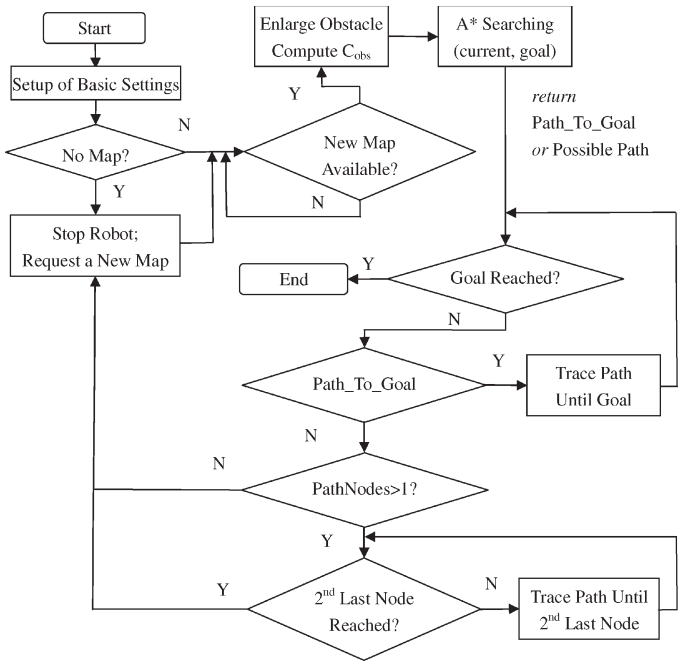


Fig. 1. Flowchart of an incremental search and planning algorithm.

Remark 1: Because it takes a while to update/receive the map supplied for search and to carry out such a graph search, we do not want to update this map while the robot is moving, or else, it may run into an obstacle or deviate from the planned path. The robot is therefore commanded to stop, and the system subsequently replans a new path (a series of waypoints) to the goal. To the best of our knowledge, grid-map-based deliberate planning approaches seldom choose to continue driving the robot during that period unless there is a global path to guide the robot to continue moving forward. Nevertheless, the system's performance (mainly the average robot speed) could be enhanced if this restriction can be relaxed.

III. Forward Kinematics and Predicted Robot Trajectory Under Robot Dynamic Constraints

This section presents modeling of forward kinematics of differential steering mobile robots, admissible robot motion subject to various dynamic/actuator constraints, and resulting

TABLE I
NOMENCLATURE

$\overrightarrow{AB}$	straight-line segment starting from point/configuration A to point/configuration B;
$\Upsilon_A^B(r)$	rectangular ray expanded from line segment $\overrightarrow{AB}$ by a radius r;
$\mathcal{C}(A,r)$	circular disk centered at point A and with radius r;
(x,y)	coordinates with respect to (w.r.t.) the global frame;
θ	angle (of the main axis of the robot) w.r.t. the global frame;
v	(translation) velocity at the reference point of the robot;
ω	angular velocity, or called rotation velocity;
a	longitudinal acceleration of the robot;
ε	angular acceleration of the robot;
r	turning radius (distance from the turning center) of the robot;
t	current time instant;
Δt	nominal time interval between two motion commands;
$s(t)$	arc length of robot trajectory from time 0 to time t.

robot trajectories that conform to forward kinematics of a robot. Table I lists some notations to be used.

A. Forward Kinematics of Differential Steering Robots

The body frame of a differential steering robot $o_b x_b y_b$ is established such that the origin is located at the midpoint of rear-axle W and that the x_b axis coincides with the main axis of the robot. The configuration of the robot is defined by $\mathbf{q} = [x \quad y \quad \theta]^{\mathrm{T}}$. The reference point (RP) that is used to trace a path is chosen at the midpoint of the rear-axle, and the robot's translation velocity v is thus defined at this point. We have

$$\dot{x} = v\cos\theta \quad \dot{y} = v\sin\theta \quad \dot{\theta} = \omega. \tag{5}$$

Forward kinematics is to predict the behavior of a mechanical system based on the inputs to that system, i.e., in this paper, how to find out the robot trajectory if the speeds are known. The robot velocities at any time are related to the velocities of the left and right wheels (v_L and v_R) as follows:

$$v(t) = \frac{v_R(t) + v_L(t)}{2} \qquad \omega(t) = \frac{v_R(t) - v_L(t)}{b} \tag{6}$$

where b denotes the wheelbase, i.e., the distance between the two rear wheels.

Let (x_0, y_0, θ_0) denote the initial pose of the robot. When the wheels turn at fixed velocities, the differential equations in (5) are solvable as follows:

$$\begin{cases} x(t) = x_0 + \frac{v_0}{\omega_0}\left(\sin(\omega_0 t + \theta_0) - \sin\theta_0\right) \\ y(t) = y_0 - \frac{v_0}{\omega_0}\left(\cos(\omega_0 t + \theta_0) - \cos\theta_0\right) \\ \theta(t) = \theta_0 + \omega_0 t \end{cases} \tag{7}$$

which implies that the trajectory that the robot follows is a circular path with its radius $r = v/\omega$.

Many research works (e.g., [15] and [19]) thus assume that the robot velocities remain constant in each control period, such that the robot path will be a straight line or a circular path. However, the velocities are normally time-varying in a control period as robots are frequently accelerated or decelerated during navigation. Thus, the differential equations in (5) must be evaluated using numerical methods. At a relatively short interval of Δt, we may take that the wheels undergo a fixed rate of acceleration or deceleration, i.e.,

$$v_L(t) = v_{L0} + a_L t \quad v_R(t) = v_{R0} + a_R t \tag{8}$$

where v_{L0} and v_{R0} and a_L and a_R are the initial velocities and accelerations of the left and right rear wheels, respectively.

Given (8), (6) can be converted to the following:

$$\begin{cases} v(t) = \frac{a_R + a_L}{2} t + \frac{v_{R0} + v_{L0}}{2} = at + v_0 \\ \omega(t) = \frac{a_R - a_L}{b} t + \frac{v_{R0} - v_{L0}}{b} = \varepsilon t + \omega_0 \end{cases} \tag{9}$$

where v_0, ω_0, a, and ε are the initial values of the translation and rotation velocities, the longitudinal acceleration, and the angular acceleration of the robot, respectively.

With (5) and (9) and the initial condition of integral as (x_0, y_0, θ_0), the solution for the robot pose is

$$\begin{cases} x(t) = \int_0^t (at + v_0)\cos\left(\frac{1}{2}\varepsilon t^2 + \omega_0 t + \theta_0\right) dt + x_0 \\ y(t) = \int_0^t (at + v_0)\sin\left(\frac{1}{2}\varepsilon t^2 + \omega_0 t + \theta_0\right) dt + y_0 \\ \theta(t) = \frac{1}{2}\varepsilon t^2 + \omega_0 t + \theta_0 \end{cases} \tag{10}$$

where the position can be obtained only through a numerical integration method such as Simpson's rule.

B. Admissible Motions Satisfying Dynamic Constraints

Due to the actuators' limits, there exist maximum limits on the robot velocities and robot accelerations. Let $a_{\mathrm{s,max}}$ denote the maximum acceleration at which the robot can comfortably and smoothly be accelerated, and let $a_{\max}$ denote the maximum permissible deceleration at which the robot can be slowed down and stopped without causing skidding to occur. Let $v_{\max}, \omega_{\max}$, and $\varepsilon_{\max}$ denote the maximum translation velocity, maximum rotational velocity, and maximum rotational acceleration of the robot, respectively. The desired translational and rotational velocities (v_1, ω_1) will be confined by the following rectangular region (also shown in Fig. 2) in order for the robot to achieve a smooth and (actuator) feasible motion, or simply called "admissible" motion:

$$\mathbf{V}_a = \{(v_1, \omega_1) | \underline{v} \le v_1 \le \overline{v}, \quad \underline{\omega} \le \omega_1 \le \overline{\omega}\} \tag{11}$$

where $\underline{v} = \max(0, v_0 - a_{\max}\Delta t)$ (if no reverse movement is allowed for the robot), $\underline{\omega} = \max(-\omega_{\max}, \omega_0 - \varepsilon_{\max}\Delta t)$, $\overline{v} = \min(v_{\max}, v_0 + a_{\mathrm{s,max}}\Delta t)$, and $\overline{\omega} = \min(\omega_0 + \varepsilon_{\max}\Delta t, \omega_{\max})$.

C. Trajectories Generated by Admissible Motion Commands

What would be the robot trajectory like within the duration of Δt if it is commanded to a certain motion, under the constraints (11)? Let us first consider the initial and final turning radii (the distance from the rotation center to the RP).

If the desired target velocities are known as $v_1 \in [\underline{v}, \overline{v}]$ and $\omega_1 \in [\underline{\omega}, \overline{\omega}]$, the turning radius at a given time instant $t \in [0, \Delta t]$

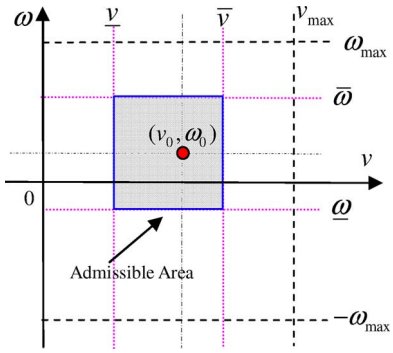


Fig. 2. Region of admissible translation and rotation velocities.

can be evaluated using the following formula:

$$r(t) = \frac{v_0 + (v_1 - v_0)t/\Delta t}{\omega_0 + (\omega_1 - \omega_0)t/\Delta t}. \tag{12}$$

Apparently, the turning radius at the initial and final robot configurations can be given by $r_0 = v_0/\omega_0$ and $r_1 = v_1/\omega_1$. Unfortunately, r_1 is often unknown beforehand as, normally, the target velocities are to be decided.

Without losing generality, suppose that $\omega_0 > 0$ and $\overline{\omega} > 0$ (like the one that is shown in Fig. 2). Table II lists the special cases of possible final turning radii if (fixed) acceleration/deceleration is imposed on v and ω. Next, let us explore some useful properties about robot trajectories.

Property 1: The value of $r(t)$ belongs to a certain range that is determined by ${}^1r(t)$, ${}^3r(t)$, and ${}^4r(t)$.[1] Specifically

$$\begin{cases} r(t) \in [{}^1r(t), {}^3r(t)], & \text{if } \underline{\omega} > 0 \\ r(t) \in [{}^1r(t), \infty] \bigcup [-\infty, {}^4r(t)], & \text{if } \underline{\omega} \le 0. \end{cases} \tag{13}$$

Proof: The aforementioned equation can be proved by evaluating (12). ■

Property 2: Arc length $s(t)$ at any time from the robot's initial position is between ${}^1s(t)$ and ${}^3s(t)$, which are arc lengths of trajectories ${}^1r(t)$ and ${}^3r(t)$, respectively.

Proof: Arc length $s(t)$ can be expressed as

$$s(t) = \int_{\theta_0}^{\theta_t} r d\theta = \int_0^t r\frac{d\theta}{dt}dt = \int_0^t r\omega(t)dt = \int_0^t v(t)dt$$
$$= \frac{v_1 - v_0}{2\Delta t}t^2 + v_0 t \tag{14}$$

from which we know that the arc length is proportional to the final translation velocity (when other variables in the equation are fixed), and thus, it can be proved that $s(t) \in [{}^1s(t), {}^3s(t)]$. ■

To more clearly illustrate this property, Fig. 3 shows the trajectories of a robot[2] with $v_0 = 0.3$ and $\Delta t = 0.5$. The left and right diagrams show the robot trajectories generated during the last 1/60 s for $\omega_0 = 1.4$ (and thus $\underline{\omega} \ge 0$) and for $\omega_0 = 0.2$ (and thus $\underline{\omega} < 0$), respectively. It is shown that the arc length of an admissible trajectory is directly related to the value of v_1: the larger v_1 is, the longer is the trajectory. It can be known in a similar way that ω_1 determines the angular displacement of the robot (for the same v_1).

[1]Prefix superscript of $r(t)$ or $s(t)$ denotes the numbering of turning radius or arc length.

[2]The robot's dynamic constraints are $v_{\max} = 0.8$ m/s, $\omega_{\max} = 2.0$ rad/s, $a_{\max} = 0.8$ m/s^2, $a_{s,\max} = 0.5$ m/s^2, and $\varepsilon_{\max} = 1.6$ rad/s^2.

TABLE II
SPECIAL CASES OF POSSIBLE ROBOT TRAJECTORIES

Final Turning Radius	Trajectory	Remarks
${}^1r_1 = \underline{v}/\overline{\omega}$	$traj\ 1$	max deceleration on v and max acceleration on ω applied.
${}^2r_1 = v_0/\omega_0$	$traj\ 2$	no acceleration.
${}^3r_1 = \overline{v}/\underline{\omega}$	$traj\ 3$	max acceleration on v and max deceleration on ω applied
${}^4r_1 = \underline{v}/\underline{\omega}$	$traj\ 4$	max deceleration on v and max deceleration on ω applied.
${}^5r_1 = \overline{v}/\overline{\omega}$	$traj\ 5$	max acceleration on v and max acceleration on ω applied.

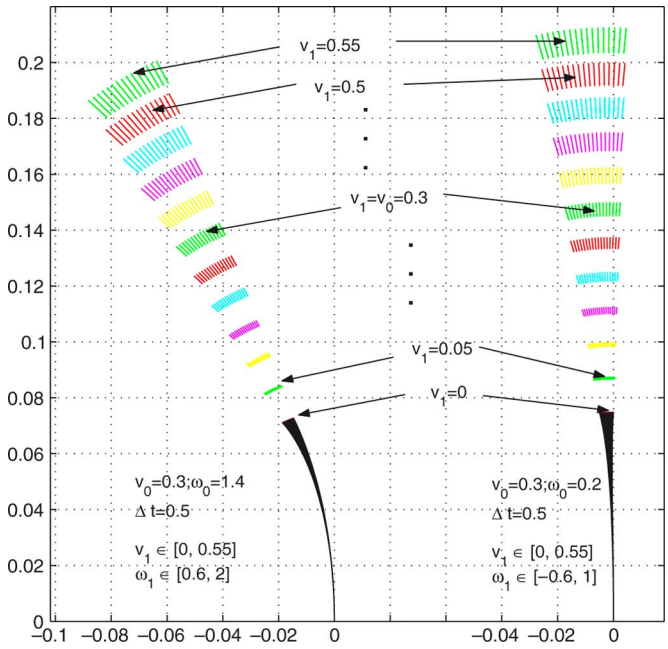


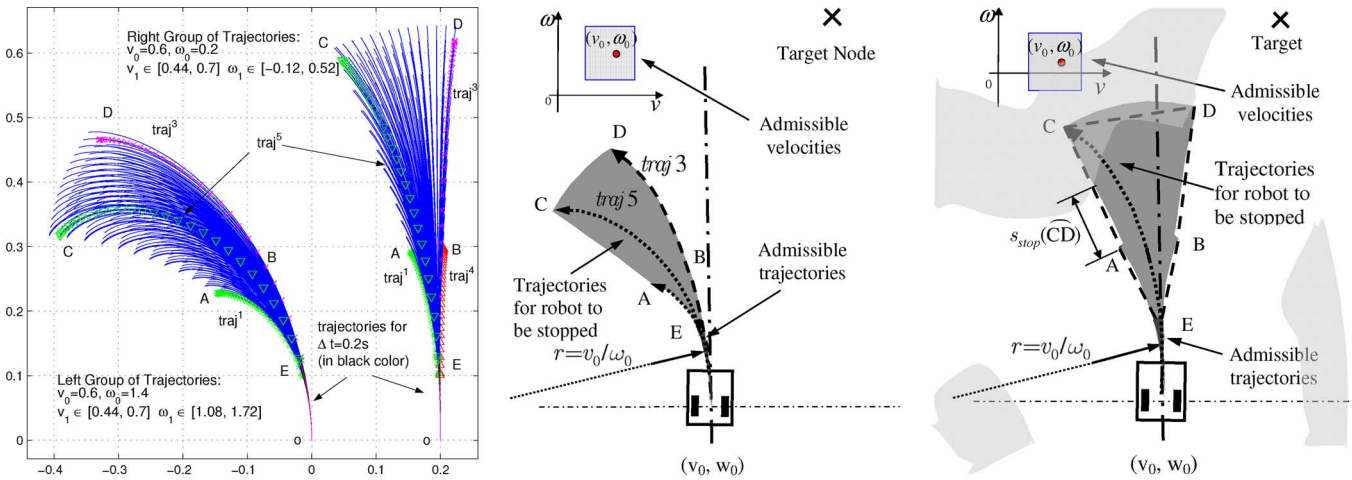
Fig. 3. Trajectories that are generated during the last 1/60 s and are distinguished by different colors according to the value of ending translation velocities. The entire trajectories for $v_1 = 0$ are plotted in black color.

IV. SITUATION-DEPENDENT MOTION OPTIMIZATION IN REDUCED VELOCITY SPACE

A. Admissible Collision Avoidance Considering Accelerations

Robot position at the end of a certain control period can be computed through numerical integration as (10). As shown in Fig. 3, in most cases, trajectories can be much different (especially when the duration is big) from the circular one and, thus, may not be able to be approximated by a circle. This paper takes into account the existence of accelerations.

To obtain a satisfactory trajectory, first the commanded velocities should be admissible, as discussed in the previous section. In addition, the selected motion command should result in a trajectory without intersecting with obstacles, and, in the meantime, allow the robot to stop before it touches an obstacle,



(a) Trajectories before robot is stopped. (b) $s_{\text{stop}}(O)$ for $\underline{\omega} \geq 0$. (c) $s_{\text{stop}}(O)$ for $\underline{\omega} < 0$

Fig. 4. Extended robot trajectories and allowed travel distance. (a) Admissible trajectories (in black color) for $\Delta t = 0.2$ s and trajectories (in blue color) for robot to be subsequently stopped. (b) and (c) Allowed travel distance $s_{\text{stop}}(O)$ for the robot to safely stop without touching obstacles.

given the current robot position and the actuator's deceleration capability of the robot. If the robot is commanded to (v_1, ω_1), the minimum time to stop the robot is obtained by decelerating the robot with maximum acceleration without causing it to skid, i.e.,

$$t_{\text{stop}} = \frac{v_1}{a_{\max}}. \tag{15}$$

Similar to the Proof of Property 2, the total arc length that is traveled by the robot, which moves for Δt and is subsequently commanded to stop, can be derived as follows:

$$\begin{aligned} s = \int_{\theta_0}^{\theta_t} r d\theta &= \int_{0}^{\Delta t} v(t)dt + \int_{\Delta t}^{\Delta t + t_{\text{stop}}} v(t)dt \\ &= \frac{v_0 + v_1}{2}\Delta t + \frac{v_1^2}{2a_{\max}} \end{aligned} \tag{16}$$

which implies that the arc length of the extended trajectory for stopping the robot is proportional to the square of v_1.

Fig. 4(a) shows the entire trajectories that are generated if the robot moves at any admissible velocities for a duration of $\Delta t = 0.2$ s and is subsequently commanded to stop. Let A, B, C, and D denote the endpoints of the extended special trajectories (which include the extended part for the robot to stop) *traj 1*, *traj 4*, *traj 5*, and *traj 3*, respectively. For collision test purposes, the set of the robot trajectories obtained in a control period (0.2 s or less) could be approximated by a circular path, starting from the robot's current position and ending at a point denoted by E.

As shown in Fig. 4, the total area that is covered by the trajectories can be approximated by a region γ that is formed by points O, A, B, C, D, and E: the circular arc passing O and E, the sector $E\widehat{AB}$, and the sector $\widehat{AB}\widehat{CD}$. Of course, if $\underline{\omega} \geq 0$, $\widehat{AB}\widehat{CD}$ is not so accurate to represent the corresponding part of the trajectories, but still sufficient for the prediction of any potential collision.

The allowed travel distance $s_{\text{stop}}(O)$ is defined as the maximum possible arc distance for the robot to travel from its current position O at any admissible velocity (under the current translation and rotation velocities, and the robot dynamic constraints) for a duration of Δt, and subsequently, to be commanded to stop without touching the obstacles (denoted as obs). It can be approximated as follows, without being overestimated:

$$s_{\text{stop}}(O) \approx \begin{cases} \infty, & \text{if obs} \bigcap \gamma = \emptyset \\ |\widehat{O\text{obs}}|, & \text{else if obs} \bigcap \widehat{OE} \neq \emptyset \\ s_{\text{stop}}(O)_{E}^{\widehat{AB}} + |\widehat{OE}|, & \text{else if obs} \bigcap E\widehat{AB} \neq \emptyset \\ s_{\text{stop}}(O)_{\widehat{AB}}^{\widehat{CD}} & \\ \quad + |\widehat{OE}| + |E, \widehat{AB}|, & \text{otherwise} \end{cases} \tag{17}$$

where $s_{\text{stop}}(O)_{E}^{\widehat{AB}}$ is part of the allowed travel distance $s_{\text{stop}}(O)$ that is counted from E to arc $\widehat{AB}$, $s_{\text{stop}}(O)_{\widehat{AB}}^{\widehat{CD}}$ is part of $s_{\text{stop}}(O)$ that is counted from arc $\widehat{AB}$ to arc $\widehat{CD}$, and $|\widehat{O\text{obs}}|$ is the arc (whose radius is approximated as v_0/ω_0) distance from O to the obstacle(s), respectively.

Then, we are able to compare the value of the allowed travel distance $s_{\text{stop}}(O)$ with that of the furthest arc length $s(t_{\text{stop}})$ [as in (16)]. Collision-free motion can be ensured if the following constraint is satisfied: $s(t_{\text{stop}}) \leq s_{\text{stop}}(O)$, i.e.,

$$\frac{v_0 + v_1}{2}\Delta t + \frac{v_1^2}{2a_{\max}} \leq s_{\text{stop}}(O). \tag{18}$$

Given that v_1, Δt, and $a_{\max}$ are all nonnegative values, we may derive v_{stop}, the limit of the maximum translation velocity due to the obstacle constraints, as follows:

$$\begin{aligned} \mathbf{V}_s = \Bigg\{ (v_1, \omega_1) | v_1 \leq v_{\text{stop}} &= -\frac{a_{\max}\Delta t}{2} \\ &+ \sqrt{a_{\max}\left(2s_{\text{stop}}(O) - v_0\Delta t\right) + \frac{a_{\max}^2 \Delta t^2}{4}} \Bigg\}. \end{aligned} \tag{19}$$

In comparison, dynamic window approaches consider velocities that do not hit an environment obstacle based on only the chosen velocity command and the assumption of circular robot motion:

$$\left\{(v_1,\omega_1)|v_1 \le \sqrt{2\mathrm{dist}(v_1,\omega_1)a_{\max}},\ \omega_1 \le \sqrt{2\mathrm{dist}(v_1,\omega_1)\varepsilon_{\max}}\right\} \tag{20}$$

where the function $\mathrm{dist}(v_1,\omega_1)$ represents the distance to the closest obstacle on the curvature that is defined by the velocity pair (v_1,ω_1), which is measured by the product of the radius of the circular trajectory $r = v/\omega$ and the angle of the circular path that touches the nearest obstacle.

By combining (11) and (19), the search space of admissible and collision-free velocities in this paper will be

$$\mathbf{V}_1 \triangleq [\underline{v_1},\overline{v_1}] \times [\underline{\omega_1},\overline{\omega_1}] = \mathbf{V}_a \bigcap \mathbf{V}_s. \tag{21}$$

B. Motion Optimization in Event of Potential Collision

Provided that the robot has physical dimensions, the admissible region which covers all admissible trajectories is expanded by a radius defined as (2) before $s_{\mathrm{stop}}(O)$ is computed. To reduce computational load, we consider only a subset of laser scans, which are of range within the maximum possible traveled distance that is determined by the current robot speed and the dynamics of the robot.

Situations of potential collision with obstacles and no potential collision are dealt with different strategies when searching for an optimized motion. In this paper, potential collision is said to exist if and only if any laser scan of the subset falls within the expanded admissible region. The main task under this situation is to avoid collision with obstacles while approaching, if possible, the intermediate waypoint (or the target, denoted as g). The following are the optimization targets when searching for a desired motion command.

1) Move to space with a bigger opening. The movement of the robot is now expected to drive it to a place that is safe from collision with obstacles. In addition, alignment with the target or convergence to it is now of low priority.
2) Direct connectivity with the target. To avoid being trapped in local minima, it is important to ensure a direct (straight-line) connectivity between the resulting robot position $\mathbf{q}_1(x_1,y_1)$ (corresponding to velocity (v_1,ω_1)) and the position of the target $\mathbf{q}_g$.

The requirement of item 1) results in a robot favoring a rotation that drives it to a space within a subset of admissible angular velocities. Before searching for an optimized motion, the rotational velocity is randomly assigned a value within the favorable set of angular velocities that drive the robot to the space with a bigger opening. This favorable set, denoted as $[\omega_\alpha,\omega_\beta]$, is determined when $s_{\mathrm{stop}}(O)$ is computed.

Item 2) is used to ensure that the robot is able to reach each target, while it is noted that generally reactive planning methods (including direction and velocity space approaches) may get trapped in local minima. A collision test is carried out between the obstacle points and the rectangular ray expanded from the line segment $\overline{\mathbf{q}_1\mathbf{q}_g}$. A cost function about connection is defined as follows:

$$\mathrm{linkcost}(\mathbf{q}_1,\mathbf{q}_g) = \begin{cases} 1, & \text{if } \mathcal{C}(\mathbf{q}_1, r_{\mathrm{enlarge}}) \bigcap \mathrm{obs} \neq \emptyset \\ \frac{1}{2}, & \text{if } \Upsilon_{\mathbf{q}_1}^{\mathbf{q}_g}(r_{\mathrm{enlarge}}) \bigcap \mathrm{obs} \neq \emptyset \\ 0, & \text{otherwise.} \end{cases} \tag{22}$$

The objective function to be evaluated in this situation is defined as follows:

$$f_{c,1} = c_v \frac{|\min(v_0,\overline{v_1})/2 - v_1|}{v_{\max}} + c_l \mathrm{linkcost}(\mathbf{q}_1,\mathbf{q}_g) \tag{23}$$

where c_v and c_l are weights that are used to adjust the importance of each item, and the first term in the right-hand side of the equation indicates a preference for the robot to travel at around half of the minimum value between the maximum allowed speed and the current speed, i.e., $\min(v_0,\overline{v_1})$.

C. Motion Optimization in Absence of Potential Collision

The robot under this situation is expected to be commanded to approach the intermediate waypoint as fast as possible. Specifically, the following are the optimization targets when searching for a suitable robot motion.

1) Better alignment with the target. The alignment of the robot heading at $\mathbf{q}_1$ with the target should be relatively favorable compared with other motion candidates.
2) Decreasing distance to the target. The distance of $\mathbf{q}_1$ from the target should decrease with respect to that of the initial robot position $\mathbf{q}_0$. This is to ensure that the robot converges to each waypoint given by the planner.
3) Direct connectivity with the target. It will be favorable if there exists a direct (straight-line) connectivity between $\mathbf{q}_1$ and the target.

First, the robot's new pose $\mathbf{q}_1$ is obtained through integration, as in (10). Item 1) can be measured by the difference between θ_g^b and θ_g^{b1}, the target direction in the current robot body frame $\{b\}$, and the target direction in the final robot body frame $\{b1\}$. In addition, the rotation should not be too large, as it accounts for localization errors (but this is of lower priority compared to the alignment to the target). A certain angular acceleration may occur when the angular velocity is adjusted, and thus, the orientation change θ_1^b should be computed as follows, rather than being evaluated by $\omega_1\Delta t$ as is commonly done:

$$\theta_1^b = \theta_1 - \theta_0 = \frac{1}{2}(\omega_0 + \omega_1)\Delta t. \tag{24}$$

Item 2) is directly evaluated based on the difference between the distances of the target from $\mathbf{q}_1$ and from $\mathbf{q}_0$. As before, item 3) can be evaluated via (22).

A minimization objective function is then designed as follows:

$$f_{c,2} = c_g \frac{\left|\theta_g^{b1}\right| - \left|\theta_g^{b}\right|}{\pi} + c_d\left(|\mathbf{q}_1\mathbf{q}_g| - |\mathbf{q}_0\mathbf{q}_g|\right) + c_l \text{linkcost}(\mathbf{q}_1, \mathbf{q}_g) + c_\theta \frac{\left|\theta_1^b\right|}{\omega_{\max}} \triangleq f_{c,2}^g + f_{c,2}^d + f_{c,2}^l + f_{c,2}^\theta \quad (25)$$

where c_g, c_θ, c_d, and c_l are weights that are used to adjust the importance of each term.

The "stage" of the robot's current motion is determined according to the distances from the robot to the target waypoint and to the start waypoint ($\mathbf{q}_s$), and is categorized as follows:

$$\text{stage} = \begin{cases} S_{\text{Target(VeryClose)}}, & \text{if } |\mathbf{q}_0\mathbf{q}_g| < d_{\text{reached}} \\ S_{\text{Target(Close)}}, & \text{else if } |\mathbf{q}_0\mathbf{q}_g| \le \max(d_{\text{close}}, v_0) \\ S_{\text{Start}}, & \text{else if } |\mathbf{q}_s\mathbf{q}_0| \le d_{\text{closetostart}} \\ S_{\text{Middle}} & \text{otherwise} \end{cases}$$

where $d_{\text{reached}} < d_{\text{closetostart}} < d_{\text{close}}$ holds for the constants.

The parameter c_d is defined as follows, such that a decreasing distance to the target is preferred in the stage of S_{Middle}

$$c_d = \begin{cases} c_d^*, & \text{if in } S_{\text{Middle}} \text{ and } |\mathbf{q}_1\mathbf{q}_g| > |\mathbf{q}_0\mathbf{q}_g| \\ 0, & \text{otherwise} \end{cases} \quad (26)$$

where c_d^* [and c_θ^* in (27)] is a predefined constant. In addition, only motions satisfying $f_{c,2}^g + f_{c,2}^d < 1$ can be accepted in order to facilitate convergence to the goal.

The parameter c_θ is defined as follows, in view that bigger rotations are allowed when the robot is close to the start/target:

$$c_\theta = \begin{cases} c_\theta^*, & \text{if stage} = S_{\text{Middle}} \\ c_\theta^*/2, & \text{otherwise.} \end{cases} \quad (27)$$

D. Optimization Algorithm in Reduced Velocity Space

Algorithm 1 presents the procedure that is used to obtain a series of optimized motions for the robot to reach the specified waypoint. Uni_rand(min, max) is a uniform random function.

Algorithm 1 MotionOptimization(waypoint $\mathbf{q}_g$)

while $\mathbf{q}_g$ is not reached **do**
  Update (v_0, ω_0) with odometry sensory data.
  if A new laser scan is not ready **then**
    continue.
  $\mathbf{V}_a \leftarrow (v_0, \omega_0)$ and robot dynamics. ▷ (11)
  $s_{\text{stop}}(O)$, $[\omega_\alpha, \omega_\beta] \leftarrow v_0, \omega_0, \mathbf{V}_a$, laser scan. ▷ (18)
  $v_{\text{stop}} \leftarrow s_{\text{stop}}(O)$; $\mathbf{V}_1 \leftarrow \mathbf{V}_a \bigcap \mathbf{V}_s$. ▷ (19), (21)
  if potential collision detected **then**
    $f_{c,1}^{old} \leftarrow 2$; OptVelFound $\leftarrow$ FALSE.
    while OptVelFound = FALSE **do**
      $\omega_1 \leftarrow (\omega_\alpha + \omega_\beta)/2 + uni\_rand(\omega_\alpha, \omega_\beta)$.
      for each $v_1 \in$ "discretized set of" $[\underline{v_1}, \overline{v_1}]$ **do**
        if $f_{c,1} < f_{c,1}^{old}$ **then**
          $f_{c,1}^{old} \leftarrow f_{c,1}$; OptVelFound $\leftarrow$ TRUE.
          break.
  else
    $f_{c,2}^{old} \leftarrow 4$; OptVelFound $\leftarrow$ FALSE; $searches \leftarrow 1$.
    while OptVelFound = FALSE **do**
      $v_1 \leftarrow choose\_speed(v_0, \underline{v_1}, \overline{v_1}, searches)$.
      $searches++$.
      for each $\omega_1 \in$ "discretized set of" $[\underline{\omega_1}, \overline{\omega_1}]$ **do**
        if $f_{c,2} < f_{c,2}^{old}$ and $f_{c,2}^g + f_{c,2}^d < 1$ **then**
          $f_{c,2}^{old} \leftarrow f_{c,2}$; OptVelFound $\leftarrow$ TRUE.
          break.
  send velocity command (v_1, ω_1) to robot.

Unlike traditional velocity space approaches, translation or rotation velocity is chosen before optimization is carried out. As shown in the procedure choose_speed (Algorithm 2), if no potential collision is detected, typically, the robot favors a relatively low but accelerated speed when it moves from the start, a relatively low and decelerated speed when it is close to the target, and a relatively high speed in the middle of moving to the target. In the event of potential collision, as shown in Algorithm 1, rotation velocity is set as the median value of the favorable set $[\omega_\alpha, \omega_\beta]$.

Algorithm 2 choose_speed $(v_0, \underline{v_1}, \overline{v_1}, searches)$

Step 1: Adjust v_1 if necessary according to $Stage$:
$S_{\mathbf{Start}}$: $v_1 \leftarrow v_0 + 0.5 \cdot a_{\text{s,max}} \cdot \Delta t$, if $v_0 < v_{\max}/4$.
$S_{\mathbf{Middle}}$: $k \leftarrow 1$ if $v_0 < v_{\max}/2$, $k \leftarrow 0.5$ otherwise.
  $v_1 \leftarrow v_0 + k \cdot a_{\text{s,max}} \cdot \Delta t$.
$S_{\mathbf{Target(VeryClose)}}/S_{\mathbf{Target(Close)}}$:
if target is the goal or the 2n last waypoint **then**
  if $S_{\mathbf{Target(VeryClose)}}$ **then**
    $v_1 \leftarrow v_{\min}$.
  else
    $v_1 \leftarrow v_0 - a_{\max} \cdot \Delta t$, if $v_0 > v_{\max}/2$.
    $v_1 \leftarrow v_0 - 0.5 \cdot a_{\max} \cdot \Delta t \cdot v_0/(v_{\max}/4)$, if $v_0 \in [v_{\max}/4, v_{\max}/2]$.
else
  $k \leftarrow 0.5$ if $v_0 > v_{\max}/2$, 0.25 if $v_0 \in [v_{\max}/4, v_{\max}/2]$, 0 otherwise.
  $k \leftarrow k \cdot 1.5$ if $S_{\mathbf{Target(VeryClose)}}$.
  $v_1 \leftarrow v_0 - k \cdot a_{\text{s,max}} \cdot \Delta t$.
Step 2: Obtain v_1 by adding some variation:
If $searches > 1$ **then**
  $v_1 \leftarrow v_1 + uni\_rand(-v_{\min}, v_{\min}) \cdot (v_1 < 2v_{\min}?0.5 : 1)$.
$v_1 \leftarrow \underline{v_1}$ if $v_1 < \underline{v_1}$; $v_1 \leftarrow \overline{v_1}$ if $v_1 > \overline{v_1}$.

Search in the 2-D velocity space is thus reduced to 1-D. Of course, if, under the chosen translation/rotation velocity, no satisfactory rotation/translation velocity can be found, another translation/rotation velocity would be chosen for a new search.

In the process of generating a motion command, alignment to the target and convergence to it are considered with high priority, which are typically not taken into account by velocity space approaches. Waypoints in this paper are designed in such a way that each pair of adjacent waypoints is visible to each other without being blocked by obstacles in the obstacle-expanded grid map. By ensuring connectivity and a decrease

in the distance/angle to the target waypoint when possible and applying the above velocity choice strategy with target-distance lookahead, the robot is able to reach a neighborhood area (radius $< d_{\text{reached}}$) of the goal or a second last waypoint (for the case of "Possible_Path"). For other kinds of waypoints, the robot needs to reach a neighborhood area of bigger size in order to perform less deceleration in the robot speed. No investigation has been made on landing the robot at a target accurately, considering it is beyond this paper's main topics.

V. Simulation and Experimental Results

The proposed approach was implemented in the Linux/C programming language. In simulations and experiments, the same set of parameter values, as shown below, were used.

1) A circular (radius 0.406 m) differential steering robot is used.
2) The robot's dynamic constraints are $v_{\max} = 1$ m/s, $\omega_{\max} = 2.0$ rad/s, $a_{\max} = 1.2$ m/s^2, $a_{\text{s,max}} = 0.5$ m/s^2, and $\varepsilon_{\max} = 2.0$ rad/s^2. In addition, $v_{\min}$ is set to 0.06 m/s.
3) A laser rangefinder (SICK LMS 291, in experiments) is mounted in the front of the robot. Its detectable range is 50 m, its angular resolution is 1°, and its sampling rate is 5 Hz.
4) The resolution of the global grid maps is 0.05 m $\times$ 0.05 m per cell.
5) The safety margin coefficient c_{enlarge} is set to 1.3.
6) The optimization parameters are set as $c_g = 1$, $c_\theta^* = 0.2$, $c_d^* = 10$, $c_v = 1$, and $c_l = 1$.
7) $d_{\text{reached}} = 0.1$ m, $d_{\text{close}} = 0.3$ m, $d_{\text{closetostart}} = 0.15$ m.
8) Computation and graphics display were performed on a Pentium IV PC (the CPU is 2.4 GHz, and the memory is 1 GB).

A. Simulation Results

In simulations, both range and azimuth errors were introduced to sensor perceptions of the environment to emulate the actual laser data acquisition process. Fig. 5(a)–(c) shows some of the sequence of incremental search (12 times of searches) in the first simulation test. Each diagram plots a grid map (supplied to the search algorithm), path nodes (i.e., waypoints, denoted by small squares in red and wired to their adjacent nodes), and robot trajectories. Green color depicts the expanded part of the grid map that is supplied to search. Fig. 5(d) shows the laser measurements and the final robot trajectories, where each robot pose is plotted as a circle of the robot size in red. Laser scans were continuously added to the plot without erasing the previous ones along with the progress of navigation. To reach the goal (11.28, -16.14), the robot took 83.44 s and traveled for a distance of 37.76 m.

The velocity profiles obtained in this simulation are shown in Fig. 6. It can be seen that the translation and rotation velocities were frequently adjusted, which was to speed up the robot when it was clear of obstacles or to avoid (potential) collision with obstacles. The average speed (which has accounted for the periods when the robot was stopped during a search) is 0.487 m/s. It is noted that, sometimes, the actual velocities be-

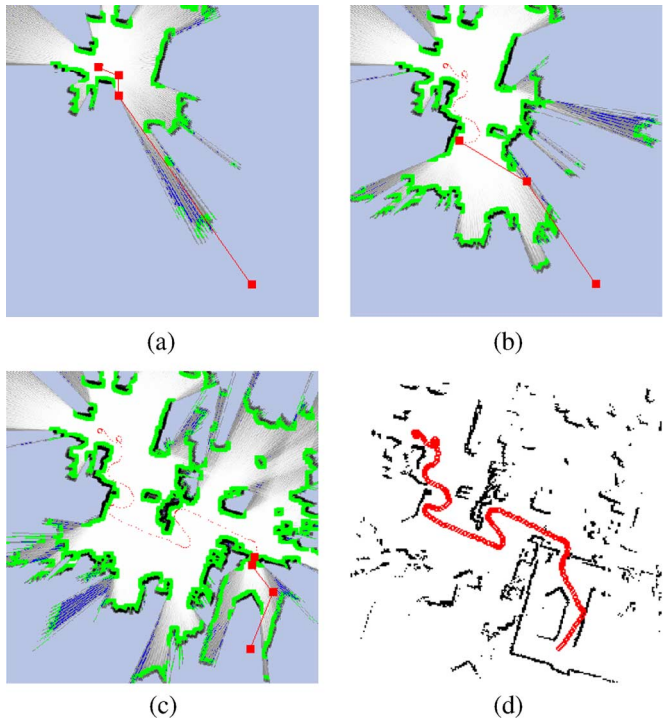


Fig. 5. Path nodes that are obtained by each search, laser scans, and final robot trajectories (from top left to bottom right of each diagram) in first simulation. (a) Result of the second search. (b) Result of the sixth search. (c) Result of the tenth search. (d) Laser scans and robot trajectories.

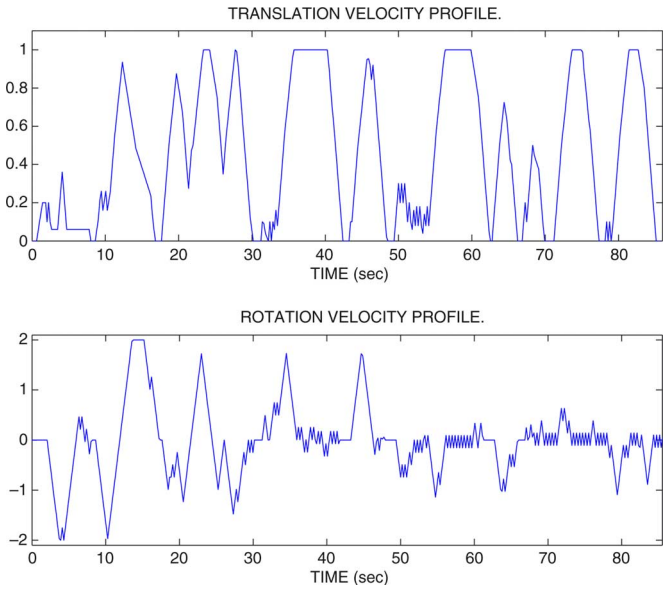


Fig. 6. Profiles of translation and rotation velocities of first simulation test.

came zero, during which the robot was stopped for replanning a new path. Another simulation test was conducted in the same environment, and the average speed is 0.469 m/s.

B. Experimental Results

In the experiments, we used a Magellan Pro robot with a payload of 9.1 kg. The robot can communicate with other computers via wireless Ethernet. The deceleration limit $a_{\max}$ was set to 1.0 m/s^2, instead of 1.2 m/s^2, because it was observed

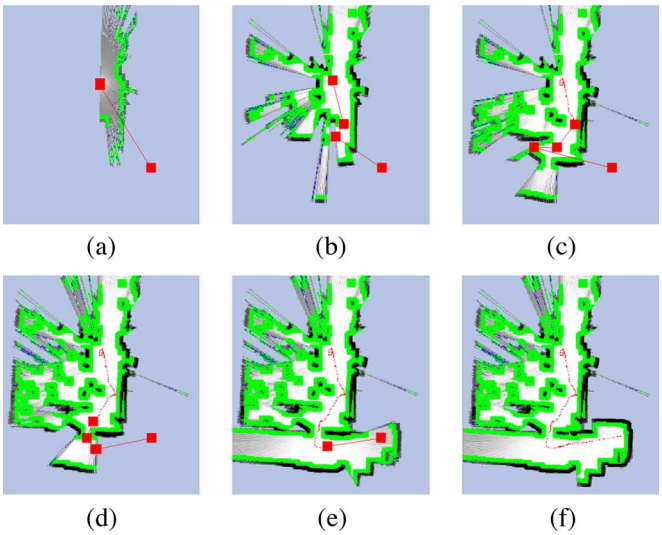


Fig. 7. Path nodes that are obtained in each search and robot trajectories in the first experiment. (a) First search. (b) Second search. (c) Third search. (d) Fourth search. (e) Fifth search. (f) Final trajectories.

that, when moving at its maximum speed limit, the robot may easily become unstable if it is exerted a very big deceleration (close to or around 1.2 m/s^2). Experiments were carried out in an unknown, unstructured laboratory environment. The size of the grid maps is 30 m $\times$ 30 m.

In the first experiment, the goal relative to the initial robot pose was set to (3.5, $-$5.2). Fig. 7 shows the robot trajectories (denoted by solid lines) right before each search and the path nodes (denoted by red squares) that are subsequently obtained by that search. It is shown that the hierarchical path planner is able to lead a robot to get closer to the goal incrementally. In addition, appropriate, optimized motions can be produced to drive the robot to its intermediate target, which might be located close to obstacles.

At the same time of online map building and displaying of the current laser scan and the traveled robot trajectories, a video[3] was taken during the experimental test. Fig. 8 shows the video captures when the robot was to search a path or when the robot was stopped.

The velocity profiles of this test are shown in Fig. 13. The average speed is about 0.289 m/s. The result of another test in the same environment is shown in Fig. 9. The average speed is about 0.337 m/s, which is a bit higher than that of the previous experiment. The translation and rotation velocities in the experiments are smoother than those obtained in the simulations. This is attributed to the fact that a robot, in reality, is not always able to rapidly track the motion commands, i.e., the actual robot velocities will change more smoothly than the commanded ones as long as the change in the commanded ones is not too big. No trajectory controller other than a PID controller has been used in this paper for the robot to track velocity commands. On the other hand, the average speeds are slightly less than those indicated in the simulations.

[3]Videos and screen captures of the first experiment test and screen captures of the first simulation test can be downloaded at http://robotics.ece.nus.edu.sg/planning/demo.htm.

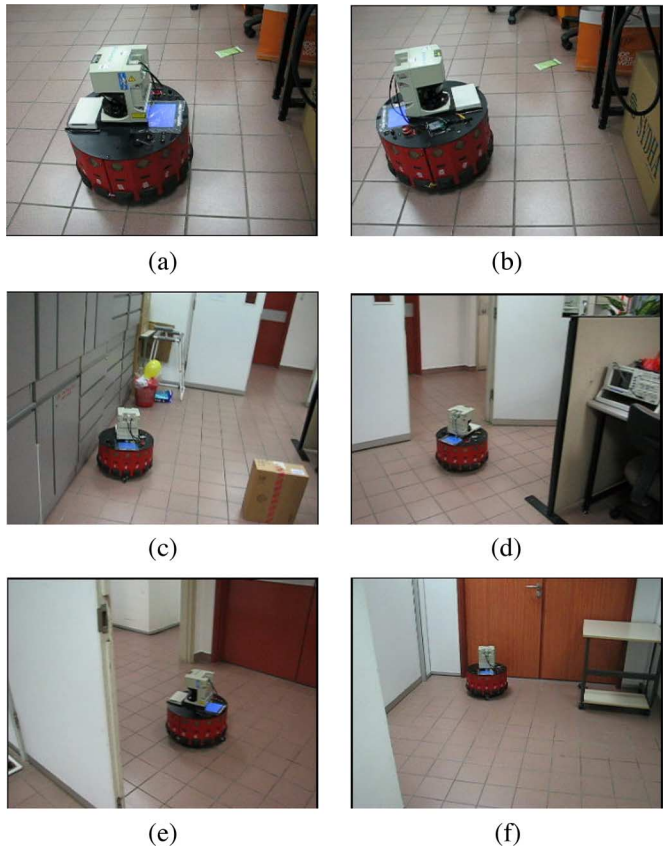


Fig. 8. Snapshots of the first experiment when the robot was to search a path or when the robot was stopped. (a) Searching for the first path. (b) Searching for the second path. (c) Searching for the third path. (d) Searching for the fourth path. (e) Searching for the fifth path. (f) Robot was stopped.

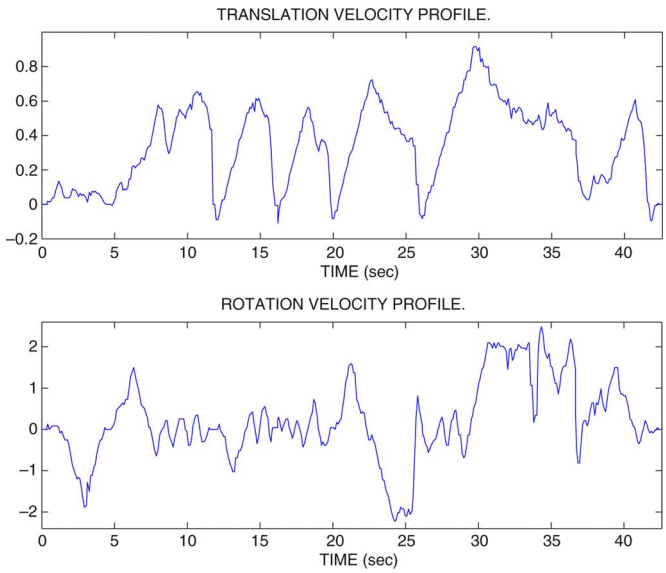


Fig. 9. Profiles of translation and rotation velocities of the second experiment.

VI. Discussions and Comparisons

A. Performance of Incremental Search

Table III shows the time that is used by each search in the four tests, where the start nodes are excluded when counting the number of path nodes (waypoints). Tens of other simulation

TABLE III
USED TIME AND NUMBER OF PATH NODES IN EACH SEARCH

	First Test		Second Test		Third Test	
Index of Search	No. of Nodes	Time Used (s)	No. of Nodes	Time Used (s)	No. of Nodes	Time Used (s)
1	2	< 0.01	2	< 0.01	2	< 0.01
2	3	0.05	3	0.96	3	0.04
3	3	0.43	2	0.24	3	0.05
4	3	0.31	3	0.50	4	0.02
5	2	0.13	3	0.31	2	0.02
6	2	0.33	3	0.30	Fourth Test	
7	2	0.41	3	0.28	2	< 0.01
8	4	0.69	2	0.18	3	0.02
9	3	0.19	3	0.32	3	0.09
10	3	0.05	3	0.37	3	0.01
11	3	0.23	3	0.06	3	0.09
12	2	0.02			2	0.01
Map Size	1200 × 1200		800 × 800		600 × 600	

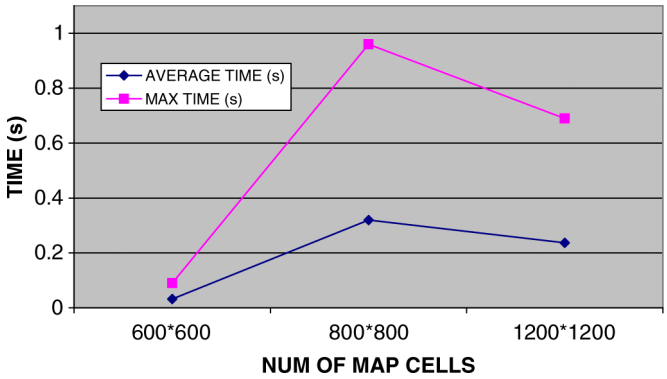


Fig. 10. Statistics of the time used versus different map scales. The "600 × 600" group includes the two experimental results, which use grid maps of the same size.

and experimental tests suggested that the time spent to find a solution is of the same level if the same map size is used.

Fig. 10 shows the statistics of the time used versus different map scales. Generally, the average and maximum values of the search time increase with map size. However, it is not a proportional relationship because search is performed on free space only (the size of which increases with each search) instead of the entire map. It is shown that a search of a map that is scaled at 800 × 800 or 1200 × 1200 (60 m × 60 m) takes an average time of about 0.32 s (or less) and a maximum time of about 0.96 s (or less). This is acceptable for most of indoor applications, considering that simultaneous mapping and path planning is involved. When the map scale is 30 m × 30 m, the search time drops to 0.032 s (average value) or 0.09 s (maximum value).

Note that display of three maps (grid map to represent the environment, map for incremental search, and accumulated laser scans) at the same time costs much computation power and time, considering that drawing/displaying is done every 0.2 s for as many as 1200 × 1200 pixels (for instance). If not for illustration purposes, the display of the three maps can be disabled in order to enhance the system's performance (such as reducing the time used by the planner). In addition, for indoor applications, the detectable range of the laser scanner can be set at a smaller value (e.g., 20 m), instead of 50 m, to further reduce the time/computation that is used by the map updating process upon receiving a new laser scan. It would be

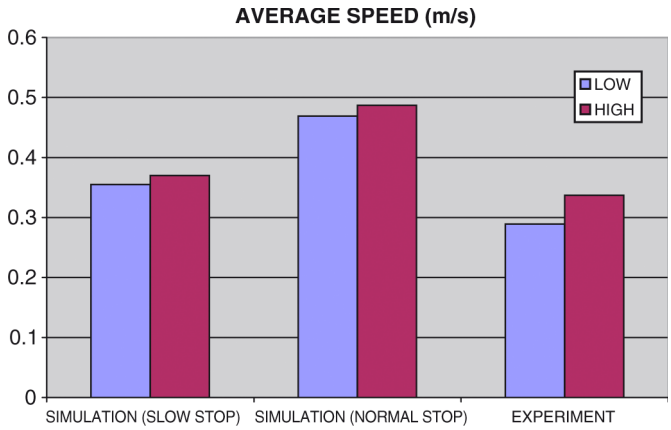


Fig. 11. Average speeds that are achieved under different dynamic settings. "Simulation (slow stop)" group shows results of using $a_{\mathrm{s,max}}$ instead of $a_{\max}$ in (15) and (19) for a smoother stop. "Simulation (normal stop)" and "experiment" groups present test results in Sections V-A and B, respectively.

beneficial to apply these two measures, although no obvious timing or computational issues have been observed on the system throughout the simulation and experimental tests.

B. Robot's Average Speed

Factors such as the robot's actuator capabilities could significantly influence the average speed that the robot may achieve. Fig. 11 shows the average robot speeds that are grouped by different dynamic settings. In the "simulation (slow stop)" group, the average speed dropped to 0.355 or 0.370 m/s, compared to 0.469 or 0.487 m/s in the "simulation (normal stop)" group. This is more obvious in the experiments: the average speed could drop to a third of the values in the "experiment" group. This observation can be explained by the fact that robot dynamics and forward kinematics have been taken into account to compute the value of $s_{\mathrm{stop}}(O)$, which, in turn, determines the value of the maximum translation velocity due to the obstacle constraints, as in (19). A decrease in $a_{\max}$ (and thus t_{stop}) may cause a significant reduction in the allowed travel distance $s_{\mathrm{stop}}(O)$, and the robot thus needs to accelerate or decelerate much more frequently even when the obstacles are faraway.

The surroundings of the robot also have much influence on the average robot speed. It, together with the maximum deceleration limit $a_{\max}$, determines the value of $s_{\mathrm{stop}}(O)$. Since the experimental environment is relatively obstacle-cluttered, the velocities were more frequently adjusted as more often there is a need to avoid (potential) collisions with obstacles. This explains why the robot's average speed in the "experiment" group (where $a_{\max}$ is set to 1.0 m/s^2) is about 0.289 or 0.337 m/s and is a bit lower than that of the simulated robot. This also explains why only a small reduction is found in the average speed obtained by the simulation tests (less obstacle-cluttered environments), in which $a_{\mathrm{s,max}}$ is used instead of $a_{\max}$.

Moreover, the maximum speed setting has some impact on the possible average speed. Nevertheless, a decrease of the maximum speed from 1 to 0.8 m/s alone may not result in a 20% decrease in the average speed. This can easily be seen from

(a) 2.8 s before stop. (b) 2.1 s before stop. (c) 1.5 s before stop. (d) .4 s before stop. (e) Robot stopped. (f) 1.7 s before. (g) .5 s before. (h) Trajectories.

Fig. 12. Snapshots of the first experiment before the third search. (a)–(e) Snapshots of the robot in the experiment. (f)–(g) Snapshots of the current laser scan and the robot's motion direction. (h) Robot trajectories when the robot was stopped.

the robot's translation velocity profile: only for a portion of the whole time is the robot able to reach the maximum speed.

It is noted that the maximum speed setting (1 m/s, which is determined based on the actuator capability of the Magellan Pro robot) and the average speed in our tests could be subpar to that of state-of-the-art, although they are already high for a commonly used indoor mobile robot. The performance of the robot is, however, not necessarily subpar, considering that the evaluation should instead be judged based on the average robot speed, smoothness of motion, robustness in collision avoidance, convergence to the goal, optimality of the path, among others, under the same conditions such as the test environment, and the availability of *a priori* map or a global path. Even when the robot speed alone is evaluated, the nature of our methodology should allow the robot to perform collision-free navigation at a higher average speed, under a higher maximum speed setting, or if a global path to the goal is provided (i.e., the robot is not required to stop in order to find a path to the goal).

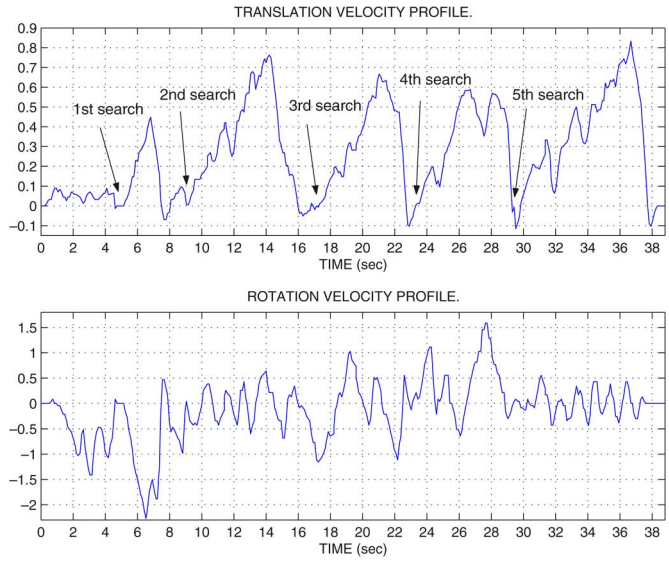


Fig. 13. Profiles of translation and rotation velocities of the first experiment.

C. Collision Avoidance When Very Close to Obstacles

The waypoints provided by the high-level planner are not always far from obstacles, although they are obtained in a way not too close to obstacles. As a consequence, in some instances, the robot needs to maneuver in obstacle-cluttered surroundings when approaching some waypoints. Fig. 12(a)–(d) shows some snapshots (with relative time stamp) of the first experiment before the robot was stopped for the third search [Fig. 12(e)]. The robot was approaching a waypoint very close to the cabinets. Fig. 12(f) and (g) shows the laser measurements, the robot's current location, and its motion direction, where both the robot dimensions and the obstacles are plotted in the same scale. The robot trajectories when the robot was stopped are shown in Fig. 12(h).

The corresponding translation velocity profile (Fig. 13) indicates that the robot started to decelerate from a speed of around 0.75 m/s at the time of about 3 s [see Fig. 12(a)] before the third search, which is located around 17.5 s in the velocity profile. From then on, the speed was observed to decrease rapidly (probably at the robot's maximum deceleration capability). Finally, the robot was able to successfully stop itself with a small distance from the cabinets, as can be seen from the robot trajectories that are shown in Fig. 12(h).

As shown in Fig. 8(d) and (e), the robot passed the door, which is a narrow passage, between the fourth and fifth searches (more accurately, during the period of approaching the first waypoint obtained by the fourth search). The translation velocity profile indicates that the robot accelerated itself when passing the door.

Since the incremental search algorithm relies heavily on the accuracy of grid maps, the robot may be unable to accurately reach the waypoints or the goal due to localization errors. Fortunately, the robot is able to efficiently avoid collision with obstacles reactively based on sensory data with the proposed approach. Fig. 12(h) shows that the cabinets are not properly plotted on the map of C-space, as some misalignment happened, but the robot was able to approach the waypoint without touching the obstacles. Nevertheless, it will be beneficial to implement a localization method other than scan matching that is used in this paper in order to correct the localization error before it evolutes to be large.

D. Comparison With Other Approaches

The nearness diagram algorithm [14] navigates a robot reactively based on situations of the surroundings and the goal's relative position to simplify the difficulty of navigation in troublesome scenarios. The work in [15] further considers shape and kinematics together in an exact manner in its obstacle avoidance process. However, being directional approaches, they may be inadequate to take the robot dynamics into account, which may result in slow or jerky movements.

Dynamic window approaches and the nearness diagram assume circular robot motion in a control period and in the period for the robot to stop. Our collision avoidance methodology has taken into account the existence of accelerations in varying the current translation and rotation velocities to the commanded ones. In this way, the predicted extended trajectories could

be more accurate, and so is the value of the allowed travel distance $s_{\text{stop}}(O)$ that is computed. By computing the limit of the translation velocity with (19), the obstacle constraints are transformed to suitable velocity limits for the robot to move as fast as possible while avoiding collision when needed. In comparison, dynamic window approaches consider the stop distance based on the chosen velocity command and the assumption of circular robot motion as in (20).

Velocity space approaches typically search in the velocity space for a velocity pair minimizing a single objective function. The optimized motion command found in this way may not be suitable for a real scenario. Our method attempts to use various strategies and objective functions to cater to different situations that the robot is currently in (e.g., the surroundings, and whether the robot is leaving or approaching a waypoint), rather than trying to design a universally applicable single objective function. Each of the weights mentioned in Section IV was roughly assigned a value (same for all the simulations and experiments) to indicate its significance relative to others. It is found out that such a parameter set, without being fine-tuned, suffices for our program to successfully run in different scenarios and for both simulations and experiments.

In [26], the effects of dynamic constraints on the torque inputs to the robot motors are considered directly using a dynamic model, instead of the kinematic model. This might result in a smoother control and a smoother robot trajectory. However, the obstacle constraints have yet to be transformed to suitable torques for the robot to accelerate/decelerate at the right time and in the right magnitude in order for high-speed navigation and effective collision avoidance. In addition, differential steering mobile robots typically accept translation and velocity control (only), and thus, the torque-based control may have to be converted back to velocity control via integration, as in the experiments in [26].

VII. Conclusion

This paper has presented a hierarchical approach for incrementally planning optimal paths and subsequently tracing them in unknown environments. The high-level planner is able to handle maps containing unknown information and robustly plan optimal paths incrementally. Situation-dependent object functions and strategies are employed to search for an optimized waypoint-directed motion in a reduced 1-D velocity space. Accelerations in varying translation and rotation velocities are taken into account, and obstacle constraints are transformed to suitable velocity limits for the robot to achieve collision-free, relatively high-speed navigation. Extensive simulation and experimental results verified the efficacy and robustness of the proposed algorithm, and thorough discussions of the test results and comparisons with other approaches are provided.

Acknowledgment

The authors would like to thank the three anonymous reviewers for providing constructive comments for this paper.

References

[1] N. J. Nilsson, *Principles of Artificial Intelligence*. Wellsboro, PA: Tioga, Jan. 1980.
[2] D. Kortenkamp, R. Bonasso, and R. Murphy, *Artificial Intelligence and Mobile Robots*. Cambridge, MA: MIT Press, 1998.
[3] L. Dorst and K. I. Trovato, "Optimal path planning by cost wave propagation in metric configuration space," in *Proc. SPIE—Mobile Robotics III*, 1989, vol. 1007, pp. 186–197.
[4] J. C. Latombe, *Robot Motion Planning*. London, U.K.: Kluwer, 1991.
[5] A. Stentz, "The focussed D* algorithm for real-time replanning," in *Proc. Int. Joint Conf. Artif. Intell.*, 1995, pp. 1652–1659.
[6] A. Yahja, S. Singh, and A. Stentz, "An efficient on-line path planner for outdoor mobile robots operating in vast environments," *Robot. Auton. Syst.*, vol. 32, no. 2/3, pp. 129–143, Aug. 2000.
[7] J. Barraquand and J. C. Latombe, "Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1991, pp. 2328–2335.
[8] Y. Hwang, P. Xavier, P. Chen, and P. Watterberg, "Motion planning with SANDROS and the configuration space toolkit," in *Practical Motion Planning in Robotics*, K. K. Gupta and A. P. del Pobil, Eds. Hoboken, NJ: Wiley, 1998.
[9] G. Oriolo, G. Ulivi, and M. Vendittelli, "Real-time map building and navigation for autonomous robots in unknown environments," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 28, no. 3, pp. 318–333, Jun. 1998.
[10] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986.
[11] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 5, pp. 1179–1187, Sep./Oct. 1989.
[12] S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," *Auton. Robots*, vol. 13, no. 3, pp. 207–222, Nov. 2002.
[13] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Trans. Robot. Autom.*, vol. 7, no. 3, pp. 278–288, Jun. 1991.
[14] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios," *IEEE Trans. Robot. Autom.*, vol. 20, no. 1, pp. 45–59, Feb. 2004.
[15] J. Minguez and L. Montano, "Abstracting vehicle shape and kinematic constraints from obstacle avoidance methods," *Auton. Robots*, vol. 20, no. 1, pp. 43–59, Jan. 2006.
[16] S. S. Ge, X. C. Lai, and A. A. Mamun, "Boundary following and globally convergent path planning using instant goals," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 2, pp. 240–254, Apr. 2005.
[17] S. S. Ge, X. C. Lai, and A. A. Mamun, "Sensor-based path planning for nonholonomic mobile robots subject to dynamic constraints," *Robot. Auton. Syst.*, vol. 55, no. 7, pp. 513–526, Jul. 2007.
[18] F. Lamiraux, D. Bonnafous, and O. Lefebvre, "Reactive path deformation for nonholonomic mobile robots," *IEEE Trans. Robot.*, vol. 20, no. 6, pp. 967–977, Dec. 2004.
[19] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.
[20] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1999, vol. 1, pp. 341–346.
[21] P. Ogren and N. Leonard, "A tractable convergent dynamic window approach to obstacle avoidance," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2002, vol. 1, pp. 595–600.
[22] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 1996, vol. 4, pp. 22–28.
[23] Y. Wang and D. M. Lane, "Solving a generalized constrained optimization problem with both logic AND and OR relationships by a mathematical transformation and its application to robot motion planning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 30, no. 4, pp. 525–536, Nov. 2000.
[24] Y. Hao and S. K. Agrawal, "Formation planning and control of UGVs with trailers," *Auton. Robots*, vol. 19, no. 3, pp. 257–270, Dec. 2005.
[25] X. C. Lai, S. S. Ge, P. T. Ong, and A. A. Mamun, "Incremental path planning using partial map information for mobile robots," in *Proc. 9th ICARCV*, Dec. 5–8, 2006, pp. 133–138.
[26] K. Pathak and S. K. Agrawal, "An integrated path-planning and control approach for nonholonomic unicycles using switched local potentials," *IEEE Trans. Robot.*, vol. 21, no. 6, pp. 1201–1208, Dec. 2005.

Xue-Cheng Lai received the B.Eng. and M.Eng. degrees from Zhejiang University, Hangzhou, China, in 1998 and 2002, respectively. He is currently working toward the Ph.D. degree at the Department of Electrical and Computer Engineering, National University of Singapore, Singapore.

His research interests include sensor-based robot navigation and motion planning, guidance/control for autonomous systems, multirobot exploration and learning, distributed data acquisition systems, signal processing, mechatronics and automation, semiconductor machine automation, and embedded systems.

Shuzhi Sam Ge (S'90–M'92–SM'00–F'06) received the B.Sc. degree from the Beijing University of Aeronautics and Astronautics, Beijing, China, and the Ph.D. degree and the Diploma of Imperial College from the Imperial College of Science, Technology and Medicine, London, U.K.

He is currently a Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. He has (co)-authored three books *Adaptive Neural Network Control of Robotic Manipulators* (World Scientific, 1998), *Stable Adaptive Neural Network Control* (Kluwer, 2001), and *Switched Linear Systems: Control and Design* (Springer-Verlag, 2005), and over 300 international journal and conference papers. He is as an Editor of the Taylor & Francis *Automation and Control Engineering Series* and an Associate Editor of *Automatica*. His current research interests include social robotics, multimedia fusion, adaptive control, and intelligent systems.

Dr. Ge has served/been serving as an Associate Editor for a number of flagship journals, including the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, and IEEE TRANSACTIONS ON NEURAL NETWORKS.

Abdullah Al Mamun (S'91–M'94–SM'04) received the B.Tech. (Hons.) degree from the Indian Institute of Technology, Kharagpur, India, in 1985, and the Ph.D. degree from the National University of Singapore, Singapore, in 1997.

He was a Research Engineer with the Data Storage Institute, Singapore, and a Staff Engineer with the Maxtor Peripherals prior to joining the Faculty of the Department of Electrical and Computer Engineering, National University of Singapore, where he is currently affiliated. His research interest includes precision servomechanism, mechatronics, intelligent control, and autonomous mobile robots.