Challenges in Real-Time Obstacle Avoidance

Bob Kohout

Veridian/Pacific-Sierra Research
1400 Key Blvd., Suite 700
Rosslyn,Va 22209
rkohout@psrw.com

Abstract

Given the importance of moving-obstacle avoidance in many mission-critical systems, the conditions under which it is possible to guarantee avoidance are poorly understood. Even more troubling is the fact that most researchers in the field address a *static* formulation of the problem, implicitly assuming that a solution to such a problem can be generalized to the dynamic demands of a real-time system. This paper exposes a serious flaw in this assumption. In the process of doing so, it also describes an important, unsolved lower-bound problem that is fundamental to real-time avoidance of moving obstacles.

Robot Motion Planning

Motion planning has been a fundamental component of robotics research from the time the first mobile robots were constructed in the late 1960's. [Nilsson 69] describes what may be the earliest such robot, and this description includes a discussion of a graph-search algorithm used to determine the shortest collision-free path from the robot's location to a goal position. In the 30 years since that time, a large body of research has focused upon the problems associated with determining how to direct a mechanical robot from one location to another, subject to constraints imposed by the application environment and the mechanical constraints of the robot itself.

Broadly speaking, motion planning problems fall into two categories. *Gross motion planning* focuses on problems involving distances much greater than the size of the robot. *Fine motion planning* revolves around the problem of moving an object in a narrow space, where the required accuracy of motion approaches, or even exceeds, the positional accuracy of the robot. This paper is solely concerned with problems and issues implicit in gross motion planning. The terms "motion planning" and "path planning" are used interchangeably throughout the paper, to refer to problems associated with planning the gross motions of mobile robots. There are a number of excellent surveys of gross motion planning research, including [Hwang92], [Latombe91] and [Yap87]. The reader should refer to these for detailed descriptions of research that falls beyond the scope of this paper.

There are two temporal aspects of motion planning that must be distinguished. Adopting terminology from [Hwang92], gross motion planning can be categorized as either *time-invariant* or *time-variant*, according to whether or not the obstacles in the domain can change position over time. Path-planning problems can also be either *static*, in which all of the obstacle information is known to the planner prior to planning, or *dynamic*, in which case information about environment becomes known to the planner over time, after the planning process in initiated, and often during the execution of a partially constructed plan.

By these definitions, most of the previous work in moving-obstacle avoidance (i.e. path planning in time-varying domains) has focused upon a "static" problem. Problem specifics are known when the algorithm for solving it begins to run, and they do not change afterwards. An implicit assumption is that such an algorithm can be easily modified to address the more realistic, "dynamic" case, where aspects of the problem become known to the control system over time, and are subject to change. This paper will demonstrate a serious flaw in this assumption.

Real-Time Motion Planning

Inside of the real-time systems community, there is a strong definition of "real time" that has yet to become part of mainstream usage in the path-planning literature. As [Musliner95] put it

> Real-time computing is *not* about building "fast" systems; it is about building systems that are predictably "fast enough" to act on their environments in well-specified ways.

Systems that are constrained by hard deadlines, which absolutely must be met to avoid catastrophe, are also known as *mission-critical*, or *hard real time* systems. Systems that are shown to meet some explicitly stated statistical performance properties, where timeliness is not strictly guaranteed, and the result of a failure to meet a deadline is not catastrophic, are generally referred to as *soft real time* systems. Systems where the statistical properties are implicit, unexamined and unstated are known as *coincidentally real time* systems. This usage will be adopted through the rest of this paper.

There has been significant research in the area of dynamic motion planning. Typically, such studies are associated with the control of actual robots, and so the convenience of the static assumption cannot be adopted. Many of the papers in this area purport to describe real-time algorithms. With only a very few exceptions, these real-time claims stem directly from the dynamic nature of the problem. In other words, most work in “real-time path-planning” is more correctly categorized as research in dynamic path planning, since the real-time aspects of the work are confined to the fact that data is gathered “online,” after system execution has begun. Such claims are not atypical, and stem generally from the looser definition of real time adopted outside of the real-time systems community. As [Garvey94] put it

> A strict definition of real time is that the system guarantees that tasks will receive up to their worst-case runtime and resource requirements, which presumably means that tasks will produce the best possible results (highest quality results). In real-time AI the focus is usually on high-level goal achievement, rather than worst-case requirements. For this reason, often in real-time AI less strict definitions of real time are used. One common (and usually implicit) definition is that the system will statistically (e.g., on average) achieve the required quality value by the required time, but no guarantee is made about any particular task .

Many recent papers reporting results in obstacle avoidance, such as [Newman91], [Ianigro92], [Hu94], and [Coombs95], make some sort of real-time claim. For example, [Coombs95] describes an impressive system that uses visual information to recognize and avoid obstacles while controlling a robot traveling at relatively high speeds. As evidence of the real-time capabilities of the system, they state:

> The robot has wandered around the lab at 30 cm/s for as long as 20 minutes without collision. To our knowledge, this is the first such demonstration of real-time wandering using only image motion cues.

Typically, as is the case with each of the papers cited above, claims of “real-time” performance are based solely upon the dynamic nature of the problem. In other words, when not all of the obstacle data is known prior to plan execution, and must be somehow sensed at run-time, especially when a robot is simultaneously moving and avoiding obstacles, researchers often claim that their system is real time. While the tendency to conflate dynamic motion-planning with real-time motion planning is epidemic in the literature, it is not universal. For example, [Fox97] describes a technique used aboard a robot that safely navigates at 95 cm/sec in cluttered environments, and they make no use of the term “real time.” According to the terminology adopting above, most real-time path-planning and obstacle-avoidance algorithms are at best coincidentally real time.

The Asteroids Avoidance Problem

The Asteroids Avoidance Problem is the best known and most often studied problem in moving-obstacle avoidance. It can be informally stated as follows:

> Given a robot, at initial location R, that is capable of instantaneous, unbounded acceleration, up to some maximum speed V_R and a set of moving obstacles, $O_1, O_2,\ldots,O_N$, that move at known, constant linear velocities, find a “safety-preserving”' path that the robot can travel to avoid being hit by any of the obstacles.

Typically this problem is cast as one of finding a path to a goal from a starting location, while avoiding moving obstacles. This is not strictly necessary: since all of the obstacles are moving in a straight line, it suffices to show that the robot is capable of getting to some one safe location, at which it can stop indefinitely without being hit. If this is the case, then eventually there will exist a linear, constant velocity, collision-free path to *any* location in the plane. [Canny87] has shown that the static formulation of the problem is ***NP***-hard. However, [Kohout96] established necessary and sufficient conditions for guaranteeing safety in this problem, and in those cases where it is possible to guarantee that a safety-preserving path exists, there are polynomial time algorithms exist to find it. The theorem for sufficiency is:

> **Theorem 1:** Let R be a point in a 2-dimensional Euclidean plane, which represents the location of a robot at time t_0. Assume that the robot can rotate and accelerate instantaneously, but is limited by a maximum speed V_R. Let $O_1, O_2,\ldots,O_N$ be a set of N circular obstacles with diameters $d_1, d_2,\ldots,d_N$ which move at known, constant velocities $v_1, v_2,\ldots,v_N$. Let V_O be the largest of the V_i. Let W be the sum of the widths of the obstacles, i.e., $W = \quad d_I$, $i = 1..N$. If
>
> $$\frac{W(V_O + V_R)}{2V_R}$$
>
> each of the obstacles is initially a distance greater than
>
> from R, then there exists a “safe harbor'” point S such that none of the O_I will touch S at any time, and the robot can move from R to S without intersecting any of the O_i.

The distance $H = W(V_O\_V_R)/2V_R$ is known as the *threat* *horizon.* [Kohout96] also shows that if obstacles are not constrained to start outside of this horizon, it is possible to construct cases where the robot cannot find a path to safety. The proof of these properties is constructive, and suggests an $O(N^3)$ algorithm for solving the problem when the requisite conditions are known to hold true, where N is the number of obstacles. A constant-time Monte Carlo algorithm can also be used, with the caveat that it has a non-zero probability p of failing to find a safe harbor. While p can be made arbitrarily close to zero, it is impossible to use this technique to guarantee a solution.

The static version of the Asteroids Problem, in which all of the relevant information about obstacle trajectories is known prior to algorithmic execution, is far and away the most studied version of the problem, and is known to be intractable. The theorem stated above describes a natural and realistic environmental constraint that both ensures that a safety-preserving path exists *and* that such a path can be found in bounded time. As in much of AI, it has been standard practice to develop algorithms to solve static problems, even when the domain of discourse is known to be dynamic, implicitly assuming that the problem of making the solution "fast enough" amounts to a performance hack.

In the Asteroids Problem, algorithms and bounds that sufficed for the static case of the problem (in a time-varying domain) are insufficient for the dynamic case, in a strong sense. The construction used to prove that H was sufficient to ensure safety in the static version of the problem *cannot* be used for the dynamic case, in which obstacles are allowed to appear over time, after execution begins. The reasons for this have important implications for real-time path planning, as well as a much larger body of algorithmic research.

To briefly summarize, H is chosen to ensure that, regardless of the configuration of obstacles, there will always exist a constant velocity path to safety. In the static formulation of the problem, requiring all obstacles to start a distance of at least H from R implies that at time t_0 all obstacles are outside of a fixed radius. Finding a safe harbor can be somewhat complex, but once it is found, the path is exceedingly simple: a straight line. In the dynamic case, constraining obstacles to appear some minimum distance from the robot says nothing about the relative position of the various obstacles. Safe harbors still exist, and they are just as easy to find, but simply finding *a* safe harbor does not imply that a path to it exists, or that it can be found.

In the case of dynamic path planning, it is not sufficient for the planner merely to find a path to the goal and/or safety. In addition the planners needs to ensure that it doesn't "paint itself into a corner," so that it cannot escape as yet undetected obstacles. More generally, in order to guarantee safety in any dynamic environment, the problem solver needs to prepare solutions that not only solve the part of the problem that is known to it at any given time. Solutions must also anticipate the impact of future events. When the goal of a system is to guarantee some minimal quality of performance, this implies that a real-time control system for a dynamic domain cannot simply choose the solution that best fits the known environment. Whatever solution it finds must also be sufficient in *all possible future worlds.*

Dodger

[Kohout96] does establish *sufficient* conditions for guaranteeing safety in the dynamic version of the Asteroids Problem, which provide a theoretical basis for reasoning over all possible futures, and guaranteeing safety in all possible eventualities. To paraphrase,

> **Theorem 2:** Assume all obstacles travel at the same speed. Let H be $W(V_O+V_R)/2V_R$, where W, V_O, and V_R are as stated above. If each of the obstacles O_i appears at some time $t_{a(i)} \geq 0$, at a distance greater than $\hat{H} = nH + W$ from the position of the robot, and no more than n obstacles can be within a distance $\hat{H}$ of R at any time, then there always exists a collision-free path from $R_{a(i)}$, the robot's location at time $t_{a(i)}$, to some point S such that none of the O_i will touch S at any time.

This section of the paper presents empirical evidence that this theoretical bound is not tight, based upon experiments with a software simulation that solves the Asteroids Problem in hard real-time. There are two things to note about $\hat{H}$, the threat horizon for the dynamic version of the Asteroids Problem. As discussed above, it not known to be necessary. Empirical evidence that this is not a tight bound on the horizon will be presented below. Second, note that $\hat{H}$ is quadratic in the number of obstacles.

Based upon this theoretical result, the author implemented a softbot that solves the dynamic Asteroids problem in hard real time. Known as "Dodger," this implementation avoided over 1.5 millions obstacles in over ten CPU-weeks of testing, without being hit a single time. All computations are made according to a fixed, pre-computed periodic schedule, and are made in strictly bounded clock time. Dodger runs on the Maruti Hard Real-Time Operating System (HRTOS), which computes component schedules, and guarantees that each reactive component executes as scheduled. The Dodger problem space was constrained to the set of problems for which solutions are guaranteed to exist, i.e., it enforces a threat horizon that grows quadratically with the number of obstacles.

In principle, there can be two main components to the system: the reactive, hard real-time competencies that are responsible for guaranteeing safety, and a non-realtime problem solving component that finds safety-preserving paths to goals. However, in its current form Dodger relies upon cyclic, time-bounded heuristic routines to find paths to goals. These *find-path* routines are heuristic in the sense that they do not explicitly search for complete safety-preserving paths to the goal, even though such paths are guaranteed to exist. One of the “heuristics” has been shown to be sufficient to ensure that progress is always made towards a goal and thus is theoretically complete. In practice, each of the heuristics employed by Dodger is sufficient to support very effective goal achievement, without compromising safety. Dodger is described in much greater detail in [Kohout99].

Briefly, the find-path heuristic used for the experiments reported in this paper is as follows: first, the algorithm determines whether or not a constant velocity path to its goal exists, and if so, the robot begins to head along the indicated trajectory. If there is no direct motion to the goal, it calculates the point at which each of the obstacles passes closest to the goal, and then considers the feasibility of going to a nearby point that is not threatened by the obstacle being considered. The robot considers such points in order of their increasing distance from the goal, and will not go to a point that is farther from the goal than its current position. If no such path can be found, the maintenance of the appropriate threat horizon constraints ensures that a safe harbor exists, and a Monte Carlo algorithm is use to find it. The intuition behind this heuristic is that, if direct motions to the goals are blocked, the robot should go to some point that is both closer to the goal and outside of the path of the obstacle(s) occluding the direct path to the goal.

Dodger was tested on a variety of pseudo-random obstacle generation schemes. In the first experiment reported in this paper, all obstacles started at a fixed “battery position,” with trajectories towards the robot's current goal, plus or minus some random noise. All obstacles had the same velocity, and obstacle/robot speed ratio was 1.5. 18 experimental trials were run, in which the number of obstacles was varied from 3-20. These 18 experiments totaled 1,671,319 seconds of runtime (19.34 days), 132,418 goals and 131,220 obstacles. During these trials, the Dodger was not hit once. The find-path computation was run once per 150 milliseconds. The worst-case running time of the find-path algorithm is plotted versus the number of obstacles in Figure 1

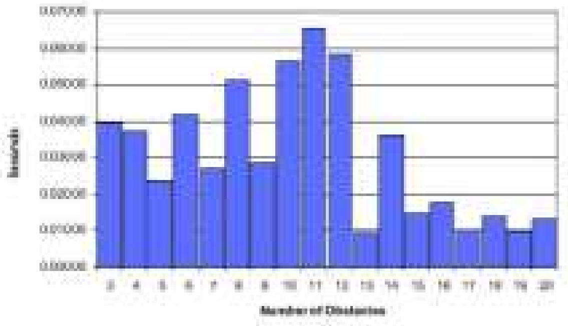

Figure 1

The variance in the plot suggests that nineteen days was not enough runtime to produce a smooth curve, but the trend is nonetheless clear. The observed worst-case performance peaked at 65.48 milliseconds in the 11 obstacle case, and quickly dropped off thereafter. This behavior is consistent with the conjecture that the threat horizon is not tight. The threat horizon in this experiment grows quadratically in the number of obstacles. Even though the worst-case algorithmic complexity of the *find-path* algorithm is linear in the number of obstacles, it apparently becomes easier for Dodger to find safety-preserving paths as the number of obstacles increases. The average-case performance in these experiments is shown in Figure 2. This curve is quite clearly linear, with a least-squares fit of 0.000266537 + 0.000153479x. Thus the average-case performance increases by approximately 1530 *microseconds* per obstacle, and will dominate the worst-case performance at approximately 425 obstacles.

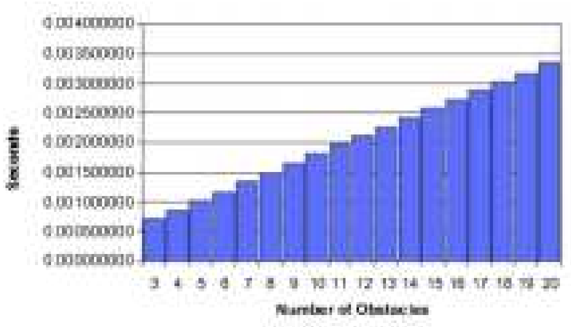

Figure 2

The basic idea of the second experiment was to reduce the threat horizon to the point where the robot starts to get hit, and to thus establish a rough idea of where the actual threat horizon lies. There are two problems with this approach: first, Dodger employs a well-defined avoidance strategy, but the threat horizon applies to *all* possible strategies. Second, just because the Dodger runs without collision for a fixed number of seconds, N, does not imply that it would not be hit at some later time. As N increases, our confidence grows, but in the absence of a mathematically grounded proof, we can never be absolutely certain that a threat horizon empirically determined in this way will suffice in all situations. The point of this investigation is not to empirically establish a threat horizon, so much as it is to get an empirical measure of how it grows with the number of obstacles.

For this experiment, obstacles started at random locations at the edge of a rectangular display that was scaled to the threat horizon. Obstacle trajectory was then randomly assigned to be towards either a) the robot's current location or b) the goal’s current location. The number of obstacles was varied from 9 to 30. For each number of obstacles, the threat horizon was systematically manipulated to find two threat horizons $H_A < H_B < \hat{H}$, such that

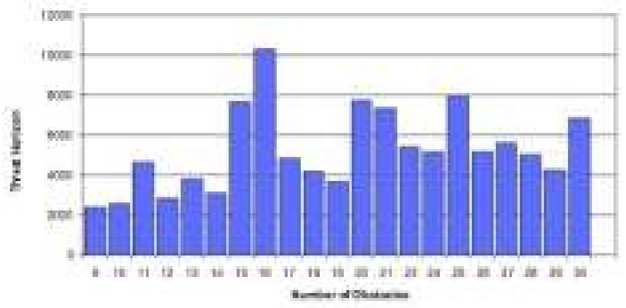

Figure 3

a) Dodger was hit at least once with threat horizon H_A.

b) Dodger avoided obstacles for two or more days with threat horizon H_B.

c) $H_B - H_A \geq \hat{H} /100$

The graph of Figure 3 plots the number of obstacles vs. the empirically determined threat horizon, in units of width. The scatter of this curve is a clear indication that a 2-day threshold is not sufficiently long to produce a smooth indication of the trend. Bear in mind, however, that each data point plotted in this figure represents roughly a week to 10 days of machine time. It took over 283 machine-days, not counting time lost to power outages and other minor disasters, to collect this data. Extending the required period of collision-free running time to a week would have taken several machine-years. Despite the scatter, there does appear to be an upward trend to the data. More significantly, consider Figure 4, which plots both the date shown in Figure 3, in dark, and the theoretical horizon in light.

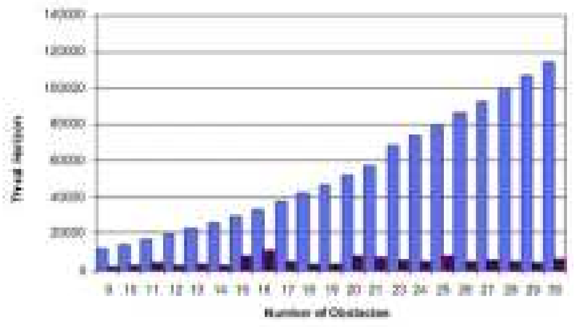

Figure 4

It is clear from this plot that the theoretical horizon greatly overestimates the empirically observed horizon. Figure 5 plots the observed threat horizon vs. the theoretical horizon for the *static* case, which grows linearly in the number of obstacles. In over 9 machine-months of testing, no failures were observed for a threat horizon greater than 5 times the horizon required for the static case. The fact that this graph does not appear to be trending upwards supports an hypothesis that the true theoretical lower bound is near-linear in the number of obstacles.

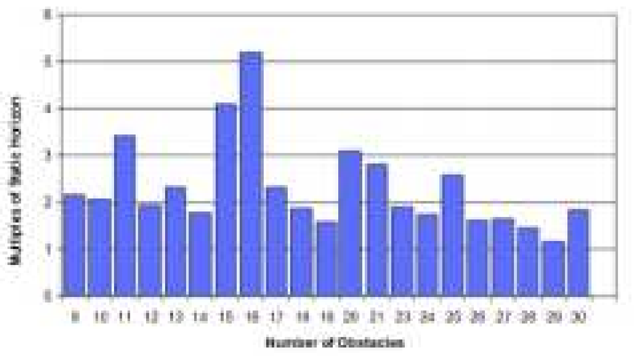

Figure 5

Discussion

These empirical results do not prove that the lower bound of the dynamic threat horizon is not quadratic. It is quite likely that one could construct an obstacle-generation strategy that is far more taxing than the randomized one employed. What would such a strategy be?

This question cuts to the core of the lower-bound problem. In order to assure that an avoidance strategy is sufficient, one must establish that it suffices in all cases, including the "worst case," whatever that may be. In most real-time applications, this worst-case is easily determined, but in the case of path planning, it is not straightforward. Without the concept of a threat horizon, it would be difficult to even define the worst-case configuration. Assuming that all obstacles are constrained to appear, dynamically over time, at a distance at least $\hat{H}$ from the robot, the question becomes:

> What (dynamic) configuration of obstacles forces $\hat{H}$ to be largest for *any possible avoidance strategy*?

Presumably, if this question can be answered, then a tight bound on $\hat{H}$ can be established, although this is not necessarily the case. This is a fundamental problem for any spatially situated mission-critical system, such as airplanes, spacecraft, and motor vehicles. All are subject to potentially catastrophic failure in the event that they collide with a moving obstacle, and all operate in far more complex domains.

Most of the constraints of the Asteroids Problem can be relaxed. [Kohout96] extends the result of Theorem 2 to the case where obstacles can have different non-zero speeds. [Kohout99] extends these results to cases where some obstacles can be stationary, and to cases where obstacles can change speed and directions subject to known bounds. However, all of these solutions make use of the established sufficiency of $\hat{H}$ to ensure safety. Consequently, these results could be improved substantially if the size of the threat horizon can be reduced.

The other main point of this paper is that the traditional approach of solving a static problem formulation, and implicitly assuming that this can somehow be made to apply in a dynamic environment is demonstrably inadequate in the case of real-time path planning. Timeliness cannot be enforced as an afterthought. Improvements in algorithmic and hardware performance belie the fact that most search, scheduling and planning algorithms developed to date are fundamentally offline procedures that can be more or less awkwardly applied in online applications. Real-time requirements do not create the need to plan, design and implement for real-world dynamism. They merely serve to highlight its importance.

Acknowledgements

The author performed the bulk of the research cited in this paper while a student of Prof. James Hendler at the University of Maryland, College Park. This research was supported in part by grants from the ARPA/Rome Laboratory Planning Initiative (F30602-93-C-0039), the ARPA I3 Initiative (N00014-94-10907), NSF (IRI-9306580), ONR (N00014-J-91-1451), AFOSR (F49620-93-1-0065),and ARPA contract DAST-95-C0037. Prof. Hendler is also affiliated with the UM Institute for Systems Research (NSF Grant NSF EEC 94-02384).

References

[Canny87] Canny,J. &Reif,J. 1987. “New Lower Bound Techniques for Robot Motion Planning Problems,” *Proceedings of the 28th IEEE Symposium of Foundations of Computer Science.*

[Coombs95] D.Coombs, M.Herman, T.Hong and M.Nashman, 1995, “Real-Time Obstacle Avoidance Using Central Flow Divergence and Peripheral Flow,” *5th International Conference on Computer Vision.*

[Fox97] Fox,D., Burgard,W. &Thrun,S.” The Dynamic Window Approach to Collision Avoidance,” *IEEE Robotics and Automation Magazine*, **4**(1).

[Garvey94] Garvey,A. and Lesser,V. 1994, “A Survey of Research in Deliberative Real-Time Artificial Intelligence” *Real-Time Systems* **6** 317-347.

[Hu94] Hu,H. and Brady,M. 1994, “A Bayesian Approach to Real-Time Obstacle Avoidance for a Mobile Robot,” *Autonomous Robots,* **1** 69-92.

[Hwang92] Hwang,H.K. and Ahuja,N. 1992 “Gross Motion Planning — A Survey,” *ACM Computing Surveys* **24**(3).

[Ianigro92] M. Ianigro, T.D'Orazio, F.P.Lovergine, E. Stella and A.Distante, 1992. “Real-Time Obstacle Avoidance Based on Sensory Information,” *SPIE Volume 1831 Mobile Robots VII.*

[Kohout96] Kohout,R., Hendler,J. and Musliner,D., 1996, “Guaranteeing Safety in Spatially Situated Agents”, *AAAI ’96.*

[Kohout99] Kohout,R. 1999, “Guaranteeing Safety in the Presence of Moving Obstacles,” University of Maryland, College Park, Technical Report CS-TR 3984, UMIACS-TR 99-06.

[Latombe91] Latombe,J.C., 1991. *Robot Motion Planning*, Kluwer Academic Publishers, Boston/Dordrecht/London.

[Musliner95] D.J.Musliner, J.A.Hendler, A.K.Agrawala, E.H.Durfee, J.K.Strosnider and C. J. Paul, 1995. “The Challenges of Real-Time AI,” *IEEE Computer,* **28**(1).

[Newman91] Newman,W.S. and Hogan,N. 1991. “High Speed Robot Control and Obstacle Avoidance Using Dynamic Potential Functions,” *Proceedings of IEEE International Conference on Robotics and Automation.*

[Nilsson 69] Nilsson,N.J. 1969. “A Mobile Automaton: An Application of Artificial Intelligence Techniques,” *IJCAI ’69.*

[Yap87] Yap,C.K. 1987. "Algorithmic Motion Planning," in *Algorithmic and Geometric Aspects of Robotics,* Schwartz&Yap,eds. Lawrence Erlbaum Associates, London.