The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Versatile Reactive Navigation

Luther A. Tychonievich
Department of Computer Science
Brigham Young University
lty@cs.byu.edu

Robert P. Burton
Department of Computer Science
Brigham Young University
rpburton@cs.byu.edu

Louis P. Tychonievich
Cleveland State University
l.tychonievich@csuohio.edu

Abstract— **Most autonomous mobile agents operate in a highly constrained environment. Despite significant research, existing solutions are limited in their ability to handle heterogeneous constraints within highly dynamic or uncertain environments. This paper presents a novel maneuver selection technique suited for both 2D and 3D environments with highly dynamic maneuvering constraints and multiple mobile obstacles. Agents may have any arbitrary set of nonholonomic control variables; maneuvers can be constrained by a broad class of function inequalities, including time-dependent constraints involving nonlinear relationships between controlled and agent-state variables. The resulting algorithm has been implemented to run in real time using only a fraction of the CPU's capacity on an ordinary notebook computer and performs well in a number of taxing simulated situations.**

I. Introduction

One of the most basic requirements of any autonomous vehicle is that it must not collide with obstacles. This task consists of three separate conceptual subtasks: identifying obstacles, selecting maneuvers to avoid these obstacles, and executing the necessary control to achieve the selected maneuvers. This paper presents a unique solution to the maneuver selection process which is fast enough to run in real time. It represents additional nonholonomic constraints[1] naturally and guarantees short-term collision avoidance.

Autonomous vehicles are in increasing demand for a wide variety of applications. In addition to the traditional robotics problems, collision avoidance has been researched for autonomous cars [1], [2], unmanned aerial and submersible vehicles [3]–[5], smart wheelchairs and walking aids [6]–[8], and many games, movies, and commercial simulations [9]–[11]. In many of these situations there are multiple moving obstacles of unknown trajectories and a variety of additional constraints such as a minimal speed to sustain lift for an airplane, engineering limitations on speed and acceleration, limited jerk-force in the presence of human occupants or delicate equipment, and nonholonomic design of many mobile platforms.

Due to the prevalence of the navigation problem, many algorithms have been published contributing to its solution. These can be organized into four broad categories:

- Application-specific solutions, often featuring monolithic design integrating sensation, maneuver selection, and low-level control [8], [12]–[14].
- Heuristic methods, particularly field-based [15]–[19], but also including various machine-learned and custom heuristics [5], [11], [20], [21].
- Plan-repair or partial-planning methods, both local [22], [23] and global [1], [9], [24], [25] in approach.
- Reactive methods, each making one of three assumptions: that obstacles are immobile [11], [26]–[29], that obstacles have fixed velocities [30]–[36], or that obstacles follow arbitrary known trajectories and agent velocity is fixed [3], [37]–[39].

Of these techniques, the planning and reactive methods offer some form of guarantee of maneuver safety in a general setting. These two approaches differ philosophically; planners proactively search for an optimal trajectory given the best available guess of the future environment, while reactors avoid only immediate problems. Planning methods have the advantage that they do not need a separate maze solver step to avoid dead ends, but gain this advantage at the cost of potentially resolving a maze at every step.

We present a novel method for reactive navigation utilizing a representation of candidate maneuvers powerful enough to encode entire trajectories. This expressive power allows us to increase candidate maneuver sophistication to take advantage of available computing resources without losing the ability to respond reactively to unexpected obstacle behaviors.

Another distinction between navigation algorithms is the breadth of constraint and agent types they can handle. Most algorithms cited above are posed for holonomic agents, which rarely exist in reality. Car-like nonholonomic agents sometimes are modeled explicitly (e.g., [9], [29], [40]) or can be modeled in static environments through a preprocessing step which transforms obstacles into the space of controllable variables, as is done in [17], [18], [41]. Most of the techniques, both for holonomic and car-like agents, accommodate only non-collision constraints; a notable exception is [3], [4] wherein a wide and powerful class of constraints on agent velocity is contemplated, including considerations such as the the impact of submersible velocity on towed sensor arrays. Additional constraints (to prevent vehicle rollover, limit jerk forces, observe traffic laws, etc.) are easily imagined but are rarely if ever mentioned in the literature.

In contrast to existing methods, a wide variety of constraints can be posed and handled within our framework, including time-dependent constraints involving nonlinear relationships between arbitrary sets of controlled and agent-

[1]Meaning here that the allowable motion and possible collisions are independent of the orientation of the agent.

state variables. Distinctions can be made between cooperative and hostile obstacles, between optimistic and pessimistic agents, and between high- and low-priority constraints.

We organize the remainder of the paper as follows. In Section II we discuss the role of reactive maneuvering within the larger robotics context. In Section III we define "agent" and "constraint" and provide an algorithm for evaluating constraints over a finite time interval. In Section IV we provide guidelines for implementing agents and constraints, including discussions of modeling, maneuver selection, and several optimizations. In Section V we provide some experimental results demonstrating real-time performance for a range of simulations. In Section VI we present areas for future development. In Section VII we close with a summary of our work.

II. Reactive Navigation Philosophy

Many solutions to maneuvering in dynamic environments are based on various high-level decisions, such as which of two vehicles approaching an intersection should wait for the other. In reactive navigation, these decisions are not made explicitly: instead, each agent makes a guess as to the likely short-term behavior of the environment and selects a safe short-term maneuver based on that guess. In a purely reactive system, no additional high-level planning is applied; each agent merely reacts in the short term as often as processor capabilities allow. Any appearance of of high-level intelligence in such systems is an emergent behavior not expressly present in the system design.

Purely reactive systems do not behave optimally in most settings. For example, a group of vehicles approaching an intersection might all slow to avoid collision when a higher-level planner would recognize that some of them could continue at full speed safely; worse, the short-term decisions of agents can lead to choosing maneuvers from which no long-term safe trajectory exists. However, reactive systems can be treated as robotic reflexes, sitting between a high-level planner and low-level control. Such a system can provide an extra level of protection against unexpected situations and functions as a mathematically simple failsafe behind a more complex high-level planner.

III. Constraining Maneuvers

Our maneuver selection technique is based on purely mathematical models of the behavior of an agent and the constraints under which it operates. Section III-A provides a mathematical abstraction of the exhibited behavior of an agent given its current state and an applied control. Section III-B describes how to pose various constraints on agent behavior as boolean predicates on the variables present in the model of the agent. Section III-C describes mathematical tools, based on Sturm's theorem and Chebyshev functional approximation, to rapidly evaluate if a proposed control signal will cause the agent to violate its constraints within a finite time window.

A. Agents

Let an **agent** with n state variables and m control variables be represented as a three-tuple, $(\vec{x}, \mathbf{s}, \mathfrak{B})$ where $\vec{x} \in \mathbb{R}^n$ is a vector of real-valued state variables, $\mathbf{s}$ is the agent's discrete state from a finite set $\mathcal{S}$, and $\mathfrak{B} : \mathbb{R}^n \times \mathcal{S} \times \mathbb{R}^m \times \mathcal{M} \to (\mathbb{R} \to \mathbb{R}^n \times \mathcal{S})$ is a higher-order "behavior" function which takes a bounded real-valued control vector $\vec{c} \in \mathbb{R}^m$ and a discrete control mode $\mathbf{m}$ from a finite set $\mathcal{M}$, together with the current agent state, and returns a function evaluating future agent state. By definition, $\mathfrak{B}$ always satisfies the equality $\mathfrak{B}(\vec{x}, \mathbf{s}, \vec{c}, \mathbf{m})(0) \equiv (\vec{x}, \mathbf{s})$.

To help solidify the definition given above, consider the following example of an airplane model:

Example 1 (airplane model):

- $\vec{x}$ is position $\vec{p}$, velocity $\vec{v}$, and orientation (consisting of pitch, roll, and yaw), implying $n = 9$;
- $\mathcal{S} = \{\text{airborne}, \text{grounded}\}$;
- $\vec{c} = (\text{left flap}, \text{right flap}, \text{rudder}, \text{throttle})$, implying $m = 4$; and
- $\mathcal{M}$ is a singleton set, since there is only one mode of operation.

The behavior function $\mathfrak{B}$ becomes a model of airplane dynamics.

B. Constraints

We define a **maneuvering constraint** to be a predicate $f : \mathbb{R}^n \times \mathcal{S} \times \mathbb{R}^m \times \mathcal{M} \to \{\texttt{safe}, \texttt{unsafe}\}$. Then a basic maneuver $\vec{c}, \mathbf{m}$ satisfies the constraint over a time window $t \in [t_0, t_1)$ if and only if $f\left(\mathfrak{B}\left(\vec{x}(t_0), \mathbf{s}(t_0), \vec{c}, \mathbf{m}\right)(t), \vec{c}, \mathbf{m}\right) = \texttt{safe}$ for all $t \in [t_0, t_1)$. A composite maneuver, which can be expressed as $\{\vec{c}_i, \mathbf{m}_i \text{ over } t \in [t_i, t_{i+1})\}$, satisfies a constraint if and only if each of its constituent basic maneuvers satisfies the constraint over the appropriate time window.

To help solidify the definition given above, consider the following example:

Example 2 (School zone speed limit): Consider a town in which the speed limit is 30 mph except for emergency vehicles with a sounding siren, but within $\frac{1}{4}$ mile of the school the limit is 20 miles per hour for all vehicles. If the school is located at $\vec{s}$, and $\mathbf{m} = e$ means "sound the emergency siren" then we can express the constraint function as

$$((\|\vec{v}\| \le 30) \vee (\mathbf{m} = e)) \wedge \left(\left(\|\vec{p} - \vec{s}\| > \frac{1}{4}\right) \vee (\|\vec{v}\| \le 20)\right).$$

In many cases it is more convenient to think of a constraint as a conjunction of several more basic constraints. Thus, we sometimes refer to an agent as having several constraints, by which we mean that f returns `safe` if and only if all of these more basic constraints return `safe`.

C. Evaluating Constraints

The preceding definitions of agents and constraints are highly expressive but do not always admit computational evaluation of constraint satisfaction. However, given the following modest restrictions on agents and constraints, rapid numerical evaluation becomes possible:

i) If $\mathfrak{B}$ permits $\mathbf{s}$ to change under constant $\vec{c}$ and $\mathbf{m}$, the time of the change can be found explicitly.

ii) The elements of the vector portions of $\mathfrak{B}(\vec{x}, \mathbf{s}, \vec{c}, \mathbf{m})(t)$ can be expressed as a ratio of polynomials in t.

iii) The predicate f can be expressed using differentiation and integration with respect to t and the elementary operations $\{\times, \div, +, -\}$, coupled with inequalities ($<, \leq, \geq, >$) and boolean operations ($\neg, \wedge, \vee$).

Conditions ii and iii are rarely satisifed for natural constraints over nonholonomic agent behavior functions. However, polynomial approximation theory is a well-developed field and allows us to create arbitrarily good approximations of $\mathfrak{B}$ and f that do satisfy conditions ii and iii. Chebyshev approximation, reviewed next, generalizes those conditions to

ii) $\mathfrak{B}(\vec{x}, \mathbf{s}, \vec{c}, \mathbf{m})(t)$ has at most a finite set discontinuities at known values of t.

iii) The predicate f is a boolean expression of inequalities of functions with at most a finite set discontinuities at known values of t.

To create a Chebyshev approximation of a function g (where g is any needed non-rational function needed to evaluate $\mathfrak{B}$ and/or f) we follow the standard Chebyshev interpolation scheme presented in any textbook on numerical methods:

1) select one or more time windows $[t_0, t_1]$ over which $g(t)$ is continuous;
2) select a polynomial order n;
3) evaluate $g(t_i)$ at the $i = 1, 2, \ldots, n+1$ Chebyshev nodes, which are

$$t_i = \frac{1}{2}(t_0 + t_1) + \frac{1}{2}(t_1 - t_0)\cos\left(\frac{2i-1}{2n+2}\pi\right);$$

4) fit an nth order polynomial to the $n+1$ points $t_i, g(t_i)$.

The resulting approximation error is bounded by $\frac{1}{2^{n-1}n!}\max_{\xi\in[t_0,t_1]}|f^{(n)}(\xi)|$ and falls super-exponentially with increasing n.

In the exact or approximate case, the resulting function f can be rearranged to a boolean expression with literals of the form $\frac{p(t)}{q(t)} < \frac{r(t)}{s(t)}$, where p, q, r, and s are all polynomials in t, q and s are both positive at t_0, and $\gcd(p,q) = \gcd(r,s) = 1$. This canonical form leads to the following theorem:

Theorem 1: Given four polynomials p, q, r, and s, where $\gcd(p,q) = \gcd(r,s) = 1$, the expression $\frac{p(t)}{q(t)} < \frac{r(t)}{s(t)}$ holds for all $t \in [t_0, t_1]$ if and only if all of the following are true

- $\frac{p(t_0)}{q(t_0)} < \frac{r(t_0)}{s(t_0)}$,
- $(p(t)s(t) - q(t)r(t)) \div \gcd(q,s)$ has no roots for $t \in [t_0, t_1]$, and
- $q(t)$ and $s(t)$ have no roots of odd multiplicity for $t \in (t_0, t_1)$.

A proof of this theorem is given in the appendix.

Theorem 1 motivates Algorithm 1, which evaluates the literals in constraints over a finite time interval.

Algorithm 1: `polynomialInequality`

```
1: Input: Polynomials $p$, $q$, $r$, and $s$; interval $[t_0, t_1]$
2: Output: Truth value of $\frac{p(t)}{q(t)} < \frac{r(t)}{s(t)}\ \forall t \in [t_0, t_1]$
3: if $\frac{p(t_0)}{q(t_0)} \geq \frac{r(t_0)}{s(t_0)}$ then return False
4: if oddRoots($q, [t_0, t_1]$) then return False
5: if oddRoots($s, [t_0, t_1]$) then return False
6: Set $h \leftarrow (p(t)s(t) - q(t)r(t)) \div \gcd(q,s)$
7: if rootCount($h, [t_0, t_1]$) $\geq 1$ then return False
8: return True
```

Algorithm 1 makes use of standard polynomial two subroutines, `rootCount` and `oddRoots`, both based on Sturm's theorem.

Definition 1: The Sturm sequence generated by a polynomial $p(t)$ is

$$\begin{aligned} S_0(t) &= p(t) \\ S_1(t) &= \frac{d}{dt}p(t) \\ S_i(t) &= -\text{remainder}(S_{i-2}, S_{i-1}) \qquad \forall\, i > 1. \end{aligned}$$

Define $\mathbb{S}(p,t)$ to be the number of sign changes in the Sturm sequence of p evaluated at t.

Theorem 2 (Sturm): The number of distinct real roots of p in the interval $t \in (t_0, t_1)$ is given by $\mathbb{S}(p, t_1) - \mathbb{S}(p, t_0)$.

A proof of this theorem may be found in [42].

A direct implementation of Sturm's theorem provides Algorithm 2, which detects the existence of roots without the effort needed to identify individual roots.

Algorithm 2: `rootCount`

```
Input: Polynomial $p(t)$; interval $[t_0, t_1]$
Output: The number of distinct real roots
Output (optional): Polynomial gcd$\left(p, \frac{d}{dt}p\right)$
Set $q \leftarrow \frac{d}{dt}p$
Set $r \leftarrow 0$
Set $(o_1, o_2) \leftarrow (p(t_0), p(t_1))$
while $q \neq 0$ do
    Set $(n_1, n_2) \leftarrow (q(t_0), q(t_1))$
    if $n_1 o_1 < 0$ then Increment $r$
    if $n_2 o_2 < 0$ then Decrement $r$
    Set $p \leftarrow$ remainder$(p \div q) \times -1$
    Swap $p$ and $q$
    Set $(o_1, o_2) \leftarrow (n_1, n_2)$
end while
return $r$ (and optionally also $p$)
```

To derive the `oddRoots` algorithm, recall that if polynomial $p(t)$ has root $t^\star$ with multiplicity $m > 1$ then $\frac{d}{dt}p(t)$ has root $t^\star$ with multiplicity $(m-1)$. Since p and $\frac{d}{dt}p$ share no other roots, $\gcd(p, \frac{d}{dt}p)$ has only the repeated roots of p, each with one lower multiplicity than in p itself. This observation motivates Algorithm 3.

```
Algorithm 3: oddRoots
  Input: Polynomial $p(t)$; interval $[t_0, t_1]$
  Output: True if $p$ has a root of odd multiplicity in $[t_0, t_1]$
  loop
    Set $n, q \leftarrow$ rootCount($p, [t_0, t_1]$)
    if $n = 0$ then return False
    Set $m, p \leftarrow$ rootCount($q, [t_0, t_1]$)
    if $m < n$ then return True
  end loop
```

To evaluate full constraints we combine calls to `polynomialInequality` using boolean algebra.

IV. Implementation

In this section we outline how the tools we presented in Section III can be used to solve a variety of practical maneuver selection problems.

A. Modeling and Constraining Agents

To model a mobile agent, we

1) identify controlled variables and their allowable ranges;
2) write $\mathfrak{B}$, mapping controlled variables to position and orientation functions;
3) create piecewise-polynomial approximations of the position and orientation functions as needed; and
4) create a piecewise-polynomial model of the boundary of the agent.

The boundary could also be a function of time if the agent is capable of self reconfiguration.

To help solidify how these steps work, we present a simple example.

Example 3 (Car-like agent): Consider modeling a wheeled vehicle with limited acceleration and turning radius. The four modeling steps are as follows:

1) We control the curvature of motion c and the forward acceleration a, assumed to be constant during each selection step. Position $\vec{p}$ and speed s are derived values restricted by the use of constraint equations.
2) Let $\vec{p}_0$ be the current position, s_0 the current speed, and θ_0 the current orientation. Defining the "right" vector to be $\vec{r}(\theta) \triangleq (\cos(\theta), \sin(\theta))$, we can then derive
$$\begin{aligned}\theta(t) &= \theta_0 + (s_0 + at)ct \\ s(t) &= s_0 + at \\ \vec{p}(t) &= \vec{p}_0 - \frac{1}{c}\vec{r}(\theta_0) + \frac{1}{c}\vec{r}(\theta(t)).\end{aligned}$$
3) The only functions we need to approximate are the sine/cosine functions. We use a piece-wise cubic Chebyshev approximation of a sine wave over $0 \le t \le \pi$; this has a relative error of only 0.03, accurate enough for most situations.
4) For simplicity, we can approximate a car with a small collection of overlapping circles. For longer vehicles we could use a simple box model: letting $\vec{f}$ be perpendicular to $\vec{r}$, the inside of a car w wide and ℓ long can be approximated by all $\vec{x}$ for which
$$\left|(\vec{p}(t) - \vec{x}) \cdot \vec{f}(t)\right| < \frac{w}{2} \quad \wedge \quad |(\vec{p}(t) - \vec{x}) \cdot \vec{r}(t)| < \frac{\ell}{2}$$

With this setup, we have access to the controlled and state variables and can impose constraints over them as discussed in Section III-B.

B. Modeled Uncertainty

One particular class of constraints provides robust guards against modeled uncertainty. Consider the region of all possible obstacle positions, $\mathcal{P}$; for example, for an agent making unknown decisions this would be $\mathcal{P}(t) = \{\mathfrak{B}(\vec{x}, \mathbf{s}, \vec{c}, \mathbf{m})(t) \mid (\vec{c}, \mathbf{m}) \text{ is a maneuver}\}$. If the obstacle's boundary is the set of point $\mathcal{B}$ then an agent can conservatively guard against all possible obstacle collisions using the Minkowski sum $\mathcal{B} \oplus \mathcal{P}$. The most liberal constraints are generated by the Minkowski difference $\mathcal{B} \ominus \mathcal{P}$. Interpolating between the two by scaling $\mathcal{P}$ can create a variety of intermediate safety guarantees depending on the optimism of the agent and the perceived hostility of the obstacle. One particular design we found useful was a time-varying approach $\mathcal{B} \oplus (1 - \alpha t)\mathcal{P}$ which is, in some sense, the definition of reactive navigation: conservative in the short term, optimistic in the long term.

One particularly useful aspect of uncertainty constraints is the ability to handle noisy, approximate sensory inputs. If a signal processor can provide a range of likely obstacle locations instead of a single best-guess input, the reactive navigation algorithm can guard against the entire range.

C. Search Strategies

Given an agent and a set of evaluable constraints, it becomes necessary to search over the decision space for maneuvers that satisfy the maneuvering constraints which are as similar to the reference commands provided by the high-level planner as possible. The selection of a search strategy should be motivated by our expectations of the environment in which the agent is maneuvering.

If we are confident in our predictions of obstacle behavior then conservative constraints will allow significant freedom and we can plan proactively, searching for a chain of maneuvering decisions that brings the agent to its end goal. An efficient search strategy in this case is the rapidly-exploring random tree which avoids exploring near-redundant states [9], [29], [40]. In highly dynamic but predictable environments it may be useful to explore space-time, not just space, because a short wait can save a lengthy detour when the obstacle is likely to move.

There is often significant uncertainty about the future, such that no long-term maneuver can guarantee agent safety. In these cases a reactive method makes more sense: at each moment, pick a short-term maneuver which is conservative over the immediate time window, but considers only the most probable situations at later times. This is similar to the quiescence search used in many programs for playing board games: since we cannot plan for every contingency explicitly we instead guard against short-term dangers and try to arrive at a "good" position a few moves in the future, where our definition of "good" is based on our expectation of what the other players are likely to do.

A third case occurs when the agent cannot be expected to satisfy all constraints; this might be an agent seeking a collision or trying to dodge many projectiles at once. Here the goal is to find a maneuver which minimizes constraint violation, either by maximizing the time until violation (a root-finding problem in our model) or by minimizing the number of high-priority constraints violated.

In all cases, we suggest using a stochastic search method. We focus on the reactive case, but expect similar principles will apply to other searches as well. We assume that the target maneuver is supplied and use the following observations to build a simple but effective search:

- The search space is strictly bounded; hence quasirandom distributions can locate an initial feasible point.
- The search space must exhibit a fair degree of locality because its complexities are created by a few polynomial inequalities of moderate order; searching near known feasible points allows rapid hill-climbing.
- Since the world typically does not change much from one iteration to the next, a previous solution can serve as a good initial guess.
- We are interested only in finding maneuvers which are more similar to the target maneuver than the best safe maneuver found so far. This defines a subset of the search space we can sample quasirandomly to escape local minima.

Thus we interleave searching near the most recently discovered safe maneuver and searching quasirandomly over the allowable more optimal region. In our experience, this converges to a near-optimal maneuver after sampling just 20–30 test vectors. Using more that 100 test vectors never resulted in improved agent behavior in our simulations.

D. Optimizations

The Minkowski sums and differences presented in Section IV-B can be used to dismiss certain obstacles before beginning a search. The smallest root of the sum of possible agent and obstacle maneuvers is the minimal time before a collision can occur, and the obstacle may be ignored safely until that time. Similarly, if subtracting the agent's possible maneuvers from the obstacle results in a constraint violation, that violation is inevitable and the agent shouldn't waste time trying to find a way to avoid it.

We gained 2–4$\times$ speedups by providing each agent with a single bounding sphere; if the bounding spheres don't collide, the actual contours need not be checked. With a similar bounding sphere for the possible obstacle locations the otherwise expensive Minkowski sums and differences become simple changes in sphere radii.

As with any stochastic search, this technique is parallelizable; both multiple search points and multiple constraints can be checked in parallel, at the possible expense of failing to short-circuit some computations.

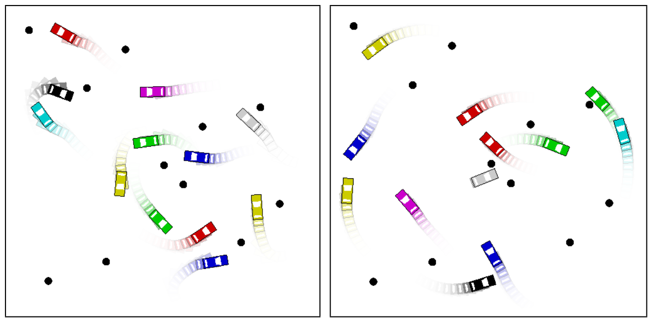
Fig. 1. Time-lapse screen shots of two real time simulations of 12 cars driving in a field containing several pillars. On the left each car has a turning radius of 4 feet, on the right the turning radius is 16 feet. All cars maintain a median speed of 30 mph and steer toward the center of the image if possible. No collisions occurred in either 15-minute simulation.

V. Results

To demonstrate the power of the approach presented in this paper we ran several reactive simulations in situations that would tax an experienced human controller. We implemented the algorithm using C++ with no external library support. Timings are measured using single-threaded execution on a 1.5GHz Core 2 Duo.

Our main tests were run with the 2D car agent described in Example 3; three of these are presented in the movie accompanying this paper and are described below. We also implemented a set of simple holonomic agents in two and three dimensions with controlled variable $\frac{d^k}{dt^k}\vec{p}$ with $k \in \{1,2,3\}$. All simulations were run for fifteen minutes of simulated time.

In our first test we place multiple cars in a mostly open space containing several pillars as shown in Fig. 1. Reference commands tried to keep the cars at 30 mph and tried to steer them toward the center of the test region. Tests were run with various turning radii (between 3 and 20 feet); in all cases each agent was allowed only 10% of a single processor, enough for a 10-sample search 30 times a second. Guarding against all possible obstacle behaviors as outlined in Sec. IV-A resulted in fewer near misses and fewer agents stopping, but no collisions resulted even without such a guard.

Another test placed a car driving at 60 mph in the same field of pillars while being shot at by 15–20 small projectiles per second, each moving at 360 mph. Each is fired from a random location through the middle of possible vehicle locations; in the worst case, six projectiles can guarantee a collision under every possible maneuver. 40% of projectiles hit the car if it takes no evasive action. Randomly generated steering actions reduced this to 15%, while our algorithm running on 50% of one processor was hit by only 1.5% of the projectiles.

A third test placed agents in a lawless intersection situation. For $k = 1$ the holonomic agents avoided all collisions, just as pedestrians typically do. For higher k or car-like agents, the fact that agents do not cooperate meant some agents (up to, depending on the their mobility) were forced off the road as a consequence of being hemmed in by other

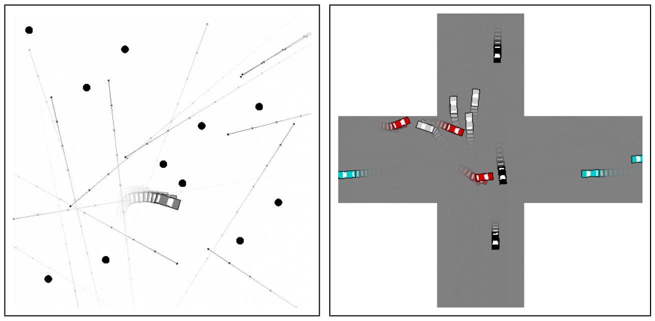

Fig. 2. Time-lapse screen shots of two simulations of cars in hostile environments. On the left the car avoided 98.5% of the projectiles with only half a second lead time. On the right no cars collided, though some were forced off the road by other cars.

agents. Snapshots of both of these tests are presented in Fig. 2.

VI. Future Work

The most obvious next step for this work is to incorporate our algorithm into a real robot. This entials selecting a particular robot and modeling it as an agent, as well as implementing a signal processing technique to provide apropriate input. While the unconstrained nature of the agent models and the handling of explicit input uncertainty should make this a relatively straightforward task, implmenting the algorithm on a physical platform will provide an additional measure of its effectiveness.

We did not implement the parallelization outlined in Section IV-D; however we anticipate it would admit a near-linear increase in performance, allowing the effective use of a more sophisticated set of maneuvers and constraints.

Many domain-specific routines are motivated by the difficulty of discerning individual obstacles with readily-available sensors. Techniques for generating constraints directly from sensor data would allow this technique to be used in such settings without the need for a customized maneuver selection algorithm.

We believe it would be fruitful to place this maneuver selection routine within a publish-subscribe architecture. The maneuvering search would continuously seek good maneuvers given constraints and goals published asynchronously by a high-level planner; the best maneuver known would then be published to a low-level control system. This would increase the modularity and parallelism of the system, but it anticipates an online search methodology more sophisticated than the simple stochastic search we have presented.

VII. Conclusion

We have developed a versatile constraint specification and evaluation technique for mobile agents, discussed in Section III. We highlighted its versatility in Section IV, where we presented techniques for handling uncertainty (IV-B); outlined how it may be used for reactive navigation, plan repair, or in overconstrained settings such as agents seeking collisions (IV-C); and discussed techniques to optimize and parallelize the routine (IV-D). We demonstrated its power in Section V with a number of reactive navigation simulations.

It would be instructive to test this method on a wider range of problems and perform a controlled comparison with other maneuvering algorithms. At present, however, the authors are unaware of any canonical set of example situations for evaluating maneuver selection algorithms.

Appendix

Proof: [Proof of Theorem 1] We first show that the inequality implies each of the three conditions:

- Trivially, $\frac{p(t)}{q(t)} < \frac{r(t)}{s(t)} \forall t \in [t_0, t_1] \implies \frac{p(t_0)}{q(t_0)} < \frac{r(t_0)}{s(t_0)}$.
- $\frac{p(t)}{q(t)} < \frac{r(t)}{s(t)} \implies \frac{p(t)}{q(t)} \neq \frac{r(t)}{s(t)} \iff \frac{ps-qr}{\gcd(q,s)} \neq 0$. The gcd term is needed for the equivalence to hold at the specific ts for which $\gcd(q,s)(t) = 0$.
- A rational function switches from positive infinity to negative infinity at each of the odd-multiplicity roots of the denominator. If only one side of the inequality switches at a given t, the inequality fails on one side of that t; if both switch at the same t the inequality is undefined at that t. Since the inequality is required does hold, neither denominator may contain an odd-multiplicity root within the interval.

We now show that the these three conditions imply the inequality. By the second condition the curves never cross; by the third condition they never jump from positive to negative infinity, so the inequality must have the same truth value over the entire interval. By the first condition it it true for t_0; thus it must be true for the entire interval. ■

References

[1] S. Petti and T. Fraichard, "Safe navigation of a car-like robot within a dynamic environment," in *Proc. of the European Conf. on Mobile Robots*, Ancona (IT), September 2005.

[2] *Proc. IEEE Intelligent Vehicles Symp.*, 1992–2008.

[3] L. P. Tychonievich, D. Zaret, J. Mantegna, R. Evans, E. Muehle, and S. Martin, "A maneuvering-board approach to path planning with moving obstacles," *Proc. 11th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1017–1021, 1989.

[4] T. C. Smith, R. Evans, L. P. Tychonievich, and J. Mantegna, "AUV control using geometric constraint-based reasoning," *IEEE Symposium on Autonomous Underwater Vehichle Technology*, pp. 150–155, 1990.

[5] K.-Y. Kim, J.-W. Park, and M.-J. Tahk, "UAV collision avoidance using probabilistic method in 3-d," in *International Conference on Control, Automation and Systems*. IEEE, October 2007, pp. 826–829.

[6] S. P. Levine, D. A. Bell, L. A. Jaros, R. C. Simpson, Y. Koren, and J. Borenstein, "The navchair assistive wheelchair navigation system," *IEEE Transactions on Rehabilitation Engineering*, vol. 7, no. 4, pp. 443–451, December 1999.

[7] S. Fioretti, T. Leo, and S. Longhi, "A navigation system for increasing the autonomy and the security of powered wheelchairs," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, no. 4, pp. 490–498, December 2000.

[8] I. Ulrich and J. Borenstein, "VFH+: Reliable obstacle avoidance for fast mobile robots," in *IEEE International Conference on Robotics and Automation*, May 1998, pp. 1572–1577.

[9] J. Go, T. Vu, and J. J. Kuffner, "Autonomous behaviors for interactive vehicle animations," in *Eurographics Symposium for Computer Animation*. ACM Press, 2004, pp. 9–18.

[10] R. A. Metoyer and J. K. Hodgins, "Reactive pedestrian path following from examples," *The Visual Computer*, vol. 20, no. 10, pp. 635–649, December 2004.

[11] C. W. Reynolds, "Steering behaviors for autonomous characters," in *Game Developers Conference*, 1999, pp. 763–782.

[12] J. Borenstein and Y. Koren, "The vector field histogram—fast obstacle avoidance for mobile robots," *IEEE Journal of Robotics and Automation*, vol. 7, no. 3, pp. 278–288, June 1991.

[13] J. Minguez and L. Montano, "Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 45–59, February 2004.

[14] P. F. Santana and L. Correia, "Survival kit: a constraint-based behavioural architecture for robot navigation," in *Progress in Artificial Intelligence*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, November 2005, vol. 3805, ch. 7—IROBOT 2005, pp. 435–446.

[15] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.

[16] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *IEEE Conference on Robotics and Automation*, April 1991, pp. 1398–1404.

[17] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[18] J. Mínguez, L. Montano, and J. Santos-Victor, "Reactive navigation for non-holonomic robots using the ego kinematic space," in *IEEE International Conference on Robotics & Automation*, May 2002.

[19] R. A. Metoyer and J. K. Hodgins, "Reactive pedestrian path following from examples," in *International Conference on Computer Animation and Social Agents*, May 2003, pp. 149–156.

[20] M. G. Lagoudakis and A. S. Maida, "Neural maps for mobile robot navigation," in *International Joint Conference on Neural Networks*, vol. 3, July 1999, pp. 2011–2016.

[21] A. F. Foka and P. E. Trahanias, "Predictive control of robot velocity to avoid obstacles in dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and System*, October 2003, pp. 370–375.

[22] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *IEEE International Conference on Robotics and Automation*, vol. 2, 1993, pp. 802–807.

[23] M. Khatib, H. Jaouni, R. Chatila, and J. P. Laumond, "Dynamic path modification for car-like nonholonomic mobile robots," in *IEEE International Conference on Robotics and Automation*, vol. 4, April 1997, pp. 2920–2925.

[24] K. Fujisawa, S. Hayakawa, T. Aoki, T. Suzuki, and S. Okuma, "Real time motion planning for control of autonomous mobile robot," in *IEEE/RSJ International Conference on Intelligent Robots and System*, October 1998.

[25] K. M. Krishna and P. V. Kalra, "Detection, tracking, and avoidance of multiple dynamic objects," *Intelligent and Robotic Systems*, vol. 33, pp. 371–408, 2002.

[26] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *IEEE International Conference on Robotics and Automation*, 1996, pp. 3375–3382.

[27] B. Yamauchi and R. Beer, "Spatial learning for navigation in dynamic environments," *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 26, no. 3, pp. 496–505, 1996.

[28] C. Schlegel, "Fast local obstacle avoidance under kinematic and dynamic constraints for a mobile robot," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 1, 1998, pp. 594–599.

[29] S. M. LaVelle and J. J. K. Jr., "Randomized kinodynamic planning," in *IEEE International Conference on Robotics & Automation*, vol. 1, 1999, pp. 473–479.

[30] F. S. Miller and A. F. Everett, *Instructions for the Use of Martin's Mooring Board and Battenberg's Course Indicator*. Authority of the Lords Commissioners of the Admiralty, 1903.

[31] K. R. Shears, *From "Where you are" to "Where you want to go"; an elementary treatise of the mooring and maneuvering board.* Annapolis, MD: United States naval institute, 1929.

[32] U. S. Government, *Radar Navigation and Maneuvering Board Manual*, 7th ed. U.S, Government, 2001.

[33] K. L. Ehresman and J. L. Frantzen, "Electronic maneuvering board and dead reckoning tracer decision aid for the officer of the deck," *ACM SIGAda Ada Letters*, vol. 21, no. 4, pp. 61–70, December 2001.

[34] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using the relative velocity paradigm," in *IEEE Conference on Robotics and Automation*, 1993, pp. 560–565.

[35] ——, "Motion planning in dynamic environments using velocity obstacles," *International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[36] A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: A collision cone approach," *IEEE Transactions on Systems, Man and Cybernetics—Part A: Systems and Humans*, vol. 28, no. 5, pp. 562–574, September 1998.

[37] Z. Shiller, F. Large, and S. Skhavat, "Motion planning in dynamic environments: obstacles moving along arbitrary trajectories," in *IEEE International Conference on Robotics and Automation*, vol. 4, 2001, pp. 3716–3721.

[38] F. Large, S. Sekhavat, Z. Shiller, and C. Laugier, "Towards real-time global motion planning in a dynamic environment using the NLVO concept," in *IEEE/RSJ International Conference on Intelligent Robots and System*, vol. 1, 2002, pp. 607–612.

[39] F. Large, D. Vasquez, T. Fraichard, and C. Laugier, "Avoiding cars and pedestrians using velocity obstacles and motion prediction," in *IEEE Intelegent Vehicles Symposium*, June 2004, pp. 375–379.

[40] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 3, October 2002, pp. 2383–2388.

[41] J. Mínguez and L. Montano, "The ego-kinodynamic space: Collision avoidance for any shape mobile robots with kinetic and dynamic constraints," in *IEEE/RSJ International Conference on Intelligent Robots and System*, October 2003, pp. 637–643.

[42] M. F. Smiley, "A proof of Sturm's theorem," *The American Mathematical Monthly*, vol. 49, no. 3, pp. 185–186, March 1942.