Proceedings of the 1998 IEEE
International Conference on Robotics & Automation
Leuven, Belgium • May 1998

Accounting for Mobile Robot Dynamics in Sensor–Based Motion Planning: Experimental Results*

J.C. Alvarez† A. Shkel V. Lumelsky

Robotics Laboratory
University of Wisconsin-Madison
Madison,Wisconsin 53706, USA

Abstract

The effect of robot dynamics on autonomous navigation is a key issue in sensor-based motion planning. In most of existing works, the solution is attempted by separating the planning and control into two sequential stages; as a minus, this may, for example, adversely effect the algorithm convergence. The strategy proposed in this paper solves the problem by combining motion planning and control within a single-stage procedure. The procedure exhibits good dynamic behavior, while providing safety (collision avoidance) and fast response. Results of testing the approach on a commercial Nomad-200 mobile robot are presented. Also discussed is the effect of model parameters on motion performance.

1 Introduction

The approaches to autonomous robot motion planning can be classified in terms of the amount of information available to the robot at each moment. In algorithms for planning with complete information, the workspace and the robot characteristics are assumed to be completely known *a priori*. This leads to the off–line calculation of the path, and reduces the problem to one in computational geometry [6]. The control is then a separate computational problem addressed within the classical control theory. This is in contrast with the paradigm of motion planning with incomplete information, where only a subset of the workspace sensed by the robot is known at each instant [8].

*The work was supported in part by the Office of Naval Research, grant 149710871, and by Sea Grant No. NA46RG048.

†On leave from the Systems Engineering & Automation Area, Electrical and Computer Engineering Dpt., University of Oviedo, Gijón 33208, Asturias, Spain

The motion of a physical robot is constrained by its kinematics, mechanical design, the properties of its drive system, and by its dynamics. As a result, a solely geometric solution to the motion planning problem is in general not feasible. For example, if a moving robot attempts to make a sharp turn, it can tip over. Among the related issues, the connection between robot dynamics and motion planning is perhaps the main one; it is also an active area of research. Considered approaches can be called *dynamic*, to distinguish them from the ones that deal solely with geometric and kinematic issues.

This paper addresses sensor-based motion planning with dynamics primarily in the context of physical reality and experimental verification. The algorithm presented below takes into account constraints imposed by the robot dynamics. An important property of the approach is that it combines the sensor-based planning and the control mechanism within a single-stage operation. This provides simplicity and preserves convergence. Parameters of the model are analyzed and identified from experimental data. Dealing with the system dynamics explicitly turns out to be crucial for assuring safety in real-time motion planning. Experiments, carried out on a commercial Nomad–200 mobile robot, demonstrate the improved system behavior under the combined single-stage planning/control procedure.

2 The approach

For the moment, consider the problem of sensor-based navigation as one consisting of two separate problems: a geometric task of path generation (call it Path Planner), for the robot to move in the workspace filled with obstacles, and a control task (call it Controller), which generates control commands to make

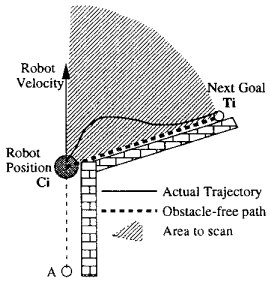

Figure 1: To handle system dynamics, planning and control must be tied together within a single-stage cycle.

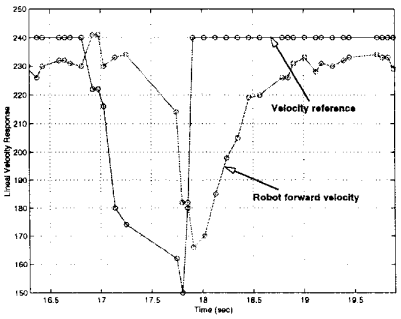

Figure 2: Forward velocity: experimental data for the (u_1, v) relation.

the robot follow that path. The input information of the Path Planner is robot current coordinates, C_i, and the description of the surrounding obstacles. Its output is an intermediate target point, T_i, and a straight-line path segment that leads to it. The Controller's input is the current state C_i, current velocity vector v_i, point T_i, and the path segment from the Path Planner which the Controller is expected to execute.

Notice a possible conflict, Figure 1. As the robot arrives at point C_i along the path AC_i, it decides on a new intermediate target, T_i, and a straight-line path segment to it. Since it arrives at C_i with a non-zero velocity, because of the system dynamics it cannot make a sharp turn suggested by the Path Planner. To preserve continuity in velocity, the real path must have a bulge around C_i (shown in Figure 1 in solid line). This curved path could in principle cut through an obstacle, which the Path Planner would not consider because it is off the intended straight line path. It would be much better if the operation of the Path Planner would directly account for the system dynamics. Besides planning the path proper, it would then also plan for the velocity and acceleration as well. In the example in Figure 1, if the robot expected an obstacle beyond point C_i, it would slow down just enough to decrease the bulge in the path and to keep closer to the line C_iT_i, while keeping the velocity at its optimum. This is the idea that inspired the strategy discussed in [10].

Under the proposed control, the robot always moves with the maximum velocity that is feasible under the circumstances. That is, if the intended path segment is a straight line, the robot will move with the maximum absolute speed – unless it senses an obstacle in this direction, in which case it slows down or/and plans a detour. If the intended path segment is curved, sensing is done within a prescribed sector; again, the velocity is adjusted so as to guarantee safety. The robot model is implicitly included in the selection of the sensor scanning sector. At all times, the robot must have a guarantee of emergency stopping path, in case an obstacle cannot be avoided in a smooth fashion.

3 Identification of Robot Dynamics

The Nomad–200 mobile robot used in this work has a synchro–drive mechanism with decoupled controls of forward velocity and orientation. We are interested in the transfer function relation between the inputs (u_1, u_2) and the robot actual motion (v, ω), corresponding to the forward velocity v, and angular (turning) velocity ω_1, respectively. We assume that the relations (u_1, v) and (u_2, ω) are mutually independent. Figure 2 shows the corresponding experimental data for the first relation. As shown, the command cycle period Δt is about 100 *msec*, which corresponds to about ten commands per second. In Figure 2 the value of the control u_1, responsible for the forward velocity motion v, changes between 15 and 24 $\frac{inch}{s}$. This produces a change in the robot actual velocity, with some time delay, and a certain acceleration rate. Similar behavior appears with the second pair (see [1]). For each pair there is an extra configurable parameter, the rate of change of velocity, set to $a = 10\frac{inch}{s^2}$ and $\alpha = 24\frac{deg}{s^2}$, respectively.

For many applications the system response can be explained with a model with constrained inputs (see, e.g. [9, 2]). This model has some range(s) within which the robot dynamics are not negligible. The robot motors, controlled by high frequency digital controllers, to set the velocities (v, ω) to the required values (u_1, u_2), work at high frequencies (1 kHz in the Nomad–200 case) under an independent microprocessor control [3]. Velocities generated by motion planning algorithms are usually commanded at lower rates

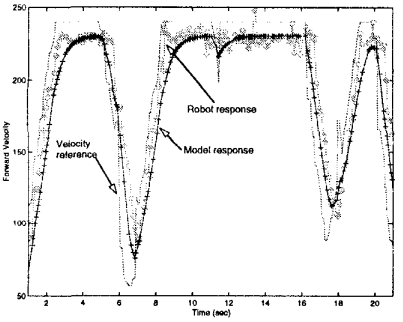

Figure 3: Comparison of (u_1, v) transfer function between the Nomad experimental data and model (4)

(for example, four commands per second [2]). This allows one to ignore the control loop dynamics with good precision, whenever the conditions above are fulfilled. When the planning frequency approaches the control frequency, the effect of the actual (rather than modelled) system dynamics is more pronounced, and larger deviations from the commanded values can be expected.

An alternative approach is to identify the input-output model from the actual robot response data. Along this line, one can estimate the system parameters under specific, experimental conditions, or carry out on-line recursive identification. To identify transfer functions between (u_1, v) and (u_2, ω), we need first to select a candidate model. The best fit, obtained with the least squares approximation of experimental data, corresponds to a first order model with unity delay (it will be used in the next section in the motion planning algorithm design):

$$G(z^{-1}) = \frac{b_1 z^{-1}}{1 + a_1 z^{-1}} \tag{1}$$

Here a_1, b_1 are the parameters to be identified. The comparison results between the experimental data and the model are shown in Figure 3. The real robot and the identified model are fed with the same references, and the responses are compared. From the identified model, the robot motion response for each control can be parameterized with (K, t_s, t_r), where K is the gain ratio, t_s the time to reach the reference (which relates to the maximum acceleration rate), and t_r is the time delay in the reference changes.

A constrained inputs model can be viewed as the discrete version of the differential equations

$$M\dot{v} = \tau_1, \qquad J\dot{\omega} = \tau_2 \tag{2}$$

where M is the robot mass and J represents the robot inertia distribution. Assuming that within the interval $\Delta t = t_1 - t_0$ the inputs $\frac{\tau_1}{M}, \frac{\tau_2}{J}$ are constant, simple integration of the model (2) produces,

$$v(t_1) = v(t_0) + \frac{\tau_1}{M} \cdot \Delta t \qquad \omega(t_1) = \omega(t_0) + \frac{\tau_2}{J} \cdot \Delta t \tag{3}$$

As mentioned above, from the control+planning point of view the main limitation of this model is that typical built-in digital motor controllers do not give one an ability of direct torque control. We can model the motor torque with the equations $\tau_1 = k_1(u_1 - v_k)$ and $\tau_2 = k_2(u_2 - \omega_k)$, with $k_1[\frac{Kg}{s}]$ and $k_2[Kg\frac{m}{s}]$ being dimensional constants. Our controls are now (u_1, u_2), and the corresponding motion equations are:

$$v(t_1) = v(t_0) + \frac{k_1}{M}(u_1 - v(t_0)) \cdot \Delta t \tag{4}$$

$$\omega(t_1) = \omega(t_0) + \frac{k_2}{J}(u_2 - w(t_0)) \cdot \Delta t \tag{5}$$

These relate directly to the difference equation identified in (1) from the experimental data, which can be considered an approximation of the differential equation

$$M\dot{v} + f_v = \bar{\tau}_1 \qquad J\dot{\omega} + f_\omega = \bar{\tau}_2 \tag{6}$$

with $f_v = k_1 v$ and $f_\omega = k_2 \omega$ being linear functions, and $\bar{\tau}_1 = k_1 u_1$, $\bar{\tau}_2 = k_2 u_2$. The relation between the identified parameters and this continuous model is given by:

$$\frac{k_1}{M} = \frac{1 + a_1}{\Delta t} \qquad k_1 = \frac{b_1}{\Delta t} \tag{7}$$

The second control is treated similarly. Model (6) had been used in robot motion control before [4]. Note that when $f_v = 0$ and $f_\omega = 0$, model (6) is reduced to (2).

4 The Algorithm

As explained in Section II, within one command cycle the navigation problem in our system is divided into two stages. First, the planner algorithm produces the next intermediate goal. Then the velocity controller, based on the sensor scan of the environment, decides what controls are to be applied in the next cycle so as to reach the intermediate goal in minimum time. The higher the command frequency, the better the system performance. (The maximum velocity achievable by Nomad-200, $v_{max} = 24[\frac{inch}{s}]$, requires about 10 control commands per second).

The planner can be implemented with any convergent sensor–based motion planning algorithm; in our

experiments, the VisBug algorithm [7] has been used. The controller makes use of the scheme proposed in [10]; in brief, it operates as follows: 1) compute controls to reach the intermediate goal in minimum time, 2) scan the workspace to check if the computed controls guarantee an (emergency) stopping path in the next robot position, 3) if so, proceed at the maximum speed; otherwise, find suboptimal controls that guarantee the stopping path. The Safe-Scan Window (SSW) is the minimum sector that the robot sensors need to scan at each planning cycle in order to assure a safe stopping path if needed, no matter what the robot velocity and direction of motion are. When designing the SSW sector, the requirements to the motion stated above – high speed, safety and convergence – must be respected. Its design will depend on the robot motion equations.

The integration of these non–holonomic motion equations is not easy because of the coupling between velocity and orientation. Alternatively we can use a worst case analysis. To simplify the implementation, the same size area, based on the worst case, will be always scanned. This sector is found based on two canonical operations that cover the worst case of an emergency stop. These are the "Panic Stop" and the "Turn Panic Stop", adopting the nomenclature in [5]. The robot will scan an area big enough to assure that these two operations are always available.

The *Panic Stop operation* occurs when the robot initial conditions are $(v,\omega)_{PS} = (v_{max}, 0)$ and an emergency stop is required. This happens when the robot senses an obstacle while moving at the maximum speed along a straight line. Intuitively, the best controls to apply in order to stop here are $(u_1, u_2)_{PS} = (0,0)$. The identified model (1) predicts a straight line stop path, with an exponential deceleration given by $v_k + a_1 v_{k-1} = 0$, or, using the model in (4):

$$v_k = \left(1 - \frac{k_1 \Delta t}{M}\right) v_{k-1} \tag{8}$$

The distance r_d to stop must include the distance traversed due to the time delay; this leads to

$$r_d = t_r \cdot v_{max} + \sum_{i}^{t_s} v_i \Delta t = v_{max}\left(t_r + \frac{t_s}{1 - a_1}\right) \tag{9}$$

Then, the linear velocity will decrease exponentially following the equation (6), giving $v = v_0 e^{-\frac{k_1}{M}t}$.

The *Turn Panic Stop operation* corresponds to an emergency stop when the robot is turning at its maximum speed. The corresponding conditions will then be $(v,\omega)_{TPS} = (v_{max}, \omega_{max})$. The necessary controls

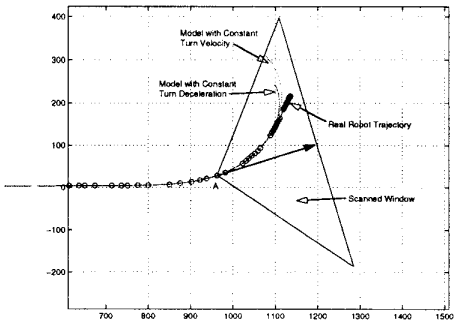

Figure 4: Turn Panic Stop operation.

will be again $(u_1, u_2)_{TPS} = (0,0)$. From the robot motion model, the linear velocity will decrease exponentially, following the same equations as those for Panic Stop. The angular velocity has the same first-order dynamics $\omega = \omega_0 e^{-\frac{k_2}{J}t}$. An approximate solution is possible if we assume that, instead of $\omega = \omega_0 e^{-\frac{k_2}{J}t}$, ω is constant during the whole maneuver. Then the orientation will increase linearly, following $\theta = \omega_0 t$; call it the Constant Turn Velocity simplification. The stop trajectory describes an spiral curve:

$$\Delta x = v_0 (e^{-p_M \Delta t} \frac{-p_M \cos(\omega_0 \Delta t) + \omega_0 \sin(\omega_0 \Delta t)}{p_M^2 + {\omega_0}^2} + \frac{p_M}{p_M^2 + {\omega_0}^2}) \tag{10}$$

$$\Delta y = v_0 (e^{-p_M \Delta t} \frac{-p_M \sin(\omega_0 \Delta t) - \omega_0 \cos(\omega_0 \Delta t)}{p_M^2 + {\omega_0}^2} + \frac{\omega_0}{p_M^2 + {\omega_0}^2}) \tag{11}$$

where $p_M = \frac{k_1}{M}$. The portion of this spiral traversed by the robot depends on the time necessary to bring its forward velocity to zero. Note that with this model, only when $p_M = \frac{k_1}{M}$ is zero, (that is, the term f_v in (6) vanishes), we can say that the forward velocity remains constant, $v = v_0$. If the robot is able to turn fast enough, it can move inside the cone given by the worst case. This has been the case in the experiments with the Nomad: the control system demanded the opposite turn; the robot was able to produce it before stopping, moving towards the inside of the safe area.

From that, and given the Nomad's sonar sensing, a triangular scan window was selected as shown in Figure 4. Two parameters define the SSW area: height (d_f) and aperture (ρ). The SSW height (d_f) is defined by the Panic Stop operation. Let us call v_k the actual robot velocity, r_k the maximum distance traveled within a control cycle, and r_d the distance needed to stop with maximum brake effort in the Panic Stop

operation. Using equation (9), the height condition is

$$r_k = v_k(\Delta t + t_r) \qquad d_f \geq r_k + r_d \tag{12}$$

This is because the distance traversed within a control cycle depends only on the robot current status v_k. Using in these expressions the worst case $v_k = v_{max}$ fixes d_f: it now depends only on the robot model parameters and can be computed off–line.

The aperture ρ of the "Turn Panic Stop" spiral is obtained from equations (10,11). It should be sufficient to guarantee a stop maneuver inside the scanned area in the worst case of turning, when $(v_{max}, \omega_{max})_k$. Two parameters define the maximum curvature of turn with these velocities: the deceleration time until a complete stop t_{dec} (given by (10)), and the minimum turning radius r_{max} (assuming no deceleration, it is $r_{max} = \frac{v_k}{\omega_k}$; it can be calculated better from the model equations). Finally, the cone angle of the scanning cone must be enough to cover the robot dimension (a circle of radius r_r), and to satisfy another restriction: $d_f \tan\rho \geq r_r$. In our case the robot characteristics are

$$v_{max} = 24[\frac{in}{s}],\ a = 10[\frac{in}{s^2}],\ \omega_{max} = 50[\frac{deg}{s}],\ \alpha = 45[\frac{deg}{s^2}].$$

The scan window will be of dimensions $r_k = 100$, $r_d = 290$ (in tenths of an inch), and $\delta \approx 30$ degrees.

The velocity control strategy has to satisfy these considerations: 1) if there are no obstacles inside the scanned area, the robot should move at the maximum velocity, 2) if an obstacle appears inside the area, the robot should have enough time for a complete stop, no matter what its current velocity, 3) when an obstacle appears on the robot's way, it should reduce its velocity or/and plan a detour. Once a detour is safely complete, the robot will accelerate again to its maximum speed. If it needs to stop due to the obstacle, a special recovery maneuver is invoked to modify the path [10]. Intuitively, this strategy realizes an idea of maximum turning, by using the extreme values of the controls:

$$u_1 = K_v, \qquad u_2 = K_w(\delta - \theta). \tag{13}$$

where δ is the angle between the current position and the intermediate goal T_i (in the robot local framework); K_v, K_w values depend on the sensory data about the environment. With no obstacles inside the scanned cone, $K_v = u_{1max}$ and K_w is chosen so as to produce maximum turning, $u_2 = u_{2max}$. If an object appears in the scanned area, the velocity is reduced lineraly, with $K_v = u_{1max} \cdot \frac{d}{d_f}$, where d is the distance to the nearest obstacle inside the area, and d_f the length of the scanned window. The robot deceleration ability is accounted for in the value of d_f. If it's clear that the deceleration is not sufficient to avoid the obstacle, the robot will come to a stop, and then initiate a detour. Such recovery is always possible for a synchro-drive robot.

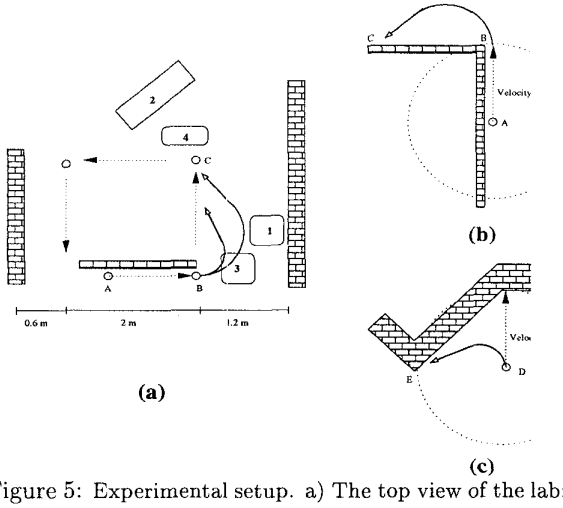

Figure 5: Experimental setup. a) The top view of the lab; objects 1,2,3,4 are additional obstacles. b,c) examples of local motion control.

5 Experimental Results

The experiments were carried out with a Nomad–200 robot, with the control algorithms implemented in the on-board 486–33MHz processor. The Safe–Scan window corresponds to a frontal sonar sensor. Based on the robot specifications, analysis above, and the energy profile of the Polaroid sonar sensor, the scan window aperture ρ was chosen at about 20 degrees. As the focus of the work is on the control strategy, some idiosyncrasies of a sonar sensor – such as false reflections from wall corners or from angular-shape obstacles – were avoided by a proper design of the test workspace.

A sketch of the lab top view is shown in Figure 5a. Obstacles are walls and other objects (chairs, table - objects 1,2,3,4). A sequence of global goals is given, which presents the corners of a 2 by 2 meter square (shown in Figure 5a by straight line arrows). Consider the moment when the robot (shown as a small circle) is moving at its maximum speed through the point A. Between points A and B only forward velocity is controlled. Given the robot model and sensory data, at each control cycle (with 20 cycles per second in our implementation) the motion planner defines a new intermediate goal T_i, which happens to be unchanged until the robot arrives in the vicinity of point B. At B, new intermediate goals appear (eventually this becomes point C), and new velocities are selected.

Note that in the corners of the square the robot moves forward while simultaneously turning towards its new intermediate target.

Tests were carried out with different combinations of obstacles, to study the system performance in various cluttered situations. Examples in Figures 5b,c show typical turning paths. The following table summarizes one set of experiments, aimed at analyzing the performance of the proposed strategy.

	Path Le.	Time	(v_m, a_m)	(w_m, α_m)
Exp 1	3200	25.78	(240,300)	(450,500)
Exp 2	3200	22.53	(240,a_M)	(450,α_M)
Exp 3	3505	24.77	(140,100)	(450,500)
Exp 4a	3422	22.92	(240,100)	(450,500)
Exp 4b	3802	22.57	(240,100)	(450,500)
Exp 4c	3990	22.88	(240,100)	(450,500)
Exp 4d	4450	22.68	(240,100)	(450,500)

Relation between the path length (in tenths of an inch) and motion time (in seconds) was studied under different velocity and acceleration patterns. In the first four experiments in the table above the environment was as shown in Figures 5a. In Experiment 1, a "stop & turn" strategy was used in order to reproduce the exact square path, Figure 5a. The robot was accelerated and decelerated along both control axes with accelerations a_m and α_m (in $\frac{0.1in}{s^2}$ and $\frac{0.1^o}{s^2}$, respectively), and velocities v_m and ω_m. The task was completed in about 26 sec; as expected, the path length was that of the $2m$x$2m$ square, about 3200 of our $.1in$ units. The same strategy is used in Experiment 2, except the robot was forced to its acceleration limits ($a_M = 900$ and $\alpha_M = 2000$). This experiment represents the robot time performance limit, achieved at the expense of slippage, unstable motion and large localization errors. In Experiment 3 the velocity was controlled without any dynamic considerations, using the control law $v = K_v$ and $\omega = K_w(\delta - \theta)$, where values K_v and K_w are design parameters chosen empirically to maximize the robot speed.

In Experiments 4a to 4d the control strategy presented in this paper has been used. Compared to Experiments 1, 2, 3, the same task is finished in shorter time, in spite of smaller accelerations and smoother motion. Figure 6 shows the top view of the resulting path. Motion starts at the lower left corner of the path square. The corresponding velocity profiles are shown in Figure 7. Experiments 4b,c,d relate to the same task and the same control strategy as in Experiment 4a, carried out in increasingly simpler environments: Experiment 4b has no obstacle 4 in its environment, Experiment 4c – no obstacles 4 and 3, Experiment 4d – no obstacles 4, 3, 2. Overall, among the experiments shown in Table I, those based on the proposed strategy showed the best performance.

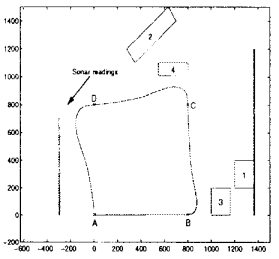

Figure 6: Exp.4a, cluttered environment; top view.

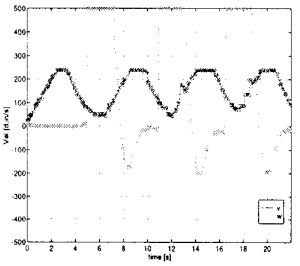

Figure 7: Experiment 4a: velocities (forward & ang.).

The experimental results suggest that the proposed approach works well within the range of speeds tested. It would be interesting to test it at higher speeds and accelerations; it is conceivable that further improvement in performance might be necessary. For such experiments, a robot with the range of velocities and accelerations wider than Nomad-200 would be needed.

References

[1] J. C. Alvarez, A. Shkel, and V. Lumelsky. Accounting for mobile robot dynamics in sensor–based motion planning. Tech.Rep. RL-97007, Rob.Lab., U.W.-Madison, jun 1997.

[2] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, april 1997.

[3] Galil Motion Control, Inc., Sunnyvale, CA 94086. *DMC-600 Series User Manual*, 1991.

[4] J. Guldner and V. I. Utkin. Tracking the gradient of artificial potential fields: sliding mode control for mobile robots. *Int. J. Control*, 63(3):417–432, 1996.

[5] A. Kelly and A. Stentz. Analysis of requeriments for high speed rough terrain autonomous mobility. In *IEEE Int. Conf. on Rob. & Aut.*, pages 3318–25, 1997.

[6] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[7] V. Lumelsky and T. Skewis. Incorporating range sensing in the robot navigation function. *IEEE Trans. Robotics and Automation*, 20(5):1059–1069, sep 1990.

[8] V. Lumelsky and A. Stepanov. Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE Trans. Autom. Control*, 31(11), nov 1986.

[9] Nomadic Techs. *Nomad 200 Users's Guide*, 1996.

[10] A. Shkel and V. Lumelsky. The Jogger's problem: Control of dynamics in real-time motion planning. *Automatica*, 33(7):1219–1233, jul 1997.