Empirical Sampling of Path Sets for Local Area Motion Planning

Ross A. Knepper and Matthew T. Mason

Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213
rak@ri.cmu.edu, mason@cs.cmu.edu

Summary. We consider the problem of online planning for a mobile robot among obstacles, where it is impractical to test all possible future paths. One approach is for the runtime task to test some subset of the possible paths and select a path that does not collide with obstacles while advancing the robot toward its goal. Performance depends on the choice of path set. In this paper we assume the path set is fixed and chosen offline. By randomly sampling the space of path sets we discover effective path sets—comparable or superior to the best previously suggested approaches. In addition, testing large numbers of randomly generated path sets yields some insights on the relation of robot performance to the choice of path set.

1 Introduction

This paper empirically explores the relation between path sampling strategies and mobile robot performance. Many autonomous vehicles, including winners of the DARPA Grand Challenges [13, 14], employ path sampling as part of a hierarchical planning strategy. In this context, a local planner performs obstacle avoidance by considering possible actions (corresponding to workspace paths) in a tight plan/execute cycle (around 10 Hz), while a low-update-rate global planner provides overall guidance.

Hard time constraints dictate that the local planner evaluate only a few paths per cycle. One approach is to define a fixed path set—a finite subset of the possible paths—which the local planner will evaluate on every cycle. Some examples of fixed path sets for a car-like robot are pictured in Figure 1. Many fielded local planners have achieved mixed results by employing a path set comprising fixed-curvature arcs (Figure 1d) [3, 4, 7, 12]. Recent work has demonstrated improved performance using other path set choices [1, 5, 8]. There has also been some investigation of how tessellation can extend a carefully-designed set of paths to the global planning context [11].

While it is clear from previous work that performance varies substantially with the choice of path set, little else is known. Green and Kelly [5] argue that lowering dispersion in the space of paths improves performance, and they present a greedy algorithm to generate such path sets. A great deal of work has been done in planning with random sequences, which are inherently low-dispersion when

O. Khatib et al. (Eds.): Experi. Robotics: The 11th Intern. Sympo., STAR 54, pp. 451–462.
springerlink.com

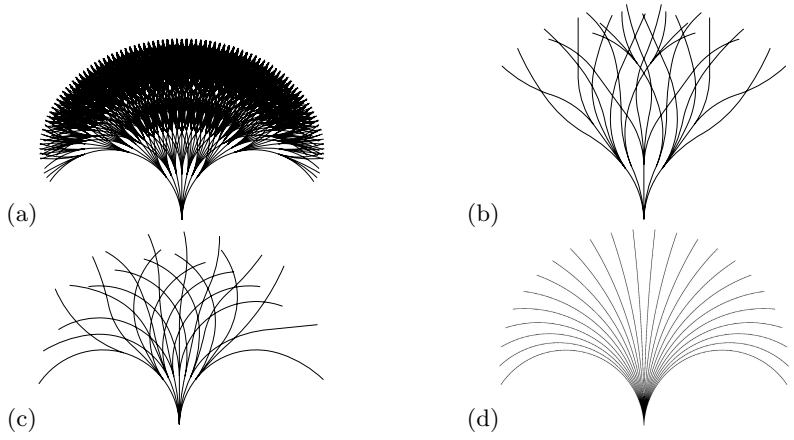

Fig. 1. Path sets: (a) The full 2,401-path data set; (b) The best randomly-generated 24-path subset; (c) The 24-path subset generated by the Green-Kelly algorithm; (d) The 24-path set of constant-curvature arcs

the sample size is large enough. A common approach to roadmap construction for global planning employs randomized state sampling, which in turn generates a low-dispersion roadmap [6, 9].

Such a randomized approach has also been applied in the context of local planning [10], but this stochastic action selection process introduces an undesirable element of uncertainty to the resultant vehicle motions. Nevertheless, we take inspiration from thees randomized approaches and explore offline random sampling in the space of path sets. By testing large numbers of random path sets, this paper explores the relation of path set to overall mobile robot performance.

Using the tools described here, we are able to estimate upper bounds on path set performance, measure the performance of previously proposed path sets, test hypotheses on path set performance, and search for high performance path sets. We began this investigation by hypothesizing that the full path set depicted in Figure 1a would outperform any path subset sampled from it, therefore providing an upper bound on performance.

2 Technical Approach

We want to determine experimentally how planner performance depends on choice of path set. We evaluate each candidate path set by trying it on several representative tasks. The main challenge is that the space of all path sets is large, consisting of about 10^{57} members in our case, and evaluating each path set requires several experiments. We address this challenge by combining physical experiments with simulations, attempting to gain the speed of simulation without sacrificing the rigor of physical experimentation.

Our approach was therefore to develop a planner on a real robot (discussed in more detail in Section 3.1 below) and then to switch to simulation for the

large-scale tests. A crucial part of the planner is the high-fidelity vehicle model, which predicts the robot's response to control inputs. It is this model (explained in Section 3.2) that enables the planner to ensure that each considered trajectory conforms to all kinodynamic constraints.

After achieving simulation results sufficiently similar to real robot behavior, we moved the experiment into simulation in order to conduct 150,000 test runs. Since exhaustive search of the path set space is not practical even in simulation, we sample the space of all path sets.

To characterize the performance of a path set, we run the local planner in simulation with a randomly generated task comprising start, goal, and arrangement of obstacles. We use two metrics: success rate and execution time. Since the robot replans continuously while moving, plan time and execution time are equivalent.

All of our path sets are subsets of the "full path set" (Figure 1a). Each path in the full path set is a sequence of four connected path segments. Each path segment is a curve of constant curvature. We assume seven different choices of curvature, i.e. seven different choices of path segment. Hence the full path set contains 7^4 or 2,401 different paths.

Each path set (Figure 1b for example) has 24 paths—about 1% of the full path set. Figure 1c shows the Green-Kelly path set, and Figure 1d shows a special path set (not a subset of the full path set) constructed to resemble prior art and comprising 24 non-branching, constant-curvature arcs.

3 Experimental Setup

3.1 Real Robot

In order to establish a rigorous basis in real-world experimentation, we used a differential-drive Nomad Scout robot (Figure 2). Since perception was not of

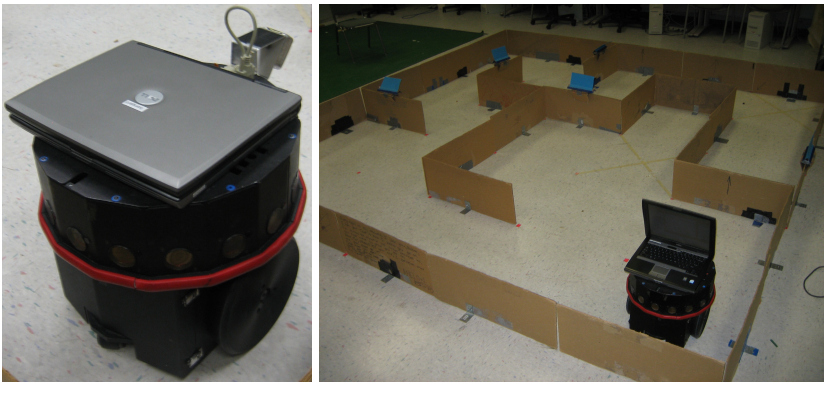

Fig. 2. Nomadic Technologies Scout robot. The robot traverses the maze, visiting each blue flag once.

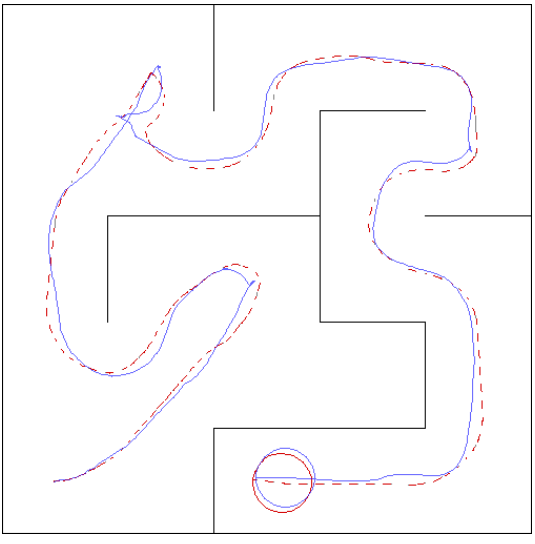

Fig. 3. The maze compares typical execution paths driven by the real (solid) and simulated (dashed) robots. The discrepancy between paths is comparable to that between any two runs of the real robot in the maze.

interest in this test, we provided the obstacle layout to the robot ahead of time. Real test environments consisted of mazes of robot-height cardboard walls. An example maze is shown in Figure 2 with the robot at its start configuration.

We set out to develop a robot simulator that accurately emulates outcomes in real-world motion planning experiments. To accomplish this feat, we faced two main problems. First, we needed a vehicle model capable of predicting the results of arbitrary commanded actions. To achieve this, a variety of actions were measured and the results incorporated into the simulation. The details of this process are explained in Section 3.2, below.

The second problem to be solved in order to achieve a close match between reality and simulation was positioning. We permitted the simulated robot to have perfect positioning, since localization is not a focus of this work. Because the Scout's dead reckoning is inadequate in these environments, we implemented a Kalman Filter to localize the robot. State variables consisted of linear and angular position and velocity. Besides the odometry updates, the filter used the robot's 16 sonar sensors to find landmarks (vertical and horizontal maze walls), which it compared to the provided map in order to generate state corrections.

3.2 The Vehicle Model

The vehicle model mentioned in the previous section serves two roles. First, the simulated experiments (Section 3.3) use the vehicle model to determine the robot's trajectory through the world based on commands issued by the planner at 5 Hz. Second, during each cycle, the planner uses the vehicle model to

anticipate where a sequence of commands issued at 1.5-second intervals would lead the robot. Only seven distinct curvature commands are available to select from at each iteration, which are uniformly discretized in the range of ±2.1 radians/meter. This car-like steering model was selected to simplify analysis of the results. So long as the robot sees a way to progress, it is commanded to drive at a constant linear velocity of 0.2 m/s.

In developing the vehicle model, a variety of classes of maneuvers were considered, from straight lines and gentle curves to tight braking spins and S-curves. We ran each maneuver on the real robot, recording odometry throughout the event. The change in pose between start and finish was measured to verify the reliability of odometry over short distances, and the vehicle model was tuned to account for the discrepancies between the commanded action and actual vehicle response.

The basic model, prior to tuning, was created based on the manufacturer's API, which permits the user to set acceleration and target velocity values individually for each wheel. We modeled the robot as being essentially kinematic, except that curvature cannot jump instantaneously because of the acceleration limit. Internally, the model mimics the robot's control loop on each wheel's speed. An Euler double-integrator generates a change in pose based on acceleration, velocity, and the passage of time. While this model is no more complicated than necessary for this experiment, it should be noted that we are modeling momentum and other kinodynamic constraints so that, for example, the vehicle has a measurable stopping distance that must be accounted for in planning to assure the robot's safety.

Tuning affected several aspects of the vehicle model. First, we noted that latency in serial communications results in a delay averaging 87 ms from sending a command to observing the response. This latency marks a further departure from a purely kinematic model. Second, the measured acceleration values do not precisely reflect what is commanded by the user. Surprisingly, a wheel's true acceleration is larger than commanded by a factor averaging about 2.4. After correcting for these two phenomena, the vehicle model proved sufficiently accurate that remaining error was overshadowed by random variation from one trial to the next. After verifying the model in extended real world tests (such as that shown in Figures 2 and 3), we carried the vehicle model into the simulation phase of the experiment.

3.3 Constructing the Planner

The focus of this experiment, the planner, was developed and tested with the real robot hardware before being carried over unchanged into simulation. This hierarchical planner consists of two levels. The high-fidelity local planner repeatedly evaluates a fixed set of paths originating from the body of the vehicle. Below, we discuss several methods of selecting this path set.

Meanwhile, the low-fidelity global planner provides a heuristic estimate of the cost-to-go in order to evaluate the available paths and choose the best one. As in most hierarchical planning systems, this global planner uses simplifying

approximations and assumptions to avoid the combinatorics of an exhaustive search. Specifically, it assumes an omnidirectional robot moving on an 8-connected grid. Prior to the first run of the local planner, the brushfire algorithm runs backwards from the goal state to populate the entire freespace grid with cost values. Besides providing an estimate of the cost-to-go, we can also filter out those planning problems that have no solution. The existence of an 8-connected grid path does not ensure success using any path set, though, because the paths are subject to kinodynamic constraints. In practice, success rates top out at about 80%.

Most of the path sets we tested in simulation were obtained by uniform random sampling from the full path set of $7^4 = 2,401$ paths. For each path selected, its mirror-symmetric path was added as well. In total, each tested path set contains 24 paths. A few special case path sets—constant-curvature arcs and the Green-Kelly path set—were generated by special means. The arc-based path set contains 24 discrete curvatures spanning the same range as the full path set. However, this tree only branches once at its root. The Green-Kelly path set is constructed according to the algorithm described in [5].

Each planning cycle consists of a breadth-first traversal of the path tree. As each segment is expanded, a robot-shaped disc is tested for collision with obstacles along the path. If that segment survives, then its endpoint, q, is evaluated using the scoring function. This function estimates the total runtime from the robot's current pose to the goal via q. The time required to traverse the path, c_p, is known. The global guidance navigation function offers a time-to-go value function, c_g. Since the global guidance cost is computed in a grid, it does not consider heading. Therefore, the bearing to the goal, θ_g, is added as an extra cost term, weighted based on the maximum angular velocity, ω_{max}. The overall cost function is

$$cost(q) = c_p(q) + c_g(q) + \theta_g(q)/\omega_{max}$$

Once the tree has been elaborated to depth 4, the best scoring node in the tree is selected. The initial command which led to that position in the tree is then sent to the robot for execution.

3.4 Experiments

The goal of the simulation experiments was to evaluate the planning performance of a large number of diverse path sets. To measure path set performance, a large number of planning tasks of comparable difficulty were needed. Toward this end, one hundred world/query pairs were generated. Each world consisted of a 100×100 grid with obstacles sampled randomly from a uniform distribution at a density of 2.5% coverage. The boundary of the world was also considered to be an obstacle. Queries consisted of a start and goal locations separated by a Euclidean distance of about 70 cells.

Each path set was evaluated by running it on all one hundred world/query pairs. Each simulated run began with the robot located at the start state and oriented in the direction of the brushfire gradient. The run ended in failure either

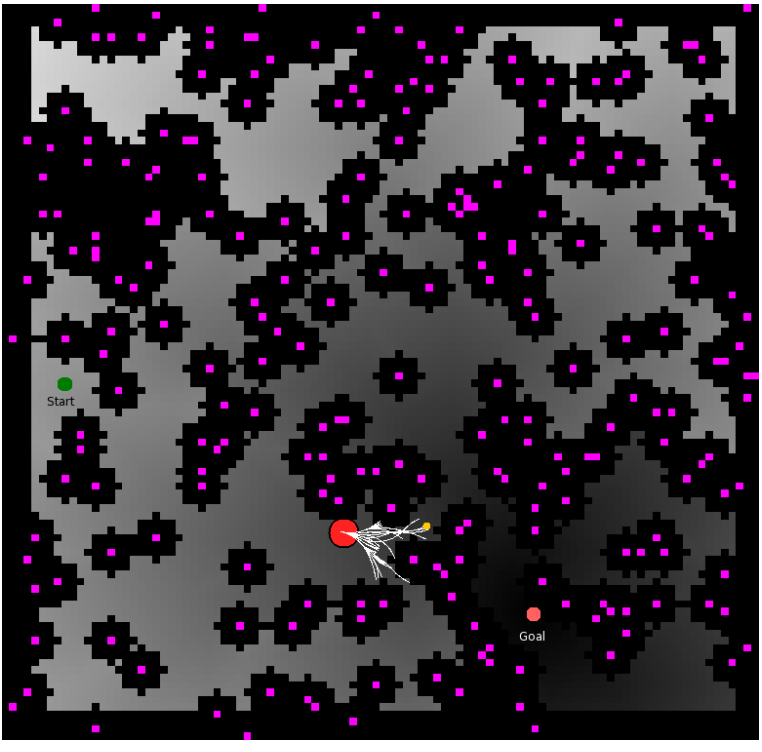

Fig. 4. Example simulation world. The small discs indicate the start and goal positions. Grey-scale shading indicates a navigation function generated by brushfire from the goal. Point obstacles are expanded by the robot's radius. Candidate paths are expanded from the robot's current pose every 0.2 sec.

when the robot came to rest after several successive iterations with no available path options or when the maximum simulated time of 400 seconds was exceeded. The run terminated successfully when the robot shape overlapped the goal.

We tested 1,500 different random path sets on each of 100 random tasks, such as the example depicted in Figure 4. Since this is such a small fraction of the overall space of path sets as we defined it, we did not expect to get an accurate picture of the total space of path sets. However, this sample size proved to be large enough to generate interesting results.

Having an objective means to compare path sets, we also evaluated several that had been discussed in other works. Constant-curvature arcs have a long history in local planning, while Green and Kelly recently proposed a more principled path set. Besides these two sets, we tested the full path set, which would normally be disqualified for having too many paths to evaluate in 0.2 seconds. Since the tests were conducted in simulation, we permitted the full path set to "cheat" with the expectation that providing it this unfair advantage would result in superior performance. That turned out not to be the case.

4 Results

4.1 Metrics on Path Sets

In comparing path sets with each other, we require at least one objective metric. The simplest, most obvious metric is success rate, which aggregates performance over all 100 worlds. A histogram of success rates for various path sets is depicted in Figure 5. The next obvious metric is simulated runtime. This metric is more nuanced since it reports 100 separate runtimes, and simple statistics may obscure trends.

These two metrics may be combined by plotting the cumulative distribution of all 100 runs for a given path set, counting failed runs as infinite runtime. This approach produces a curve from which both runtime and success rate are easily interpreted. Figure 6 shows highlights from the 1,503 tested path sets. The top-left curve depicts the best random path set we tested, and it also happens to represent an approximate envelope for achievable performance based on the sample of paths taken here.

While the cumulative distribution is an effective tool for comparing the performance of several path sets, it is still useful to condense each path set's overall performance into a single score. One simple, effective method is to measure the area beneath the curve. Thus, both a higher success rate and lower runtimes increase the score. Of course, the integral of the unbounded cumulative distribution function is infinite, so we chose to place an upper bound at 90 seconds. This upper bound was the maximum runtime observed over all of the experiments. Recognizing that the choice of upper bound trades off between the relative weighting

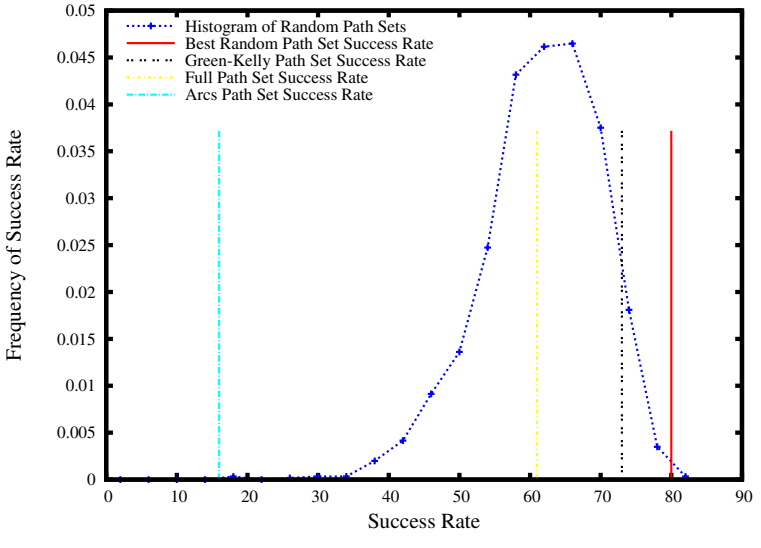

Fig. 5. Frequency distribution histogram showing success rate of various path sets. A higher success rate indicates a more robust path set in planning experiments.

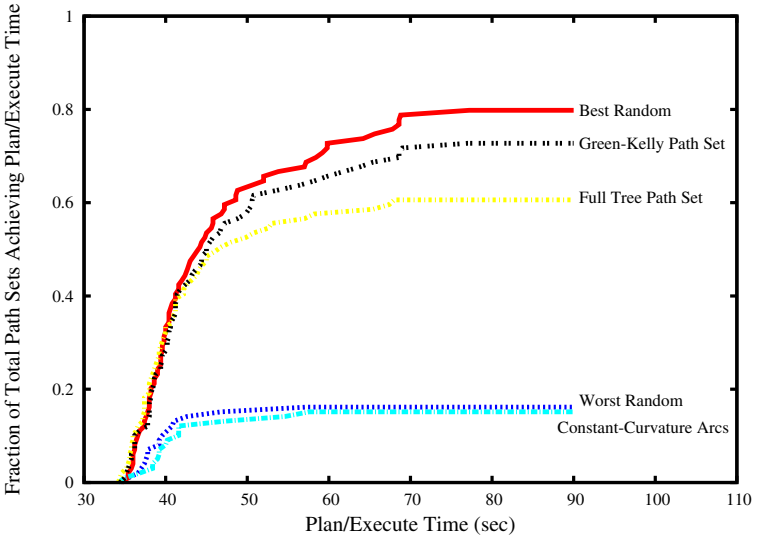

Fig. 6. Cumulative distribution plot summarizing results from several path sets. The area under each curve translates to a general score for that path set's performance across obstacle configurations.

of success rate and runtime, several alternative upper bounds were considered. In each case, the quantitative score values changed without altering the quantitative results relating the performance among path sets. Figure 7 shows the frequency distribution of score values for the set of random path sets and the special path sets mentioned previously.

4.2 Findings

Here are some of our findings on path set performance. First, it should be noted from Figures 5 – 7 that the constant-curvature arc planner performed quite badly overall. Our control, the full path set, did better. Surprisingly, many random path sets outperformed the full set. To verify this result, we retested the best-performing path set with 100 novel sets of 100 tasks (Figure 7). The worst and best path sets are shown in Figure 8. The best previously-generated path set we are familiar with, the Green-Kelly path set, scored 3323, while our best randomly-generated path set scored 8% better at 3604.

To examine the statistical significance of this finding, we reran the Green-Kelly and best random path sets for 1000 trials (unique world/query pairs) each. The Green-Kelly path set succeeded in 781 trails, while the best random path set succeeded in 803 trials. A chi-squared test tells us with 98.3% confidence that this performance difference does not result simply from random chance. Since these two path sets are comparatively close in performance, it naturally follows that performance differences over the full spectrum of path sets tested in over

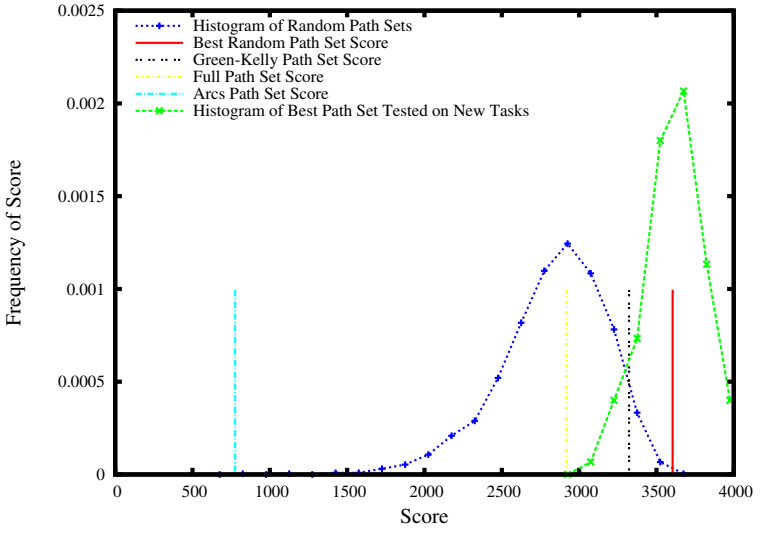

Fig. 7. Frequency distribution histogram of performance score comparing variation over path sets and variation over a range of planning problems for the best tested path set. Higher scores correspond to better planning performance.

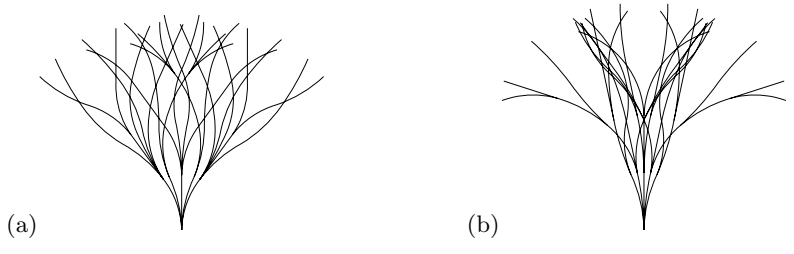

Fig. 8. Best (a) and worst (b) randomly generated path sets

150,000 trials reflect the importance of careful path set selection in obstacle avoidance.

5 Discussion and Future Work

The first significant outcome of this work is that random path set sampling is capable of generating a fairly dramatic range in performance (approaching a factor of three in our case). When searching for the best path sets, many bad ones must be rejected. One might therefore argue, as Branicky and LaValle [2] observed, that carefully constructed quasi-random sequences can have lower dispersion than a true random sequence, and that we could therefore produce improved results with such a sequence. But as we don't know what properties of a path set lead to good performance, we believe it is precisely the higher variance observed

in a random sequence (as compared with a Hammersley-Halton sequence) that widens the tail on both sides of our frequency distribution plot in Figures 5 and 7, thereby pushing out the best and worst performances.

The two most surprising outcomes are: (1) that a randomly generated path set would substantially out-perform the Green-Kelly path set; and (2) that the full path set would perform worse than half of the random subsets we tested.

We are still exploring the comparison with the Green-Kelly approach, by testing additional Green-Kelly path sets, and by measuring dispersion of our randomly generated path sets. If the difference of performance is substantiated by further tests, we wish to explore the metric used to define dispersion in the path space, and whether the inherently sub-optimal performance of a greedy approach might be responsible.

The poor performance of the full path set is perhaps the most intriguing outcome. We expected that a planner's performance can only improve when it is given more choices to evaluate, but instead we are seeing that more choices can be worse. Because of the limited planning horizon and the heuristic nature of the global planner's guidance, more choices also means more bad choices. Evidently, sub-sampling the full path set introduces a mechanism whereby bad choices can be hidden from the planner. We are exploring the details of this mechanism.

Acknowledgments

This work is sponsored by the Defense Advanced Research Projects Agency. This work does not necessarily reflect the position or the policy of the Government. No official endorsement should be inferred.

We would also like to thank Alonzo Kelly and Michael Erdmann for their contributions to this work.

References

1. Branicky, M.S., Knepper, R.A., Kuffner, J.J.: Path and trajectory diversity: Theory and algorithms. In: Proc. Intl. Conf. Robotics and Automation, Pasadena, CA (2008)
2. Branicky, M.S., LaValle, S.M., Olson, K.: Quasi-randomized path planning. In: Proc. IEEE International Conference on Robotics and Automation, vol. 2 (2001)
3. Daily, M., Harris, J., Keirsey, D., Olin, D., Payton, D., Reiser, K., Rosenblatt, J., Tseng, D., Wong, V.: Autonomous cross-country navigation with the ALV. In: Proc. of IEEE Intl. Conf. Robotics and Automation, Philadelphia, PA, pp. 718–726 (1988)
4. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robotics and Automation 4(1) (1997)
5. Green, C., Kelly, A.: Toward optimal sampling in the space of paths. In: 13th International Symposium of Robotics Research, Hiroshima, Japan (November 2007)
6. Kavraki, L.E., Svestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4), 566–580 (1996)

7. Kelly, A., Stentz, A.: Rough terrain autonomous mobility - part 2: An active vision, predictive control. Autonomous Robots 5(2), 163–198 (1998)
8. Lacaze, A., Moscovitz, Y., DeClaris, N., Murphy, K.: Path planning for autonomous vehicles driving over rough terrain. In: Proc. of IEEE Intl. Symp. Intelligent Control (1998)
9. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. The International Journal of Robotics Research 20(5), 378 (2001)
10. Team MIT. A perception driven autonomous urban vehicle. Journal of Field Robotics (to appear, 2008)
11. Pivtoraiko, M., Knepper, R.A., Kelly, A.: Optimal, smooth, nonholonomic mobile robot motion planning in state lattices. Technical Report CMU-RI-TR-07-15, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (May 2007)
12. Simmons, R.: The curvature-velocity method for local obstacle avoidance. In: International Conference on Robotics and Automation (April 1996)
13. Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., et al.: Stanley: The robot that won the DARPA grand challenge. Journal of Field Robotics 23(9), 661–692 (2006)
14. Urmson, C., Baker, C., Darms, M., Dolan, J., Ferguson, D., Salesky, B., Whittaker, W., Rybski, P.: Boss: A robust urban driving autonomous vehicle. In: Proc. of Intl. Symp. Experimental Robotics (July 2008)