2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Anytime, Dynamic Planning in High-dimensional Search Spaces

Dave Ferguson
Intel Research Pittsburgh
4720 Forbes Avenue
Pittsburgh, PA 15213
dave.ferguson@intel.com

Anthony Stentz
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
tony@cmu.edu

***Abstract*— We present a sampling-based path planning and replanning algorithm that produces anytime solutions. Our algorithm tunes the quality of its result based on available search time by generating a series of solutions, each guaranteed to be better than the previous ones by a user-defined improvement bound. When updated information regarding the underlying search space is received, the algorithm efficiently repairs its previous solution. The result is an approach that provides low-cost solutions to high-dimensional search problems involving partially-known or dynamic environments. We discuss theoretical properties of the algorithm, provide experimental results on a simulated multirobot planning scenario, and present an implementation on a team of outdoor mobile robots.**

I. INTRODUCTION

Path planning for real-world robotic systems involves overcoming several challenges. In particular, the information available concerning real-world environments is usually incomplete or imperfect, the time available for planning is usually limited, and the environment itself is often dynamic. These challenges are further exacerbated by the complexity of the search spaces commonly involved (for instance, when planning for complicated robotic vehicles or teams of vehicles).

For lower-dimensional problems involving fairly simple single robots, several discrete planning approaches have been devised for coping with these challenges. These algorithms plan over graph-based representations of the search problem and use heuristics to focus their search towards the most promising areas of the search space. Two very common such algorithms for planning in known environments are Dijkstra's search [1] and A* [2].

To cope with imperfect information or dynamic environments, efficient replanning variants of these algorithms have been developed that correct previous solutions based on updated information [3], [4]. These algorithms maintain optimal solutions for a fraction of the computation required to generate such solutions from scratch. To cope with very limited deliberation time, suboptimal or anytime variants of these algorithms have been developed. These algorithms generate an initial, possibly highly-suboptimal solution very quickly, then, in the case of the anytime approaches, concentrate on improving this solution while deliberation time allows [5], [6]. To deal with both features, an anytime, replanning algorithm called *Anytime Dynamic A** has recently been developed that effectively provides both anytime and replanning capabilities [7].

However, all of these discrete approaches are limited by their reliance on an underlying search graph that encodes the search space (either explicitly or implicitly). For very high-dimensional problems such as those we address in this paper, generating and planning over such a graph can become prohibitively expensive. Further, they all rely on heuristics to guide their searches; in many high-dimensional problems, generating useful heuristics may not be possible.

In response to these limitations, researchers have developed sampling-based analogs that generate solutions in very high-dimensional search spaces. One of the most widely-used of these algorithms is the Rapidly-exploring Random Tree (RRT) algorithm [8]. This algorithm grows a search tree out from an initial position in the search space (the initial configuration) and uses random sampling of the search space to bias the growth of this tree towards unexplored regions. Consequently, it explores the space extremely efficiently and can be easily modified to focus efforts towards a particular goal configuration. Because randomization is used to grow the tree, the algorithm copes well with both very high-dimensional search spaces and very large branching factors.

Unfortunately, the RRT algorithm provides no replanning capability and, although it generates feasible solutions, it cannot control or improve over time the quality of these solutions. Recently, two modified versions of this algorithm were developed with these individual features [9], [10]. Yet some of the most interesting real-world problems are those that are *both* dynamic or involve partially-known information (and thus require replanning) *and* involve complex, non-uniform cost search spaces (so that anytime performance has great benefit).

In this paper, we present a sampling-based planning algorithm that provides both anytime and replanning capabilities. Our algorithm, *Anytime Dynamic RRTs*, continually improves its solution while deliberation time allows, and efficiently repairs its solution when new information is received. It is thus ideally suited to several real-world problems involving complex search spaces and partially-known information.

We begin by introducing our motivating problem of constrained exploration. We then review the original RRT algorithm and two recent algorithms, Dynamic RRTs and Anytime RRTs, that provide replanning and anytime per-

1-4244-0602-1/07/$20.00 ©2007 IEEE.

formance. We then introduce our novel algorithm, Anytime Dynamic RRTs, and provide intuitive examples and experimental results demonstrating its benefits in both simulation and on a team of outdoor robotic vehicles.

II. Coordinated Motion Planning

Consider a team of robots tasked with exploring key areas in some outdoor environment. For safety or communication reasons, assume we require that the robots always remain within line-of-sight communication of each other at all times. This is known as the *Constrained Exploration Problem* [11]. In order to successfully complete such a task, the actions of each robot need to be tightly-coordinated. When planning for such tight-coupled teams, it is often necessary to plan actions for the entire team at once, rather than each robot individually. As the number of robots in the team increases, this becomes an increasingly complex planning problem. In fact, both the dimensionality and the complexity of the search space quickly overwhelm discrete planning algorithms. Sampling-based approaches are attractive alternatives, since they are not nearly as encumbered by complex, high-dimensional search spaces.

III. Sampling-based Planning

Sampling-based planning algorithms search through the continuous search space or configuration space rather than a discrete representation of the space, and they use intelligent sampling techniques to efficiently explore large search spaces. In particular, the RRT algorithm is very effective for planning paths through high-dimensional search spaces, and has been applied to a wide range of motion planning tasks [8], [12].

A. Rapidly-exploring Random Trees (RRTs)

The standard RRT algorithm is shown in Figure 1. This algorithm grows a search tree out from the initial robot configuration q_{start} towards the goal configuration q_{goal}. To grow the tree, first a target configuration q_{target} is randomly selected from the configuration space using the function *ChooseTarget*. Then, a *NearestNeighbor* function selects the node $q_{nearest}$ in the tree closest to q_{target}. Finally, a new node q_{new} is created in an *Extend* function by growing the tree some distance from $q_{nearest}$ towards q_{target}. If extending the tree towards q_{target} requires growing through an obstacle, no extension occurs. This process is repeated until the tree grows to within some user-defined threshold of the goal (line 3). A very nice property of this method of construction is that the tree growth is strongly biased towards unexplored areas of the configuration space. Consequently, exploration occurs very quickly. The RRT algorithm can also be significantly more efficient if the search is focused towards the goal (lines 10 and 11): with probability $1-p$, q_{target} is randomly selected; with probability p, q_{target} is set to the goal configuration.

```
InitializeRRT(rrt T)
 1  T.add(q_start);
GrowRRT(rrt T)
 2  q_new = q_start;
 3  while (Distance(q_new, q_goal) > distance-threshold)
 4    q_target = ChooseTarget();
 5    q_nearest = NearestNeighbor(q_target, T);
 6    q_new = Extend(q_nearest, q_target, T);
 7    if (q_new ≠null)
 8      T.add(q_new)
ChooseTarget()
 9    p = RandomReal([0.0, 1.0]);
 10   if (p < goal-sampling-prob)
 11     return q_goal;
 12   else
 13     return RandomConfiguration();
Main()
 14 InitializeRRT(tree);
 15 GrowRRT(tree);
```

Fig. 1. The RRT Algorithm.

B. Dynamic RRTs

The standard RRT algorithm requires perfect initial information concerning the environment; if new information is received after it has generated its original plan, a new plan has to be generated from scratch. In mobile robot navigation scenarios involving real-world environments, such information is received frequently (e.g. from onboard sensors) and replanning from scratch can be very computationally expensive and in many cases intractable.

A recent extension to the RRT algorithm allows efficient repair of the tree when changes are made to the configuration space [9]. Known as *Dynamic RRTs*, this approach first generates a standard RRT from an initial configuration to a goal configuration. When changes occur to the configuration space, all the parts of the RRT that are invalidated by these changes are trimmed from the search tree. The remaining tree is then grown out until the goal configuration is reached once more.

By maintaining as much as possible of the previous tree when changes are observed, the Dynamic RRT algorithm generates new solutions much more efficiently than replanning from scratch. For more details on this algorithm, its application to planning in partially-known environments, and its advantage over competing approaches, see [9].

C. Anytime RRTs

One of the most significant limitations of single-shot sampling-based algorithms is their inability to incorporate cost into their searches in order to produce high quality solutions. As a result, they can often produce very expensive or suboptimal paths. Another RRT-based algorithm known as *Anytime RRTs* addresses this limitation [10]. Anytime RRTs generate a series of RRTs, with each RRT after the first using novel search techniques to produce a new solution that is guaranteed to be less expensive than any of the previous

solutions by a user-defined improvement factor ϵ_f. Thus, a valid solution is returned as quickly as by the standard RRT algorithm, but the quality of this solution is then improved while deliberation time allows. The resulting algorithm provides similar benefits to discrete anytime algorithms but plans over much larger, higher-dimensional search spaces.

The Anytime RRT algorithm first generates an initial RRT without considering costs. It then records the cost $\mathcal{C}_s$ of the solution returned by this RRT. Next, it generates a new RRT and ensures that the solution produced by this new RRT is better than the previous solution by limiting the nodes added to the tree to only those that could possibly contribute to a solution with a lower overall cost than $\mathcal{C}_s$. This cost bound $\mathcal{C}_s$ can also be multiplied by some factor $(1 - \epsilon_f)$, where $0 \leq \epsilon_f < 1$, to ensure that the next solution will be at least ϵ_f times less expensive than the previous solution. The algorithm then updates the cost bound $\mathcal{C}_s$ based on the cost of the new solution and repeats the process until time for planning runs out.

This algorithm has a number of nice properties, such as guarantees on the improvement in path quality between successive solutions. It also produces much better solutions than standard RRTs in non-uniform cost search spaces and has been applied to the constrained exploration problem in known environments [10].

IV. Anytime Dynamic RRTs

The Anytime RRT algorithm generates a series of improving RRTs and the Dynamic RRT algorithm efficiently repairs RRTs when changes are made to the configuration space. Path planning in the real world, however, often requires an algorithm with both features. Such an approach would provide a comprehensive sampling-based framework capable of dealing with the most challenging, most realistic real-world problems, where we have imperfect information and limited time, and we are concerned with generating and executing high quality solutions.

To this end we have developed an algorithm called *Anytime Dynamic Rapidly-exploring Random Trees* (Anytime Dynamic RRTs) that couples these approaches. First, the Anytime RRT algorithm provides the best solution possible within the time available for planning. This solution is then executed by the agent or team, and can be further improved upon by the Anytime RRT algorithm while the agent or team is in motion. When changes are observed in the environment, these changes may invalidate the current solution path being executed. However, by maintaining the search tree associated with this solution, the Dynamic RRT algorithm can efficiently repair the tree and generate a new solution. Once this new solution has been produced, the Anytime RRT algorithm can improve it, and the entire process can continue until the goal is reached.

This coupling has significant advantages. First, we can take advantage of the low-cost solutions generated by the Anytime RRT algorithm, and can continually improve over time the quality of the solution being executed. Second, when changes are observed, rather than having to start the entire anytime process over from scratch, we can maintain the current low-cost solution and repair it. This results in a much better solution than we would otherwise have generated from scratch, and since we are only repairing the portions of the current solution that have been invalidated, this new solution is generated very quickly. Further, since we are continually improving the solution by generating new trees through the configuration space, the overall amount of trimming required when new information is received is minor. Thus, by combining both anytime and replanning capabilities, we provide better solutions than either approach alone, and we provide these solutions with less computation.

An example application of the approach to a single agent planning problem is shown in Figure 2. In this example, the agent plans an initial path in an anytime fashion, then as it is about to traverse its least-costly path, it observes a new obstacle that invalidates this path. It then updates its solution to take into account this obstacle. As with the Dynamic RRT algorithm, it can be beneficial for Anytime Dynamic RRTs to search backwards from the goal configuration so that the tree can be efficiently trimmed when new information is received in the vicinity of the agent or team of agents.

Pseudocode of the *Main* function of the Anytime Dynamic RRT algorithm is provided in Figure 3. Two trees are used: the tree T is used to construct improved solutions over time and the tree R is used to store the best solution generated thus far. When changes take place, the affected nodes in R are trimmed and R is repaired to generate a new solution. The *GrowRRT* function used during the anytime planning phase (line 4) and within the *RegrowRRT* function is the anytime version used in the Anytime RRT algorithm. This combined algorithm has number of nice theoretical properties, including bounds on the cost of the improved solutions, which are discussed in [13].

V. Anytime Dynamic RRT Results

We have applied Anytime Dynamic RRTs to constrained exploration in non-uniform cost, partially-known environments and in environments with dynamic elements (such as moving obstacles or other agents). To analyse its performance, we simulated traverses for a team of 3 agents navigating across a series of random environments containing both communication obstacles and areas of non-uniform traversability cost. During execution, new information was received concerning the environment and the solutions were updated. We describe three different testing scenarios below.

A. Constrained Exploration in Partially-known Environments

For performing constrained exploration in partially-known environments, initial paths were planned for the team through these environments that incorporated all initial information available to the team. During execution, the agents observed navigation obstacles that were randomly placed in the environment. When these obstacles invalidated the current solution, the team was forced to generate a new solution. We

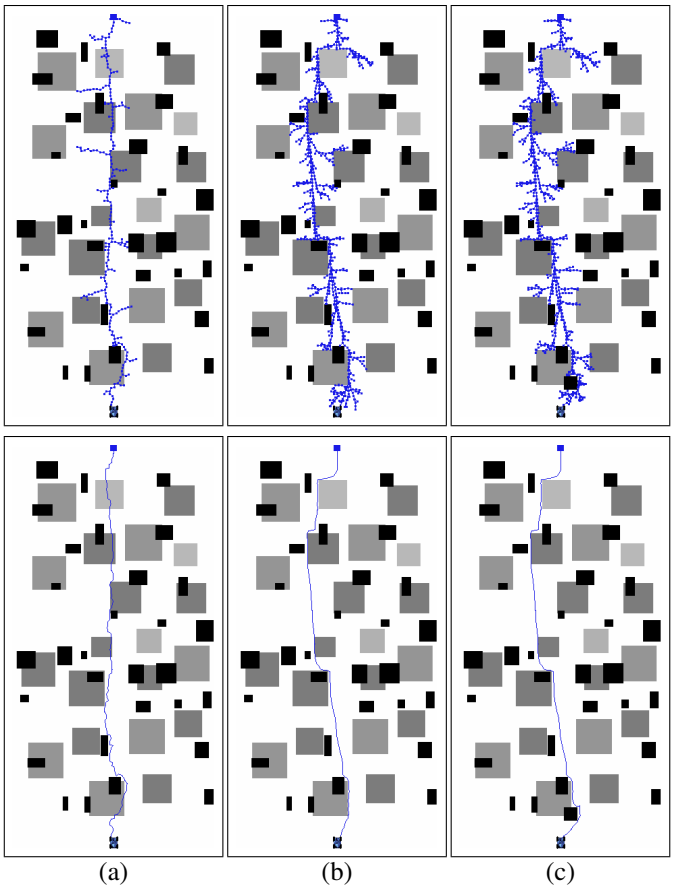

Fig. 2. Anytime Dynamic RRTs used for single robot path planning. The robot begins at the bottom of the environment and plans to its goal, the square at the top. Shaded regions represent higher-cost areas to traverse; black regions represent obstacles. The top images show the trees while the bottom images show the solution paths. (a) Initial RRT generated without cost consideration. (b) Final RRT generated in initial planning time, using costs of previous solutions and nodes of the current tree to bias the growth of the current tree. (c) A new obstacle is discovered in front of the agent, invalidating the final solution path and a portion of the search tree. The tree is repaired to account for the new obstacle and a new solution is generated. The majority of the low-cost solution remains unaffected.

Main()
1 $T.d_b = 1$; $T.c_b = 0$; $T.\mathcal{C}_s = \infty$;
2 forever
3 **ReinitializeRRT**(T);
4 $T.\mathcal{C}_n =$ **GrowRRT**(T);
5 if ($T.\mathcal{C}_n \neq$ **null**)
6 $R = T$;
7 **PostCurrentSolution**(R);
8 $R.\mathcal{C}_s = T.\mathcal{C}_s$; $R.d_b = T.d_b$; $R.c_b = T.c_b$;
9 $T.\mathcal{C}_s = (1 - \epsilon_f) \cdot T.\mathcal{C}_n$;
10 $T.d_b = T.d_b - \delta_d$;
11 if ($T.d_b < 0$)
12 $T.d_b = 0$;
13 $T.c_b = T.c_b + \delta_c$;
14 if ($T.c_b > 1$)
15 $T.c_b = 1$;
16 for each new obstacle o in configuration space
17 **InvalidateNodes**(R, o);
18 if significant obstacle changes were observed
19 increase $R.\mathcal{C}_s$, $R.d_b$, and $T.d_b$; decrease $R.c_b$ and $T.c_b$;
20 if solution path of R contains an invalid node
21 **RegrowRRT**(R);
22 **PostCurrentSolution**(R);
23 $T.\mathcal{C}_s = (1 - \epsilon_f) \cdot R.\mathcal{C}_n$;

Fig. 3. The Anytime Dynamic RRT Algorithm: Main function

compared Anytime Dynamic RRTs to the original RRT algorithm, Dynamic RRTs, and Anytime RRTs. Each approach was allowed to run for a total initial planning time of 10 seconds (on a 1.5 GHz Powerbook G4), and each individual RRT was allowed to take up to 2 seconds. During execution, each approach planned from a configuration three steps ahead of it on the current path. The time taken to traverse each edge was set to 0.5 seconds, so that a total of 1.5 seconds was available for planning before the best solution was taken and the first three steps in this path were executed. Figure 4 plots the average cost of the team traverses following each of the four approaches. For these experiments, the Anytime Dynamic RRT and Anytime RRT approaches used an ϵ_f value of 0.1. When the solution was invalidated due to newly-observed obstacles, the Anytime RRT approach reset its C_s, d_b, and c_b values, while the Anytime Dynamic RRT approach increased these values by an amount that was dependent on how much of the search tree and solution was affected. For details, see [13]. We employed Euclidean distance as the heuristic used by the algorithm and our available extensions were straight-line segments for each agent.

By combining the anytime capability of Anytime RRTs and the replanning capability of Dynamic RRTs, Anytime Dynamic RRTs provide better solutions than either of these approaches, and much better solutions than the standard RRT algorithm. Both Anytime RRTs and Anytime Dynamic RRTs outperform the non-anytime approaches because they take the cost of the solution into account when searching. Further, they use information from previous searches to help guide and restrict the next search towards cheaper solutions. Anytime Dynamic RRTs produce lower-cost traverses than Anytime RRTs because they cope with new information much more intelligently. Rather than resetting the anytime parameters and generating a brand new solution from scratch, Anytime Dynamic RRTs maintain the current low-cost solution and repair just the portions of this solution that have been affected by the new information. This means both that Anytime Dynamic RRTs produce a valid solution more efficiently than Anytime RRTs and that the solution produced is better.

B. Constrained Exploration in Dynamic Environments

We have also applied the Anytime Dynamic RRT approach to constrained exploration in environments containing dynamic elements, such as moving obstacles or other agents. To compare the relative performance of Anytime RRTs and Anytime Dynamic RRTs against standard RRTs, we had each approach plan traverses for a team of 3 agents

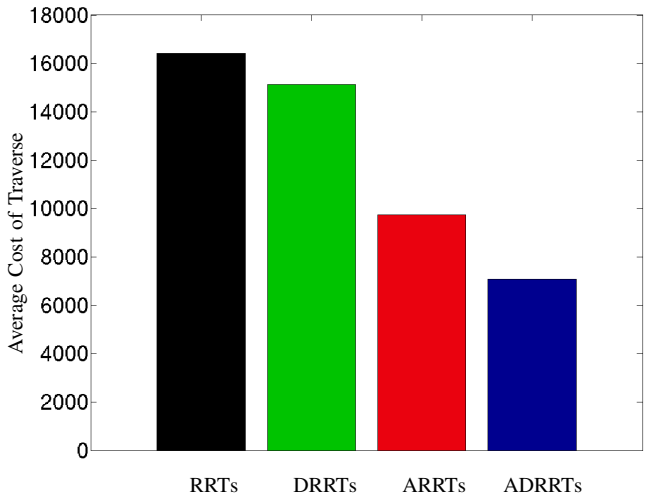

Fig. 4. Anytime Dynamic RRT results from our 3 agent constrained exploration experiments in partially-known environments.

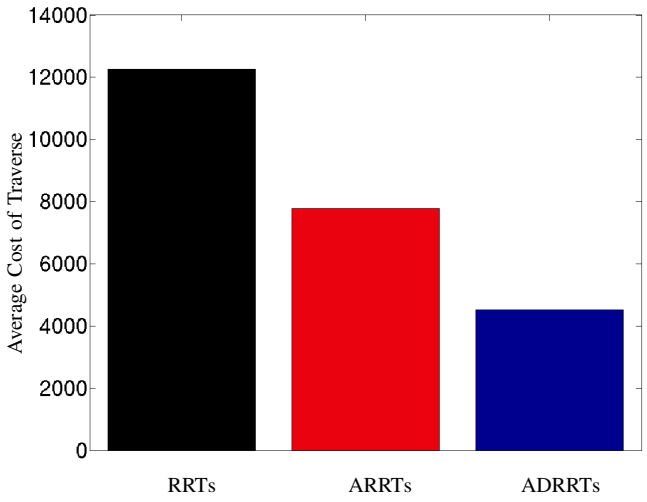

Fig. 5. Comparative results from our 3 agent constrained exploration experiments in dynamic environments.

navigating across a series of random environments containing communication obstacles, areas of non-uniform traversability cost, and dynamic obstacles. These traverses were similar to those performed above except that the environment now contained dynamic obstacles and new information was received concerning these dynamic obstacles as the team executed its traverse. We left Dynamic RRTs out of our comparison because they were already shown to be far less effective than Anytime Dynamic RRTs. For our experiments, we created 20 different environments containing randomly generated communication obstacles and high-cost areas, and for each of these environments we used 10 different collections of dynamic obstacles, each having a randomly-generated trajectory. Each of these collections contained 8 dynamic obstacles whose trajectories were unknown and needed to be estimated by the team, and 4 dynamic obstacles for which fairly accurate trajectories were initially provided to the team.

As the team traversed through the environment, updated information on the actual trajectories of both types of dynamic obstacles was received. Each of the dynamic obstacles with unknown trajectories changed their course 20 times, and each of the other dynamic obstacles updated their full trajectories 4 times. Whenever new information was received concerning any of the dynamic obstacles, each planning approach checked whether this information invalidated its current solution. If so, a new solution was generated, with each approach using its associated method for producing this solution.

Each approach was allowed to run for a total initial planning time of 20 seconds (on a 1.5 GHz Powerbook G4), and each individual RRT was allowed to take up to 10 seconds. As in our first set of experiments, each approach planned from a configuration three steps ahead of it on the current path.

Figure 5 plots the average cost of the team traverses and shows that Anytime Dynamic RRTs maintain their performance edge over the other approaches when dynamic obstacles are added to the environment. Although new information regarding the trajectories of these obstacles can affect states in the configuration-time space that are widely-separated in terms of distance, Anytime Dynamic RRTs are still able to repair the solution when this new information is received more efficiently than regenerating a series of new solutions from scratch.

C. Implementation on Outdoor Vehicles

We have also implemented Anytime Dynamic RRTs on a team of 3 autonomous E-gator vehicles performing constrained exploration in partially-known outdoor environments. As in simulation, the team began with a prior map of the environment that contained a collection of communication obstacles and areas of non-uniform traversal cost. The Anytime Dynamic RRT algorithm used this prior map to plan an initial path from the team's starting position to its desired goal position. As the agents traversed their paths, they observed with onboard lasers any navigation obstacles such as trash cans and bushes that were not in their prior map. When these obstacles interfered with their current solution paths, the team's path was repaired to account for the obstacles. This path was also continually being improved in an anytime fashion throughout the team's traverse. Figure 6 shows a sequence of images from the team navigating across an 80×100 meter environment.

VI. Conclusion

In order to provide high quality solutions when deliberation time is limited and the environment is imperfectly-known, we have combined the Anytime RRT algorithm with the Dynamic RRT algorithm to create a sampling-based anytime, replanning algorithm which we call *Anytime Dynamic RRTs*. The Anytime Dynamic RRT algorithm both improves and repairs the solution over time, and is particularly well-suited to path planning in high-dimensional search spaces for

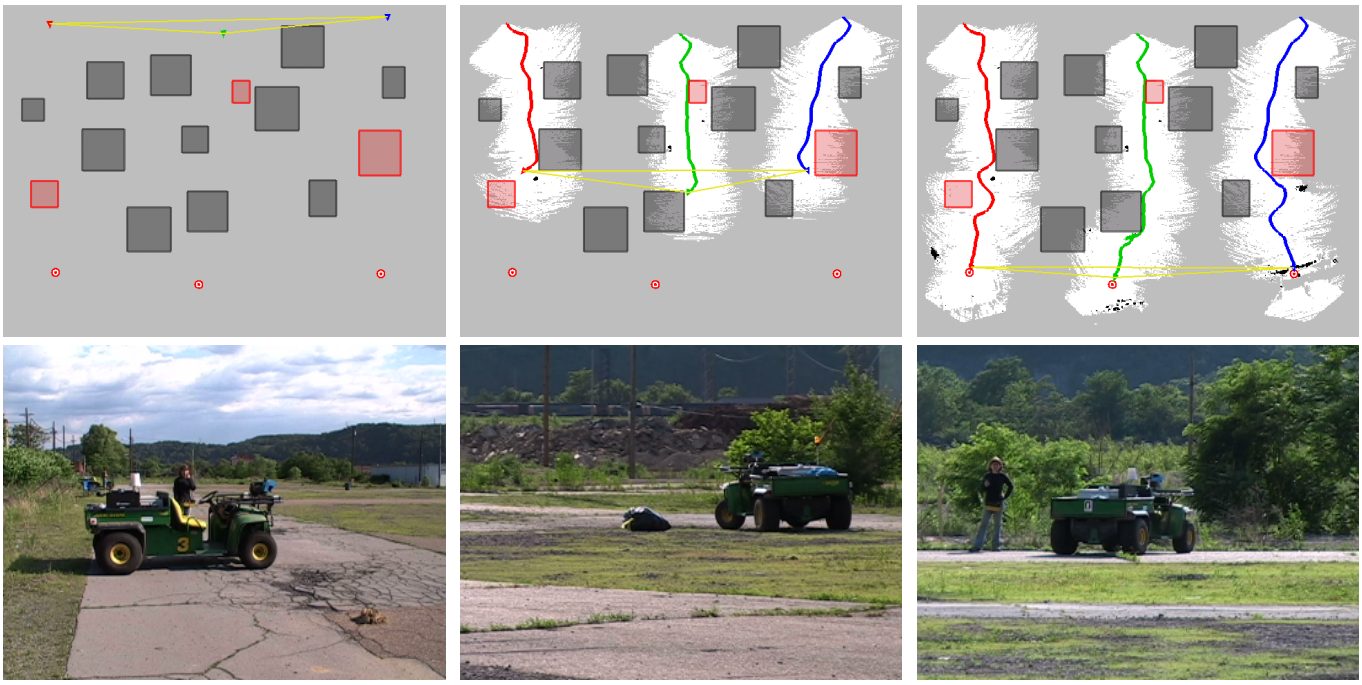

Fig. 6. Anytime Dynamic RRTs used for constrained exploration by a team of 3 vehicles in an outdoor environment. The left-most images show the initial positions of the vehicles and the prior map used by the team. The top-left image shows the map, along with the agents (the red, green, and blue arrows), the communication obstacles (in red), the high-cost regions (in dark grey), the goals for the agents (the red circles), and the line-of-sight links across the team (in yellow). The bottom-left image shows the center (green) vehicle in the foreground and the right-most (blue) vehicle in the background. The center images show the team after the left-most (red) vehicle has detected an obstacle in its path and the solution for the team has been repaired. The right-most images show the end of the run, with the top image of the pair showing the full paths traversed by the vehicles and the obstacles observed in the environment.

teams of agents navigating partially-known or dynamic environments. We have presented results highlighting its benefits in both simulation and from a team of 3 robotic platforms used to perform constrained exploration in partially-known outdoor environments.

Approaches that improve and repair their solutions over time can be extremely useful for complex planning problems involving imperfect information. The results provided here show this holds true for the Anytime Dynamic RRT algorithm. By extending RRTs to exhibit both anytime and replanning behavior, Anytime Dynamic RRTs efficiently produce low-cost solutions to very high-dimensional planning problems in partially-known or dynamic environments.

VII. Acknowledgments

The authors would like to thank Nidhi Kalra for significant contributions to the final manuscript. This work was sponsored by the U.S. Army Research Laboratory, under contract "Robotics Collaborative Technology Alliance" (contract number DAAD19-01-2-0012). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies or endorsements of the U.S. Government. Dave Ferguson was previously supported in part by a National Science Foundation Graduate Research Fellowship.

References

[1] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[2] N. Nilsson, *Principles of Artificial Intelligence*. Tioga Publishing Company, 1980.

[3] A. Stentz, "The Focussed D* Algorithm for Real-Time Replanning," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.

[4] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[5] R. Zhou and E. Hansen, "Multiple sequence alignment using A*," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2002, Student abstract.

[6] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in *Advances in Neural Information Processing Systems*. MIT Press, 2003.

[7] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime Dynamic A*: An Anytime, Replanning Algorithm," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2005.

[8] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[9] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with RRTs," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[10] D. Ferguson and A. Stentz, "Anytime RRTs," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[11] N. Kalra, D. Ferguson, and A. Stentz, "Constrained Exploration for Studies in Multirobot Coordination," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[12] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Motion planning for humanoid robots," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2003.

[13] D. Ferguson, "Single Agent and Multi-agent Path Planning in Unknown and Dynamic Environments," Ph.D. dissertation, Carnegie Mellon University, 2006.