Noname manuscript No.
(will be inserted by the editor)

On Responsiveness, Safety, and Completeness in Real-Time Motion Planning

Kris Hauser

the date of receipt and acceptance should be inserted later

Abstract: Replanning is a powerful mechanism for controlling robot motion under hard constraints and unpredictable disturbances, but it involves an inherent tradeoff between the planner's power (e.g., a planning horizon or time cutoff) and its responsiveness to disturbances. This paper presents an adaptive time-stepping architecture for real-time planning with several advantageous properties. By dynamically adapting to the amount of time needed for a sample-based motion planner to make progress toward the goal, the technique is robust to the typically high variance exhibited by replanning queries. The technique is proven to be safe and asymptotically complete in a deterministic environment and a static objective. For unpredictably moving obstacles, the technique can be applied to keep the robot safe more reliably than reactive obstacle avoidance or fixed time-step replanning. It can also be applied in a contingency planning algorithm that achieves simultaneous safety-seeking and goal-seeking motion. These techniques generate responsive and safe motion in both simulated and real robots across a range of difficulties, including applications to bounded-acceleration pursuit-evasion, indoor navigation among moving obstacles, and aggressive collision-free teleoperation of an industrial robot arm.

1 Introduction

Robots must frequently adjust their motion in real-time to respond to unmodeled disturbances. A common approach to deal with nonlinear dynamics and hard state and control constraints is to reactively replan at each time step.This basic approach has been studied under various nomenclature (model predictive control, receding horizon control, or real-time planning) and using various underlying planners (numerical optimization, forward search, or sample-based motion planners), and is less susceptible to local minima than myopic potential field approaches. But replanning, in all its forms, faces a fundamental tradeoff based on the choice of time limit: too large, and the system loses responsiveness; too short, and the planner may fail to solve difficult problems in the allotted time, which sacrifices global convergence and safety. Empirical tuning by hand is the usual approach. But the time needed to solve a planning query can vary by orders of magnitude not only between problems, but also between different queries in the same problem, and even on the same query (in the case of randomized planners). Unless variability is addressed, the safety and completeness of real-time replanning is in doubt.

School of Informatics and Computing, Indiana University, hauserk@indiana.edu

The first contribution of this paper is to demonstrate that a real-time replanning framework that adapts the time steps devoted to planning can achieve higher tolerance to variability in planning time than constant time steps. Section 3 presents the adaptive time-stepping with exponential backoff (ATS+EB) algorithm, which uses a sample-based planner to build partial plans whose endpoints monotonically improve an objective function that is given as input. ATS+EB adaptively learns a suitable time step on-the-fly by observing whether the planner is able to make progress within the time limit. In systems with deterministic dynamics it guarantees safe motion by construction, and furthermore we prove that the robot plans a path to a global optimimum of the objective function in expected finite time for a large class of systems (e.g., reversible systems). We apply it to real-time obstacle avoidance for a 6DOF industrial robot in a cluttered environment both in simulation

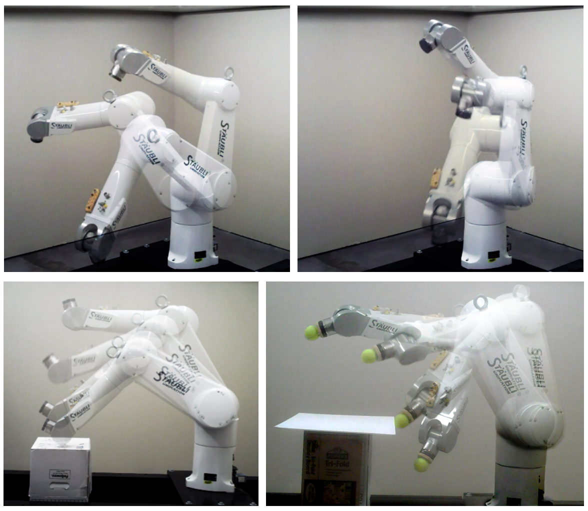

Fig. 1: Assisted teleoperation demonstrations on the Staubli TX90 robot. Top: a circular motion in free space is followed closely. Bottom left: a commanded motion through a box is halted within 1cm of hitting the box. Bottom right: although the command passes through the table top, the robot finds a motion around it.

and on a real robot (Figure 1). The same approach can also be easily applied to variable network latency (jitter) when the robot is controlled by a remote controller. We inject artificial latency into our network to demonstrate dynamically feasible, collision-free motion can be controlled in the presence of arbitrary network delays.

The second contribution of this paper is to demonstrate several advantages of adaptive time-stepping in unpredictable dynamic environments. Here it is apparent that there exist difficult scenarios in which safety and completeness cannot be guaranteed under finite computational resources. Yet adaptive time stepping may be used to schedule resources in order to minimize the risks of failure. A faster response may be needed in a time-critical situation, while more powerful advanced resoning may be needed in a complex one. We present two additional algorithms in Section 4 based on this approach. The first algorithm is a minor variant of ATS+EB applied to optimizing a safety objective. The second is a new conservative contingency planning algorithm that guarantees safety with a pessimistic plan while also seeking an objective function, and we apply it to pursuit-evasion and indoor obstacle avoidance problems.

Several simulations suggest that adaptive time stepping is more consistent than constant time stepping in both deterministic and nondeterministic environments. It can even outperform reactive approaches in obstacle avoidance scenarios where reactive approaches are traditionally considered to work quite well. We conclude with some discussions about the limitations of replanning architectures and avenues for future work.

An earlier version of this work was presented in [9].

2 Related Work

Because planning can be computationally expensive, planning before execution can lead to long delays in movement. To address this problem, *real-time planning* considers interleaving planning with execution, and hence the agent may decide to compute only partial local plans in order to avoid delays in the onset of movement. Real-time planning systems differ in many respects, such as the choice of which underlying planner to use, how much time to spend in each iteration, how detailed to make each plan, how to choose between candidate partial plans, and how to reuse computation between iterations. For any such system we must ask whether the agent will always remain safe (correctness), will always reach a goal (completeness), and how long the agent will take to reach a goal (optimality). Safety is trivially guaranteed in systems without drift, whereas safety in systems with drift can be addressed using slightly more sophisticated planning (see below). As for completeness and optimality, few performance bounds exist outside of relatively restrictive settings (e.g., finite state spaces [24], linear systems with convex constraints [15], or known, large terminal sets [1]). The ATS+EB algorithm attains a general probabilistic completeness condition for continuous, nonlinear systems and arbitrary objective functions over configuration space. Consideration of path optimality is left for future work.

Bounded Rationality in Real-Time Agents. Real-time planning architectures have a long history of study in artificial intelligence, control theory, and robotics, but few have explicitly addressed the problem of "bounded rationality", where limited computational resources hamper an agent's ability to produce timely, optimal plans. A notable exception is the CIRCA real-time agent architecture [17] that separates the agent's control into high-level planning and low-level reactive control tasks. The high-level task conveys controller specifications to the low-level task whenever planning is complete. The disadvantage of this approach is that uncertainty in computation time causes uncertainty in state even in deterministic environments because the system moves during planning. This leads to harder planning problems. By contrast our approach avoids state uncertainty by ensuring, via a hard cutoff, that planning only affects the portion of the trajectory to be executed after the planning completion time, much like [5, 11, 19].

Replanning Applications and Implementations. Model predictive control (MPC), aka receding horizon control, is a form of replanning that at each time step formulates a optimal control problem truncated at some horizon. Such techniques have been successful in robot navigation [2, 23]; for example the classic dynamic windowing technique introduced for indoor mobile robot navigation is essentially MPC by another name [23]. In nonlinear systems, truncated optimal control problems are often solved using numerical optimization or dynamic programming [1, 15]. In discrete state spaces, efficient implementations of replanning algorithms include the D* and Anytime A* algorithms which are based on classic heuristic search [14, 24, 25].

Sample-based motion planners such as randomly-exploring random trees (RRTs) and expansive space trees (ESTs) have been applied to real-time replanning for dynamic continuous systems [4, 5, 11, 28]. RRT and EST variants have been applied to 2D helicopter navigation [5], free-floating 2D robots [11], and and car-like vehicles [19] among moving obstacles, as well as exploring an unknown environment [3] and distributed communication in multi-robot systems [8]. Our algorithms also use sample-based planners because they can be practically applied to high-dimensional spaces, but to our knowledge no prior technique addresses the problem of completeness in predictable environments.

Time Stepping in Replanning. Although many authors have proposed frameworks that can handle nonuniform time steps [3, 5, 19, 28], few actually *exploit* this capability to adapt to the power of the underlying planner. We are aware of one paper in the model predictive control literature [20] that advances time exactly by the amount of time taken for replanning. The weakness of this approach is that if replanning is slow, the actions taken after planning are based on outdated state estimates, leading to major instability and constraint violations. Our work, like many others, avoids this problem by setting planner cutoffs and projecting state estimates forward in time at the start of planning. To adapt the replanning cutoffs we follow the exponential backoff strategy that has proven highly successful in network switching algorithms [16]. The key insight is to treat the uncertainty inherent in sample-based planning much like the uncertainty of an unreliable network.

Safety Mechanisms. Several mechanisms have been proposed to address the safety of replanning agents in dynamic environments. Feron et al introduced the notion of τ-safety, which indicates that a trajectory is safe for at least time τ [5]. Such a certificate establishes a hard deadline for replanning. Hsu et al introduced the notion of an "escape trajectory" as a contingency plan that is taken in case the planner fails to find a path that makes progress toward the goal [11]. We use a contingency planning technique for unpredictable environments that is much like a conservative escape trajectory approach, except that it always ensures the conservative path is followed in case of a planning failure.

The notion of inevitable collision states (ICS) was introduced by Petti and Fraichard to the problem of real-time planning for a car-like vehicle among moving obstacles [19]. An ICS is a state such that no possible control can recover from a collision, and considering ICS as virtual obstacles prevents unnecessary exploration of the state space. In practice, testing for ICS can only be done approximately, and the conservative test proposed in [19] may prevent the robot from passing through states that are actually safe. Our work provides similar safety guarantees without explicit testing for ICS. The basic approach is to extend a search tree by applying random controls as well as braking actions. Solution paths are constrained to terminate at rest states.

Speeding up Replanning. Many approaches have sought to improve responsiveness by simply reducing average replanning time. Some common techniques are to reuse information from previous plans [4], to use precomputed coarse global plans to essentially reduce the depth of local minima [2, 12, 23, 26], or a combination [3, 28]. These approaches are mostly orthogonal to the choice of time step and can be integrated with the adaptive time stepping approach proposed in this paper.

3 Replanning in Deterministic Environments

This section presents the adaptive time-stepping technique as applied to deterministic environments. We note that deterministic environments may be not be static; dynamic environments are also addressed here as long as the environment's motion is predictable. Theoretical and simulation results are described, as well as an application to real-time assisted teleoperation of a robot manipulator.

3.1 Assumptions and Notation

The state of the robot x lies in a state space S, and its motion must obey differential constraints $\dot{x} \in U(x, t)$ (note that this is simply a more compact way of writing control constraints). We assume that the robot has a possibly imperfect model of the environment and how it evolves over time, and let $F(t) \subseteq S$ denote the subset of feasible states at time t. We say that a trajectory $y(t)$ is *τ-safe* if $y(t) \in F(t)$ and $\dot{y}(t) \in U(y(t), t)$ for all $0 \leq t \leq \tau$. If so, we say $y(t)$ is an F_τ trajectory.

F_τ trajectories are guaranteed to remain feasible until time τ, and in the presence of planning time uncertainty it is important to remain on trajectories with high τ. This section will restrict the robot to F_∞ trajectories in order to ensure hard safety guarantees. (We will relax this requirement in Section 4 because it is unrealistic for unpredictable environments, but we will lose safety guarantees.) F_∞ feasibility can be achieved by ensuring that each trajectory terminates at a feasible stationary state. For certain systems with dynamics, such as cars and helicopters, a "braking" control can be applied. This paper will not consider systems like aircraft that cannot reach zero velocity, although terminal cycles may be considered as a relatively straightforward extension.

We address the problem of reaching a global minimum of a continuous time-invariant potential function $V(x)$ via an F_∞ trajectory $y(t)$ starting from the initial state x_0. Assume the global minimum is known and is attained, without loss of generality, at $V(x) = 0$. We say any trajectory that reaches $V(x) = 0$ is a *solution trajectory*. We do not consider path cost, and define the cost functional $C(y)$ that simply returns the value of $V(x)$ at the terminal state of the trajectory y. It is important to note that when we refer to an optimal solution in this paper, we are referring to the *optimality of the terminal point*, not the trajectory taken to reach it.

3.2 Real-Time Replanning Architecture

In real-time replanning the robot interleaves threads of *replanning* and *execution*, in which the robot executes a partial trajectory (possibly using a high rate feedback controller) that is intermittently updated by the replanning thread (at a lower rate) without interrupting execution.

The communication between the execution and replanning thread is subject to the *real-time constraint* that no portion of the current trajectory that is being executed can be modified. So, if a replan is instantiated at time t and is allowed to run for time Δ, then no portion of the current trajectory before time $t + \Delta$ may be modified. Thus when the architecture chooses a time cutoff Δ, the planner starts from a state propagated in the future by time Δ. The prediction step is important because otherwise the new plan may start at an out-of-date state estimate, leading to a loss of safety or convergence.

We assume that the *underlying planner* has the following "any-time" characteristics:

```
**ATS+EB**. Replanning with an Adaptive Time-Step
*Initialization*:
0a. $y(t)$ is set to an $F_\infty$ initial trajectory starting from $t = 0$.
0b. $\Delta_1$ is set to a positive constant.

*Repeat for* $k = 1, \ldots$:
1. Measure the current time $t_k$
2. Initialize a plan starting from $y(t_k + \Delta_k)$, and plan for $\Delta_k$ time
3. If $C(\hat{y}) \leq C(y) - \epsilon$ for the best trajectory $\hat{y}(t)$ generated so far, then
4.     Replace the section of the path after $t_k + \Delta_k$ with $\hat{y}$
5.     Set $\Delta_{k+1} = 2/3\Delta_k$
6. Otherwise,
7.     Set $\Delta_{k+1} = 2\Delta_k$
```

Fig. 2: Pseudocode for the replanning algorithm.

1. The planner iteratively generates F_∞ trajectories starting from an initial state and time $y(t_0)$ given as input.
2. Planning can be terminated at any time, at which point it returns the trajectory that attains the least value of the cost functional $C(y)$ found so far.
3. If the planner is given no time limit on any query that admits a solution trajectory, then the planner finds a solution in expected finite time.

A variety of underlying planning techniques (e.g., trajectory optimization, forward search, and sample-based motion planning) can be implemented in this fashion. All simulations and experiments in this paper are conducted with minor variants of the sampling-based planners RRT [13] and SBL [21], which grow trees using forward integration of randomly-sampled control inputs. Adaptive time-stepping is particulary important for sample-based planners because their running time is variable across runs on a single query, and can vary by orders of magnitude with the width of narrow passages in the feasible space.

3.3 Adaptive Time-Stepping With Exponential Backoff

Here we describe our adaptive-time step replanning algorithm and a simple but effective exponential backoff strategy for adapting to an appropriate time step. Pseudocode is listed as ATS+EB in Figure 2. The replanning thread takes time steps $\Delta_1, \Delta_2, \ldots$. In each time step, the planner is initialized from the state on the current trajectory at time $t_k + \Delta_k$, and plans until Δ_k time has elapsed (Line 2). If the planner finds a trajectory with lower cost than the current trajectory (Line 3) then the new trajectory is spliced into current trajectory at the junction $t_k + \Delta_k$ (Line 4). Otherwise,

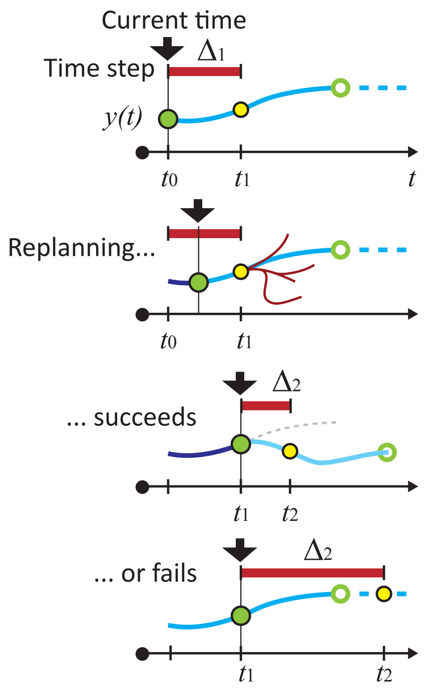

Fig. 3: Each replanning iteration chooses a time step (left), initiates a plan starting from the predicted future state (center), and either succeeds or fails. Upon success, the robot progresses on the new plan and the time step is contracted. Upon failure, the robot retains the original plan and the time step is increased.

the current trajectory is left unaltered and replanning repeats.

Note that the condition in Line 3 requires a decrease in C by some small amount $\epsilon > 0$. This simplifies later analysis by preventing the theoretical occurrence of an infinite number of infinitesimal cost improvements.

Lines 5 and 7 implement a simple exponential backoff strategy for choosing the time cutoff. This permits recovery from a local minimum of $V(x)$ in case several planning failures are encountered in sequence. Such strategies are widely used in protocols for handling network congestion [16], and there is a rough analogy between uncertainty in planning time and uncertainty in message delivery over an unreliable network. The idea is simple: if the planner fails, double the time step (Line 7). If it succeeds, contract the time step (Line 5). The constant 2/3 that we use in the contraction strategy does not need to be chosen particularly carefully, although if it is near 1 the system will be slow to adapt to easier planning subproblems. In practice, resetting Δ_{k+1} to a small value works well too. Figure 3 illustrates one iteration of the protocol.

3.4 Safety, Completeness, and Competitiveness Analysis

Here we will prove the straightforward result that ATS+EB keeps the robot in a feasible state. We will also prove that ATS+EB is probabilistically complete for static goals as long as the robot never reaches a state where the goal becomes unreachable.

Theorem 1 *ATS+EB will never drive the robot to an infeasible state.*

Proof By construction, the robot is driven only along F_∞ trajectories. Since F_∞ never leaves the feasible set for any t, the theorem holds. A corollary is that the robot is never driven into an inevitable collision state (ICS) as defined in [19]. □

Theorem 2 *If the environment is deterministic and perfectly modeled, and the goal is reachable from any state that is reachable from the start, then ATS+EB will find a solution trajectory in finite expected time.*

Proof Let $\mathcal{R}$ be the set of states reachable from the start, and let $P_x(t)$ denote a function describing the probability that the planner finds a solution trajectory within time t starting from state $x \in \mathcal{R}$. It must hold that $P_x(t)$ is nondecreasing with $P_x(0) = 0$. Integrating by parts, the expected running time of the planner $E[T(x)]$ starting from x is therefore

$$E[T(x)] = \int_{t=0}^{\infty} tP'_x(t)dt = \int_{t=0}^{\infty} (1 - P_x(t))dt. \quad (1)$$

Because expected running time was assumed finite in Section 3.1, we have $\lim_{t\to\infty} P_x(t) = 1$ for all x. Now define $P(t)$ as the minimum value of $P_x(t)$ over all $x \in \mathcal{R}$. We then define a bound T_{max} on expected running time using the equation:

$$T_{max} = \int_{t=0}^{\infty} (1 - P(t))dt. \quad (2)$$

First we will show that the time until a successful plan update has a finite expected value. The planning cutoff on the k'th unsuccessful iteration is $2^{k-1}\Delta_1$ and so the probabilty of updating the plan by the k'th iteration is at least $P(\Delta_k)$ where $\Delta_k = 2^{k-1}\Delta_1$. If the algorithm succeeds on the k'th iteration, then $(2^k - 1)\Delta_1$ time will have elapsed. Now define the partial sums

$$S_N = \Delta_1 P(\Delta_1) + \sum_{k=2}^{N} (2^k - 1)\Delta_1 (P(\Delta_k) - P(\Delta_{k-1})) \quad (3)$$

so that S_∞ is an upper bound on the expected time before ATS+EB finds a successful plan update. With some rearranging of sums we can show that

$$S_N \le \sum_{k=1}^{N-1} 2^k \Delta_1 (P(\Delta_N) - P(\Delta_k)). \tag{4}$$

We have that $P(\Delta_N) \le 1$ and $P(\Delta_k) \ge P(t)$ for $t \in [\Delta_{k-1}, \Delta_k]$, so that

$$(P(\Delta_{N+1}) - P(\Delta_k)) \le \frac{1}{\Delta_{k-1}} \int_{t=\Delta_{k-1}}^{\Delta_k} (1 - P(t))dt \tag{5}$$

Substituting this into (4), we have

$$\begin{aligned} S_N &\le \sum_{k=1}^{N} 4 \int_{t=\Delta_{k-1}}^{\Delta_k} (1 - P(t))dt \\ &= 4 \int_{t=0}^{\Delta_N} (1 - P(t))dt \end{aligned} \tag{6}$$

As N goes to ∞ we find that $S_\infty \le 4T_{max}$, which is finite.

The number N of plan updates needed to reach a global minimum is finite since the initial trajectory has finite cost, and each plan update reduces cost by a significant amount. So, the total expected running time of ATS+EB is bounded by $4NT_{max}$, which is finite. $\square$

We remark that the bound $4NT_{max}$ is extremely loose, and seemingly poor compared to the performance bound T_{max} of simply planning from the initial state until a solution is found. In practice, replanning iterations are comprised primarily of small, greedy advances in $C(y)$ that are planned quickly, along with a few planning queries of high difficulty corresponding to escaping deep local minima of C. The running time of ATS+EB will tend to be dominated by the few difficult queries.

3.5 Failure Cases

In systems that violates the assumptions of Theorem 2, ATS+EB may not converge to the global objective. We will distinguish between Type I and Type II errors, which can be thought of, respectively, as errors of local and global foresight:

1. Type I. Time-varying goals. For example, if the global minima of V might alternate quickly between two locally easy but globally difficult problems, then the algorithm will forever just make local progress and will thereby keep the time step short.
2. Type II. The robot is inadvertently driven into a dead end from which it cannot escape — a condition we call an *inevitable failure state*. This is a violation of the assumption of Theorem 2 that the goal can be reached by all states in $\mathcal{R}$.

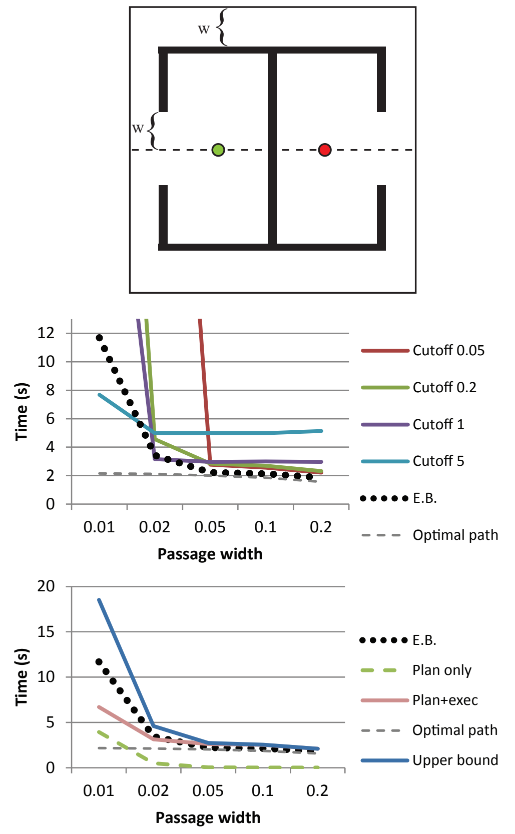

Fig. 4: (a) A 2D benchmark problem. Passage width, and hence, difficulty, is parameterized by w. (b) The time to reach the goal is sensitive to passage width, and short cutoffs (0.05 and 0.2) fail completely on difficult problems. The adaptive exponential backoff strategy (E.B.) is more consistent across problem variations. The execution time of the optimal path is shown for comparison. (c) The average time for adaptive time stepping compared to the theoretical bound (Bound) of Theorem 2. Offline planning time (Plan Only) and offline planning + execution (Plan+Exec) time are plotted for comparison.

We note that Type II errors do not exist in reversible systems, and that neither type of failures cause safety violations.

3.6 Completeness and Sensitivity Experiments

We evaluated the performance of the adaptive strategy against constant time stepping strategies on a static 2D benchmark across varying problem difficulties. Consider a unit square state space S where the state is subject to velocity constraints $||\dot{x}|| \le 1$. Obstacles partition the feasible space $F(t)$ into two "rooms" with oppositefacing doorways, which are connected by hallways (see Figure 4). The state must travel from $(0.3, 0.5)$ to $(0.6, 0.5)$, and the potential function $V(x)$ simply measures the distance to the goal. For replanning we use a unidirectional RRT planner [13], which, like other sample-based planners, is sensitive to the presence of narrow passages in the feasible space. We control the difficulty of escaping local minima by varying a parameter w, and set the hallway widths to w and the doorway widths to $2w$.

We measure performance as the overall time taken by the robot to reach the goal, averaged over 100 runs with different random seeds. If it cannot reach the goal after 120 s, we terminate the run and record 120 s as the running time. Experiments compared performance over varying w for constant cutoffs 0.05, 0.2, 1, and 5 s and the exponential backoff algorithm starting with $\Delta_1 = 0.1$ (performance was relatively insensitive to choice of Δ_1).

Figure 4 plots the performance of several strategies. It can be seen that shorter cutoffs are unreliable on hard problems because the planner is unable to construct paths that escape the local minimum of the initial room. On the other hand, longer cutoffs waste time on easier problems. The adaptive strategy outperforms constant cutoffs on most problems except the narrowest passage, in which it was 45% slower than the best constant cutoff. This extra time is consumed while it learns an appropriately large time step while stuck in the local minimum of the first room. Running times are also consistent with the theoretical bounds in Theorem 2, and are only slightly slower than planning before execution.

3.7 Assisted Teleoperation Experiments on a 6DOF Manipulator

Replanning interleaves planning and execution, so motion appears more fluid than a pre-planning approach. This is advantageous in human-robot interaction and assisted teleoperation applications where delays in the onset of motion may be viewed as unnatural. We implemented a teleoperation system for a dynamically simulated 6DOF Staübli TX90L manipulator that uses ATS+EB for real-time obstacle avoidance in assisted control. The robot is able to reject infeasible commands, follow commands closely while near obstacles, and does not get stuck in local minima like potential field approaches. We have also applied it to the real robot as well, using the adaptive time-stepping approach to handle variable network latency between the planner and the robot.

In this system, an operator controls a 3D target point (for example, using a mouse, joystick, or a laser pointer), and the robot is instructed to reach the point using its end effector. The robot's state space consists of configuration $\times$ velocity. Its configuration is subject to joint limit and collision constraints, and acceleration and velocity are bounded. The objective function for the planner is an unpredictably time-varying function $V(x, t)$ which measures the distance from the end effector to the target point. It should be noted that the robot is not trying to follow the *trajectory and speed* of the target point, but rather just the position. Nevertheless in this scheme the robot follows the speed of the trajectory quite closely in unconstrained regions. If user moves the target too quickly or through an obstacle, then the system will deviate to obey constraints.

Our underlying planner is a unidirectional variant of the SBL motion planner [21] that is adapted to produce dynamically feasible paths. We made the following adjustments to the basic algorithm:

- We extend the search tree by sampling extensions to stationary configurations sampled at random. The local planner constructs dynamically feasible trajectories that are optimal in obstacle free environments (a similar strategy was used in [5]). To do so, we use analytically computed trajectories that are time-optimal under the assumption of box-bounds on velocity and acceleration [10].
- For every randomly generated sample, we generate a second configuration using an inverse kinematics solver in order to get closer to the target.
- SBL uses a lazy collision checking mechanism that improves planning time by delaying edge feasibility checks, usually until a path to the goal is found. We delay edge checks until the planner finds a path that improves $C(y)$.
- To improve the fluidity of motion, we devote 20% of each time step to trajectory smoothing. We used the shortcutting heuristic described in [10] that repeatedly picks two random states on the trajectory, constructs a time-optimal segment between them, and replaces the intermediate portion of the trajectory if the segment is collision free.

The simulation environment is based on the Open Dynamics Engine rigid-body simulation package, where the robot is modeled as a series of rigid links controlled by a PID controller with feedforward gravity compensation and torque limits. The simulation does perform collision detection, but in our tests the simulated robot did not collide with the environment.

We simulated a user commanding a target to follow a circular trajectory that passes through the robot and obstacles (Figure 5). The circle has radius 0.8 m and a period of 20 s. The upper semicircle is relatively unconstrained and can be followed exactly. Targets along the lower semicircle are significantly harder to reach; at several points they pass through obstacles, and at other

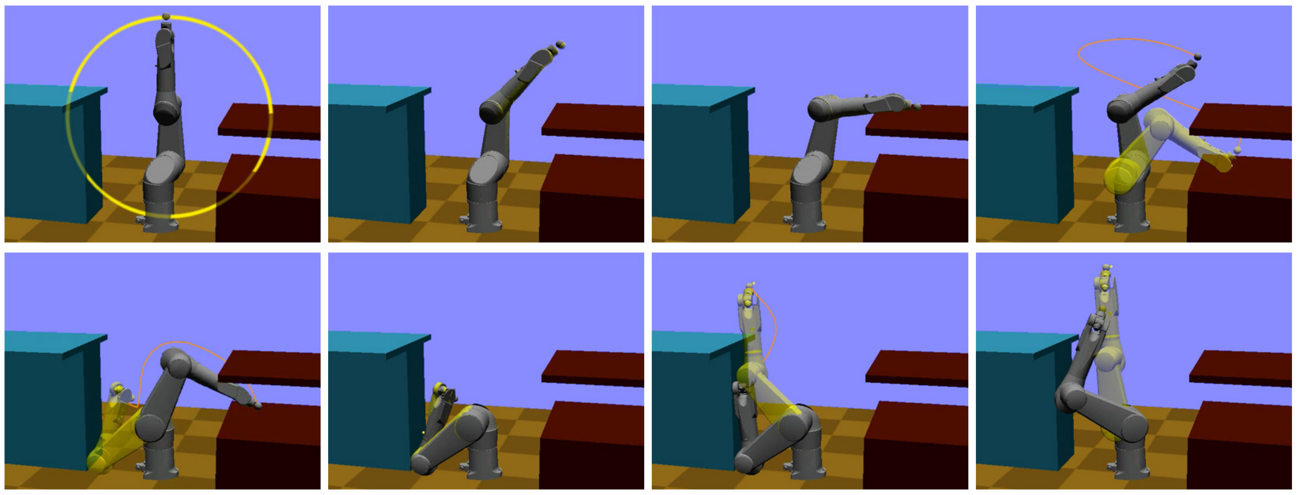

Fig. 5: A simulated Staübli TX90L manipulator is commanded in real time to move its end effector in a clockwise circle in a cluttered environment. The robot responds reactively to the target's motion. Along the upper semicircle, rapid replanning with a short time step allows the target to be followed closely. When obstacles are encountered on the lower semicircle, planning becomes more difficult. Adaptive time stepping gives the planner sufficient time to enter and escape deep narrow passages. The current plan is drawn in orange, and its destination configuration is drawn transparently.

points they require the robot to execute contorted maneuvers through narrow passages in the feasible space. Experiments show that ATS+EB can reach a large portion of the lower semicircle while tracking the upper semicircle nearly perfectly (Figure 6). Examining the mean squared error between the target and the actual end effector positions, we find that ATS+EB attains far lower error than long constant time steps and slightly lower error than short ones. We also compare mean squared objective function values for the robot's current path. This value can be viewed as the resulting MSE if the target stopped to let the robot catch up. We find that ATS+EB and longer time steps attain far better objective function values than shorter time steps.

The scheme is quite usable. In recent work we have studied how novice users perform reaching tasks using a variety of assisted teleoperation schemes [27]. Compared to several other schemes, the real-time planner enabled users to solve reaching tasks in cluttered environments twice as fast as other techniques while avoiding collision. Furthermore, it incurred no significant negative impressions of "loss of control" relative to reactive control.

We implemented the scheme on the real Staubli TX90L as well (Figure 1). In these figures a CAD model of the environment was constructed by hand and provided to the planner. In future work we hope to use a model captured from a laser scanner. The planner resides on an offboard PC and communicates with the robot's real-time controller via Ethernet. The network imposes a latency that typically varies between 10–20 ms. To handle latency both the controller and planner store a representation of the robot's trajectory on a global clock. The planner replans the trajectory from a predicted state along the trajectory as usual, adding the estimated latency to the allocated time step. The new trajectory segment is transmitted over Ethernet and the controller returns an "accept" or "reject" signal depending on whether it arrived completely before the beginning of the segment. If accepted, both the controller and planner update their trajectories. Otherwise, the new trajectory is rejected and the planner adjusts the estimated latency. We have tested the scheme with synthetic latencies of up to 1 s without incurring collisions or loss of dynamic feasibility.

4 Replanning in Unpredictable Environments

In contrast to the prior section, no safety guarantees are possible in general unpredictable environments. For example, there is nothing from stopping a determined adversary from crashing into cars on a highway. So the safety and completeness arguments of the prior section do not apply. Nevertheless replanning is highly practical and has proven successful in practice. We show that adaptive time stepping is very useful in this setting, because planning time can be reduced to produce urgent responses to imminent threats and increased to allow long-term exploration once the robot reaches a safer location.

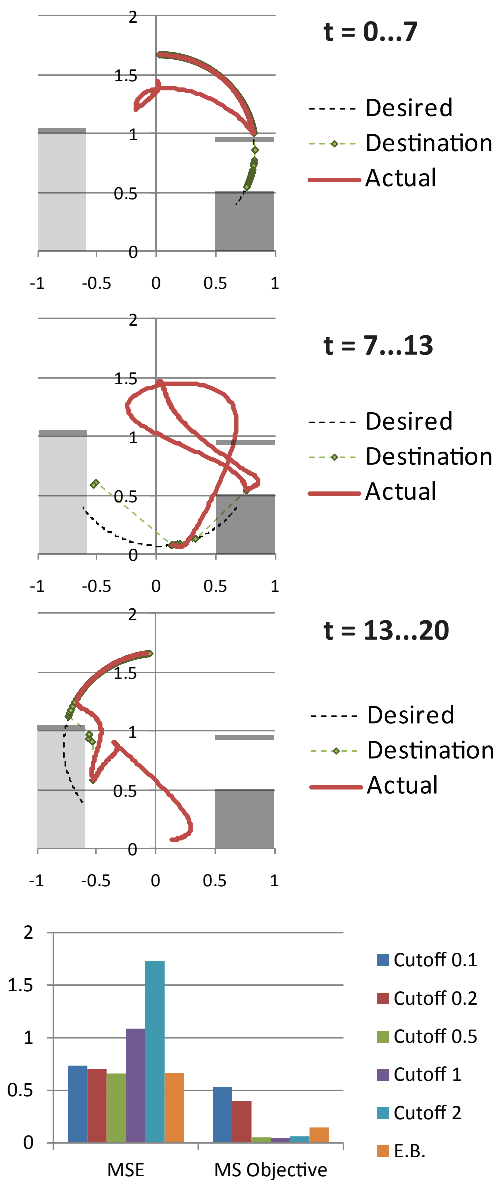

Fig. 6: Top: Traces of the end effector's desired position (Desired), the position at the current plan's destination configuration (Destination), and actual position as executed by the robot (Actual) for the simulation in Figure 5. Bottom: Comparison of mean squared errors for a variety of constant cutoffs and exponential backoff over five consecutive circles. The second column indicates the mean-squared objective function value for the robot's planned trajectories.

We first illustrate the principle on a variant of ATS+EB, called ATS+S, that optimizes a safety measure given bounded uncertainty, and allows for a high probability of replanning within a certain time limit — the time to potential failure, or TTPF — in which safety is guaranteed. To address the more practical task of achieving a goal while priortizing safety, we present a algorithm, ATS+C, that performs contingency planning of both optimistic and pessimistic trajectories. Experiments in simulated multi-agent scenarios demonstrate advantages of adaptive time stepping for both algorithms.

4.1 Conservative Safety-Seeking Algorithm

We assume that the robot has access to conservative bounds on the uncertainty of the environment, and let E_k denote the environment model estimated by the robot's sensors at k'th time step. Let $F(t; E_k)$ denote the feasible set with the current model, and let $\tilde{F}(t; E_k)$ denote the set of states that is guaranteed to be feasible at time t under the conservative uncertainty bounds. For example, if obstacle velocities are bounded, then one can consider a conservative space-time "cone" of possible obstacle positions that grows as time increases.

Definition. A trajectory $y(t)$ has *time to potential failure* (TTPF) T if it is safe for some duration T under conservative bounds on uncertainty. That is, $y(t) \in \tilde{F}(t; E_k)$ for all $t \in [t_k, t_k + T]$.

ATS+S implements a purely safety-seeking robot by adapting ATS+EB to use the TTPF as an optimization criterion. The robot will remain safe as long as planning can increase the TTPF faster than it is consumed. On iteration k, ATS+S performs the following steps:

1. Sense the current environment E_k and compute the TTPF T of the current trajectory.
2. Set $\Delta_k \leftarrow \max(\Delta_k, T/2)$.
3. Plan a path $\hat{y}$ that is F_∞ in $\tilde{F}(t; E_k)$ and that achieves a TTPF greater than T.
4. Reduce the time step if the new TTPF is at least $T + \Delta_k$; otherwise increase it.

The only significant changes to ATS+EB are in Steps 2 and 4. Step 2 picks a time step such that planning is completed before the TTPF. If the constraint is active it induces a sort of "panic mode" in which the robot tries repeatedly to increase TTPF with shorter time steps in the hopes that planning will succeed or that a potential hazard goes away. The factor of 1/2 here increases the chance of finding a feasible path, because it places the initial state of the planning problem away from the boundary of $\tilde{F}(t + \Delta_k, E_k)$. It also improves the chances of exploiting non-worst-case behavior of the environment. Step 4 requires that a significant increase in the TTPF induces a reduction in the time step, which ensures that time steps will be reduced only if the TTPF can be improved at a roughly constant rate. If the TTPF cannot be improved, then the time step increases. Although this might seem counterintuitive, it increases the chance of finding a large improvement via

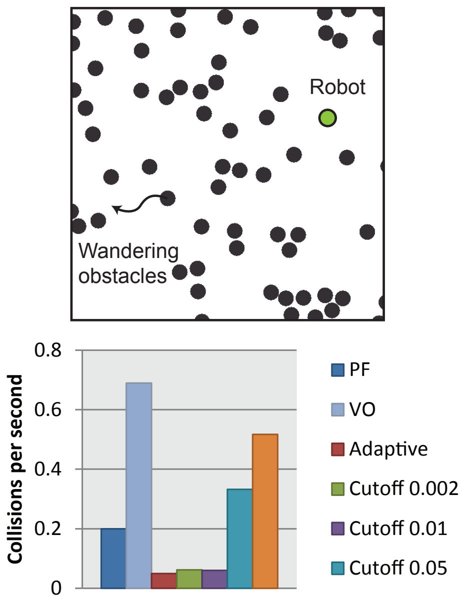

Fig. 7: Safety-seeking in a problem with 63 moving obstacles with unpredictable, randomized behavior but bounded velocity. Robot has bounded acceleration. Here the collision rate of adaptive time-stepping is slightly lower than the best constant time-stepping strategy, and is lower than the reactive potential field (PF) and velocity obstacle (VO) approaches [6,7]. Data was gathered over 100 s of execution.

a deep search. If no such improvement exists eventually the time step will be restricted by the constraint in Step 2 and the behavior becomes more myopic and reactive.

We tested ATS+S in a challenging obstacle avoidance simulation with dozens of circular obstacles in the unit square. The robot has maximum speed 1 and maximum acceleration 10. Obstacles have an unpredictable, nonadversarial "wandering" behavior, bounce elastically, and have a known maximum speed of 0.5. It should be noted that despite the robot's speed advantage, multiple obstacles may surround the robot and cause an unavoidable collision. Figure 7 compares collision rates between a reactive potential field (PF) of Ge and Cui (2002) [7], the Velocity Obstacles (VO) formulation of Fiorini and Shiller (1998) [6], ATS+S with adaptive time-step, and ATS+S with various constant time-steps. Results over 100 seconds of simulation demonstrate that adaptive time-stepping performs equally well as the best constant-time step and far better than PF and VO. We suspect this is the case because it is conservative, unlike VO, and it is not myopic, like PF.

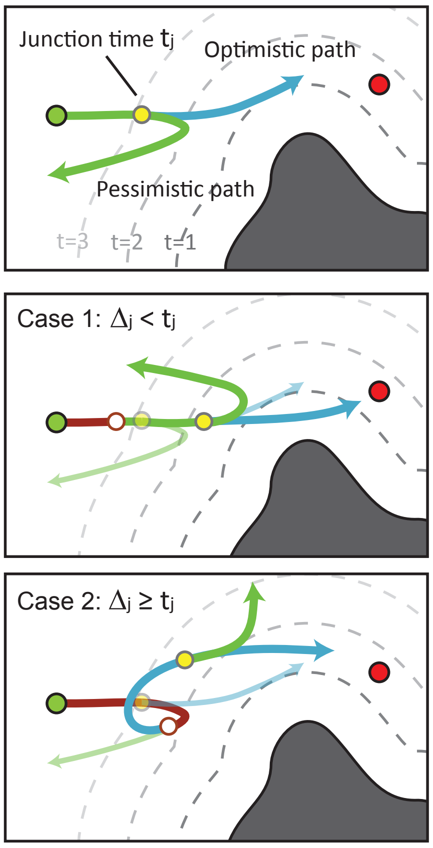

Fig. 8: Above, an illustration of the contingency plan data structure. Level sets (dotted lines) for the time-to-potential-failure function for a bounded-velocity obstacle (gray) are drawn. Below, the two main cases of the contingency planning algorithm are illustrated. In the first case a new plan is branched from the optimistic path. In the second case a new plan is branched from the pessimitic path.

4.2 A Contingency Replanning Algorithm

If the robot must also seek to optimize an objective $V(x)$, it must sacrifice some degree of safety in order to do so. Below we describe a contingency planning framework where the robot's path has a similarly high probability of safety as the above scheme, but the planner seeks to simultaneously increase the TTPF and makes progress towards reducing $V(x)$.

In our contingency planning algorithm ATS+C, the robot maintains both an optimistic and a pessimistic trajectory that share a common prefix (Figure 8). The role of the pessimistic trajectory is to optimize the TTPF, while the role of the optimistic trajectory is to encourage consistent progress toward the goal.

Pseudocode is listed in Figure 9. The pessimistic trajectory $y(t)$ is maintained and followed by default.

```
ATS+C. Contingency Replanning with Adaptive Time Steps
Initialization:
0a. $y(t) \leftarrow$ an initial trajectory starting from $t = 0$.
0b. $y^o(t) \leftarrow$ nil.
0c. Junction time $t_j \leftarrow 0$

Repeat for $k = 1, \ldots$:
1. Measure the current time $t_k$.
2. Pick a pessimistic and optimistic time limit $\Delta_k^p$ and $\Delta_k^o$. Let $\Delta_k = \Delta_k^p + \Delta_k^o$
3. If $t_j > t_k + \Delta_k$ (branch the new plan from the optimistic path)
4.    Plan an improved optimistic path starting from $y_o(t_j)$.
5.    Plan a pessimistic path $\hat{y}$ starting from $y_o(t_j + \Delta_k)$.
6.    If Line 5 is successful, then
7.       Set $y(t) \leftarrow y_o(t)$ for $t \leq t_j + \Delta_k$, and $y(t) \leftarrow \hat{y}(t)$ for $t \geq t_j + \Delta_k$.
8.       Set $t_j \leftarrow t_j + \Delta_k$.
9.    End
10. Otherwise, (branch the new plan from the pessimistic path)
11.    Plan an optimistic path starting from $y(t_k + \Delta_k)$.
12.    If successful, then
13.       Plan a pessimistic path $\hat{y}$ starting from $y_o(t_k + 2\Delta_k)$.
14.       If successful, then
15.          Set $y(t) \leftarrow y_o(t)$ for $t_k + \Delta_k \leq t \leq t_k + 2\Delta_k$, and $y(t) \leftarrow \hat{y}(t)$ for $t \geq t_k + 2\Delta_k$.
16.          Set $t_j \leftarrow t_k + 2\Delta_k$.
17.       End
18.    Otherwise,
19.       Plan a pessimistic path $\hat{y}$ starting from $y(t_k + \Delta_k)$.
20.       If successful, set $y(t) \leftarrow \hat{y}(t)$ for $t \geq t_k + \Delta_k$.
```

Fig. 9: Pseudocode for the contingency replanning algorithm.

The optimistic trajectory $y^o(t)$, if it exists, is identical to $y(t)$ until the "junction" time t_j. Each iteration of the replanning loop begins by establishing time limits for the optimistic and the pessimistic planners, with sum Δ_k (Line 2). Then a top-level decision is made whether to initiate the new plan from the optimistic or the pessimistic trajectory:

- *From the optimistic trajectory* (Lines 4–9). To continue progress along y^o after time t_j, the robot must generate a pessimistic trajectory that branches out of y^o at some time after t_j. An improvement to the optimistic plan is attempted as well.
- *From the pessimistic trajectory* (Lines 11–20). To progress toward the target, the planner will attempt to branch a new pessimistic and optimistic pair out of the current pessimistic trajectory at time $t_k + \Delta_k$. The new junction time will be $t_k + 2\Delta_k$. If this fails, the planner attempts an extension to the pessimistic path.

To improve the optimistic path, the planner constructs a path in the optimistic feasible space $F(t; t_c)$ based on the current environment model. A query is deemed successful if, after time limit Δ_k^o, $C(y^o)$ is improved over the current optimistic path if it exists, or otherwise over the current pessimistic path. If the query fails, y^o is left untouched. Pessimistic queries are handled exactly as in the prior section.

To choose planning times, we again use an adaptive time stepping scheme using the exponential backoff strategy of Section 3.3. Pessimistic and optimistic planning times are learned independently. We also make adjustments in case the candidate time step exceeds the finite TTPF of our paths. First, if we find that Δ_k exceeds the TTPF T of the pessimistic path, that is, failure may occur before planning is complete, we set $\Delta_k^p = T/2$ and $\Delta_k^o = 0$. Second, if we are attempting a modification to the optimistic trajectory, and the TTPF of the optimistic trajectory T^o is less than $t_j + \Delta_k$, then we scale Δ_k^p and Δ_k^o to attempt a replan before T^o (otherwise, the pessimistic replan is guaranteed to fail).

4.3 Failure Cases

For unpredictable environments in general, there are no guarantees on safety. there are again two types of failure that may lead to violations of safety (as well as completeness).

1. Type I. States for which escaping failure is difficult for the planner. Consider a state on the verge of becoming an inevitable failure state. Without enough time to successfully plan an escaping path, there is no escape. Moreover some escape plans may require an intractably large number of contingencies for possible future events.
2. Type II. Inevitable failure states.

In general, addressing Type I errors may be a simple matter of increasing computational power or tuning replanning techniques, while Type II errors seem to require larger architectural changes to the algorithms to incorporate global knowledge of the problem structure. A major benefit of sample-based replanning is that a factor n increase in computational power results in a sharp reduction in Type I failure rate simply because if n independent planning processes have individual failure probability p, the probability that they all fail simultaneously is p^n. Such an approach would be quite powerful should it be able to exploit parallel computing in the underlying planner, for example in recent work in sample-based planning on graphics hardware [18].

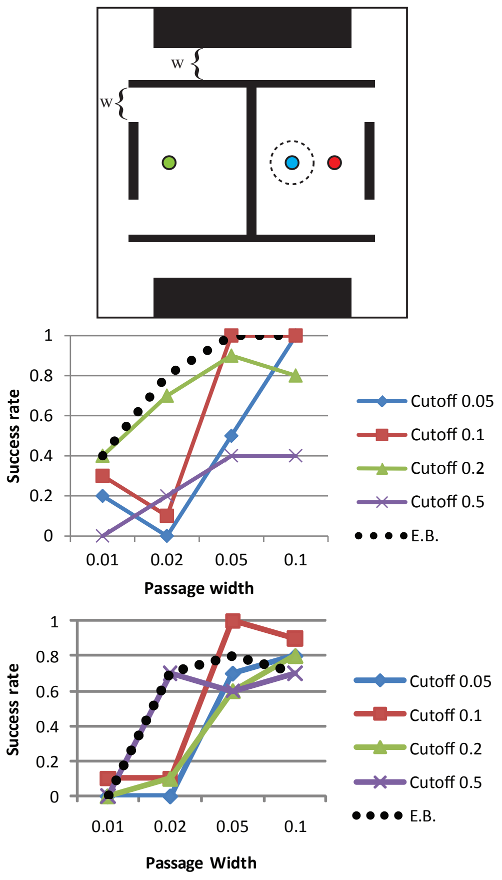

Fig. 11: (a) A pursuit-evasion problem. Narrow passages, and hence, difficulty, are parameterized by w. The evader (green) must try to reach the target (red) within 10 s while avoiding the pursuer (blue), with capture radius 0.05. Success rates for (b) nonadversarial and (c) adversarial pursuer behaviors. In the nonadversarial case the pursuer is allowed to pass through obstacles. A shorter time step (Cutoff 0.2) performs well in the nonadversarial case, but a longer time step (Cutoff 0.5) performs better in the adversarial case. The adaptive strategy works well in both cases.

Increases in computational power are also likely to decrease the rate of Type II errors, but here the analysis is much more complex. So, practical methods for detecting and avoiding inevitable failure states would be extremely valuable as a subject of future work.

4.4 Pursuit-Evasion Simulation Experiments

Figure 10 depicts how ATS+C builds contingency plans in a planar pursuit-evasion scenario in the unit square in which the evader and pursuers move at maximum speeds 1 and 0.5, respectively, and the evader's maximum acceleration is 10. The evader's goal is to reach a target while being chased by the pursuers. The evader is able to make progress on the optimistic plan only by building up a large distance between itself and the pursuers. In experiments we also observe that this can be accomplished for high-speed, low-acceleration pursuers.

Experiments were also conducted to evaluate how the time stepping affects the evader's performance in an environment in which the evader must traverse a narrow passage to reach its target (Figure 11). The evader treats the pursuer as an unpredictable obstacle with bounded velocity and uses ATS+C. The evader's conservative model of $\tilde{F}(t, E_k)$ does not consider walls to be impediments to the pursuer's possible movement. Mere survival is not challenging (in all experiments survival over 10s was over 90%), but reaching the target requires the evader to choose a different hallway than the pursuer. We tested a nonadversarial pursuer behavior in which it "wanders" with velocity varying according to a random walk, and is allowed to pass through walls. Figure 11(b) shows that in this case, the success rate is highly dependent on problem difficulty, and no constant cutoff performs uniformly well across all width variations. Similar variations were found using an adversarial pursuer, in which the pursuer treats the evader as an unpredictably moving target and uses ATS+EB to reach it. (Figure 11(c)). The adaptive strategy performed nearly as well as the best constant cutoff across both problem variations.

4.5 Avoiding Unpredictable Obstacles in Indoor Navigation

Finally we apply ATS+C to a more complex indoor navigation problem where the robot has a coarse model of worst-case obstacle behavior. Coarse models are important in multi-agent settings because the behavior and intentions of pedestrians or other robots can be complex and difficult to predict.

Figure 12 depicts a run of our simulation with 10 agents in a map of Intel Research Lab taken from the OpenSLAM dataset [22]. The map is scaled to the unit square. The agents traverse a roadmap of the environment by choosing a goal node at random, and moving along the shortest path toward the goal. Their accelerations are also perturbed by Gaussian noise. Agents dwell at their goals for a randomly chosen period of time.

Rather than have the robot model this process, the robot only knows the agents' velocity and acceleration bounds ($0.5\,\mathrm{s}^{-}1$ and $10\,\mathrm{s}^{-}2$, respectively). The robot is taken to have the same acceleration and velocity bounds as before, and it senses all obstacle positions and velocities within a radius of 0.3.

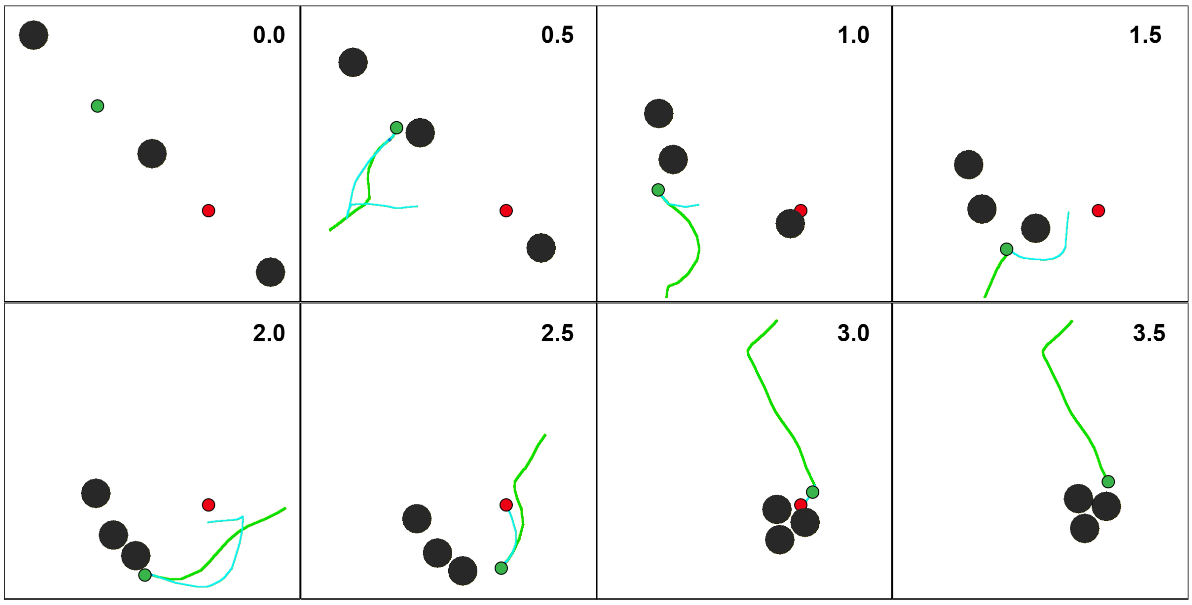

Fig. 10: Snapshots taken at half-second intervals from a pursuit-evasion simulation in the unit square. Three pursuers (grey circles) seek the evader (green) greedily at half the speed of the evader. The evader knows the pursuers' velocity bound but not their behavior. The evader replans a pessimistic path (green) to avoid the pursuers in the worst case, and replans an optimistic path (cyan) in order to reach the target (red). Both paths share a common prefix.

As Figure 12 depicts, the robot takes conservative motions (e.g., moving into rooms) when necessary to avoid obstacles. While this behavior might be excessively conservative in the presence of pedestrians that themselves avoid the robot, it may be perfectly reasonable for avoiding rushing crowds or unintelligent robots. One of the strengths of the ATS+C framework is that sample-based planners can easily incorporate better predictive models of agent-agent interactions as they become available.

5 Conclusion

Real-time replanning architectures have been slow to catch on in large part by the responsiveness/completeness dilemma that is posed in the presence of runtime variance and unpredictable environments. This paper introduces an adaptive time-stepping approach to addresses this dilemma by tolerating and exploiting run-time variance by estimating an appropriate time step on-the-fly. Safety and completeness can be guaranteed in deterministic dynamic environments, and simulation results demonstrate a low failure rate in challenging unpredictable environments. Experiments on shared control for an industrial robot arm and on simulated pursuit-evasion examples suggest that replanning may be a viable mechanism for real-time navigation and obstacle avoidance in dynamic environments. Additional videos of simulations and experiments can be found on the web at http://www.iu.edu/∼motion/realtime.html.

An important question for future work is the problem of path optimality, which introduces a third performance criterion in addition to responsiveness and completeness. The impact of tradeoffs between replanning suboptimal paths quickly or optimal plans slowly is poorly understood. Another important question is detecting and avoiding inevitable failure states in non-reversible systems, which is needed to preserve hard guarantees on safety in unpredictable scenarios.

References

1. F. Allgöwer and A. Zheng. *Nonlinear Model Predictive Control (Progress in Systems and Control Theory)*. Birkhäuser, Basel, 2000.
2. S. J. Anderson, S. C. Peters, K. D. Iagnemma, and T. E. Pilutti. A unified approach to semi-autonomous control of passenger vehicles in hazard avoidance scenarios. In *Proc. IEEE Int. Conf. on Systems, Man and Cybernetics*, pages 2032–2037, San Antonio, TX, USA, 2009.
3. K. Bekris and L. Kavraki. Greedy but safe replanning under kinodynamic constraints. In *Proc. IEEE Int. Conference on Robotics and Automation (ICRA)*, pages 704–710, Rome, Italy, April 2007.
4. J. Bruce and M. Veloso. Real-time randomized path planning for robot navigation. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, October 2002.

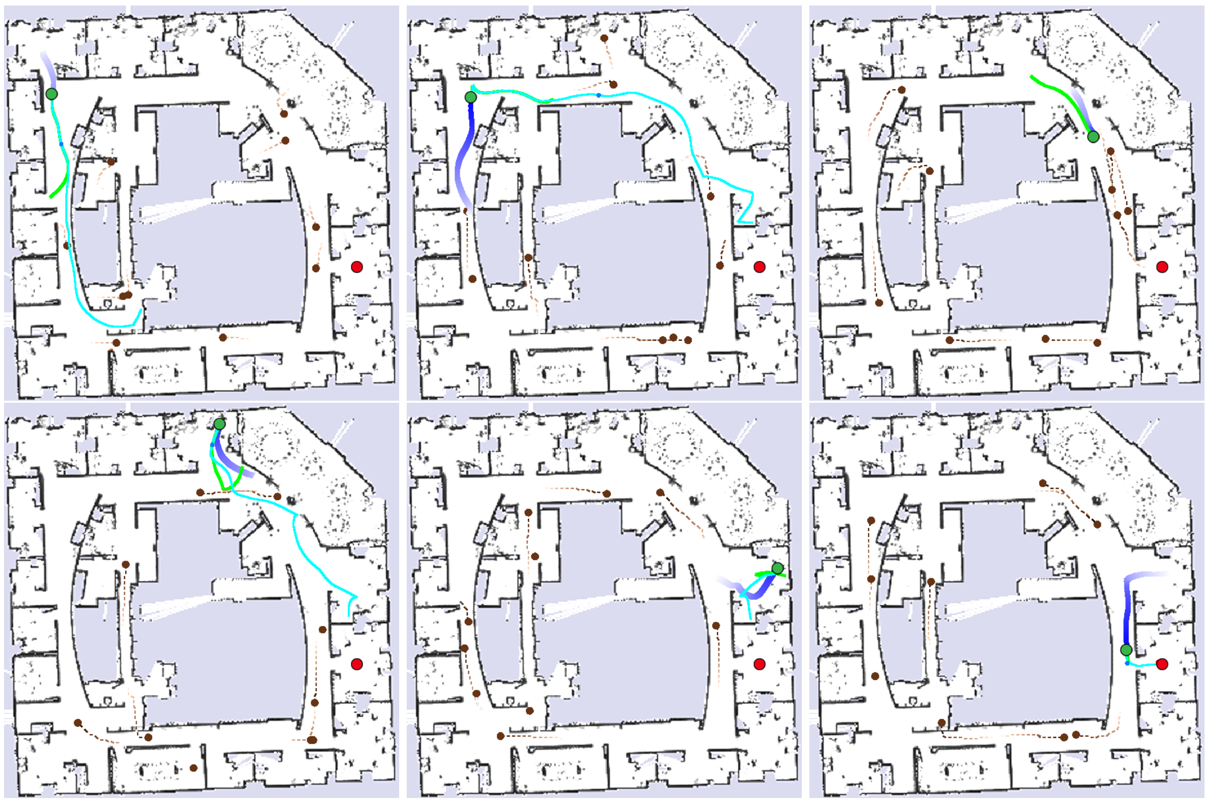
Fig. 12: A robot (green circle) navigates a building to a target (red circle) among unpredictable moving obstacles using ATS+C. The robot only has knowledge of obstacle acceleration and velocity bounds. Its pessimistic plan is shown in green, and the optimistic plan is in cyan. First the robot plans a route down the left hallway (frame 1). The robot finds an obstacle coming up the hallway and plans an alternate route (frame 2). After proceeding down the hallway, the robot backtracks into a room to avoid the obstacles crossing in the hall (frames 3 and 4). The robot avoids one other obstacle and proceeds to the goal (frames 5 and 6).

5. E. Feron, E. Frazzoli, and M. Dahleh. Real-time motion planning for agile autonomous vehicles. In *AIAA Conference on Guidance, Navigation and Control*, Denver, USA, August 2000.
6. P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *Int. Journal of Robotics Research*, 17(7):760–772, 1998.
7. S. S. Ge and Y. J. Cui. Dynamic motion planning for mobile robots using potential field method. *Auton. Robots*, 13:207–222, November 2002.
8. D. K. Grady, K. E. Bekris, and L. E. Kavraki. Asynchronous distributed motion planning with safety guarantees under second-order dynamics. In *Algorithmic Foundations of Robotics IX*, pages 53–70. Springer, Berlin / Heidelberg, 2011.
9. K. Hauser. Adaptive time stepping in real-time motion planning. In *Workshop on the Algorithmic Foundations of Robotics*, 2010.
10. K. Hauser and V. Ng-Thow-Hing. Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. In *Proc. IEEE Int. Conference on Robotics and Automation (ICRA)*, Anchorage, USA, 2010.
11. D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Kinodynamic motion planning amidst moving obstacles. *Int. J. Rob. Res.*, 21(3):233–255, Mar 2002.
12. M. Kallmann and M. Mataric. Motion planning using dynamic roadmaps. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Apr. 2004.
13. S. LaValle and J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 473–479, 1999.
14. M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun. Anytime dynamic a*: An anytime, replanning algorithm. In *In Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS*, 2005.
15. D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.
16. R. M. Metcalfe and D. R. Boggs. Ethernet: distributed packet switching for local computer networks. *Commun. ACM*, 19:395–404, July 1976.
17. D. J. Musliner, E. H. Durfee, and K. G. Shin. Circa: A cooperative intelligent real-time control architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23:1561–1574, 1993.
18. J. Pan, C. Lauterbach, and D. Manocha. g-planner: Real-time motion planning and global navigation using gpus. In *AAAI Conf, on Artificial Intelligence*, 2010.
19. S. Petti and T. Fraichard. Safe motion planning in dynamic environments. In *IEEE International Conference on*

Intelligent Robots and Systems (IROS), pages 3726–3731, 2005.
20. I. Ross, Q. Gong, F. Fahroo, and W. Kang. Practical stabilization through real-time optimal control. In *American Control Conference*, page 6 pp., jun. 2006.
21. G. Sánchez and J.-C. Latombe. On delaying collision checking in PRM planning: Application to multi-robot coordination. *Int. J. of Rob. Res.*, 21(1):5–26, 2002.
22. C. Stachniss, P. Beeson, D. Hähnel, M. Bosse, J. Leonard, B. Steder, R. Kümmerle, C. Dornhege, M. Ruhnke, G. Grisetti, , and A. Kleiner. Laser-based slam datasets and benchmarks.
23. C. Stachniss and W. Burgard. An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 508–513, 2002.
24. A. Stentz. The focussed d* algorithm for real-time replanning. In *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.
25. J. van den Berg, D. Ferguson, and J. Kuffner. Anytime path planning and replanning in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2366 – 2371, May 2006.
26. J. van den Berg and M. Overmars. Roadmap-based motion planning in dynamic environments. *IEEE Trans. Robot.*, 21(5):885–897, October 2005.
27. E. You and K. Hauser. Assisted teleoperation strategies for aggressively controlling a robot arm with 2d input. In *Proc. Robotics: Science and Systems*, Los Angeles, USA, July 2011.
28. M. Zucker, J. Kuffner, and M. Branicky. Multipartite rrts for rapid replanning in dynamic environments. In *Proc. IEEE Int. Conf. Robotics and Automation*, April 2007.