A Generic Infrastructure for Benchmarking Motion Planners

Benjamin Cohen, Ioan A. Şucan, Sachin Chitta

***Abstract*— Randomized planners, search-based planners, potential-field approaches and trajectory optimization based motion planners are just some of the types of approaches that have been developed for motion planning. Given a motion planning problem, choosing the appropriate algorithm to use is a daunting task even for experts since there has been relatively little effort in comparing the plans generated by the different approaches, for different problems. In this paper, we present a set of benchmarks and the associated infrastructure for comparing different types of motion planning approaches and algorithms. The benchmarks are specifically designed for robotics and include typical indoor human environments. We present example motion planning problems for single arm tasks. Our infrastructure is designed to be easily extensible to allow for the addition of new planning approaches, new robots, new environments and new metrics. We present results comparing the performance of several motion planning algorithms to validate the use of these benchmarks.**

I. Introduction

Motion planning is a search problem that requires finding a continuous path between given start and goal states, for a particular system, subject to a variety of constraints. Under this broad definition, solving the motion planning problem has applications in robotics, protein folding, games, model checking, among others [1], [2]. In this paper however we focus on robotics applications. The variety of problems, even within robotics alone, is large. For example, computing motion plans for the legs of a biped is done very differently than computing plans for an omni-directional wheeled base. At the same time, planning motions for a car moving at high velocity is done differently than computing motion plans for a robot arm moving at low velocities. The variety of problems in robotics naturally stems from the large number of different types of robots and the various constraints that arise in the tasks for these robots. A wide range of motion planners have been developed and used to address these problems in robotics [1], [2].

There are multiple approaches to solving various instances of the motion planning problem. We enumerate just a few of the more prominent ones:

1) Sampling-based motion planners use a randomized approach that iteratively constructs an approximation of the robot's state space (or configuration space) [3], [4]. This class of algorithms usually includes probabilistically complete algorithms (i.e., a solution will eventually be found, if one exists, but the lack of a solution cannot be decided).

B. Cohen is with the GRASP Laboratory at the University of Pennsylvania, Philadelphia, USA (bcohen@seas.upenn.edu). I. Şucan and S. Chitta are with Willow Garage Inc., Menlo Park, USA (isucan,sachinc)@willowgarage.com.

Fig. 1. An example scenario where the PR2 robot is manipulating objects in a kitchen environment.

2) Search-based planners operate on a discretized representation of the continuous space [5]. These approaches can provide optimality guarantees with respect to the discretization of the actions and the state space, and they are resolution complete (i.e., if a solution exists, it will be found given that the resolution is sufficiently refined).
3) Potential fields techniques use attractive and repulsive components to draw a robot to its goal while keeping it away from obstacles (e.g., [6], [7]).
4) Trajectory optimization techniques (e.g., [8], [9]) start with an initial guess for a path and try to refine it to get a collision-free optimal solution.

The list above is of course not exhaustive. For example, work has been done on complete algorithms that solve the motion planning problem [10]. However, the complexity of such algorithms is usually prohibitively high.

Each of the motion planning approaches mentioned above encompasses several algorithms. Sampling-based motion planning, for instance, includes many tens of related but different approaches. A problem that arises at this point is choosing the appropriate algorithm for a given problem. Given the large number of existing algorithms, this is a daunting task even for an expert. In this paper, we propose a benchmarking system that allows the dissemination of benchmark results for a variety of problems using a variety of algorithms. This system is open source, is part of the MoveIt! [11] framework (built using the ROS [12] middleware), and is intended to receive contributions from the community in the form of both implementations of planning algorithms and problem scenarios to include in benchmarks.

Our intention is not to identify the best algorithm, as such

a notion is difficult to define and in fact may not make sense, given the variety of motion planning problems. We intend to present relevant information for each of the methods we have an implementation for, and to allow the user to choose the planner that best suits their particular needs. We believe that avoiding the use of carefully chosen benchmark problems for particular planning algorithms and simply running all implemented algorithms on all problems from a database of problems that everyone can contribute to, will allow more meaningful results to stand out.

II. Related Work

The study and development of most new motion planning algorithms is usually accompanied by tests against a variety of benchmarks. A widely cited benchmark is the *alpha puzzle*, which was used to test the performance of motion planners for *narrow passage* problems. A number of well-known benchmarks for motion planning, including the alpha puzzle, can be found in [13]. An early attempt at defining benchmarks can be found in [14]. Iossifidis et. al. [15] presented a methodology for building benchmarks for pick and place tasks. Geraerts et. al. [16] benchmarked a set of probabilistic planners and their components, including multiple uniform and non-uniform samplers in six different environments. A set of humanoid benchmark problems for randomized planners was presented in [17]. A series of benchmarks for testing GPU based planners were also developed in [18]. Comparisons of collision detectors, which are a significant component of any motion planning framework, were done in [19].

Several open source software implementations of motion planning algorithms (e.g., OMPL [20] and OpenRAVE [21]) have also included a form of internal benchmarking capabilities. However, there is still a lack of easily available benchmarking infrastructure to which new problems, new motion planning techniques and new environments can be easily added. In particular, as pointed out in [22], a lot of the benchmarking efforts have been discontinued and are no longer well-supported. Furthermore, even with the large number of motion planning algorithms developed over the last two decades, there is very little insight available to new or expert users on the suitability of different motion planning algorithms for different types of problems.

Our work differs from previous work and addresses the issues mentioned above in multiple ways. We present an open source benchmarking framework explicitly designed for comparing motion planning algorithms in robotics. We are not restricted to one category of motion planners (e.g., only sampling-based or only search-based). Our framework builds on a widely available and easily accessible infrastructure (ROS) allowing for the easy addition of new environments, new motion planning problems and new motion planning algorithms, thus encouraging contributions from the community. Our benchmarking framework runs all the available motion planners against all the motion planning problems (as much as possible), and presents informative statistics. Note that we do not try to find the best motion planning algorithm, but intend to highlight the relative strengths and drawbacks of the different approaches in different situations and let the user choose the approach best suited to the problems they need to solve.

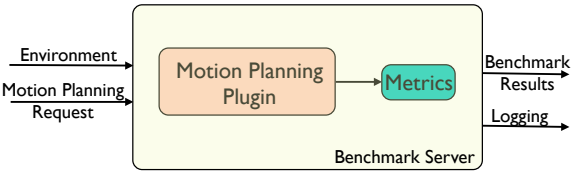

Fig. 2. A schematic representation of the benchmark server. All interfaces are implemented using ROS.

The rest of this paper is organized as follows. In Section III we present the infrastructure for computing benchmarks. This includes a discussion of the types of metrics that will be computed for the benchmark problems. In Section IV we present a set of example benchmark problems. In Section V we present detailed results and analysis for these benchmarks computed across multiple randomized and search-based planning algorithms. We conclude in Section VI.

III. Benchmarking Infrastructure

The infrastructure for our approach is currently built on top of the ROS middleware [12], as part of the MoveIt! project [11]. Our infrastructure is designed to be easy to use, allowing new motion planning algorithm designers to quickly test new approaches over a wide range of scenarios. We will now describe in detail the different components of our infrastructure.

A. Benchmarking Server

Fig. 2 shows a schematic representation of the benchmarking server. Each benchmark test consists of two inputs to the benchmark server, an environment representation and a motion plan request. The motion plan request includes the start state for the motion plan, the goal region and possible constraints that may be applied during execution (e.g., orientation constraints that ensure a grasped object remains upright during motion) and the type of planner to be used for planning.

B. Motion Planning Interface

Fig. 3 shows how the benchmark server views the planning algorithm it evaluates. The planning algorithm must offer a C++ interface that provides a "solve" function. The benchmark server will pass relevant information to the motion planner: the environment in which motion planning will be carried out, the robot's starting state, its goal region, and any constraints to be maintained along the planned path. The motion planner may include multiple post-processing steps, e.g., a shortcutting step and/or a smoothing step to improve the quality of plans. The number and types of operations performed by these additional post-processing steps are not restricted. The explicit representation of these intermediate steps is used to record the behaviour of the planner along its processing pipeline, so that more informative benchmark results can be presented. This reporting of information at

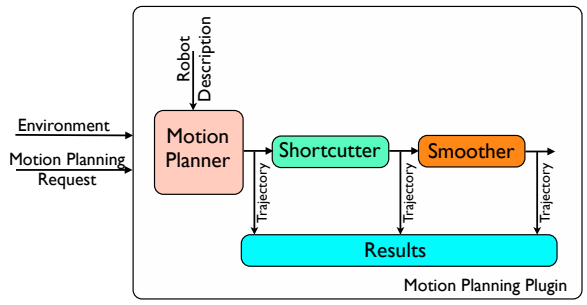

Fig. 3. A schematic representation of the motion planning plugin. The plugin is implemented as a C++ interface and allows easy configuration of different types of planners.

intermediate steps is the only additional functionality that a motion planning library will (optionally) need to implement for use with the benchmarking server.

C. Collected Data

The main result from the planning interface is a set of trajectories (or plans) that each satisfy the given motion planning problem. The planning interface allows the inspection of intermediate steps in the planning process. It can thus measure the quality of raw motion plans as well as post-processed plans and trajectories. Post-processing is a significant step for most motion planning approaches and could often take more time to perform than the planning process itself [23]. Each plan or trajectory returned by the planner has an identifier associated to it which indicates the type of post-processing performed. Shortcutting and smoothing are examples of possible such processing. Shortcutting is a process typically employed by randomized planners for reducing the length of raw planned paths, and smoothing refers to an operation where sharp edges in a trajectory are rounded off [23].

D. Computed Metrics

A series of metrics are used to measure the performance, reliability and quality of paths generated by all the motion planners. The metrics currently implemented include the following:

- **Plan computation times** - Separate computation times are computed for each stage of the planning process. This includes the time for computing raw plans and any path simplification that may be performed by the planner (τ_r) and the time for other processing (including any pre-processing of the environment) τ_o. Post-processing can typically involve shortcutting, interpolation, smoothing or other operations and a separate metric that could be measured is the time for post-processing.
- **Path length** - The length l_p of the path is measured as the total distance in the space in which the motion planner is operating.
- **Smoothness of plans** - This metric κ' measures how smooth the paths generated by the planners are. Let α_i represent the angle between two consecutive segments of a plan with n segments. Then, $\kappa' = \frac{1}{n}\sum_{i=2}^{n}\alpha_i^2$. Note that there are other measures of smoothness that could be reported for other types of robotic systems and problems. Our system can be easily extended to incorporate them.
- **Clearance** - This metric d_n measures the average minimum distance from the waypoints along computed plans to environment obstacles.
- **Success Rate** - The success rate s_r represents the percentage of trials for which the motion planning computation was successful within the specified time.

TABLE I

THE METRICS COMPUTED FOR EACH TRAJECTORY IN OUR BENCHMARKS

Metric	Description
τ_t	Total computation time (in s)
τ_r	Time taken (in s) to compute raw plans and any path simplification
τ_o	Time taken (in s) for "other" processing, e.g., setting the environment
d_n	The average minimum clearance (in m) from the environment
l_p	Path length (in radians)
κ'	Path smoothness
s_r	Success rate (in fixed amount of time)

Table I summarizes the set of metrics we use. Our benchmarking approach is not limited to the set of metrics described above. Including additional metrics is easy and our system allows the inclusion of metrics specific to individual planning algorithms. We will now describe the environments and motion planning problems that we have implemented as sample benchmarking problems in our system.

IV. Benchmark Problems

The benchmark problems we define here require the specification of four components:

1) **Environment Model** - The environment to be used for motion planning. This is specified as a collection of mesh and primitive objects representing the actual structure of the environment, e.g., rooms, shelves, etc. and the objects in the environment. The environment representation can also include a voxel grid representation of the occupied parts of the environment (which can optionally be represented as an octree [24]). The environment could also be generated artificially (e.g., using graphics and CAD tools) or could represent data obtained directly from the sensors on a robot.
2) **Robot Model** - The particular robot used for the benchmark experiments is specified using the ROS robot description formats. In addition, objects from the environment can be attached to the robot model (e.g., robot is carrying an object).
3) **Goals** - Associated with the environment is a set of goals for motion planning (e.g., goals may be specified in areas of the environment where objects may be typically found). In our framework goals are specified as sets of constraints. Different types of constraints are supported, and more can be added, but for the purposes of this paper we only used position and orientation constraints. A position constraint specifies the bounding volume where a reference point on a robot link needs to arrive. An orientation constraint

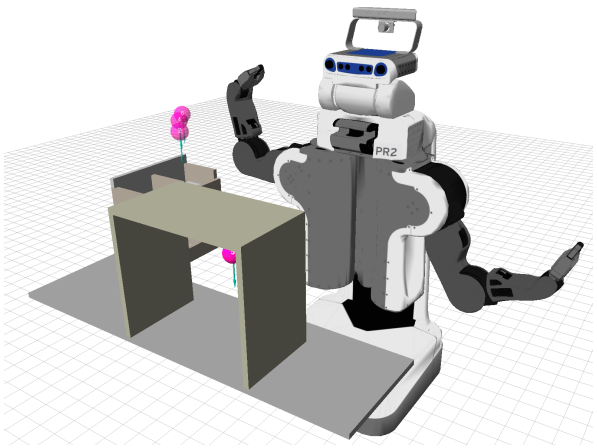

Fig. 4. The INDUSTRIAL benchmark environment: The pink spheres indicate desired goal locations for the right end-effector of the PR2 robot. The arrows indicate the desired orientation of the end-effector.

specifies the bounds in orientation (roll, pitch, yaw) for a particular robot link. Often, position and orientation constraints are used in conjunction to specify goal regions for robot links such as end-effectors.

4) **Robot Start State** - The robot start state is specific to each robot and provides complete information on the state of the robot at the start of a motion plan.

The benchmark problems are designed to exercise the planners through a wide-range of scenarios. Some of the environments were chosen with the express intention of testing the performance of planners in difficult situations, e.g., the narrow passage problem. Note that we do not intend for this set of benchmarks to be the final set of benchmarks that all planners will be run against. Instead, we hope that with the help of the community we will build upon this set and include environments and problems that exercise other capabilities of planners, e.g., dealing with dynamic obstacles or uncertainty. We will now describe the set of example benchmark experiments that we have currently designed and evaluated.

A. *Testing Environments*

Fig. 4, Fig. 5, Fig. 6 and Fig. 7 illustrate the sample environments that we use in all our experiments. The robot used in all experiments is the simulated PR2 robot. The PR2 is a state-of-the-art mobile manipulation robot with an omni-directional base, two 7 DOF arms and parallel jaw grippers. The robot also has multiple sensors including a RGB-D sensor, two laser scanners, an IMU, tactile sensors and stereo cameras.

The environments represent just a sample of the types of scenarios that mobile manipulation robots can expect to find in typical human environments. The first environment (Fig. 4) represents a factory scenario where the robot needs to pickup objects from a shelf containing multiple parts (e.g., kitting applications). The second environment (Fig. 5) represents a typical kitchen scenario where the robot may be expected to pickup objects from multiple places on the countertop or in the shelves. The third environment (Fig. 6)

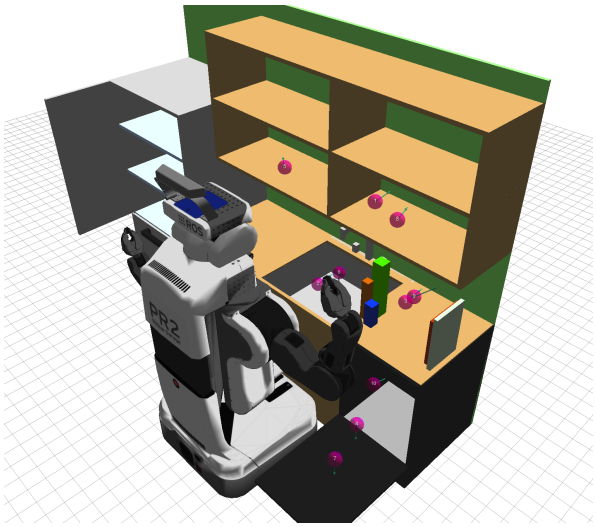

Fig. 5. The KITCHEN benchmark environment (human-scale).

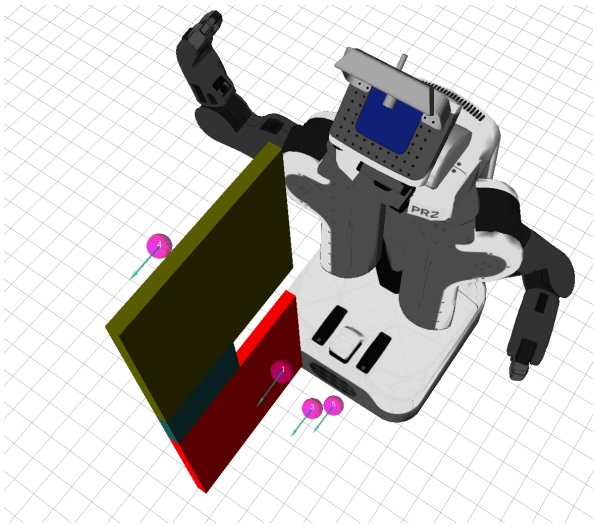

Fig. 6. The NARROW PASSAGE benchmark environment.

represents a scenario where the robot is expected to move its arm through a narrow passage, a problem that is typically difficult for sampling-based planners. The position of the wall restricts the workspace available for the arm considerably. The fourth environment (Fig. 7) again represents a typical kitchen environment where the robot may be expected to move an object above and below a table.

The motion planning problems we chose to use for this paper are derived from typical pick and place tasks in human environments. Our problems specify desired goal configurations for the PR2's right arm end-effector in typical positions from where objects may be picked up or where objects may be placed down. Fig. 4, Fig. 5 and Fig. 7 show some of the end-effector poses at the desired goal configurations, indicated by the red numbered markers. The goals in Fig. 6 are designed to force the right arm of the robot to move through the narrow constrained space between the robot and the wall. Additional constraints may be separately specified in the motion planning problem, e.g.,

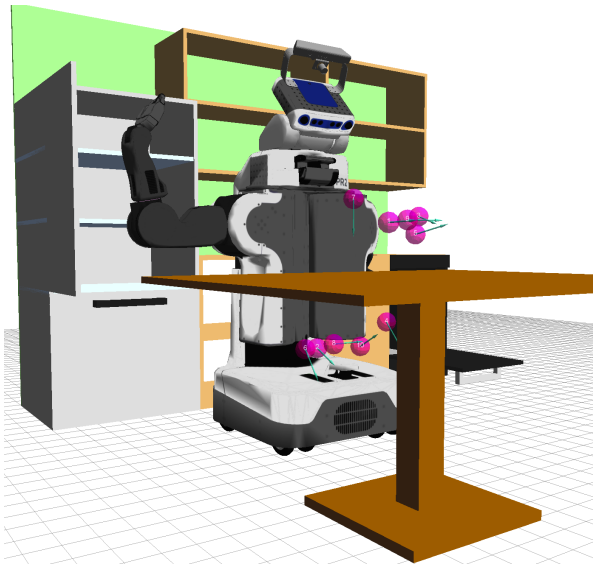

Fig. 7. A set of goals for manipulation tasks on a table in the TABLETOP environment.

a dual-arm motion of the robot may require the two arms to move in a constrained manner. Note that the desired goals do not constrain the redundant degrees of freedom of the PR2 robot arm in any manner. Each planner is thus free to choose the joint configuration that satisfies the desired goal configuration, e.g., by using inverse kinematics.

B. Planners

To validate our benchmarking approach, we tested it with two types of motion planners: randomized planners from the OMPL library [20] and search-based planners from the SBPL library [25].

The randomized planning approaches tested here are KPIECE [26], LBKPIECE [26], SBL [27], RRT* [28], RRT and RRT-Connect [29]. The cost function for RRT* is the distance traveled in joint space. RRT* was configured to return the first solution that was found. All these planners are implemented in the OMPL library and use the FCL collision checking library [30]. Sampling-based planners usually depend on problem-specific parameters that can significantly influence performance. In this work we did not set any of these parameters and only the default settings provided by OMPL were used.

The underlying search-based planner used here is the ARA* planner [31]. A set of discretized motion primitives in joint space are used with the planner to generate motion plans for the arms of a robot [32], [33]. An inflated heuristic derived from a 3D breadth first search in the space of the end-effector is used to accelerate this planning process. The ARA* planner provides bounds on sub-optimality with respect to the graph that represents the planning problem. We configured the search-based planners to return the first solution they find and we set ϵ, the sub-optimality bound parameter, to 100. The ARA* planner is an anytime planner capable of improving the solutions that it finds over time but we chose not to use these capabilities of the planner. The

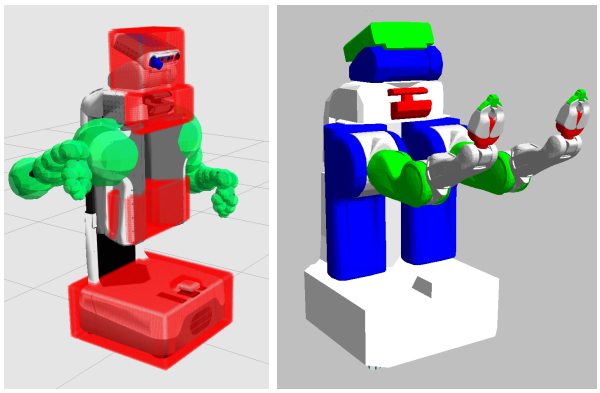

Fig. 8. The collision checking models used by the two planning libraries - SBPL (left) and OMPL (right)

cost function for the search-based planner incorporated the distance traveled in joint space.

It should be noted here that collision checking is a critical component of motion planning and often the most expensive. The two motion planning libraries we benchmark use different collision representations for the robot (Figure 8) and also employ different approaches to the collision representation for the environment. The collision representation used by the search-based planners for representing the robot is more conservative but an order of magnitude faster than the mesh-based representation used for the randomized planners. However, the representation of the environment for the search-based planners uses a distance field [34], and generating that distance field requires additional time (included in the metric τ_o), thus slowing down the overall time taken to generate plans. Integrating both collision representations to offer planners a common representation to work from is a subject of ongoing work.

C. Experiments

The experiments were run with the simulated environments presented in this paper. A set of goals were specified for each environment and the planners were instructed to plan motions for the right arm of the robot between successive goal positions (starting from a pre-defined start position for the first goal). As noted earlier, the goals were not specified as individual poses, but instead as goal regions. The translational tolerance for the goal was specified using a 2 cm cube within which the end-effector was supposed to end up. An orientation tolerance of 0.05 radians was also specified for each goal, i.e., the goal was considered to have been achieved if the final orientation of the end-effector of the robot was within a roll, pitch and yaw of 0.05 radians of the desired orientation. This goal representation allows planners to optionally sample individual goal poses inside the goal region, thus increasing the chances of finding valid joint configurations that reach the goal region. The number of goal regions used for the different environments is: 5 for INDUSTRIAL, 10 for KITCHEN, 5 for NARROW_PASSAGE and 10 for TABLETOP.

Each planner execution, between a start and goal position, was carried out 30 times for each sampling-based planner and

TABLE II

MEAN AND STANDARD DEVIATIONS FOR METRICS FOR DIFFERENT PLANNERS RUNNING IN THE KITCHEN ENVIRONMENT.

Metric	ARA*	RRTC	LBKP	SBL	RRT	KP	RRT*
τ_t (Mean,s)	0.46	0.14	0.26	0.42	1.33	0.52	2.67
τ_t (σ,s)	0.00	0.04	0.08	0.16	2.15	0.47	4.40
τ_r (Mean,s)	0.28	0.13	0.25	0.42	1.32	0.51	2.66
τ_r (σ,s)	0.00	0.04	0.08	0.16	2.15	0.47	4.40
τ_o (Mean,s)	0.18	0.01	0.01	0.01	0.01	0.01	0.01
τ_o (σ,s)	0.00	0.00	0.00	0.00	0.01	0.00	0.00
d_n (Mean,m)	0.03	0.03	0.03	0.03	0.03	0.03	0.03
d_n (σ,m)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
l_p (Mean,radians)	10.75	12.69	10.52	9.54	12.10	10.31	10.72
l_p (σ,radians)	0.00	2.27	1.50	1.31	2.14	1.70	1.60
l_e (Mean,m)	1.21	1.19	1.21	1.15	1.21	1.19	1.17
l_e (σ,m)	0.00	0.19	0.20	0.12	0.19	0.17	0.18
κ' (Mean,radians)	1.54	0.36	0.45	0.43	0.31	0.40	0.31
κ' (σ,radians)	0.00	0.57	0.56	0.44	0.46	0.56	0.46
s_r	100%	100%	100%	100%	96%	100%	93%

TABLE III

MEAN AND STANDARD DEVIATIONS FOR METRICS FOR DIFFERENT PLANNERS RUNNING IN THE INDUSTRIAL ENVIRONMENT.

Metric	ARA*	RRTC	LBKP	SBL	RRT	KP	RRT*
τ_t (Mean,s)	0.17	0.10	0.19	0.25	0.49	0.22	1.62
τ_t (σ,s)	0.00	0.03	0.07	0.10	0.97	0.15	3.98
τ_r (Mean,s)	0.08	0.10	0.18	0.24	0.48	0.21	1.61
τ_r (σ,s)	0.00	0.03	0.07	0.10	0.97	0.15	3.98
τ_o (Mean,s)	0.09	0.00	0.01	0.01	0.01	0.01	0.01
τ_o (σ,s)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
d_n (Mean,m)	0.07	0.06	0.05	0.05	0.06	0.05	0.06
d_n (σ,m)	0.00	0.02	0.01	0.01	0.02	0.01	0.01
l_p (Mean,radians)	6.60	10.10	9.20	7.38	9.99	8.82	8.30
l_p (σ,radians)	0.00	2.49	2.95	1.26	2.91	2.72	1.37
l_e (Mean,m)	0.98	1.08	1.31	0.99	1.13	1.29	1.05
l_e (σ,m)	0.00	0.20	0.59	0.20	0.29	0.60	0.18
κ' (Mean,radians)	1.49	0.36	0.38	0.38	0.37	0.40	0.29
κ' (σ,radians)	0.00	0.51	0.53	0.52	0.58	0.65	0.41
s_r	100%	100%	100%	100%	100%	100%	100%

TABLE IV

MEAN AND STANDARD DEVIATIONS FOR METRICS FOR DIFFERENT PLANNERS RUNNING IN THE TABLETOP ENVIRONMENT.

Metric	ARA*	RRTC	LBKP	SBL	RRT	KP	RRT*
τ_t (Mean,s)	0.77	0.07	0.11	0.18	0.18	0.11	0.35
τ_t (σ,s)	0.00	0.02	0.02	0.04	0.13	0.04	0.30
τ_r (Mean,s)	0.72	0.07	0.11	0.17	0.17	0.11	0.34
τ_r (σ,s)	0.00	0.02	0.02	0.04	0.13	0.04	0.30
τ_o (Mean,s)	0.05	0.01	0.01	0.01	0.01	0.01	0.01
τ_o (σ,s)	0.00	0.00	0.00	0.00	0.00	0.01	0.00
d_n (Mean,m)	0.06	0.07	0.07	0.06	0.08	0.08	0.08
d_n (σ,m)	0.00	0.03	0.03	0.02	0.03	0.03	0.03
l_p (Mean,radians)	10.33	12.85	10.06	9.31	11.95	10.03	10.14
l_p (σ,radians)	0.00	2.43	1.65	1.29	2.44	1.64	1.52
l_e (Mean,m)	1.55	1.66	1.67	1.60	1.68	1.69	1.71
l_e (σ,m)	0.00	0.18	0.19	0.13	0.18	0.19	0.20
κ' (Mean,radians)	1.28	0.37	0.32	0.31	0.23	0.30	0.35
κ' (σ,radians)	0.00	0.65	0.48	0.45	0.30	0.45	0.70
s_r	100%	100%	100%	100%	100%	100%	100%

TABLE V

MEAN AND STANDARD DEVIATIONS FOR METRICS FOR DIFFERENT PLANNERS RUNNING IN THE NARROW PASSAGE ENVIRONMENT.

Metric	ARA*	RRTC	LBKP	SBL	RRT	KP	RRT*
τ_t (Mean,s)	0.48	1.88	0.83	1.01	17.19	1.62	22.03
τ_t (σ,s)	0.00	1.23	0.35	0.24	8.22	0.81	8.86
τ_r (Mean,s)	0.41	1.87	0.82	1.00	17.15	1.58	22.01
τ_r (σ,s)	0.00	1.23	0.35	0.24	8.22	0.83	8.86
τ_o (Mean,s)	0.07	0.01	0.01	0.01	0.03	0.04	0.02
τ_o (σ,s)	0.00	0.00	0.01	0.00	0.02	0.10	0.01
d_n (Mean,m)	0.02	0.01	0.01	0.01	0.01	0.01	0.01
d_n (σ,m)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
l_p (Mean,radians)	15.21	18.95	18.20	17.34	18.40	17.96	16.00
l_p (σ,radians)	0.00	2.89	1.63	1.81	2.36	1.93	1.46
l_e (Mean,m)	1.41	1.76	1.65	1.59	1.77	1.59	1.66
l_e (σ,m)	0.00	0.27	0.16	0.19	0.26	0.17	0.17
κ' (Mean,radians)	0.94	0.37	0.36	0.39	0.37	0.33	0.27
κ' (σ,radians)	0.00	0.43	0.28	0.24	0.39	0.25	0.30
s_r	100%	100%	100%	100%	89%	40%	52%

once for ARA*. The results are averaged for the sampling-based motion planners, to account for the randomness of the methods. Each planner execution was given a maximum of 60 seconds to complete. All resulting trajectories were logged and metrics were computed on them. The averaged metrics are presented in the next section. All experiments were carried out on a quad-core Intel i7-2600 CPU (3.40 GHz) with 8 GB of RAM running the Ubuntu Precise distribution of Linux and ROS Fuerte.

V. RESULTS

Tables II, III, IV, V show results from all the benchmark tests[1]. We remind the reader that these results are based on implementations that use different environment representations between ARA* and the randomized algorithms, which make collision checks one order of magnitude faster for ARA*. The data for these tables was compiled from the individual planner runs on each environment. In addition to the metrics described in Table I, we include the metric l_e, which is particular to our set of experiments and corresponds to the distance traveled by the end-effector. Smoothness information was computed on plans resulting after post-processing. All reported means are over the complete set of successful motion plans for a particular environment. Because there are different goals for each environment, the standard deviation was instead computed for each goal (for successful motion plans), and the average standard deviation for the goals of a particular experiment is reported.

The tables show that all the planners are able to deal reasonably well with the environments they are benchmarked against. In the KITCHEN environment, the randomized planners are very fast in planning between the different goals. The search-based planner is slower but delivers better path lengths and also displays less variance in its path lengths (l_p, l_e). Note that both the search-based planner and RRT* are only run until a first solution is obtained, giving them less time to improve the quality of the solution. The sampling-based planners also spend lower amounts of time for *other* processing (τ_o) while the search-based planning approach needs to spend this extra time creating and processing the distance field for the environment. Note that this difference in processing times for the environment could be reduced by having each planner use a common collision representation.

The results in the INDUSTRIAL environment and the TABLETOP environment are also similar to those for the KITCHEN environments. Again, the search-based planners generate, on average, the shortest paths on the first try but take longer than the sampling based planners. The NARROW

[1]In the tables, RRTC indicates the RRTConnect algorithm, LBKP indicates the LBKPiece algorithm, and KP indicates the KPiece algorithm.

PASSAGE results are different than those for the other 3 environments. In this case, the randomized planners are generally slower than the search based planner which also generates significantly shorter paths.

VI. CONCLUSIONS

We presented a benchmarking system for comparing motion planners. Our primary motivation is to create a set of benchmarks that are easily accessible, easy to contribute to, and can be downloaded and run for new motion planning algorithms. We have shown sample results for motion planning for a mobile manipulator arm using a wide variety of planners. We hope that the informative statistics we have presented in this paper, coupled with knowledge about the environments that the robot will be operating in, will allow users of motion planning algorithms to make informed decisions when choosing an approach.

With the help of the community, we hope to add new environments, new motion planning problems and new robots to this benchmarking system. We also intend to add more informative metrics that attempt to quantify and capture other properties of motion planning that might be important. In particular, we would like to add benchmarks that test how motion planners can deal with information from real sensors and uncertainty due to noise in sensor data. We would also like to include dynamic obstacles such as humans moving through the environment and noisy data from real sensors.

Future work will also include the ability to specify cost functions for use by the planners that can handle them, e.g., the search-based planners or RRT*. We will also incorporate more true mobile manipulation tasks, i.e., tasks where a mobile base may have to move around in a cluttered environment while carrying an object and possibly manipulating it with its arms. Finally, we will also allow for easier specification of more types of constraints on the robot, e.g., visibility constraints or torque constraints.

REFERENCES

[1] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, June 2005.

[2] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[3] S. Carpin, "Randomized motion planning - a tutorial," *Intl. Journal of Robotics and Automation*, vol. 21, no. 3, pp. 184–196, 2006.

[4] I. A. Şucan and L. E. Kavraki, "On the implementation of single-query sampling-based motion planners," in *IEEE Intl. Conference on Robotics and Automation*, Anchorage, May 2010, pp. 2005–2011.

[5] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice-Hall, Englewood Cliffs, NJ, 2003.

[6] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Transactions on Man and Cybernetics*, vol. 22(2), pp. 224–241, 1992.

[7] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions onRobotics and Automation*, vol. 8, pp. 501–518, 1992.

[8] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *IEEE Intl. Conference on Robotics and Automation*, Kobe, Japan, May 2009, pp. 489–494.

[9] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Intl. Conference on Robotics and Automation*, Shanghai, China, May 2011.

[10] J. Canny, "Some algebraic and geometric computations in PSPACE," in *Annual ACM Symposium on Theory of Computing*. Chicago, Illinois, United States: ACM Press, 1988, pp. 460–469.

[11] S. Chitta, I. A. Şucan, and S. Cousins, "MoveIt! [ROS Topics]," *IEEE Robotics and Automation Magazine*, vol. 19, no. 1, pp. 18–19, March 2012. [Online]. Available: http://moveit.ros.org

[12] "Robot Operating System," http://www.ros.org.

[13] T. P. Lab, "Algorithms and Applications Group Motion Planning Puzzles," http://parasol-www.cs.tamu.edu/groups/amatogroup/benchmarks/mp/.

[14] J. Baltes, "A Benchmark Suite for Mobile Robots," in *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2000.

[15] I. Iossifidis, G. Lawitzky, S. Knoop, and R. Zllner, "Towards Benchmarking of Domestic Robotic Assistants," vol. 14. Springer Press, 2004, pp. 403–414.

[16] R. Geraerts, "Sampling-based motion planning: Analysis and path quality," Ph.D. dissertation, Utrecht University, 2006.

[17] J. Kuffner, S. Kagami, M. Inaba, and H. Inoue, "Performance benchmarks for path planning in high dimensions," in *JSM Conference on Robotics and Mechatronics*, June 2001.

[18] J. Pan, C. Lauterbach, and D. Manocha, "g-Planner: Real-time Motion Planning and Global Navigation using GPUs," in *AAAI Conference on Artificial Intelligence*, 2010.

[19] M. Reggiani, M. Mazzoli, and S. Caselli, "An Experimental Evaluation of Collision Detection Packages for Robot Motion Planning," in *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2002.

[20] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics and Automation Magazine*, 2012. [Online]. Available: http://ompl.kavrakilab.org

[21] R. Diankov and J. J. Kuffner, "OpenRAVE: A Planning Architecture for Autonomous Robotics," The Robotics Institute, Carnegie Mellon University, Technical Report CMU-RI-TR-08-34.

[22] W. Nowak, A. Zakharov, S. Blumenthal, and E. Prassler, "BRICS Deliverable D3.1: Benchmarks for Mobile Manipulation and Robust Obstacle Avoidance and Navigation, Technical Report," 2010.

[23] R. Geraerts and M. Overmars, "Creating high-quality paths for motion planning," *Intl. Journal of Robotics Research*, vol. 26, pp. 845–863, 2007.

[24] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010, software available at http://octomap.sf.net/.

[25] M. Likhachev, http://www.ros.org/wiki/sbpl, 2010.

[26] I. A. Şucan and L. E. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 116–131, 2012.

[27] G. Sánchez and J.-C. Latombe, "On delaying collision checking in PRM planning: Application to multi-robot coordination," *Intl. Journal of Robotics Research*, vol. 21, no. 1, pp. 5–26, 2002.

[28] S. Karaman and E. Frazzoli, "Sampling-based algorithms for Optimal Motion Planning," *Intl. Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[29] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE Intl. Conference on Robotics and Automation*, San Francisco, April 2000, pp. 995–1001.

[30] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *IEEE Int. Conference on Robotics and Automation*, Minneapolis, Minnesota, 2012, pp. 3859–3866.

[31] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in *Advances in Neural Information Processing Systems (NIPS) 16*. MIT Press, 2003.

[32] B. J. Cohen, G. Subramanian, S. Chitta, and M. Likhachev, "Planning for Manipulation with Adaptive Motion Primitives," in *Proceedings of the IEEE Intl. Conference on Robotics and Automation*, 2011, pp. 5478–5485.

[33] B. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for dual-arm manipulation with upright orientation constraints," in *IEEE Int. Conference on Robotics and Automation*, IEEE. Minneapolis, Minnesota: IEEE, 2012, pp. 3784–3790.

[34] Q. Ye, "The signed euclidean distance transform and its applications," in *9th Intl. Conference on Pattern Recognition*, 1988, pp. 495–499.